This book offers a critical reconstruction of the fundamental ideas and methods of artificial intelligence research. Through close attention to the metaphors of AI and their consequences for the field's patterns of success and failure, it argues for a reorientation of the field away from thought in the head and toward activity in the world.

By considering computational ideas in a philosophical framework, the author eases critical dialogue between technology and the humanities and social sciences. AI can benefit from new understandings of human nature, and in return, it offers a powerful mode of investigation into the practicalities and consequences of physical realization.

Computation and human experience

Computation and human experience

PHILIP E. AGRE
University of California, San Diego

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 2RU, UK

Published in the United States of America by Cambridge University Press, New York

www.cambridge.org
Information on this title: www.cambridge.org/9780521384322

First published 1997

A catalogue record for this publication is available from the British Library

Library of Congress Cataloguing in Publication data
Agre, Philip E.
Computation and human experience / Philip E. Agre.
p. cm. – (Learning in doing)
ISBN 0-521-38432-X (hardback). – ISBN 0-521-38603-9 (pbk.)
1. Artificial Intelligence – Data processing. 2. Human behavior.
I. Title. II. Series.
Q336.A37 1997
006.3′01 – dc20 96-14826
CIP

ISBN-13 978-0-521-38432-2 hardback
ISBN-10 0-521-38432-X hardback

ISBN-13 978-0-521-38603-6 paperback
ISBN-10 0-521-38603-9 paperback

Transferred to digital printing 2006

Joshu asked Nansen: "What is the path?"
Nansen said: "Everyday life is the path."
Joshu asked: "Can it be studied?"
Nansen said: "If you try to study, you will be far away from it."
Joshu asked: "If I do not study, how can I know it is the path?"
Nansen said: "The path does not belong to the perception world, neither does it belong to the nonperception world. Cognition is a delusion and noncognition is senseless. If you want to reach the true path beyond doubt, place yourself in the same freedom as sky. You name it neither good nor not-good."

At these words Joshu was enlightened.

Mumon's comment: Nansen could melt Joshu's frozen doubts at once when Joshu asked his questions. I doubt though if Joshu reached the point that Nansen did. He needed thirty more years of study.

In spring, hundreds of flowers; in autumn, a harvest moon;
In summer, a refreshing breeze; in winter, snow will accompany you.
If useless things do not hang in your mind,
Any season is a good season for you.

Ekai, *The Gateless Gate,* 1228

Contents

Preface

Artificial intelligence has aroused debate ever since Hubert Dreyfus wrote his controversial report, *Alchemy and Artificial Intelligence* (1965). Philosophers and social scientists who have been influenced by European critical thought have often viewed AI models through philosophical lenses and found them scandalously bad. AI people, for their part, often do not recognize their methods in the interpretations of the critics, and as a result they have sometimes regarded their critics as practically insane.

When I first became an AI person myself, I paid little attention to the critics. As I tried to construct AI models that seemed true to my own experience of everyday life, however, I gradually concluded that the critics were right. I now believe that the substantive analysis of human experience in the main traditions of AI research is profoundly mistaken. My reasons for believing this, however, differ somewhat from those of Dreyfus and other critics, such as Winograd and Flores (1986). Whereas their concerns focus on the analysis of language and rules, my own concerns focus on the analysis of action and representation, and on the larger question of human beings' relationships to the physical environment in which they conduct their daily lives. I believe that people are intimately involved in the world around them and that the epistemological isolation that Descartes took for granted is untenable. This position has been argued at great length by philosophers such as Heidegger and Merleau-Ponty; I wish to argue it technologically.

This is a formidable task, given that many AI people deny that such arguments have any relevance to their research. A computer model, in their view, either works or does not work, and the question is a purely technical one. Technical practice is indeed a valuable way of knowing, and my goal is not to replace it but to deepen it. At the same time, AI has had great trouble understanding certain technical impasses that philosophical methods both predict and explain. The problem is one of consciousness: the AI community has lacked the intellectual tools that it needs to comprehend its own difficulties. What is needed, I will argue, is a critical technical practice – a technical practice for which critical reflection upon the practice is part of the practice itself. Mistaken ideas about human nature lead to recurring patterns of technical difficulty; critical analysis of the mistakes contributes to a recognition of the difficulties. This is obvious enough for AI projects that seek to model human life, but it is also true for AI projects that use ideas about human beings as a heuristic resource for purely technical ends.

Given that no formal community of critical technical practitioners exists yet, this book necessarily addresses two very different audiences, technical and critical. The technical audience consists of technical practitioners who, while committed to their work, suspect that it might be improved using intellectual tools from nontechnical fields. The critical audience consists of philosophers and social scientists who, while perhaps unhappy with technology as it is, suspect that they can make a positive contribution to its reform.

In my experience, the first obstacle to communication between these audiences is the word "critical," which for technical people connotes negativity and destruction. It is true that critical theorists are essentially suspicious; they dig below the surface of things, and they do not expect to like what they find there. But critical analysis quickly becomes lost unless it is organized and guided by an affirmative moral purpose. My own moral purpose is to confront certain prestigious technical methodologies that falsify and distort human experience. The purpose of critical work, simply put, is to explain how this sort of problem arises. Technical people frequently resist such inquiries because they seem to involve accusations of malfeasance. In one sense this is true: we all bear some responsibility for the unintended consequences of our actions. But in another sense it is false: critical research draws attention to structural and cultural levels of explanation – to the things that happen through our actions but that exist beneath our conscious awareness.

In the case of AI, I will argue that certain conceptions of human life are reproduced through the discourses and practices of technical work. I will also argue that the falsehood of those conceptions can be discerned in their impracticability and that new critical tools can bring the problem into the consciousness of the research community. The point, therefore, is not to invoke Heideggerian philosophy, for example, as an exogenous authority that supplants technical methods. (This was not Dreyfus's intention either.) The point, instead, is to expand technical practice in such a way that the relevance of philosophical critique becomes evident *as a technical matter.* The technical and critical modes of research should come together in this newly expanded form of critical technical consciousness.

I am assuming, then, that both the technical and critical audiences for this book are sympathetic to the idea of a critical technical practice. Writing for these two audiences simultaneously, however, has meant reckoning with the very different genre expectations that technical and critical writing have historically entailed. Technical texts are generally understood to report work that their authors have done; they are focused on machinery in a broad sense, be it hardware, software, or mathematics. They open by making claims – "Our machinery can do such and such and others' cannot" – and they confine themselves to demonstrating these claims in a way that others can replicate. They close by sketching future work – more problems, more solutions. Critical texts, by contrast, *are* the work that their authors have done. Their textuality is in the foreground, and they are focused on theoretical categories. They open by situating a problematic in an intellectual tradition, and they proceed by narrating their materials in a way that exhibits the adequacy of certain categories and the inadequacy of others. They close with a statement of moral purpose.

When a technical audience brings its accustomed genre expectations to a critical text or vice versa, wild misinterpretations often result, and I have spent too many years revising this text in an attempt to avoid seeming either scandalous or insane. The result of this effort is a hybrid of the technical and critical genres of writing. By intertwining these two strands of intellectual work, I hope to produce something new – the discursive forms of a critical technical practice. This being a first attempt, I will surely satisfy nobody. Everyone will encounter whole chapters that seem impossibly tedious and other chapters that seem unreasonably compressed. I can only ask forbearance and hope that each reader will imagine the plight of other readers who are approaching the book from the opposite direction.

This amalgamation of genres has produced several curious effects, and rather than suppress these effects I have sought to draw them out and make them explicit. One such effect is a frequent shifting between the levels of analysis that I will call reflexive, substantive, and technical. In one section I will explicate the ground rules for a contest among substantive theories, and on the next I will advocate one of these theories over the others. In one chapter I will criticize technical uses of language, and in the next I will start using language in precisely those ways in order to show where it leads.

Another such effect is a frequent overburdening of terms. Sometimes, as with "logic," a term has evolved in different directions within the critical and technical traditions, so that it is jarring to bring the divergent senses together in the same text. In other cases, as with "problem" and "model," technical discourse itself employs a term in at least two wholly distinct senses, one methodological and one substantive. And in yet other cases, technical and critical vocabulary together have drawn so many words out of circulation that I cannot help occasionally using some of them in their vernacular senses as well. I have tried to make the senses of words clear from context, and in most cases I have provided notes that explain the distinctions.

The peculiarity of my project might also be illustrated by a comparison with Paul Edwards's (1996) outstanding recent history of AI. In the language of social studies of technology (Staudenmaier 1985), Edwards opposes himself to "internalist" studies that explain the progress of a technical field purely through the logic of its ideas or the economics of its industry. He observes that internalist studies have acquired a bad name from their association with the sort of superficial, self-justifying history that Kuhn (1962) lamented in his analysis of "normal science." In response to this tendency, Edwards (1996: xiv) positions his work as a "counterhistory," drawing out the interactions among cultural themes, institutional forces, and technical practices that previous studies have inadvertently suppressed.

While I applaud this kind of work, I have headed in an entirely different direction. This book is not only an internalist account of research in AI; it is actually a work *of* AI – an intervention within the field that contests many of its basic ideas while remaining fundamentally sympathetic to computational modeling as a way of knowing. I have written a counterhistory of my own. By momentarily returning the institutional dimensions of AI to the periphery, I hope to permit the esoteric practices of the field to emerge in their inner logic. Only then will it be possible to appreciate their real power and their great recalcitrance.

Not only is the daily work of AI firmly rooted in practices of computer system design, but AI researchers have also drawn upon deeper currents in Western thought, both reproducing and transcending older ideas despite their conscious intentions. Would-be AI revolutionaries are continually reinventing the wheel, regardless of their sources of funding, and I hope to convey some idea of how this happens. AI is not, however, intellectually consistent or static; to the contrary, its development can be understood largely as successive attempts to resolve internal tensions in the workings of the field. I want to recover an awareness of those tensions through a critical exhumation of their sources.

My expository method is hermeneutic. I want to exhibit AI as a coherent totality, and then I want to turn it inside out. To do this, I have painted a big picture, examining the most basic concepts of computing (bits, gates, wires, clocks, variables, seriality, abstraction, etc.) and demonstrating their connection to seemingly more contentious ideas about such matters as perception, reasoning, and action. I will pass through some of these topics several times in different ways. My purpose is not to produce an exhaustive linear history but to assemble a complex argument. Technical ways of knowing are irreducibly intuitive, and each pass will open up a new horizon of intuition based on technical experience. AI has told variations on a single story about human beings and their lives; I believe that this story is wrong, and by forcing its tensions to the surface I hope to win a hearing for a completely different story. My goal, however, is not to convince everyone to start telling that same story. Computer modeling functions as a way of knowing only if the modelers are able to hear what the materials of technical work are trying to tell them, and if they respond by following those messages wherever they lead. The only way out of a technical impasse is through it.

I owe many debts. John Brace taught me mathematics, and Chuck Rieger and Hanan Samet introduced me to the field of AI. At MIT, many of the ideas in this book arose through conversations with David Chapman. He is also the principal author of the computer program that I discuss in Chapter 13. John Batali, Gary Drescher, Ken Haase, and Ian Horswill were early members of the debating society in which I first practiced the arguments. I appreciate the supportively skeptical comments of Randy Davis, Ken Forbus, Pat Hayes, and Dan Weld, and the philosophical assistance of Jon Doyle.

As the internal problems in AI became clear, I went looking for people who could explain them to me. Hubert Dreyfus, Harold Garfinkel, and Lucy Suchman introduced me to the phenomenological tradition; Jean Comaroff, John Comaroff, Bill Hanks, and Jean Lave introduced me to the dialectical tradition; and George Goethals introduced me to the psychoanalytic tradition.

Parts of this book began life in dissertation research that I conducted at the MIT Artificial Intelligence Laboratory, and I am indebted to Mike Brady, Rod Brooks, Gerald Jay Sussman, and Patrick Winston for their guidance. Aside from the people I have already mentioned, Jonathan Amsterdam, Mike Dixon, Carl Feynman, and Eric Saund wrote helpful comments on drafts of that work. I was supported for most of that time by a graduate fellowship from the Fannie and John Hertz Foundation, and support for the AI Laboratory's artificial intelligence research was provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K-0124.

Work on this manuscript began at the University of Chicago, and I appreciate the support and comments of Tim Converse, Kris Hammond, Charles Martin, Ron McClamrock, and the other members of the University of Chicago AI group. As the manuscript evolved and became a book, it benefited greatly from extensive comments by Steve Bagley, David Chapman, Julia Hough, Frederic Laville, Lucy Suchman, and Jozsef Toth, and from the assistance of John Batali, Margaret Boden, Bill Clancey, David Cliff, Mike Cole, Bernard Conein, Johan de Kleer, Bruce Donald, Yrjö Engeström, Jim Greeno, Judith Gregory, David Kirsh, Rob Kling, Jim Mahoney, Ron McClamrock, Donald Norman, Beth Preston, Stan Rosenschein, Penni Sibun, Rich Sutton, Michael Travers, and Daniel Weise. Paul Edwards and Brian Smith were kind enough to provide me with drafts of their own books prior to publication. Mario Bourgoin directed me to the epigraph (which is taken from Paul Reps, ed., *Zen Flesh, Zen Bones* [Anchor Press, n.d.], by permission of Charles E. Tuttle Publishing Company of Tokyo, Japan). My apologies to anybody I might have omitted.

1 Introduction

Activity

Computational inquiry into human nature originated in the years after World War II. Scientists mobilized into wartime research had developed a series of technologies that lent themselves to anthropomorphic description, and once the war ended these technologies inspired novel forms of psychological theorizing. A servomechanism, for example, could aim a gun by continually sensing the target's location and pushing the gun in the direction needed to intercept it. Technologically sophisticated psychologists such as George Miller observed that this feedback cycle could be described in human-like terms as pursuing a purpose based on awareness of its environment and anticipation of the future.[1] New methods of signal detection could likewise be described as making perceptual discriminations, and the analytical tools of information theory soon provided mathematical ways to talk about communication. In the decades after the war, these technical ideas provided the intellectual license for a counterrevolution against behaviorism and a restoration of scientific status to human mental life. The explanatory power of these ideas lay in a suggestive confluence of metaphor, mathematics, and machinery. Metaphorical attributions of purpose were associated with the mathematics of servocontrol and realized in servomechanisms; metaphorical attributions of discrimination were associated with the mathematics of signal and noise and realized in communications equipment; and metaphorical attributions of communication were associated with the mathematics of information theory and realized in coding devices. The new psychology sought to describe human beings using vocabulary that could be metaphorically associated with technologically realizable mathematics.

The development of the stored-program digital computer put this project into high gear. It is a commonplace that the computer contributed a potent stock of metaphors to modern psychology, but it is important to understand just how these metaphors informed the new research. The outlines of the project were the same as with servocontrol, signal detection, and information theory: a bit of metaphor attached to a bit of mathematics and realized in a machine whose operation could then be narrated using intentional vocabulary.[2] But the digital computer both generalized and circumscribed this project. By writing computer programs, one could physically realize absolutely any bit of finite mathematics one wished. The inside of the computer thus became an imaginative landscape in which programmers could physically realize an enormous variety of ideas about the nature of thought. Fertile as this project was, it was also circumscribed precisely by the boundaries of the computer. The feats of physics and chemistry that supported the digital abstraction operated inside the computer, and not outside.

In this way, a powerful dynamic of mutual reinforcement took hold between the technology of computation and a Cartesian view of human nature, with computational processes inside computers corresponding to thought processes inside minds. But the founders of computational psychology, while mostly avowed Cartesians, actually transformed Descartes's ideas in a complex and original way. They retained the radical experiential inwardness that Descartes, building on a long tradition, had painted as the human condition. And they retained the Cartesian understanding of human bodies and brains as physical objects, extended in space and subject to physical laws. Their innovation lay in a subversive reinterpretation of Descartes's ontological dualism (Gallistel 1980: 6–7). In *The Passions of the Soul*, Descartes had described the mind as an extensionless res cogitans that simultaneously participated in and transcended physical reality. The mind, in other words, interacted causally with the body, but was not itself a causal phenomenon. Sequestered in this nether region with its problematic relationship to the physical world, the mind's privileged object of contemplation was mathematics. The "clear and distinct ideas" that formed the basis of Descartes's epistemology in the *Meditations* were in the first instance *mathematical* ideas (Rorty 1979: 57–62; cf. Heidegger 1961 [1927]: 128–134). Of course, generations of mechanists beginning with Hobbes, and arguably from antiquity, had described human thought in monistic terms as the workings of machinery (Haugeland 1985: 23). But these theorists were always constrained by the primitive ideas about machinery that were available to them. Descartes's physiology suffered in this way, but not his psychology. Although they paid little heed to the prescriptive analysis of thought that Descartes had offered,[3] the founders of computational psychology nonetheless consciously adopted and reworked the broader framework of Descartes's theory, starting with a single brilliant stroke. The mind does not simply contemplate mathematics, they asserted; the mind is *itself* mathematical, and the mathematics of mind is precisely a technical specification for the causally explicable operation of the brain.

This remarkable proposal set off what is justly called a "revolution" in philosophy and psychology as well as in technology. Technology is in large measure a cultural phenomenon, and never has it been more plainly so than in the 1950s. Computational studies in that decade were studies of faculties of *intelligence* and processes of *thought*, as part of a kind of cult of cognition whose icons were the rocket scientist, the symbolism of mathematics, and the computer itself.[4] The images now strike us as dated and even camp, but we are still affected by the technical practice and the interpretation of human experience around which artificial intelligence, or AI, was first organized.

I wish to investigate this confluence of technology and human experience. The philosophical underside of technology has been deeply bound up with larger cultural movements, yet technical practitioners have generally understood themselves as responding to discrete instrumental "problems" and producing technologies that have "effects" upon the world. In this book I would like to contribute to a *critical technical practice* in which rigorous reflection upon technical ideas and practices becomes an integral part of day-to-day technical work itself.

I will proceed through a study in the intellectual history of research in AI. The point is not to exhaust the territory but to focus on certain chapters of AI's history that help illuminate the internal logic of its development as a technical practice.[5] Although it will be necessary to examine a broad range of ideas about thought, perception, knowledge, and their physical realization in digital circuitry, I will focus centrally on computational theories of action. This choice is strategic, inasmuch as action has been a structurally marginal and problematic topic in AI; the recurring difficulties in this computational research on action, carefully interpreted, motivate critiques that strike to the heart of the field as it has historically been constituted. I aim to reorient research in AI away from *cognition* – abstract processes in the head – and toward *activity* – concrete undertakings in the world. This is not a different subject, but a different approach to the same subject: different metaphors, methods, technologies, prototypes, and criteria of evaluation. Effecting such a reorientation will require technical innovation, but it will also require an awareness of the structure of ideas in AI and how these ideas are bound up with the language, the methodology, and the value systems of the field.

Roughly speaking, computational research into activity seeks technical ideas about action and representation that are well suited to the special requirements of *situated, embodied agents* living in the physical world. The "agents" could be robots we would like to build or creatures we would like to understand. The word *agent,* though common in AI, does not appeal to everyone. Its advantage is its ambiguity – robots, insects, cats, and people are all agents.[6] Such vocabulary tacitly promises, of course, that computation provides useful ways of talking about robots, insects, cats, and people at the same time without reducing all of them to a bloodless technical order. In any event, I will have little to say about insects and cats. To say that an agent is *situated* is to emphasize that its actions make little sense outside of the particular situation in which it finds itself in the physical and social world; it is always provided with particular materials and involved with particular other agents. To say that an agent is *embodied* is simply to say that it has a body. Even better, following Merleau-Ponty (1962 [1945]), it *is* a body or *exists as* a body. As a physical being, it has a definite location, limited experience, and finite abilities. It is *in* the world, *among* the world's materials, and *with* other agents. The claim is not simply that these things are true (hardly anybody would deny them), but also that taking them seriously requires an overhaul of basic ideas about both computation and activity.

My project is both critical and constructive. By painting computational ideas in a larger philosophical context, I wish to ease critical dialogue between technology and the humanities and social sciences (Bolter 1984; Güzeldere and Franchi 1995). The field of AI could certainly benefit from a more sophisticated understanding of itself as a form of inquiry into human nature. In exchange, it offers a powerful mode of investigation into the practicalities and consequences of physical realization.

My recommendation of a shift of focus from cognition to activity converges with a number of other intellectual trends, each of which is also founded in a critique of Cartesianism. These include the otherwise disparate traditions that descend from Heidegger's phenomenological analysis of routine activity, Vygotsky's theory of human development, and Garfinkel's studies of the interactional construction of social reality.[7] Each of these schools of thought has attempted to replace the philosophical opposition between a self-contained perceiving subject and an independent external object by describing our relationships to things as fundamentally bound up with their role in our ongoing projects, which in turn are defined by our cultures, located in forms of embodied activity, and acquired through socialization into a system of cultural practices. As AI reorients itself toward the study of activity, it will be able to engage in mutually beneficial dialogue with these traditions of research. This process begins with computational ways of thinking about routine activity.

Planning

Although the AI tradition has placed its principal emphasis on processes it conceives of as occurring entirely within the mind, there does exist a more or less conventional computational account of action. The early formulation of this account that had the most pervasive influence was George Miller, Eugene Galanter, and Karl Pribram's book, *Plans and the Structure of Behavior* (1960).[8] These authors rejected the extreme behaviorist view that the organized nature of activity results from isolated responses to isolated stimuli. Instead, they adopted the opposite extreme view that the organization of human activity results from the execution of mental structures they called Plans. Plans were *hierarchical* in the sense that a typical Plan consisted of a series of smaller sub-Plans, each of which consisted of yet smaller sub-Plans, and so forth, down to the primitive Plan steps, which one imagines to correspond to individual muscle movements. (Miller, Galanter, and Pribram capitalized the word "Plan" to distinguish their special use of it, especially in regard to the hierarchical nature of Plans, from vernacular usage. Subsequent authors have not followed this convention. I will follow it when I mean to refer specifically to Miller, Galanter, and Pribram's concept.)

What is a Plan? "A Plan is any hierarchical process in the organism that can control the order in which a sequence of operations is to be performed" (Miller et al. 1960: 16). They state, as a "scientific hypothesis" about which they are "reasonably confident," that a Plan is "essentially the same as a program for a computer," a connotation the term has carried to the present day. Shortly thereafter, though, they state that "we shall also use the term 'Plan' to designate a rough sketch of some course of action, just the major topic headings in the outline, as well as the completely detailed specification of every detailed operation" (Miller et al. 1960: 17). Thus a new Plan's hierarchical structure need not initially reach down to the most primitive actions, though the hierarchy must be constructed in full detail by the time any given step of it is executed. They define execution by saying that "a creature is executing a particular Plan when in fact that Plan is controlling the sequence of operations he is carrying out" (Miller et al. 1960: 17).

Miller, Galanter, and Pribram applied the term "Plan" as broadly as they could. In considering various aspects of everyday life, they focused everywhere on elements of intentionality, regularity, and goal-directedness and interpreted each one as the manifestation of a Plan. As with the servos, radars, and codes that first inspired Miller and his contemporaries in the 1940s, the concept of a Plan combined the rhetoric of structured behavior with the formalisms of programming and proposed that the latter serve as models of biological systems. A great difficulty in evaluating this proposal is the imprecise way in which Miller, Galanter, and Pribram used words like "Plan." They demonstrated that one can find aspects of apparent planfulness in absolutely any phenomenon of human life. But in order to carry out this policy of systematic assimilation, important aspects of activity had to be consigned to peripheral vision. These marginalized aspects of activity were exactly those which the language of Plans and their execution tends to deemphasize.

These ideas had an enormous influence on AI, but with some differences of emphasis. Although they occasionally employ the term "planning," Miller, Galanter, and Pribram provide no detailed theory of the construction of new Plans. The AI tradition, by contrast, has conducted extensive research on plan construction but has generally assumed that execution is little more than a simple matter of running a computer program. What has remained is a definite view of human activity that has continued, whether implicitly or explicitly, to suffuse the rhetoric and technology of computational theories of action. In place of this view, I would like to substitute another, one that follows the anthropologically motivated theoretical orientations of Suchman (1987) and Lave (1988) in emphasizing the situated nature of human action. Let me contrast the old view and the new point by point:

- Why does activity appear to be organized?

  Planning view: If someone's activity has a certain organization, that is because the person has constructed and executed a representation of that activity, namely a plan.

  Alternative: Everyday life has an orderliness, a coherence, and patterns of change that are emergent attributes of people's interactions with their worlds. Forms of activity might be influenced by representations but are by no means mechanically determined by them.

- How do people engage in activity?

  Planning view: Activity is fundamentally planned; contingency is a marginal phenomenon. People conduct their activity by constructing and executing plans.

  Alternative: Activity is fundamentally improvised; contingency is the central phenomenon. People conduct their activity by continually redeciding what to do.

- How does the world influence activity?

  Planning view: The world is fundamentally hostile, in the sense that rational action requires extensive, even exhaustive, attempts to anticipate difficulties. Life is difficult and complicated, a series of problems to be solved.

  Alternative: The world is fundamentally benign, in the sense that our cultural environment and personal experiences provide sufficient support for our cognition that, as long as we keep our eyes open, we need not take account of potential difficulties without specific grounds for concern. Life is almost wholly routine, a fabric of familiar activities.

The alternative view of human activity that I have sketched here contains a seeming tension: how can activity be both improvised and routine? The answer is that the routine of everyday life is not a matter of performing precisely the same actions every day, as if one were a clockwork device executing a plan. Instead, the routine of everyday life is an emergent phenomenon of moment-to-moment interactions that work out in much the same way from day to day because of the relative stability of our relationships with our environments.

My sketched alternative also denies a central role to the use of plans. People certainly use plans. But real plans are nothing like computer programs. Sensibly organized goal-directed activity need not result from the use of a plan. And plans never serve as direct specifications of action. Instead, a plan is merely one resource among many that someone might use in deciding what to do (Suchman 1987). Before and beneath any use of plans is a continual process of moment-to-moment improvisation. "Improvisation," as I will employ the term, might involve ideas about the future and it might employ plans, but it is always a matter of deciding what to do *now*. Indeed, the use of plans is a relatively peripheral phenomenon and not a principal focus here.[9]

To speak of a "planning view" is misleading in one respect: few people are aware of having committed themselves to such a view. Future chapters will explain more precisely the sense in which the planning view has governed research in AI. For the time being, it will be helpful to consider Heidegger's (1961 [1927]) account of why the emergence of something like the planning view is nearly inevitable. Most of us, Heidegger observes, spend our days immersed in practical concerns. We are concerned with the traffic, the paperwork, the dust, the celery – with the objects that we encounter as we pursue our goals and enact our identities. We find it natural, therefore, to see the world as a constellation of objects. Moreover, the occasions on which particular objects really come to our attention are not representative of activity as a whole. Sometimes we momentarily detach ourselves from our daily concerns to contemplate an object in a special way – as, for example, a work of art. And sometimes an object simply becomes obstinate; perhaps it is broken, or missing, or not the right size. In these situations, we confront the object as a stranger – as something very much separate from us. It is *problems* that attract our attention, and problems play a wildly disproportionate role in the stories we tell about our lives. We hardly notice the vast background of ordinary, routine, unproblematic activities from which our lives are largely made. Even when a problem does arise, the detection and resolution of the problem both consist of concrete activities that are mostly routine. Because this unproblematic background of routine activity goes largely unnoticed, we can succumb to the illusion that life is basically a series of problems, that situations in life typically require thinking and planning, and that our normal way of relating to objects involves detached contemplation. This illusion does not simply arise in individual experience; it is also handed down through metaphors, narrative conventions, philosophical systems, and other cultural constructs. It is this illusory view of life – the planning view – that first crystallized in its modern form in Descartes and that originally defined the tacit agenda for research in AI. Yet, I will argue, the planning view is inadequate both as an account of human life and as an approach to computational modeling.

It is hard to know, of course, how to evaluate Heidegger's argument. Perhaps we should treat it as a just-so story; Heidegger, in any case, presented it as a phenomenological description and a reconstruction of the history of philosophy, not a logical deduction from premises or a scientific inference from evidence. For our purposes here, though, that is enough. Heidegger's story is useful in several ways. It confers an overall sense on the more detailed analyses of Descartes and other theorists of mechanism. It also directs our attention heuristically to technical difficulties that might otherwise have gone undiagnosed or misunderstood. Above all, it helps us cultivate an awareness of our own experience as human beings. Heidegger's crucial insight is that philosophical ideas tend to formalize the ways we experience our lives; if we experience our lives in superficial ways then our philosophies will be correspondingly superficial. The same reasoning applies to computational models, which (whether the model-builders realize it or not) are derived from philosophical theories and guided in their development by experiences of everyday life. Better descriptions of everyday life do not disprove technical ideas, but they do motivate different intuitions, and they also help evaluate the appeals to everyday intuition that are found throughout AI research. AI's pervasive focus on problems, for example, aligns with the unreflective emphasis on problems that Heidegger finds in the modern experience of everyday life. By failing to place problems in the context of an unproblematic background, AI may fall prey to a mistaken conception of them and an excessive emphasis on attempts to solve them. The point in each case is not to debunk AI or technology in general, but to gain what Heidegger would call a "free relation" to it, so that technological modes of thinking do not colonize our awareness of our lives (Dreyfus 1995). Let us turn to the methodological issues that arise as AI research is rethought in this way.

Why build things?

Every discipline has its distinctive ways of knowing, which it identifies with the activities it regards as its own: anthropologists do fieldwork, architects design buildings, monks meditate, and carpenters make things out of wood. Each discipline wears its defining activity as a badge of pride in a craftworker's embodied competence. It will be said, "You can read books all your life, but you don't really know about it until you do it." Disciplinary boundaries are often defined in such ways – you are not an anthropologist unless you have spent a couple years in the field; you are not an architect unless you have built a building; and so forth – and neighboring disciplines may be treated with condescension or contempt for their inferior methods. Each discipline's practitioners carry on what Schön (1983: 78) would call "reflective conversations" with their customary materials, and all of their professional interactions with one another presuppose this shared background of sustained practical engagement with a more or less standard set of tools, sites, and hassles. Defining a discipline through its own special activity carries risks. If a disciplinary community cultivates invidious contrasts between its own methods and those of other fields, it will surely become inbred and insular, emptied by hubris and intolerance. If it is guided by critical reflection on its practices and presuppositions, however, it has at least a chance of continually deepening its self-awareness, renewing itself through interdisciplinary dialogue without losing its distinctive advantages. The culture of any particular discipline will presumably be found somewhere between these extremes.

The discipline in question here is computational modeling, and specifically AI. Although I will criticize certain computational ideas and practices at great length, my enterprise is computational nonetheless. AI's distinctive activity is building things, specifically computers and computer programs. Building things, like fieldwork and meditation and design, is a way of knowing that cannot be reduced to the reading and writing of books (Chapman 1991: 216–217). To the contrary, it is an enterprise grounded in a routine daily practice. Sitting in the lab and working on gadgets or circuits or programs, it is an inescapable fact that some things can be built and other things cannot. Likewise, some techniques scale up to large tasks and others do not; and some devices operate robustly as environmental conditions fluctuate, whereas others break down. The AI community learns things by cultivating what Keller (1983) calls a "feeling for the organism," gradually making sense of the resulting patterns of what works and what does not. Edwards (1996: 250, italics in the original) rightly emphasizes that much of AI's practitioners' technical framework "emerged not abstractly but *in their experiences with actual machines.*" And Simon (1969: 20) speaks of computing as an "empirical science" – a science of design.

I take an unusual position on the nature of computation and computational research. For my purposes, *computation* relates to the analysis and synthesis of especially complicated things.[10] These analytic and synthetic practices are best understood as nothing less grand or more specific than an inquiry into physical realization as such. This fact can be lost beneath ideologies and institutions that define computation in some other way, whether in terms of Turing machines, mathematical abstraction, intentionality, symbolic reasoning, or formal logic. Nonetheless, what truly founds computational work is the practitioner's evolving sense of what can be built and what cannot. This sense, at least on good days, is a glimpse of reality itself. Of course, we finite creatures never encounter this "reality" except through the mediation of a historically specific ensemble of institutions, practices, genres, ideologies, tools, career paths, divisions of labor, understandings of "problems" and "solutions," and so forth. These mediating systems vary historically through both practical experience and broader shifts of social climate. Nonetheless, at each point the technologist is pushing up against the limits of a given epoch's technology, against the limits of physical reality conceived and acted upon in a specific way. These limits are entirely real. But they are not simply a product of reality-in-itself; nor are they simply internal consequences of the idea-systems on their own, considered in abstraction from particular attempts to get things to work.

This is the sense in which people engaged in technical work are – and, I think, must be – philosophical realists. The something-or-other that stands behind each individual encounter with the limits of physical realization I would like to call *practical reality.* Practical reality is something beyond any particular model or ontology or theory. A given model might seem like the final word for years or centuries, but ultimately the limit-pushing of technical work will reveal its margins. The resulting period of stumbling and improvisation will make the previously taken-for-granted model seem contingent: good enough perhaps for some purposes, but no longer regarded (if it ever has been) as a transparent description of reality. Much of the sophistication of technical work in mechanical engineering and semiconductor physics, for example, lies in the astute choice of models for each purpose.

Technical communities negotiate ceaselessly with the practical reality of their work, but when their conceptions of that reality are mistaken, these negotiations do not necessarily suffice to set them straight. Computational research, for its part, has invested an enormous amount of effort in the development of a single model of computation: the dual scheme of abstraction and implementation that I will describe in Chapter 4. This framework has motivated a multitude of technical proposals, but it has also given rise to recurring patterns of technical trouble. Although computationalists do possess a certain degree of critical insight into the patterns of trouble that arise in their work, they also take a great deal for granted. Beneath the everyday practices of computational work and the everyday forms of reasoning by which computationalists reflect on their work, a vast array of tacit commitments lies unexamined. Each of these commitments has its margins, and the field's continual inadvertent encounters with these margins have accumulated, each compounding the others, to produce a dull sense of existential chaos. Nobody complains about this, for the simple reason that nobody has words to identify it. As successive manifestations of the difficulty have been misinterpreted and acted upon, the process has become increasingly difficult to disentangle or reverse.

In trying to set things right, a good place to start is with AI researchers' understanding of their own distinctive activity: building computer systems. AI people, by and large, insist that nothing is understood until it has been made into a working computer system. One reason to examine this insistence critically is its association with research values that disrupt interdisciplinary communication. This disruption goes in two directions – from the inside out (i.e., from AI to the noncomputational world) and from the outside in (i.e., the other way round) – and it is worth considering these two directions separately.

Research based on computer modeling of human life often strikes people from other fields as absurd. AI studies regularly oversimplify things, make radically counterfactual assumptions, and focus excessive attention on easy cases. In a sense this is nothing unusual: every field has to start somewhere, and it is usually easier to see your neighbor's leading assumptions than your own. But in another sense things really are different in AI than elsewhere. Computer people believe only what they can build, and this policy imposes a strong intellectual conservatism on the field. Intellectual trends might run in all directions at any speed, but computationalists mistrust anything unless they can nail down all four corners of it; they would, by and large, rather get it precise and wrong than vague and right. They often disagree about *how much* precision is required, and *what kind* of precision, but they require ideas that can be assimilated to computational demonstrations that actually get built. This is sometimes called the *work ethic:* it has to work. To get anything nailed down in enough detail to run on a computer requires considerable effort; in particular, it requires that one make all manner of arbitrary commitments on issues that may be tangential to the current focus of theoretical interest. It is no wonder, then, that AI work can seem outrageous to people whose training has instilled different priorities – for example, conceptual coherence, ethnographic adequacy, political relevance, mathematical depth, or experimental support. And indeed it is often totally mysterious to outsiders what canons of progress and good research *do* govern such a seemingly disheveled enterprise. The answer is that good computational research is an evolving conversation with its own practical reality; a new result gets the pulse of this practical reality by suggesting the outlines of a computational explanation of some aspect of human life. The computationalist's sense of bumping up against reality itself – of being *compelled* to some unexpected outcome by the facts of physical realizability as they manifest themselves in the lab late at night – is deeply impressive to those who have gotten hold of it. Other details – conceptual, empirical, political, and so forth – can wait. That, at least, is how it feels.

How, then, can we keep what is good about AI's methods without falling into disciplinary chauvinism? We can start by rejecting the idea, derived from the institutions of engineering, that the sole test of computational research is whether it produces *solutions* to *problems.* These terms presuppose and conflate two unfortunate doctrines: Turing's model of computation (problems as input–output mappings) and the instrumentalism of engineering (problems as the unquestioned goals of one's employer).[11] On this view, the work ethic is both right and wrong. It is right in the sense that building things is a valuable way of knowing. It is wrong, however, in that "working" is too narrow as a criterion of success. It is well known that a technical method can be perfectly well defined and perform exactly according to its specification, while at the same time being as wrongheaded as you like (Jirotka and Goguen 1994). The point of a critical technical practice is that learning from technical experience takes sustained, sophisticated thought. Technical methods do not simply "work" or "fail to work." The picture is always mixed. Every method has its strengths and weaknesses, its elegance and its clumsiness, its subtle patterns of success and failure. These things ought to be the subject of intellectual discussion and dispute, both in particular cases and as general matters of methodology – and not as a separate category of research but as something continuous with the presentation of technical projects.

The work ethic needs to be qualified in another way as well. To understand what is implied in a claim that a given computer model "works," one must distinguish two senses of "working." The first, narrow sense, again, is "conforms to spec" – that is, it works if its behavior conforms to a pregiven formal-mathematical specification. Since everything is defined mathematically, it does not matter what words we use to describe the system; we could use words like "plan," "learn," and "understand," or we could use words like "foo," "bar," and "baz." In fact, programmers frequently employ nonsense terms like these when testing or demonstrating the logical behavior of a procedure. Local programming cultures will frequently invent their own sets of commonly used nonsense terms; where I went to school, the customary nonsense terms also included "blort," "quux," and "eep." But nonsense terms are not adequate for the second, broad sense of "working," which depends on specific words of natural language. As I mentioned at the very beginning, an AI system is only truly regarded as "working" when its operation can be narrated in intentional vocabulary, using words whose meanings go beyond the mathematical structures. When an AI system "works" in this broader sense, it is clearly a discursive construction, not just a mathematical fact, and the discursive construction succeeds only if the community assents.[12] Critics of the field have frequently complained that AI people water down the meanings of the vernacular terms they employ, and they have sought to recover the original force of those terms, for example through the methods of ordinary language philosophy (Button, Coulter, Lee, and Sharrock 1995). But these critics have had little influence on the AI community's own internal standards of semantic probity.

The community is certainly aware of the issue; McDermott (1981: 144), for example, forcefully warns against "wishful mnemonics" that lead to inflated claims. But these warnings have had little practical effect, and the reward systems of the field still depend solely on the production of technical schemata – mathematically specified mechanisms and conventions for narrating these mechanisms' operation in natural language. The point, in any case, is that the practical reality with which AI people struggle in their work is not just "the world," considered as something objective and external to the research. It is much more complicated than this, a hybrid of physical reality and discursive construction. The trajectory of AI research can be shaped by the limitations of the physical world – the speed of light, the three dimensions of space, cosmic rays that disrupt memory chips – and it can also be shaped by the limitations of the discursive world – the available stock of vocabulary, metaphors, and narrative conventions. Technical tradition consists largely of intuitions, slogans, and lore about these hybrids, which AI people call "techniques," "methods," and "approaches"; and technical progress consists largely in the growth and transformation of this body of esoteric tradition. This is the sense in which computers are "language machines" (e.g., Edwards 1996: 28).[13] Critical reflection on computer work is reflection upon both its material and semiotic dimensions, both synchronically and historically.

More specifically, the object of critical reflection is not computer programs as such but rather the *process* of technical work. Industrial software engineering is governed by rigid scripts that are dictated more by bureaucratic control imperatives than by the spirit of intellectual inquiry (Kraft 1977), but research programming is very much an improvisation – a reflective conversation with the materials of computational practice. As it expands its collective understanding of this process, AI will become aware of itself as an intellectual enterprise whose concerns are continuous with those of numerous other disciplines. We are a long way from that goal, but any journey begins with wanting to go somewhere, and above all it is that desire itself that I hope to cultivate here.

To sum up, programming is a distinctive and valuable way of knowing. Doing it well requires both attunement to practical reality and acuity of critical reflection. Each of these criteria provides an indispensable guide and reinforcement to the other. Research always starts somewhere: within the whole background of concepts, methods, and values one learned in school. If our existing procedures are inadequate, practical reality will refuse to comply with our attempts to build things. And when technical work stalls, practical reality is trying to tell us something. Listening to it requires that we understand our technical exercises in the spirit of reductio ad absurdum, as the deconstruction of an inevitably inadequate system of ideas. Technique, in this sense, always contains an element of hubris. This is not shameful; it is simply the human condition. As the successive chapters of this book lay out some technical exercises of my own, the attentive reader will be able to draw up an extensive intellectual indictment of them, consisting of all the bogus assumptions that were required to put forth *some* proposal for evaluation. But the point of these technical exercises does not lie in their detailed empirical adequacy, or in their practical applicability; they do not provide canned techniques to take down from a shelf and apply in other cases. Instead, each exercise should be understood in the past tense as a case study, an attempt in good faith to evolve technical practice toward new ways of exploring human life. What matters, again, is the process. I hope simply to illustrate a *kind* of research, a way of learning through critical reflection on computational modeling projects. Others are welcome to form their own interpretations, provided that the practice of interpretation is taken seriously as a crucial component of the discipline itself.

How computation explains

Artificial intelligence, then, is a potentially valuable enterprise in need of considerable reform. It remains, however, to make precise the sense in which computation *explains* anything. A crucial conceptual difficulty is the complex manner in which AI straddles the boundary between science and engineering. Even though science and engineering have different goals and methods and vocabularies, the AI literature has drawn from both of them, often without any clear distinction. This is understandable enough, given the cross-fertilization that has contributed to the progress of computational science and engineering. (It is also understandable in the institutional context of early AI research, which was conducted largely by psychologists who were funded by instrumentally oriented military research agencies.) Still, it is sometimes important to distinguish clearly between the two projects, at least as ideal types.

Science explains things that exist, aiming to learn the truth; engineering builds new things, aiming to serve instrumental goals. When a word like "planning" or "knowledge" or "reasoning" becomes a technical term, science and engineering pull it in different directions: science toward *human* planning and knowledge and reasoning; engineering toward whatever phenomena can profitably be realized within the existing technology (Woolgar 1987: 320). Yet the term "artificial intelligence" refers to a tradition of research that includes both scientific and engineering projects. Despite its ambiguity, I will use the term in the conventional way to refer to this tradition, and I will use the term "computational psychology" to name the scientific project of using computational methods to explain human phenomena. The term "cognitive science" will refer to the interdisciplinary scientific movement that began in the 1950s and gained steam in the 1980s, not all of which uses computational methods. The adjective "technical" will relate to any discipline of design that employs mathematical formalization (including any sort of digital design or computer software), whether for engineering design or for scientific modeling.

My own intention is to do science, or at least to learn something about human nature, and not to solve industrial problems. But I would also like to benefit from the powerful modes of reasoning that go into an engineering design rationale. One way to reconcile the claims of science and engineering is to posit that people are, in some sense, well engineered. If this is actually the case, then the engineering task of building synthetic intelligence might discover truths of "human design" as well. This is a tricky proposition, given that engineering is geared to the construction of devices with well-defined instrumental purposes. People, of course, have no such well-defined purposes and can be spoken of as "engineered" only in some special and limited sense. In particular, it is not necessary to embrace the dangerous notion that people are "optimal" from the point of view of some instrumental criterion. It is enough to understand how the existence of creatures such as ourselves might be possible at all.

How, then, can computation explain things about people? The most common proposal is that human beings instantiate some mechanism, psychology tries to discover what that mechanism is, and success is judged by matching the input–output behavior of hypothesized mechanisms to the input–output behavior of human beings (Fodor 1968: 121–152). This view of computational explanation had great appeal in the early days of computational psychology, since it promised to bring precision to a discipline that had long suffered from the vagueness and expressive poverty of its concepts. For many, a computer program *was* a theory whose predictions were its outputs. Such was the view of Feigenbaum and Feldman in their introduction to the influential anthology *Computers and Thought* (1963). In one of the papers in that volume Newell and Simon pursued the input–output matching view of computational theorizing, comparing the operation of the General Problem Solver (GPS) program step by step against the reports of an experimental subject. (I will return to GPS repeatedly in subsequent chapters.) This was an extraordinary advance in psychological method: computer programming had made it imaginable to explain singular events, not just statistical averages over large populations. Newell and Simon went to great lengths to fashion an experimental situation for which a comparison could be made (Dreyfus 1972: 112–114). They chose a task with definite formal rules and a well-defined goal, limited the subject's choice of actions, entrusted the execution of the actions to the experimenter, ensured that the subject had no experience with the task before the experiment began, and generally arranged for the subject to conform to all of the premises of the theory. Subsequent research in their school has mounted impressive campaigns of this kind, systematically constraining aspects of theories through precisely contrived experimental setups (Anderson 1983; Newell 1990).[14]

This approach has an honorable history, but it also assumes a definite conception of research. Bound to the artificial and easily manipulated conditions of the laboratory, it gains precision at the risk of detaching itself from the ordinary reality of human life; models that are wildly incongruent with that reality may seem reasonable in the laboratory. I would like to pursue a different approach, one that preserves a connection between technical practice and ordinary experience. In doing so, I must somehow replace the whole system of research values that begins with the testing of input–output predictions. I doubt if it is possible yet to draw meaningful comparisons between computers and people across the vast majority of human activities. Looser comparisons will suffice, however, provided that we bring some analytical rigor to the question of why human beings are the way they are. Forty years of computational studies have shown that it is easy to build systems that resemble human activities in one way or another. What is harder is to specify the *computational principles* that might make such a resemblance significant.

A computational principle is a verity of design, some kind of conclusion about the practicalities of the physical realization of complicated things. One example of a computational principle is commonly known as *least commitment* (Marr 1982: 106; Stefik 1981). This is the idea that a system should, other things being equal, only derive the logically necessary conclusions of each new item of information, thus keeping its options open to the greatest possible extent. Another is the slogan that "a good representation exposes domain constraint," meaning that a good representation scheme expresses things in a way that makes it easy to formulate patterns of deduction that have utility within a given activity.[15] Another is the concept of *nearly decomposable* systems (Simon 1969: 100), which suggests that well-designed artifacts will consist of loosely coupled components whose interactions are simple enough to be understood. Computational principles are, of course, always open to debate and reformulation. Indeed, each of these principles contains assumptions that would be worth reconsidering. (For example, the first two are often interpreted as requiring that computation be understood in terms of mathematical logic.) Nonetheless, principles like these help articulate a design community's accumulated experience in a way that can offer provisional guidance to theorizing. A theory cast in these terms will seek to portray hypothesized mechanisms not simply as real but as *necessary*, at least in particular aspects, due to the constraints of physical realization. The complexity and variety of psychological phenomena make such a mode of explanation especially valuable.

The most influential conception of computational principles comes from David Marr (1982), who prescribed a definite format for computational theories. For Marr, each theory had three tiers: a *computational theory* of some problem, a specification of the *algorithm* by means of which the brain solves this problem, and the *implementation* of this algorithm in neural machinery. This scheme is attractive because it offers a way to decompose the research task, one problem at a time, and because it demands clarity about what the machinery is supposed to be explaining. Computational principles mediate between the analysis of a problem and the design of its solution, and an analysis of the problem can indeed clarify many issues without all the complications that mechanisms introduce. Unfortunately, Marr's conception of computation presupposes particular forms of machinery. Marr interprets the word "problem" in roughly the same way as the conventional theory of computation, as a mathematical function from an input representation to an output representation. In Marr's case these representations are derived from retinal images, but they could also be derived from auditory inputs or databases or a variety of other things. In any event, the technical notion of a problem found in both Marr and Turing presupposes a restrictive view of an agent's relationships to its surroundings, with single isolated inputs mapped to single isolated outputs. This might be a good way to think about the earliest stages of visual processing, but it is not (I will argue) a good way to think about human activity in general. Computational inquiry into human activity thus requires a broader conception of computation itself.

Chapter 3 sketches a way of thinking about computation that provides an alternative to the conventional notion of a problem and is better suited to the study of activity. Whereas Marr proposed focusing research on distinct mappings from inputs to outputs, I propose focusing research on aspects of agents' interactions with their environments. Computational principles, on my view, relate the analysis of a form of interaction to the design of machinery. These principles are irreducibly intuitive, taking precise form only in the context of particular technical proposals. In computational psychology, they attempt to explain something about human nature starting from basic facts: that we have bodies, that our environment is extensively adapted to our culture's ways of doing things, and that everyday life contains a great deal that is cyclical, repetitive, and routine.

Critical orientation

Several previous authors have cast a critical eye on AI research. Weizenbaum (1976), for example, draws on the critique of technology in the Frankfurt School. Focusing on the culture of computer programmers and their use of machine metaphors for human thought, he argues that AI promotes an instrumental view of human beings as components in a rationalized society. In doing so, he largely accepts as practicable the construction of rationality found in AI and other engineering fields, even though he rejects it on ethical grounds.

Other authors have argued that AI as traditionally conceived is not just wrong but impossible, on the grounds that its technical methods presuppose mistaken philosophies. The first and most prominent of these critics was Dreyfus (1972), who pointed out that symbolic methods in AI are all based on the construction of rules that gloss English words and sentences as formally defined algorithms or data structures. Although these rules seem perfectly plausible when presented to audiences or displayed on computer screens, Dreyfus argued that this plausibility was misleading. Since philosophers such as Heidegger and Wittgenstein had shown that the use of linguistic rules always presupposes an embodied agent with a tacit background of understanding, attempts to program a computer with formal versions of the rules would necessarily fail. Unable to draw on tacit understandings to determine whether and how a given rule applied to a given situation, the computer would be forced into a regressive cycle of rules-about-how-to-apply-rules (Collins 1990). Later, Winograd and Flores (1986) extended this argument by describing the numerous ways in which language use is embedded in a larger way of life, including an individual's ceaseless construction of self and relationship, that cannot itself be framed in linguistic terms except on pain of a similar regress.

The AI community has, by and large, found these arguments incomprehensible. One difficulty has been AI practitioners' habit, instilled as part of a technical training, of attempting to parse all descriptions of human experience as technical proposals – that is, as specifications of computing machinery. Given the currently available schemata of computational design, this method of interpretation will inevitably make the theories of Dreyfus and of Winograd and Flores sound naive or impossible, or as deliberate obscurantism, or even, in many cases, as mystical rejections of the realizability of human thought in the physical, causal world.[16]

Another difficulty has been the hazardous procedure, shared by practitioners and critics alike, of "reading" computer programs and their accompanying technical descriptions as if they encoded a framework of philosophical stances. Of course, technical *ideas* and *discourses* do encode philosophical stances, and these stances generally *are* reflected in the programs that result. But as I have already observed, the programs themselves – particular programs written on particular occasions – inevitably also encode an enormous range of simplifications, stopgap measures, and practical expedients to which nobody is necessarily committed. As a result, many members of the AI community do not believe that they have actually embraced the philosophical stances that their critics have found wanting. In fact, AI's engineering mindset tends to encourage a pragmatic attitude toward philosophical stances: they are true if they are useful, they are useful if they help to build things that work, and they are never ends in themselves. If a particular stance toward rules, for example, really does not work, it can be abandoned. Instead, the fundamental (if often tacit) commitment of the field is to an inquiry into physical realization through reflective conversations with the materials of computer work. This is not always clear from the rhetoric of the field's members, but it is the only way I know to make sense of them.

Dreyfus as well as Winograd and Flores have conducted their critiques of AI from a standpoint outside of the field. Dreyfus is a philosopher by background, though in recent work with Stuart Dreyfus he has increased his constructive engagement with the field by promoting connectionism as a philosophically less objectionable alternative to the symbolic rule-making of classical AI (Dreyfus and Dreyfus 1988). Winograd began his career as a prominent contributor to AI research on natural language understanding and knowledge representation (1972), but his critical writing with Flores marked his departure from AI in favor of research on computer systems that support cooperative work among people (Winograd 1995). Dreyfus and Winograd both define themselves against AI as such, or the whole realm of symbolic AI, and they advocate a wholesale move to a different theory. Each of them effectively posits the field as a static entity, doomed to futility by the consequences of an impracticable philosophy.

Another approach, which I adopt in this book, takes its point of departure from the tacit pragmatism of engineering. I regard AI as a potentially valuable enterprise, but I am equally aware that right now it is a misguided enterprise as well. Its difficulties run deep: we could sink a probe through the practices of technology, past the imagery of Cartesianism, and into the origins of Western culture without hitting anything like a suitable foundation. And yet it is impossible simply to start over. The troubles run deeper than anyone can currently articulate, and until these troubles are diagnosed, any new initiative will inevitably reproduce them in new and obscure forms. This is why we need a critical technical practice.[17] The word "critical" here does not call for pessimism and destruction but rather for an expanded understanding of the conditions and goals of technical work. A critical technical practice would not model itself on what Kuhn (1962) called "normal science," much less on conventional engineering. Instead of seeking foundations it would embrace the impossibility of foundations, guiding itself by a continually unfolding awareness of its own workings as a historically specific practice. It would make further inquiry into the practice of AI an integral part of the practice itself. It would accept that this reflexive inquiry places all of its concepts and methods at risk. And it would regard this risk positively, not as a threat to rationality but as the promise of a better way of doing things.

One result of this work will be a renewed appreciation of the extent to which computational ideas are part of the history of ideas. The historicity of computational ideas is often obscured, unfortunately, by the notion that technical work stands or falls in practice and not in principle. Many times I have heard technical people reject the applicability of philosophical analysis to their activities, arguing that practical demonstration forms a necessary and sufficient criterion of success for their work. Technical ideas are held to be perfectly autonomous, defined in self-sufficient formal terms and bearing no constitutive relationship to any intellectual context or tradition. The Cartesian lineage of AI ideas, for example, is held to be interesting but incidental, and critiques of Cartesianism are held to have no purchase on the technical ideas that descend from it. This view, in my opinion, is mistaken and, moreover, forms part of the phenomenon needing explanation. Technical practitioners certainly put their ideas to the test, but their understandings of the testing process have placed important limits on their ability to comprehend or learn from it. Advances in the critical self-awareness of technical practice are intellectual contributions in their own right, and they are also necessary conditions for the progress of technical work itself.

Limitations in a certain historical form of technical practice do not, however, result from any failings in the people themselves. Such is the prestige of technical work in our culture that the AI community has attracted a great many intelligent people. But they, like you and I, are the products of places and times. The main units of analysis in my account of technical practice are discourses and practices, not the qualities of individual engineers and scientists. A given individual can see only so far in a fog. Periodic moments of clarity are but the consolidation of changes that have been gathering in the works; their full-blown emergence in the composition of great books is a convenient outcome to an orderly process. Equipped with some understanding of the mechanics of this process, critical inquiry can excavate the ground beneath contemporary methods of research, hoping thereby to awaken from the sleep of history.

In short, the negative project of diagnosis and criticism ought to be part and parcel of the positive project: developing an alternative conception of computation and an alternative practice of technology. A critical technical practice rethinks its own premises, revalues its own methods, and reconsiders its own concepts as a routine part of its daily work. It is concerned not with destruction but with reinvention. Its critical tools must be refined and focused: not hammers but scalpels, not a rubbishing of someone else but a hermeneutics and a dialectics of ourselves.

Outline

Chapter 2 states the book's theses: the reflexive thesis (which concerns the role of metaphors in computer modeling), the substantive thesis (which proposes replacing one set of metaphors with another), and the technical thesis (which describes the basis in technical experience for proposing such a shift). It then discusses the reflexive thesis at length, developing a vocabulary for analyzing the metaphors in technical research. The point is not simply to discover the right set of metaphors but to encourage a critical awareness of the role of metaphors in research. The chapter concludes with a sketch of reflexive issues that must await further work.

Chapter 3 concerns the substantive thesis. It describes the metaphor system of mentalism, which portrays the mind as an abstract territory set apart from the "outside world." Put into practice in day-to-day technical work, mentalism participates in characteristic patterns of success and failure, progress and frustration. An alternative is to ground AI in interactionist metaphors of involvement, participation, and reciprocal influence. I introduce some vocabulary and methodological ideas for doing interactionist AI research.

Chapter 4 analyzes the mentalist foundations of computing, starting with the tension between abstraction and implementation in conventional computer science. The technical notion of a variable provides an extended case study in this tension that will turn up in subsequent chapters. The tension between abstraction and implementation is also evident in the history of cognitive science, and its outlines provide some motivation for interactionist alternatives.

Chapter 5 continues the analysis of conventional computer science with a critical introduction to the workings of digital logic. Computers these days are made of digital logic, and throwing out digital logic altogether would leave little to build models with. Instead, this chapter prepares the way for a critical engagement with digital logic by describing the peculiar ideas about time that have accompanied it.

Chapter 6 shifts into a more technical voice, developing a set of fairly conventional ideas about the relationship between digital logic and human reasoning. It is a costly and difficult matter to think anything new, and so "dependencies" provide a means of recording, storing, and automatically recapitulating common lines of reasoning. In addition to presenting dependencies as a technical proposal, this chapter also briefly recounts the tradition of ideas about habit and learning from which they arise.

Chapter 7 introduces a simple rule-based programming language called Life that aids in the construction of artificial agents whose reasoning can be accelerated through dependency maintenance. Execution of Life programs depends on some formal properties of conventional AI rule languages that I define just well enough to permit an expert to reconstruct the details. Some cartoon programming examples demonstrate the Life language in use.

Chapter 8 presents a detailed analysis of the early planning literature, from Lashley's (1951) "serial order" paper to Newell and Simon's (1963) GPS program to Miller, Galanter, and Pribram's (1960) theory to the STRIPS program of Fikes, Hart, and Nilsson (1972). Through this history, a complicated pattern of difficulties develops concerning the relationship between the construction and execution of plans. Viewed in retrospect, this pattern has pushed AI research toward a different proposal, according to which activity arises through improvisation rather than the execution of plans.

Chapter 9 develops this proposal in more detail by introducing the notion of a running argument, through which an agent improvises by continually redeciding what to do. This scheme will not work in a chaotic world in which novel decisions must be made constantly, but it might work in a world in which more routine patterns of activity are possible. A set of Life rules is described through which an agent might conduct arguments about what to do. The chapter concludes by describing an architecture for such an agent, called RA.

Chapter 10 demonstrates RA in action on a series of simple tasks drawn from AI's conventional "blocks world." The chapter detects a series of difficulties with the program and, in the spirit of reductio ad absurdum, traces these back to common assumptions and practices. In particular, difficulties arise because of the way that the system's ideas about the blocks are connected to the blocks themselves.

Chapter 11 takes heed of this conclusion by reexamining the mentalist understanding of representation as a model of the world. The shortcomings of this view emerge through an analysis of indexical terms like "here" and "now," but they also emerge through the technical difficulty of maintaining and reasoning with such a model. The path to interactionist alternatives begins with the more fundamental phenomenon of intentionality: the "aboutness" of thoughts and actions. Some phenomenologists have proposed understanding intentionality in terms of customary practices for getting along in the world.

Chapter 12 attempts to convert this idea into a technical proposal. The basic idea is that an agent relates to things through time-extended patterns of causal relationship with it – that is, through the roles that things play in its activities. The concept of deictic representation makes this proposal concrete.

Chapter 13 describes a computer program called Pengi that illustrates some of these ideas. Pengi plays a video game calling for flexible actions that must continually be rethought. As with RA, reflection on the strengths and weaknesses of this program yields lessons that may be valuable for future theorizing and model-building. Some of these lessons concern the tenacity of mentalism in the face of attempts to replace it; others concern the role of attention in the organization of improvised action.

Chapter 14 summarizes a variety of other research projects whose approaches converge with my own. It also offers some reflections on the reflexive thesis concerning the role of metaphor in technical modeling.

2 Metaphor in practice

Levels of analysis

The Introduction has sketched the notion of a critical technical practice, explored the distinctive form of knowledge associated with AI, and described a reorientation of computational psychology from a focus on cognition to a focus on activity. It should be clear by now that I am proceeding on several distinct levels at once. It is time to systematize these levels and to provide some account of the theses I will be developing on each level.

On the *reflexive* level, one develops methods for analyzing the discourses and practices of technical work. Reflexive research cultivates a critical self-awareness, including itself among its objects of study and developing useful concepts for reflecting on the research as it is happening. To this end, I will begin by suggesting that technical language – that is, language used to investigate phenomena in the world by assimilating them to mathematics – is unavoidably metaphorical. My reflexive thesis is that predictable forms of trouble will beset any technical community that supposes its language to be precise and formally well defined. Awareness of the rhetorical properties of technical language greatly facilitates the interpretation of difficulties encountered in everyday technical practice. Indeed, I will proceed largely by diagnosing difficulties that have arisen from my own language – including language that I have inherited uncritically from the computational tradition, as well as the alternative language that I have fashioned as a potential improvement.

On the *substantive* level, one analyzes the discourses and practices of a particular technical discipline, namely AI. Chapter 1 has already outlined my substantive thesis, which has two parts. Part 1: For about forty years, the discourse of AI has been organized around a particular metaphor system according to which the mind is a space with an inside, an outside, a boundary, and contents. The resulting project suffers from recurring difficulties whose form can be predicted in some detail. Part 2: A better starting point is the complementary metaphor system organized by the notion of interaction. A good understanding of the interactional dynamics of a particular form of activity can greatly simplify the machinery necessary to explain it. The point, however, is not simply to substitute new metaphors for old metaphors, but to employ the new metaphors with a reflexively critical awareness of the role that metaphors play in technical work.

On the *technical* level, one explores particular technical models, employing a reflexive awareness of one's substantive commitments to attend to practical reality as it becomes manifest in the evolving technical work. My particular case study will explore the relationships between action, perception, and representation in the context of certain highly simplified improvised activities. Specifically, I will argue that the phenomena of human activity normally understood in terms of representation need to be partitioned under two headings: the first and more fundamental of the two defines things in terms of their roles in conventionalized activities; the second and more complex involves internalized symbolic representations. One confirmation of this distinction is its utility in phenomenological analysis. Another is that it leads to simpler forms of machinery.

The balance of this chapter will develop a reflexive analysis of technical language and technical work. Chapter 3 will then employ these reflexive ideas to develop a substantive analysis of AI, including a critique of its existing metaphor system and a proposed alternative.

Language in practice

Philosophers and linguists have long asked, do we speak language or does language speak us? In other words, is language a tool for the expression of thoughts and actions, or do the forms of a given language determine the forms of thought and action of its speakers? Views on this matter in Western thought might be sorted into two broad categories. On one side is an Enlightenment tradition, for which the principles of reason are universal, and Chomsky (1966, 1979) in particular, for whom the generativity of grammar is a mark of human freedom. On the other side is a counter-Enlightenment tradition, including theorists of culture from Vico (1968 [1774]; see Berlin 1976) through Wilhelm von Humboldt (see

R. Brown 1967) to Whorf (1956 [1941]) to numerous modern-day anthropologists. This alternative tradition holds that in learning to speak the language of one's culture, one acquires that culture's way of living in and seeing the world.

This latter view has attracted considerable abuse, particularly from those American philosophers who equate it with idealism.[1] At issue is the way in which people can use language to represent the world. If language represents reality through a transparent mapping of grammatical structure to the structure of the world, then it is indeed hard to imagine how different cultures could legitimately speak about the world in different ways: different representations would imply different realities. But is that the nature of language? Does anyone ever describe a given piece of the world in such a transparent way? The question becomes particularly urgent if one believes, as many do, that competent action depends on knowledge of the world, that knowledge consists in true descriptions, and that truth consists in a commonality of form between descriptions and the things described. The theory of knowledge as commonality of form can be found in Aristotle,[2] but in the period after Descartes it came increasingly to be overlaid with a specific, mathematical notion of "form." With this shift, and particularly with the contemporaries of Newton, comes a view of language founded not on poetry or rhetoric or dialogue but on mathematics (Markley 1993). The world, on this view, does not exceed language. Rather, the world effectively possesses the same mathematical forms as language, so that any inexactitude of linguistic expression is merely a matter of error, and any incompleteness of linguistic expression is merely a matter of degree (Burtt 1959 [1924]: 96–110, 302–305).

Modern philosophical logic has systematized this approach to language. No longer the empirical investigation of the forms and properties of reasoning, logic as a field of study is now understood as specifying the *conditions* for any empirical investigation, and indeed for inquiry in general. Barwise and Perry (1985), for example, regard their theory of *situation semantics* not simply as a systematic account of the structures of language but also literally as a series of discoveries about reality itself. The logical form of language is, in this sense, simultaneously the logical form of the world. Though Barwise and Perry's theory represents a substantial advance over previous logical theories of semantics (see Chapter 11), it will nonetheless strike the scientist as metaphysical: its discoveries are not presented as contingent facts but rather as analytic conclusions about the logical form of *both* language and reality. In this way, philosophical logic increasingly presupposes that reality enjoys the same kind of perfectly determinate existence as that of mathematics, and consequently that scientific language is the prototype of human language because its explicitly mathematical nature ensures its transparency in relation to that reality.

This interpretation of language and reality and the relationship between them is not explicitly or uniformly adhered to by all technical people, but it is nonetheless an important element of the culture of present-day technical practice. This influence is easily seen in the method of theory construction that I described in the introduction: lashing a bit of metaphor to a bit of mathematics and embodying them both in computational machinery. The everyday work of AI consists, to a large extent, in the alignment of these disparate resources (Suchman and Trigg 1993) and, specifically, in an unrelenting reconstruction of ordinary language in technical terms. How this enterprise proceeds will depend, to an equal extent, on the ideas about language that underlie it. Just *how* the enterprise's progress depends on its ideas about language is a difficult matter that deserves investigation.

In recent years the counter-Enlightenment tradition has grown another branch, starting with authors like Foucault, for whom language is part and parcel of a certain material organization of the world. Whorf and Foucault share certain concerns; both, for example, applied their views to the analysis of technical practices – Whorf in his anecdotes from his day job as an insurance inspector (1956 [1941]) and Foucault in historical studies of biology, linguistics, and medicine (1970, 1973). The common accusation of idealism against these theories presupposes a particular understanding of the operation of language in activity, namely, the semantic notion of structural correspondence to reality. Yet something more is clearly required to account for historical episodes in which practices have achieved and maintained an obstinate stability over long periods despite their adherence to the most disparate discourses imaginable. What remains is the philosophical, anthropological, and historical task of understanding what relationships *did* obtain between the forms of language and practice in question.

Whorf and Foucault understood the relationship between language and practice in different ways. Whorf investigated how the grammatical forms of language shape perception, reasoning, and action; Foucault viewed language as part of the locally organized means of producing whatever counts as truth in a given setting. For Whorf, language is a distinct and bounded phenomenon; for Foucault, language is simply one element of the much more expansive category of "discourse." Foucault's notion of discourse, then, includes language, but it also includes technologies, practices, and any other resources that people draw on to construct reality. Because my own argument will turn on the properties of language as such, however, I want to retain the term "discourse" for something narrower: the forms and structures of language that a given community uses in organizing its customary activities. I will then need to specify some of the relationships between the discourses and practices of technical modeling in AI.

Before moving along to that work, however, let us briefly consider the path I have not taken. Edwards (1996) employs Foucault's more expansive concept of discourse in his history of the origins of AI. He is concerned with the institutional and cultural context within which the core ideas and practices of AI arose, and he organizes his discussion around two discourses that situate AI within this larger context. These discourses are *closed world discourse* and *cyborg discourse.* Closed world discourse is roughly the discourse of the Cold War: treating the earth as a single closed system, the United States as a bounded territory to be defended, and the Soviet Union as a closed society to be contained. Cyborg discourse is the discourse of human–machine symbiosis: treating the whole world as a militarized command-and-control structure that operates on rational, technological principles, from the highest levels of global strategy to the lowest levels of individual cognition. Edwards defines the notion of discourse in this way:

> [D]iscourse goes beyond speech acts to refer to the entire field of *signifying* or *meaningful practices:* those social interactions – material, institutional, and linguistic – through which reality is interpreted and constructed for us and with which human knowledge is produced and reproduced. *A* discourse, then, is a way of knowledge, a background of assumptions and agreements about how reality is to be interpreted and expressed, supported by paradigmatic metaphors, techniques, and technologies and potentially embodied in social institutions. (1996: 34, emphasis in the original)

Observe that these definitions of *discourse* and *a discourse* reproduce the divide that I just described: discourse is defined as a matter of practices –

social interactions – and *a* discourse is defined as a worldview – "assumptions and agreements" – that is "supported" by other things. In practice, Edwards, like Foucault, treats discourse as an assemblage of "heterogeneous" elements: techniques, artifacts, practices, experiences, fictions, and language, including metaphors. Foucault's purpose in defining discourse so broadly is to disrupt any sense that knowledge consists in a collection of linguistic representations. Instead, Foucault wishes to emphasize that knowledge is something produced in the material world. He resists any formula that would derive knowledge by combining language and practice in fixed proportions, and Edwards (1996: 38) quotes his famous description of discourse as "a series of discontinuous segments whose tactical function is neither uniform nor stable" (Foucault 1980: 100). This sense of locality – the improvised assembly of parts into wholes in various sites – serves Edwards well in describing the emergence of the fundamental discourses of AI. For example, he traces lines of intellectual and technical development that did not mature, or that remained marginalized for many years, because they were not able to gain traction in the larger institutional and cultural context of the times. My own study is concerned with the lines that survived this early winnowing and stabilized in canonical forms at a handful of well-funded research institutions. The resulting schools of research were products of the Cold War in certain important senses, but they also drew on older and deeper trends in Western intellectual history, and they evolved with a significant degree of autonomy and a complex inner logic.

I will return to the substantive particulars of that research in the next chapter. For the moment, I want to explore in greater detail the relationship between discourse – in the narrower sense, now, of the forms and structures of a community's language – and practice – a community's forms of action upon the objects of its work. In terms of the two divergent branches of the counter-Enlightenment tradition, I have returned to the anthropological notion of language as shaping a worldview for a community. One objection to this notion has some force: if language shapes a culture's understanding of the world, how might people attain any critical consciousness of the role of language in their lives? At the beginning of his most influential paper, for example, Whorf quotes this passage from Sapir:

Human beings do not live in the objective world alone, nor alone in the world of social activity as ordinarily understood, but are very much at the mercy of the particular language which has become the medium of expression for their society. It is quite an illusion to imagine that one adjusts to reality essentially without the use of language and that language is merely an incidental means of solving specific problems of communication or reflection. The fact of the matter is that the "real world" is to a large extent unconsciously built up on the language habits of the group. . . . We see and hear and otherwise experience very largely as we do because the language habits of our community predispose certain choices of interpretation. (1956 [1941]: 134)

Language, on this account, limits consciousness. We have a great deal of choice in how we interpret the world, yet we are often unaware of *making* any choices. Sapir does not claim that our interpretations of the world are completely determined by our language, only "very largely" so. Language's effect on consciousness is thus a matter of degree. What Sapir leaves unclear is whether the magnitude of the effect is fixed or whether it varies historically. Can we become more aware of our interpretive choices? Can the study of rhetoric, for example, confer any immunity to the manipulations of politics and advertising? Perhaps an explicit understanding of language's ability to shape experience, or even a relatively crude understanding, can make apparent the contingent nature of a cultural or professional worldview. Every culture has *some* explicit understandings of language use – a *linguistic ideology* (Woolard and Schieffelin 1994). Silverstein (1979) has argued that a given culture's linguistic ideology stands in a dialectical relationship with the language itself, exerting a long-term influence on it and being influenced reciprocally by it.[3] The linguistic ideology will be open to all of the social influences of any other ideology, so that one might trace a chain of reciprocal influence between a society and its language, with the linguistic ideology as one critical link. Sapir's fatalistic assertion that we are "at the mercy" of our language may well be false, at least in this one sense: that research into the workings of language in society might help ameliorate some of the limitations of our understanding of the world.

Metaphors in technical work

In order to apply these ideas to technical work, it is necessary to understand technical communities as analogous to discrete cultures (cf. Forsythe 1993b). Such analogies operate on two levels. The twentieth century has seen the rise of hundreds of technical discourses, each organized by a different confluence of metaphor and mathematics: information theory, control theory, chaos theory, general systems theory, operations research, rational expectations theory, and so forth. Inasmuch as these disciplines are embedded in a larger culture, their local configurations of language correspond to distinct worldviews only in some restricted sense. More fundamentally, though, technical language as such encodes a cultural project of its own: the systematic redescription of human and natural phenomena within a limited repertoire of technical schemata that facilitate rational control.[4] Technical communities are strikingly uniform in their sociologies and methods, considering the heterogeneity of their metaphors. But it is precisely this phenomenon that makes it especially important to investigate the role of metaphors in technical practice.[5]

Several authors have described the role of metaphors in organizing scientific research programs. Mirowski (1989), for example, argues that metaphors of equilibrium from physics provided a model for the creation of neoclassical economics; as economic questions arose, economists could look to physics as a metaphorical resource in formulating theories to address them. Likewise, E. Martin (1987) describes how cultural constructions of gender are transferred into reproductive physiology, and Young (1985) traces the use of sociological metaphors in evolutionary biology. Schön (1979) in particular suggests using techniques from literary criticism to identify the *generative metaphors* of the "stories" told in his own field of social policy. A metaphor establishes an open-ended mapping from one discursive domain to another (economics and physics, reproductive physiology and cultural gender roles, evolutionary biology and social structures), and a metaphor is "generative" in the sense that a research community can extend its own discourse by carrying one element after another through the mapping.[6] The discovery that disciplinary discourses frequently employ generative metaphors gives critical reflection a reliable starting place. But metaphors are not simply a literary device. Schön emphasizes the role of metaphors in the complex process of constructing a version of social reality:

Each story constructs its view of social reality through a complementary process of *naming* and *framing*. Things are selected for attention and named in such a way as to fit the frame constructed for the situation. Together, the two processes construct a problem out of the vague and indeterminate reality which John Dewey called the "problematic situation." They carry out the essential problem-setting functions. They select for attention a few salient features and relations from what would otherwise be an overwhelmingly complex reality. They give these elements a coherent organization, and they describe what is wrong with the present situation in such a way as to set the direction for its future transformation. (1979: 264–265)

When this construction is not understood *as* a construction, Schön argues, descriptions flow far more easily into prescriptions than a real critical awareness of the process would permit (1979: 268).[7] The identification of a generative metaphor is often a liberating experience, giving concise expression to a previously diffuse sense of being oppressed by unarticulated false assumptions. Unfortunately, the most common response to such a discovery is to place the blame on metaphors themselves, rather than on the unreflexive neglect of metaphors and their consequences.[8]

This view of metaphor as pathological does not account for the constructive role that generative metaphors play in organizing scientific inquiry. Boyd (1979) has described the place of *theory-constitutive* metaphors in the construction of scientific theories. Opposing the conventional view that scientific language should be as precise as possible, Boyd argues that the open-ended nature of metaphors is a virtue. Extending Black's (1962) *interaction theory* of metaphor, he emphasizes the ability of metaphors to draw in new themes and pose new questions when the outlines of a given phenomenon are still unclear (1979: 403).[9] As a result, the referential properties of scientific language must be understood in dialectical terms, as the result of a sustained conversation with the phenomena rather than an unproblematic relation of designation between discrete terms and discrete things.

Reference has an essential dynamic and dialectical aspect. Changes in language use – when they reflect the dialectics of accommodation [of theories to phenomena] – do not represent changes of reference in any philosophically puzzling sense of that term. Instead, such dialectical changes of reference are characteristic of referential continuity and represent perfectly ordinary vehicles for the reporting of new discoveries. (Boyd 1979: 382)

One consequence of Boyd's view is that the essence of scientific terminology lies not in fixed relations to reality but in changing relations to scientific enterprises:

Theory-constitutive metaphorical terms – when they refer – refer implicitly, in the sense that they do not correspond to explicit definitions of their referents, but instead indicate a research direction toward them. The same thing is apparently true of theoretical terms in science generally. (Boyd 1979: 406)

In short, it is impossible to extract the referential function of scientific language from the ongoing research project within which a given network of terms is evolving.

An example of Boyd's account of scientific metaphors can be found in the history of the clock metaphor in psychology. As McReynolds (1980) has observed, the metaphor of human beings as clockwork devices helped legitimize the emerging mechanistic philosophy of the seventeenth century. Numerous authors in this period suggested that a mainspring "drives" a clock in the same way that inner motivations "drive" people. Thus, seventeenth-century clocks played a rhetorical role somewhat analogous to the twentieth-century servomechanisms that provided one of the principal inspirations for cognitivism.[10] As McReynolds points out, metaphors like human-being-as-clock do not themselves generate scientific innovations. Nor, on the other hand, do the metaphors simply summarize or encode an insight previously given explicit articulation in a philosophy or technical proposal. Instead, a metaphor will emerge from a cultural stock in order to give shape to a scientific intuition that is still inchoate. Once it has taken hold, such a metaphor may become the generative principle of a new discourse, facilitating the elaboration of a scientific picture across a range of topics (McReynolds 1980: 108–109). The simple prevalence of a given metaphor, then, does not explain the circumstances that permitted it to rise up and flourish at a particular point in history.

Boyd's account of theory-constitutive metaphors raises significant issues that run throughout the philosophy and sociology of science. Kuhn (1979), for example, argues that Boyd's account has no explanatory force except on the assumption of an objective system of things-in-themselves upon which the dialectical process of establishing reference settles down. The problem is epistemological: unless he can explain how scientists navigate the open-ended space of potential meanings that a metaphor opens up within their research, the argument collapses into simple relativism. Rather than take that path, Kuhn observes, Boyd veers instead into simple realism by speaking of scientists' "epistemic access" to the matters they are studying (cf. Bloor 1976).

Bono (1990) provides another explanation of why Boyd's argument goes wrong. Bono observes that Boyd tries to delimit the phenomenon of scientific metaphor, dissociating it from the "unruliness" (Bono 1990: 66) of literary metaphor and confining its operation within the boundaries of a particular scientific community. Bono asserts that neither of these moves is tenable. The whole purpose of a metaphor is to join two distinct semantic fields, and metaphors operate as a "medium of exchange" (Bono 1990: 72) between different parts of society. As a result, shifts in the social environment can destabilize an otherwise esoteric discipline. The novel technologies of World War II, for example, contributed to a revolution in psychology by expanding and giving new life to the ancient metaphor of human-being-as-machine. Boyd (1979: 361) had cited this case to suggest that "at least for a time, [the] cognitive content [of scientific metaphors] cannot be made explicit." But whereas Boyd viewed the dialectical establishment of reference as a unidirectional process, Bono follows Arbib and Hesse (1986) in arguing that scientific metaphors always exist in a state of tension:

> To use Kuhnian terminology, in the development of science a tension always exists between normal and revolutionary science: normal science seeks to reduce instability of meaning and inconsistency and to evolve logically connected theories; revolutionary science makes metaphoric leaps that are creative of new meanings and applications and that may constitute genuine theoretical progress. (Arbib and Hesse 1986: 157)

For Bono, the two features of scientific metaphor – its connection to other social domains and the internal continuity of the tension between the "normal" and "revolutionary" modes – combine to create a powerful framework for historical analysis.[11] Individual scientists are always the inheritors of a rich constellation of metaphors, many of which are likely to stand in stable relationships to established practices of a field. In applying those metaphors to new circumstances, however, these individual scientists must reckon with the whole elaborate economy of meaning in the contemporary intellectual environment (Bono 1990: 77). Since the formations that result are "hybrid" in character, "even the most coherent of them will contain inherent tensions, if not contradictions, which are usually kept submerged but are never completely hidden" (78). The formation that Edwards identifies as cyborg discourse, for example, arises historically in just this way, through the adaptation of old themes to an ensemble of new circumstances. The rise and evolution of this discourse are driven not only by institutional circumstances, but (I will argue at length) in large part by tensions that become inevitable as a particular system of metaphors is put into practice.

Centers and margins

My own account of scientific metaphors builds on those of previous authors by describing some of the dynamics of a particular class of scientific projects: those that proceed through technical modeling. This class includes much of cognitive science, as well as an increasing number of other fields. Technical model-making is always in part a discursive enterprise: the model-maker establishes the relevance of a given model to reality by demonstrating that the operation of both model and reality can be narrated in a common vocabulary.[12] As I suggested in Chapter 1, computational model-building proceeds through the application of a repertoire of schemata, each of which joins a metaphor to a bit of mathematics that can be realized on a computer. And model-building is an ongoing enterprise, proceeding through the incremental, generative application of metaphors to new issues and questions. As Boyd pointed out for the application of metaphors in general, the complex process of model-building needs to be understood *as* a process, as inherently in motion.

This section concerns the *practical logic* of technical model-building. By "practical logic" I mean what happens when people who possess some definite worldview, for example one that has been shaped by certain generative metaphors, put that worldview into practice.[13] Technical ideas develop through a roughly cyclical process: ideas are made into techniques, these techniques are applied to problems, practitioners try to make sense of the resulting patterns of promise and trouble, and revised ideas result. Inasmuch as the practitioners' perceptions are mediated by their original worldview, and given the near immunity of such worldviews from the pressures of practical experience, the result is likely to be a series of steadily more elaborate versions of a basic theme.[14] The generative principle behind this process will become clear only once the practitioners' worldview comes into question, that is, when it becomes possible to see this worldview as admitting alternatives. Until then, the whole cycle will make perfect sense on its own terms, except for an inchoate sense that certain general classes of difficulties never seem to go away.

The practical logic of technical work has numerous facets, including training regimens, funding pressures, career paths, cultural fashions, and academic politics. I will focus on one particular facet: the relation between the practical logic of technical projects and those projects' generative metaphors. In doing so, I will draw on some of the analytic devices that Derrida introduced in his analyses of Western philosophy, namely *presence, effacement, margin, reversal,* and *displacement.*[15] I intend these terms seriously: it will be the gist of my account that technical model-building projects deconstruct themselves.

The target of Derrida's critical project is the notion that human ideas stand in a transparent correspondence to reality (cf. Arbib and Hesse 1986: 159). Derrida refers to this notion as "presence," after one of its variants, the idea that perception simply records what is present to the senses. Notions of presence can be found in more or less obvious forms throughout philosophy. One particularly obvious form, already remarked upon, is the idea that reality can be mapped perfectly onto a mathematical structure. Such a mathematical representation would be so accurate that it would no longer have the central property of a representation: the necessity of interpreting it. By standing in unmediated correspondence to the world, it would be "effaced" as a contingent interpretation. Perhaps the most striking philosophies of effacement are those formal-logical conceptions of natural language semantics which hold representation and reality to be, in a mathematical sense, isomorphic.

One way to understand the tacit philosophy of presence in mathematization is through Husserl's account of the origin of geometry. For Husserl (1970 [1954]), the history of geometry held clues to a "crisis" in science. Husserl did not mean that the accomplishments of the mathematizing sciences are illusory. Instead, he wished to suggest that these accomplishments have been misunderstood and require a more complex analysis than they have usually been given. The key, for Husserl, is the difference between ideal mathematical forms and the concrete phenomena that people, scientists included, encounter in their activities. An abstract geometric shape such as a triangle, for example, is quite different in kind from a concrete triangular object such as a plot of land. Yet this difference is frequently covered over by scientists in their daily practice. The beginnings of this pattern, Husserl contends, can be found in Galileo's use of mathematics, particularly in the highly developed practical arts that he had inherited from generations of craft workers. Galileo's research, as embodied work, consisted in part of complex practical activities that drew on traditional methods for describing and measuring geometrical shapes. Precisely because those methods were unproblematic, Galileo could have a sense of "seeing" the ideal forms whose existence his theories posited. That is, he could describe nature as consisting of certain abstract geometric shapes, and he could operationalize this description by going out and "finding" those shapes in his experimental apparatus and astronomical observations.

What Galileo lost, in Husserl's view, was any awareness of the relationship between the abstract forms of his theories and the concrete phenomena from which these forms were derived. Once the concrete phenomena were lost from view, it became possible to understand the universe as a multitude of abstract forms – ideal entities with no necessary connection to history or culture and no relationship to the phenomena of embodied activity. Husserl conceded that the universe was essentially mathematical in its basic spatiotemporal structure. He also accepted that the practical work of *technē* required scientists to become lost periodically in the manipulation of ideal forms and their mathematical representations. His concern was simply with the view of the universe, and of individual subjectivity, that arose when these ideal forms were mistaken for the phenomena of experience. In this way, Husserl asserted, the rise of science simultaneously uncovered and obscured reality. He did not doubt the power of its discoveries, but he simultaneously regretted the progressive detachment from concrete human experience that accompanied them.

The point of Husserl's theory of science, then, is not that ideal forms are false representations of reality, much less that science as such is a pathological enterprise. He believed that the passing-over of concrete phenomena called for a kind of philosophical reform – not just a reform of ideas, but a reform of experience itself, gradually restoring awareness of the phenomena from which scientific ideality has distracted us. Husserl's reform project prefigures later empirical research into laboratory science whose purpose is not to debunk science or relativize its discoveries, but simply to recover an awareness of the lived work by which those discoveries are produced (Latour and Woolgar 1986 [1979]; Lynch 1985; Pickering 1984).

Husserl's theory of ideal forms and concrete experience is one precursor of Derrida's notion of effacement. Derrida (1978 [1962]) argued that

Husserl failed, and necessarily so, in his attempt to reactivate the originary perception of concrete phenomena – the direct presence of things – that supposedly lies beneath the obfuscations of abstraction. Nonetheless, the analogies between the two authors are extensive. The whole point of effacement is that abstractions can take on a false transparency when they are mistaken for apodictic representations, or indeed when they are mistaken for the things themselves. Husserl's prototype of effacement is mathematization, and Derrida's prototype is found in philosophical language. Both of them initiate a search for something that has been lost, but they are heading in opposite directions: Husserl suggests that the scientist loses track of concrete experience while absorbed in abstractions, while Derrida (1976 [1967]: 20) suggests that the ordinary speaker of language loses track of the materiality of the signifier while absorbed in speaking. Each project of recovery speaks to the irreducibly contingent nature of representation: it is always something constructed in human activity, and yet it always exceeds any attempt to fasten it to a determinate phenomenon. Each project, moreover, provides a corrective to the other: Husserl encourages us to investigate the lived activity of mathematization, but Derrida cautions us against expecting to recover this activity as a bounded category of experience; Derrida directs our attention to the spaces between signifiers and the impossibility of transparent representation, but Husserl cautions us against lapsing at this point into agnosticism, simply quitting our work because the signifiers want to play.[16] In studying computational work, the critique of representation is located exactly within this productive dialogue between Husserl's project and Derrida's. Computer systems are defined mathematically, like Galileo's apparatus, but they are also designed to embody ideas about language, action, learning, communication, and so on whose nature is specified using language. Computer systems are thus, among other things, also philosophical systems – specifically, mathematized philosophical systems – and much can be learned by treating them in the same way.

These considerations encourage us to recover a full sense of the place of language in the construction of technology. As Knoespel puts it:

> [T]he reification of geometry in architecture and technology has enormous implications for language. Once geometry becomes manifest in artifacts, these artifacts retain an authority radically different from that accessible to natural language. By virtue of their being manifested as physical objects they acquire

what appears as an autonomy utterly separated from language. The apparent separation of both architecture and technology from language has great significance, for it works to repress the linguistic framework that has allowed them to come into being. (1987: 42)

Having employed certain discursive forms in designing technological artifacts such as buildings and computers, it becomes possible – indeed, almost inevitable – to treat the linguistic forms as if they were perfectly embedded in the artifacts. A computer might be constructed so that its operation can be narrated using words like "knowing" or "reminding," but those words have a life that goes far beyond – and may even conflict with – the artifacts that are supposed to embody them. When this is forgotten, Smith (1996) speaks of an "inscription error": inscribing one's categories onto an artifact and then turning around and "discovering" them as if they had already been there. And Derrida speaks of the effacement of signifiers. The point is not that technological artifacts inherently suppress language, but that a historically specific experience of artifacts has arisen and needs to be undone.

Derrida seeks to discover the effacement of signifiers not through a generalized argument but through a practice of deconstruction (Culler 1982). He wants to demonstrate how a particular system fails in its attempt to establish unmediated correspondences between symbols and phenomena. Deconstruction does not portray itself as an ultimate theory standing upon an Archimedian point external to all theories. Nor does it proceed by comparing each particular symbolic system with another, supposedly more accurate description of reality. (Either of these approaches would obviously fall prey to deconstructive analysis themselves.) Instead, it argues within the premises of a given system, exhibiting the reductio ad absurdum that follows from that system's version of effacement. This strategy makes its point in each case without being founded on a systematic set of premises of its own.

The arguments by which Derrida deconstructs particular proposals take numerous forms, some of which might be taken to exemplify heuristic formulae for producing such arguments. I will concentrate on one such heuristic here. This heuristic applies when a philosophical system employs a *hierarchical opposition:* that is, a classification of phenomena into two categories, one of which is central, normal, and well behaved, and another of which is peripheral, abnormal, and ill behaved. Simplifying Derrida's terminology, I will refer to these as the *center* and the *margin* of the opposition. A given center–margin distinction might be an explicitly established partition, as with the distinction between literal and figurative uses of language, or it might be tacit, as when all the examples provided in some exposition are central, "easy" cases of a theory. It is extraordinarily common for a philosophical system to elevate some central category as the "normal" case, so that the integrity of the system depends on its success in hiding or explaining away the associated marginal category.[17] A theory of truth based on the correspondence between literal sentences and reality, for example, immediately faces the problem of figurative sentences. Figurative sentences might be regarded as paraphrases of literal ones, as meaningless, as false, as immature or provisional stages in the development of literal concepts, or in some other way as inessential to the central, normal phenomenon of literal truth. It is probably impossible to demonstrate that all such attempts must fail, but it has proved possible to demonstrate the internal inconsistency of each one in turn.

Systems that are founded on hierarchical oppositions tend to exhibit characteristic difficulties, simply because the supposedly marginal cases refuse to go away. Systematizers are often found elaborately rationalizing, trivializing, obfuscating, or ignoring the marginal areas of their systems. This is not to say that systematizers are dishonest. When a scholar attempts to work out a system, the system will undergo one revision after another until no more problems are apparent to the thought of that day. As a result, the difficult bits of intellectual systems tend to be located in obscure places: in footnotes, systematic ambiguities, or supposedly unproblematic assumptions. These small symptoms may be hard to find. Once discovered, though, they frequently reveal systemic fissures within the discourse: deeply rooted tensions or contradictions hidden beneath tacit equivocations, bland evasions, or a problematic shifting of the trouble into a lost origin, an idyllic future, or a parallel realm of ideals.

One of the heuristics for discovering such things is called "reversal" (Derrida 1981: 41–44, 1982b: 329).[18] It is simple enough: given a system based on a hierarchical opposition, propose another system based on a reversal of the same opposition. The point of this exercise is to make obvious the ways in which the supposedly central cases of the original system depend on the supposedly marginal cases. A classification formerly held up as natural and obvious is thereby exposed as artificial and obscure. In the case of the literal–figurative opposition, for example, the very notion of literality turns out to be defined in a figurative way. In general, the goal of the strategy of reversal is not to replace an existing theory but rather to displace the whole practice of theory-making, that is, to promote a reflexive awareness of the complex role of language in practical action. Specifically, displacement makes evident the fact that theories are *interpretations* of their supposed referents. These interpretations are not underwritten by an objective correspondence, but they are also far from being arbitrary. To someone who equates reasoned thought with some philosophy of presence, this whole project will sound like either idealism or nihilism. But this is a mistake. The impossibility of transparent knowledge does not render all practices of inquiry incoherent. These practices have logics that can themselves be made into objects of hermeneutic inquiry.

A particularly subtle class of hierarchical oppositions is found in systems that assimilate all phenomena in a given territory to a particular metaphor. In these cases, the opposition is tacit and more of a gradient than a sharp divide: in the center are those phenomena that are readily assimilated to the system's generative metaphor; in the margin are those phenomena that can be assimilated to the generative metaphor only by means of unreasonable convolutions. A generative metaphor also induces centers and margins within the construction of particular phenomena: every metaphor constructs a mapping between two semantic fields, and in doing so it inevitably draws certain elements of each field into the foreground and relegates others to the background. Metaphors that liken society to an organism, for example, draw attention to issues of structure, symbiosis, and reproduction, while also distracting attention from issues of consciousness, conflict, and social mobility. Every generative metaphor, therefore, has a margin, which consists of those practical phenomena for which the metaphor provides no ready account. Marginality, of course, is a matter of degree; one will find phenomena being shoe-horned into explanatory frameworks through various ad hoc extensions and extrapolations that are not parsimonious but that are not easily refuted either.

Margins in practice

The application of these ideas to technical discourse is straightforward; their application to technical *practice* is not.

First, discourse. Technical people are systematizers too, even if a computer system does not immediately seem to resemble a philosophical system. But the analogy between technical and philosophical system-building is close: both are intended to assimilate some territory of phenomena in a systematic and complete way to a formal scheme. Technical work is special in that, unlike most philosophical enterprises, the "formal scheme" in question is "formal" in the strong sense of "mathematical." At the same time, technical work assimilates phenomena to the metaphors associated with established model-building techniques. The generative metaphors of any given technical project thereby define a hierarchical opposition between central and marginal cases, that is, between those phenomena that are readily assimilated to the metaphor and those that are not. When the technical practice shaped by a given technical discourse runs into trouble, therefore, its difficulties may conform to a characteristic pattern. Specifically, it will persistently fail to produce reasonable accounts of the phenomena it treats as marginal.

Then, practice. A discourse suggests a way of talking about the world. As Feenberg (1995: 144) points out, it is frequently instructive to watch what happens when a discourse is put into practice. How can we characterize the practical logic of research projects that embody a given discourse in technical artifacts, and specifically in computer models? I have suggested that a technical practice will get itself into trouble around the margins of its generative metaphors. This seems obvious enough when stated abstractly. But it is much more complicated in practice, as researchers move through the cycle of building systems, testing their performance, interpreting what happens, revising their ideas, and building new systems. A research community may accumulate a great deal of experience through this cycle without ever understanding its own underlying discourse or its margins. This does not, in itself, imply that the technical proposals themselves are necessarily misguided (as scientific theories) or useless (as engineering techniques). It could be that the margins of a particular theory really are distinctive in some way, so that they do deserve different treatment than the central cases. (Chapter 14 will return to the point.) More prosaically, an engineering technique that works well 90 percent of the time might be an enormous advance, provided that it fails safely the other 10 percent of the time. This reckoning of percentages, of course, requires the same kind of critical analysis as any other technical practice; the point is that a technical project need not employ representations that transparently grasp the project's practical reality in order to engage it productively.

It is now possible to provide a reasonably precise statement of the reflexive thesis of the book. A technical research program based on a particular generative metaphor will encounter difficulties around its margins. When these difficulties are recognized at all, they will not have words like "margin" written on them; instead they will seem like new technical problems to be identified as topics for future research through technical methods. In particular, when a technical research community encounters manifestations of trouble, it will interpret them within its existing discourse. That is, the community will try to assimilate these manifestations to its customary metaphors and thereby motivate technical solutions to them. Different sections of a community might frame the difficulties in different ways, depending on their particular slants. But the metaphors themselves will not come into question unless the community is aware that they exist and possesses the intellectual tools to analyze them. When this sort of reflexive awareness is lacking, the metaphors will hover in the background, acting as arbiters of plausibility without taking any of the blame. As a result, the research process will resemble a treadmill, in the form of a cyclic incorporation of margins. Every turn of this cycle will seem like technical progress, and indeed it often will be, within certain limits. Yet fundamental commitments will go unquestioned and fundamental difficulties will go unaddressed. Given an awareness of these phenomena, the patterns of difficulty in a technical project can serve as diagnostics, leading the research community to articulate and evaluate previously implicit commitments.

Let me summarize the argument in another way. Scientific inquiries based on technical modeling should be guided by a proper understanding of the nature of models. A model is, before anything else, an *interpretation* of the phenomena it represents. Between the model and the putative reality is a research community engaged in a certain discursive operation, namely, glossing some concrete circumstances in a vocabulary that can be assimilated to certain bits of mathematics. The discourses within which this process takes place are not transparent pictures of reality; nor are they simply approximations of reality. On the contrary, such discourses have elaborate structures and are thoroughly metaphorical in nature. These discourses are not simply ways of speaking; they also help organize mediated ways of *seeing*. They provide the vocabulary for formulating models, interpreting results, and then choosing among revised models. It is thus important that model-builders understand the properties of metaphor systems. This is not a simple matter, since a generative metaphor will have an utterly pervasive influence on the techniques, methods, and priorities of a field. The procedure of diagnosing generative metaphors and deconstructing the discourses they shape does not *disprove* a theory. Instead, it encourages an awareness of the *nature* of theories. The theory may still be useful; it may still be the best theory available; but its development risks getting mired in epicycles unless its developers are fully aware of the dynamics of model-building. Attention to a research project's margins is useful not because the theory is necessarily any less true in its margins, but rather because the systemic "bias" or "spin" of a model's metaphors will become most apparent in its margins.

This version of the reflexive thesis is, unfortunately, incomplete. Let me sketch some of the additional issues, which I hope to take up in another publication. The reflexive thesis, as it is, sets up a dichotomy between a completely unconscious existing practice and a perfectly self-aware alternative practice. Things are obviously more complicated than that. Existing research communities do have reasonably useful ways of talking about the margins of their practices. For example, a given technical method is often spoken of as making "assumptions," which serve to mark out the space of situations within which the method can be applied. Or a theory might involve "idealizations" that cause it to "approximate" the reality it attempts to explain. A more detailed theory of technical work would describe how the practical logic of a given community's projects interacts with its particular modes of self-awareness. The vocabulary of assumptions and approximations reflects one kind of self-awareness; the vocabulary of practical logics and margins reflects another. Neither one is perfect, but the former has the important limitation that the various assumptions and approximations are themselves formulated within the reigning discourse of the community. Far from calling into question the consequences of language, they are part of the intellectual apparatus by means of which the various practical manifestations of a given practice's margins are managed. Thus, a technique that works 90 percent of the time and fails safely 10 percent of the time is readily understood as making certain assumptions that are sometimes false, without at the same time casting doubt on the generative metaphors that lie behind entire classes of such methods.

Another shortcoming of my account of the practical logic of technical work is that my focus on metaphors, while useful, is nonetheless limiting. Before proceeding to substantive applications of my account, I will sketch a more satisfactory account, the elaboration of which will have to await future work. Technical discourses entail a practice of systematic assimilation, by means of which a large number of phenomena are described in a special vocabulary. In cognitive science, this vocabulary includes terms like "knowledge," "planning," and "reasoning." The workings of these terms are not exhausted by laying out an underlying metaphor system. What these terms share with metaphors, however, is a sort of doubled meaning. Just as a metaphor (e.g., lion-hearted or insipid) establishes a point of contact between two normally distinct territories (animals and people; cooking and art), each of these technical terms establishes a point of contact between the world of computational formalism and the world of human existence.[19] It is frequently said that technical practice employs an especially precise and well-defined form of language, but this is misleading. In fact, terms like "knowledge," "planning," and "reasoning" are simultaneously precise and vague. Considered as computational structures and processes, these terms are as precise as mathematics itself. Considered as descriptions of human life, however, they are profoundly *im*precise.[20] AI continually tries to assimilate the whole of human life to a small vocabulary. Human life, though, is far more complex and varied than any known means of describing it. As a result, technical discourses in cognitive science routinely employ their terminology in an extremely vague way, the better to assimilate a broad range of phenomena to a narrow range of themes. Many technical people are painfully aware of the vagueness of their vocabulary. Their attempts to solve the problem, unfortunately, are founded on a misguided conception of linguistic clarity that requires ever more ambitious and elaborate formalizations. This procedure exacerbates the problem by proliferating vague terminology. As a result, it has become spectacularly difficult to think clearly with computational ideas. This is too bad, and I hope that a better understanding of technical language can alleviate the problem.

3 Machinery and dynamics

Mentalism

As a substantive matter, the discourse of cognitive science has a generative metaphor, according to which every human being has an abstract inner space called a "mind." The metaphor system of "inside," which Lakoff and Johnson (1980) call the CONTAINER metaphor, is extraordinarily rich. "Inside" is opposed to "outside," usually in the form of the "outside world," which sometimes includes the "body" and sometimes does not.[1] This inner space has a boundary that is traversed by "stimuli" or "perception" (headed inward) and "responses" or "behavior" (headed outward). It also has "contents" – mental structures and processes – which differ in kind from the things in the outside world. Though presumably somehow realized in the physical tissue of the brain, these contents are abstract in nature. They stand in a definite but uncomfortable relation to human experiences of sensation, conception, recognition, intention, and desire. This complex of metaphors is historically continuous with the most ancient Western conceptions of the soul (Dodds 1951; Onians 1954) and the philosophy of the early Christian Platonists. It gradually became a secular idea in the development of mechanistic philosophy among the followers of Descartes. In its most recent formulation, the mind figures in a particular technical discourse, the outlines of which I indicated in Chapter 1.

This metaphor system of inside and outside organizes a special understanding of human existence that I will refer to as *mentalism.* I am using the term "mentalism" in an unusually general way. The psychological movements of behaviorism and cognitivism, despite their mutual antagonism, both subscribe to the philosophy of mentalism. Behaviorism, on this account, comprises several forms of mentalism that employ the inside–outside language while perversely negating some of its terms:

speaking of internal mental entities, various behaviorists held, is either senseless, or bad methodology, or just scientifically premature. Indeed, the densely organized conflict between behaviorism and cognitivism is the clearest indication of their underlying unity. The victory over behaviorism has become codified as the cognitivist movement's origin myth. Like all such myths, conventional narratives of this forty-year-old fight unintentionally obscure the movement's premises by redirecting attention to secondary issues. (Since I am concerned entirely with those underlying premises, I will bother to distinguish between mentalism and cognitivism only when specifically discussing the cognitivists' debate with behaviorism.) It is striking, for example, that the literature on psychological metaphors exhaustively catalogs metaphors for the mind's internal workings without ever mentioning the inside–outside metaphors that constitute the very idea of the mind (Berman 1989; Edwards 1996: 147–173; Hampden-Turner 1981; Leary 1990; Turbayne 1991; West and Travis 1991b).[2] To my knowledge, the only author to have noted the central role of inside–outside metaphors in AI is K. Gardner (1991), who employed Lakoff and Johnson's (1980) image schema theory to analyze the metaphors of expert systems research. Gardner notes that this literature treats an expert's knowledge as a collection of objects stored in a container; knowledge is structured according to various ordering schemas (surface–deep, part–whole, center–periphery), and knowledge engineers "acquire" it through forcible extraction.[3] AI has elaborated an enormous space of metaphors to describe the internal workings of minds – as static hierarchies or bustling anarchies, constantly innovative or mechanically rote, homogeneous or heterogeneous, largely innate or wholly constructed (Lugowski 1985). This proliferation of *surface metaphors* (Schön 1979: 267) can seem chaotic if the deep metaphor lying beneath them is not clear. The practical logic of mentalist research begins with this pattern.

The metaphor system of inside and outside weaves through a massive vocabulary for talking about structures and processes inside machines. This vocabulary has often been characterized in terms of computer metaphors for human thought, or else human metaphors for the operation of computers (e.g., Davis and Hersh 1986: 240–254; Edwards 1996; G. Miller 1974). But as West and Travis (1991a) have pointed out, things are more complex than this. The word "computers" does not name a fixed quantity; rather, understandings of machinery and of minds have influenced one another for centuries within a deeper discursive field. (Both of them have also influenced, and been influenced by, ideas about social structure.) Moreover, the mentalist tradition has drawn computational vocabulary from a wide range of resources. In comparing these vocabulary items with their earlier meanings, two related patterns emerge:

1. A word that once referred to something in the world now refers to a structure in the computer. Common examples include "situation," "pattern," "context," "object," "list," "map," "structure," and "problem." Individual AI researchers have defined hundreds of others.
2. A word that once referred to an activity conducted by agents in the world now refers to a process occurring entirely in the computer. Examples include "search," all verbs for operations on data structures ("construct," "manipulate," "inspect," "point at," "traverse," "collect," "recycle"), and many predicates on the internal operations of technical entities.

Concisely put, mentalism provides a simple formula that gives plausible answers to all questions of psychological research: put it in the head. If agents need to think about the world, put analogs of the world in the head. If agents need to act in situations, put data structures called "situations" in the head. If agents need to figure out what might happen, put simulations of the world in the head. The tacit policy of mentalism, in short, is to reproduce the entire world inside the head.[4] In planning research, this policy is found in the notion of a *world model,* according to which reasoning about the world depends on having access to a sufficiently detailed simulacrum (see Chapters 8 and 11).[5] It is also found in the notion of *inverse optics* (Edelman and Poggio 1989; Hurlbert and Poggio 1988), whereby the visual system works backward from retinal images to compute a model of the objects in the world that caused them. The development of computational methods over the past forty years has been shaped by the steady encouragement of this discursive pattern. The sophisticated structures and processes that form the basis for AI research are not geared to living in the world; they are geared to *replacing* it.

Mentalism is often closely associated with another doctrine that tends to duplicate the world in abstract form, namely Platonism. Although it comes in numerous varieties, the core idea of Platonism is that alongside the material world there stands a parallel world of ideal forms. Theories of logic and language, for example, frequently veer between psychologism, according to which things like "meanings" are mental structures, and Platonism, according to which those things rest in the timeless, extensionless world of ideals. (See Chapter 11.) What is most striking is how little this distinction has mattered in practice. The mind would seem to be located in space and time – in virtue of being causally associated with someone's head – but theories of the mind have always had extraordinary difficulty relating the mind's internal contents to the external circumstances that those contents represent. The source of this difficulty, often enough, is that theories of human conduct have evolved into Platonic formalisms, losing all reference to human bodies and lives along the way. In other cases, theories of mind are really Platonic theories of ideal form that have simply been transplanted into the head. So, for example, West and Travis (1991a) have also noticed the tendency of AI research to reproduce the world in the head. Yet they attribute the effect not to mentalism but rather to the use of "context-free symbols" in computational models (cf. Collins 1990; Dreyfus 1972). Their explanation might be understood as the flip side of mine, with context-free symbols as a variety of Platonic form. Husserl (1970 [1954]: 60–69) argues that mentalist dualism and the world of abstract idealities in its modern, quasi-mathematical guise are, in fact, internally related, having arisen through the same historical process. If Husserl is right, in retrospect it seems nearly inevitable that thought should have been understood as the manipulation of abstract idealities – context-free symbols – that mirror concrete forms in the world.

Interactionism

Mentalism, like any other discourse with a generative metaphor, has a center and a margin. The central phenomena of mentalist discourse are the varieties of abstract, detached mentation that are loosely grouped under the notion of "cognition"; their paradigm is the culturally organized experience of standing apart from things, orienting oneself to some complex practical difficulty, and engaging in the internalized speech that is sometimes called "thinking." The marginal phenomena of mentalist discourse are those that involve any complex interaction between an individual and an "outside" world. The center and margin of mentalism are not, however, predefined sets of already existing phenomena. Instead, mentalism must be understood as partly a discursive enterprise in which the phenomena of human life are described using a particular metaphor system. The margins of mentalism show themselves in the frequent strangeness of these descriptions. Among the strangest, as will become clear, are the descriptions of phenomena that involve interactions between a person and an environment, even to a minimal degree.

To deconstruct mentalism and explore alternatives, it will be useful to reverse it. Doing so brings us to a perspective on human existence that might generically be called *interactionism.* The generative metaphor here, "interaction," suggests that two entities are acting reciprocally upon one another, back and forth, continually and symmetrically. Interaction is a metaphor, and not just an abstract description, to the extent that it describes both entities, in this case individual and environment, as taking *actions,* not just causing effects. Individual and environment are still understood as different things, but it becomes impossible to understand them except as participants in a third figurative "thing," namely the interaction itself. Interactionism, therefore, is more than the study of interaction; it is, more fundamentally, the use of theoretical categories (units of analysis) that are defined *in terms of* interaction. Whereas mentalism gives a central role to mentation and a marginal role to interaction, interactionism does the opposite. Interactionist words – "interaction," "conversation," "involvement," "participation," "feedback," "metabolism," "regulation," "cooperation," "improvisation," "turntaking," "symbiosis," "management," and so forth – shift attention from cognition to activity. They lead to the positing of structures and processes that cross the boundaries of agents' heads. If some structures and processes *are* entirely inside of agents' heads, interactionism would regard them as simply an unusual special case with no particular privilege.

Later chapters will describe some interactionist computational ideas. The concept of describing human life using metaphors of interaction is, of course, hardly new. After all, one of the lines of computational research that symbolic AI largely displaced in the 1960s – cybernetics – employs a particular set of qualitatively limited but mathematically complex interactionist technical schemata (Ashby 1956; Edwards 1996: 180–187; Hayles 1990; Heims 1991; Wiener 1948). Indeed, interactionism is not really a specific doctrine but a space of possibilities, constructed to facilitate a reorientation of AI. My project has accomplished something if it allows AI people to converse with the disparate intellectual traditions –

dialectical, feminist, phenomenological, biological, and so forth – that have already developed valuable interactionist theories of one sort or another.[6]

Given the options of mentalism and interactionism, how can one decide between them? Considered simply as discursive formations, neither of them has any special advantages over the other. If interactionism someday attains the unreflectively hegemonic status in AI that mentalism enjoys now, the best antidote will no doubt be a mentalist revival. The grounds for deciding between them are more complex; they lie in each discourse's relationship to the technical practice within which it guides day-to-day work. Moreover, principled comparison between the two frameworks is frustrated by their contrasting metaphor systems. They are not simply means of description; they are also means of choosing problems, evaluating solutions, judging plausibility, setting priorities, and parceling out phenomena to subfields. An interactionist AI would be nearly incommensurable with the mentalist one.

Nonetheless, we can obtain some provisional guidance by further analyzing the discourse of mentalism. What kinds of troubles would plague mentalist research if interactionism were a better way to talk about human activity? Interactionism constantly directs attention to the connections between inside and outside; it offers constant reminders that inside and outside are inextricably bound up with each other. If mentalist research always starts by drawing a sharp distinction and firm boundary between mind and world, one should expect inside and outside to try to reunite in some covert way. And this is exactly what happens when mentalism tacitly reconstructs the world inside the head. The attempt to partition inside from outside subverts itself in practice as the mentalist research program is driven to blur the *difference* between inside and outside. The problem is remarkably systematic. The blurring need not be deliberate, of course; it is more likely to operate through the ambiguous use of language and the subtle redefinition of terms. It also happens when computer simulations of thought are designed as if their physical environments were actually part of their mental models of the world.

The early cognitive models of Newell and Simon illustrate this effect. In one report on GPS (Newell, Shaw, and Simon 1960), for example, they introduced the notion of *problem solving* this way:

A *problem* exists whenever a problem solver desires some outcome or state of affairs that he does not immediately know how to attain. . . . A genuine problem-solving process involves the repeated use of available information to initiate exploration, which discloses, in turn, more information until a way to attain the solution is finally discovered. (257, emphasis in original)

The language here suggests some activity in the world: obtaining a desired outcome by exploring and acquiring information. These connotations persist when the authors introduce the GPS program on the next page:

GPS operates on problems that can be formulated in terms of objects and operators. An operator is something that can be applied to certain objects to produce different objects (as a saw applied to logs produces boards). The objects can be characterized by the features they possess, and by the differences that can be observed between pairs of objects. . . .

Various problems can be formulated in a task environment containing objects and operators: to find a way to transform a given object into another; to find an object possessing a given feature; to modify an object so that a given operator may be applied to it; and so on. (258)

"Objects" would seem to be physical things in the world ("the task environment") and "operators" would seem to be physical operations (such as sawing) that can be performed upon them. To solve "problems" would seem to entail that objects are physically discovered or manufactured. The examples they provide, however, are more ambiguous:

In chess, for example, if we take chess positions as the objects and legal moves as the operators, then moves produce new positions (objects) from old. . . .

The problem of proving theorems in a formal mathematical system is readily put in the same form. Here the objects are theorems, while the operators are the admissible rules of inference. To prove a theorem is to transform some initial objects – the axioms – into a specified object – the desired theorem. (258)

The significance of chess positions and mathematical proofs is that they can be represented wholly within the machine: the object and the internal representation of the object are the same. Of course this is not really true, given that a machine representation will fail to capture numerous aspects of chess-playing and theorem-proving as embodied activities. Chess, for example, has rules about touching the pieces and the clock. But the narrative conventions of chess normally delete these details without comment, and so do Newell and Simon. To be sure, when describing a problem that GPS solves, the authors will often (as in the first quotation) use a formula like "find a way to transform" that leaves it ambiguous whether the program will actually do the transforming or whether it will simply specify a procedure by which the transforming might be done. But just as often (as in the second quotation) they will speak as if (for example) "the problem of proving a theorem" is precisely "to transform some initial objects" into desired products. In general, the terms "object" and "operators" are ambiguous: they refer both to real things and actions and to the *representations* of things and actions within the program.[7]

Another example of this phenomenon is found in Simon's (1969: 24–25) famous discussion of the ant on the beach. He begins by suggesting that organized activity arises from the interaction between relatively simple creatures with complex but benign worlds. Yet one page later, he asserts that he is interested only in cognition and not in embodied agency, and he moves on to discuss his studies of human performance on cryptarithmetic puzzles. Hoc (1988: 16) similarly declares, "Throughout this study of planning mechanisms, psychological activity will be seen in an interactionist perspective." But perhaps because it is specifically *psychological* activity that concerns him, he defines an activity not as the interaction between an agent and a world, but as "the interaction between a *subject* and a *task*" (16; emphasis in original). The word "task" is defined in a manner similar to Newell and Simon's problem-solving framework. His substantive theory of plans and their execution, which resembles that of Wilensky (1983), lies within the mainstream AI planning tradition. He defines a plan (1988: 84) as "a schematic and/or hierarchical representation whose function is to guide activity."

The point of this exercise is not that mentalist AI research necessarily blurs its concepts in this way. Although this particular pattern is fairly common, other research projects might manifest the underlying tensions of mentalism in quite different ways. Each project must be considered on its own terms, and my purpose here is simply to offer a vocabulary for analyzing such projects. Nor do I wish to say that mentalism is useless. Considered as an engineering framework, it is likely that applications exist for which devices conceived in a mentalist fashion would be useful. Simple, computationally tractable world models are often sufficient for devices that represent specific environments for specific practical purposes, especially environments that have been deliberately constructed or reconstructed to suit the needs of the device. A critical technical practice would attempt to understand these phenomena, working out better ways of recognizing the practical logic underlying the impasses and trade-offs of engineering work. My own concern, however, is with the scientific question of which framework works best for the description of human beings and their lives. The promise of computational psychology is that the technical practice of psychological model-building, done in a critical way, can cast light on the phenomena of human life. I cannot provide an a priori argument that such a thing is possible. Instead, beginning with the next section, I will develop the outlines of a computational psychology, and indeed a version of computation itself, that takes the interactions of human beings with their environment to be the central phenomenon for computational study.

Machinery and dynamics

Reinventing computation is a considerable task. Whole vocabularies, institutions, curricula, artifacts, workspaces, value systems, and industrial organizations are built around a conception of computation as abstract processes and structures inside of machines. What I can do here is sketch an alternative vocabulary and methodology for AI, motivating them in a loose way with some stories about ordinary activities. Later chapters will analyze fundamental concepts of AI in more detail, leading to some experiments with implemented systems.

I propose thinking about computation in terms of *machinery* and *dynamics*. A machine, as usual, is a physically realized, formally specified device. It is an object in the physical world that obeys the laws of physics. It might have some capacity for change. Its means of operation might be analog or digital or both. The mass noun "machinery" is intended to suggest particular metaphors for thinking about the machinery's physical realization: its operation depends on a configuration that is more or less fixed. Talk of inside and outside applies to machinery, but only in the straightforward physical sense: we have brains in our heads. It might make sense to say that the machinery has "inputs" and "outputs," but these words must be construed cautiously, in a minimal way; as Maturana and Varela (1988: 169) point out, if the machinery engages with its environment differently in the course of different activities, it may be impossible to define a fixed array of "inputs" and "outputs" that are always useful in describing it.

The notion of dynamics is less familiar. It concerns the interactions between an individual (robot, ant, cat, or person) and its surrounding environment. The word might be used in a number of ways. In making dynamic explanations one often isolates a particular "dynamic," which is a common regularity in the way that some sort of individual interacts with some sort of world. One might employ the word as an adjective by speaking of "dynamic explanations," "dynamic theories," and the like. One might also speak of the "dynamics" of an interaction between a particular class of agent and a particular class of world. This simply means "everything one might say about the way the agent and world interact."

A few examples of specific dynamic regularities will motivate some general points about the notion of dynamics.

> In your kitchen cupboard you probably have a set of bowls stored in a stack. If you use one of them, you are likely to return it to the top of the stack. Over time, the bowls you never use (or no longer use, or have not begun using) tend to sink to the bottom. If you use some of the bowls and not others, then the bowls you often use and the bowls you rarely use become segregated.

These sinking and segregation effects are dynamics. Though they are regularities in the interaction between a cook and a kitchen, they are neither explicit policies followed by cooks nor inherent properties of kitchens. Instead, they are the joint product of actions aimed at other goals and environments organized according to customs. The bowls do not sink and segregate because the dynamics act upon them as an outside force; they do so because on a series of occasions you returned the bowls you have used to the top of the stack. The effect is an emergent property of particular concrete interactions.

This notion of the emergent organization of activity has not been central to computational discourse. Among AI people, therefore, the notion of a dynamic risks being confused with the notion of a plan. But a dynamic is not a structure in any agent's head. Theorists might describe dynamics, but these descriptions need not be realized in a data structure or a plan or a mental object of any kind. Having identified a recurring form of interaction between an agent and its world, one can set about determining what kinds of machinery might be compatible with it. The agent *participates in* the dynamic but is not solely *responsible for* it. Here is another example of this point:

> When I was first writing this book, I had a record player and a shelf of records in my office. Since I played upward of two dozen records a day when I was working, a routine slowly arose for getting a record down off the shelf, removing it from its sleeve, picking a side to play, putting it down on the turntable, cleaning it and setting it going, removing it from the turntable when it's done, returning it to its sleeve, and returning it to its place in alphabetical order on the shelf. Though this routine happened quickly (about fourteen seconds to put the record on and about twelve seconds to put it back), it was enormously complex, involving several changes of grip and orientation. One evening I went through this routine about thirty times and wrote down every detail of it. In doing so, I was able to satisfy myself of the accuracy of the following hunch about it: if I always played only side A (say) of a given record, then side A would always be facing up at the point in my routine where, having just removed the record from its sleeve, I checked whether the side I wished to play was facing upward or downward. This is unfortunate; since I happened to be holding the record from underneath just then, it would be much less clumsy to turn it over in the course of putting it down on the turntable. This invariant in my interactions with my office is a dynamic.

A dynamic continues occurring only ceteris paribus; many different events might interrupt it. You might use all the bowls in a stack for a party one evening so that the next morning the newly reconstituted stack will be scrambled; or someone might play my records in my absence. In studying an activity, it is probably best to concentrate on its more stable dynamics. But any dynamic description that would aspire to the status of natural law is limited by the possibility that any of a thousand additional factors might arise to change the outcome next time.

A given dynamic can depend on a wide variety of facts about both the individual and the world. It depends on the world in that stacks require gravity, objects are often designed to stack, stacks tend to stay orderly unless disturbed, my records are at eye level whereas my turntable is at waist level, and so on. These particular dynamics depend only on some rather vaguely described aspects of the individual who participates in them (puts the stacked objects back reliably, checks record labels). Many others depend on more specific aspects of the individual. Consider, for example, the following:

> This story covers four days one winter. One morning upon arriving at my office, I decided I was tired of my coat cluttering my office, so I decided to leave the coat lying on top of the file cabinet just outside my office door. Shifting my concern to the day's work, I walked into my office and pushed the door shut behind me as always – except that today it didn't slam behind me as usual. Investigating, I found that an edge of the coat had caught in the door jamb, preventing the door from closing. I moved the coat out of the way, closed the door, and went back to work. The next day I left the coat on top of the file cabinet, headed into my office, and pushed the door shut as always – and it didn't slam behind me again. This time, though, I immediately knew what the problem was. The next day I left the coat on top of the file cabinet as before, but as soon as I turned to head into my office I realized that the coat was liable to get caught, so I moved the coat out of the way again. The fourth day, I was aware of the problem as I was placing the coat down on the file cabinet, so I made a point of placing it as far as practicable from the door jamb. On each day, a bit of the previous day's insight had drifted back toward the beginning of the routine.

A dynamic description is not simply a description of an agent's outward behavior. Instead, a dynamic pertains to a recurring causal chain whose path passes back and forth between components of the individual's machinery and objects in the world. The dynamic in this case is the "drift" of certain recurring elements back toward the beginning of a cycle. This dynamic is the sort of thing that classical psychology called *transfer.* For mentalist psychology, transfer occurs between one isolated situation and another. The interactionist approach, by contrast, is to view transfer as part of the larger dynamics of activity in the world of everyday life. (Transfer and the other phenomena in these stories are discussed further in Chapter 6.)

The continued existence of a dynamic can also depend on the dynamic remaining unnoticed. It is common to notice and describe an interactional invariant in one's life (though few people call them "interactional invariants" or have any need to make scientific generalizations about them). The dynamic picture might then become more complicated if you begin deliberately doing things differently or if you go out of your way to encourage or retard the effect. Having noticed your stack of bowls segregating, for example, you might deliberately put a more valuable bowl back toward the bottom of the stack to ensure that more expendable bowls are subjected to the risks of everyday use.

Interactionist methodology

The examples in the preceding section all illustrate the idea that the organization of everyday activity is a product of interactions between agents and environments. They are not a proof, of course, but rather some initial hints toward the cultivation of a worldview. In particular, they are intended to help establish a new and different connection between technical ideas and the experience of everyday life. Just as mentalist AI has been influenced by a Cartesian preoccupation with personal isolation, an interactionist AI will require an increased awareness of personal involvement in the physical and social world. New modes of experiences will lead to new technical intuitions. In particular, attention to the dynamics of everyday activity can impress us with the inherent orderliness of everyday life. Once intuitions are revised in this way, the agenda for AI research begins to change.

Interactionist AI is a broad and ecumenical activity. Its fundamental question is: what kinds of machinery get into what kinds of dynamics in what kinds of environments? We can ask this question about robots in factories, insects in factories, insects in the jungle, cats in the jungle, cats in the suburbs, people in the suburbs, people in factories, or whatever and wherever we like. As these examples make evident, the answers to this fundamental question will depend in important ways on the kinds of agents and worlds we choose to investigate. The question, moreover, leads quickly to other, older questions about biology and culture and history. AI does not replace these older questions; nor does it supplant the older answers. Quite the contrary, AI ought to participate in a dialogue with a variety of other intellectual projects. What it brings to each such conversation is a body of experience investigating issues of physical realization. In using the term "machinery," I am not restricting attention to engineered artifacts, much less to factory machines. Nor, in using the term "dynamics," am I restricting attention to processes that can be characterized using differential equations or any other sort of mathematical description. The intuitions and methods of AI had their origins in engineering, but with a sufficiently critical attitude they can be much more general than that, applying just as well to scientific enterprises.

Interactionist research methodology consists of a sustained inquiry into issues of machinery and dynamics. Good research will move back and forth between the two of them, using insights about each to deepen inquiry into the other. Machinery and dynamics constrain one another: only certain kinds of machinery can participate in given kinds of dynamics in a given environment. For example, an agent that always puts its tools back where they belong may need simpler means of finding those tools than an agent that leaves them lying about. Likewise, an agent whose machinery is immutable can change its interactions with the world only by changing the world or its relationship to the world.

The mutual constraint between machinery and dynamics is the subject matter of computational intuitions and of the principles that give these intuitions heuristic form. In practice, research will generally start from certain leading assumptions about machinery, for example that the agents in question employ a particular architecture whose properties have been explored by the existing literature. Given this starting point, design issues pertaining to any aspect of that machinery (such as timing, internal state, speed, digital vs. analog, or noise tolerance) will make predictions about the dynamics of that agent's interactions with particular environments. Questions about dynamics can be addressed to the empirical phenomena, either by formal appeal to existing fields of study (anthropology, phenomenology, biology) or by informal collection of anecdotes such as the ones I reported earlier. Anecdotes prove little on their own, of course, but computational research is sterile if it proceeds in the abstract, without *some* kind of experiential contact with the phenomena. Research is progressing if design issues are steadily raising questions about dynamics and observations about dynamics are steadily deepening the designer's understanding of the practical reality of design.

The most important principle of interactionist methodology is *machinery parsimony:* postulate the simplest machinery that is consistent with the known dynamics. This principle is critical when, as so often in technical work, the temptation arises to build some very general type of machinery to model some broadly characterized phenomenon. Instead of designing steadily more general-purpose machinery, it is better to understand the dynamics that people (or cats or insects, etc.) actually get into. Often such inquiries will turn up empirical distinctions that correspond to distinctions of technical practicality (Horswill 1995). Attempts to equip an agent with elaborate schemes for mapping its environment, for example, are often halted by the combinatorial complexity of the requisite algorithms. But general-purpose mapping devices seem less necessary after some empirical study of the signs and sight lines and sources of advice that the world makes available; a few simple way-finding strategies might interact with these helpful resources to achieve the same effect. All of the technical arguments in this book are instances of the principle of machinery parsimony. Deeper understandings of dynamics can lead to simpler theories of machinery.

Why might an understanding of an agent's interactions with its world lead to simpler hypotheses about its machinery? Far from the Cartesian ideals of detached contemplation, real agents *lean on the world.* The world is its own best representation and its own best simulation (Brooks 1991). Interactions with the world, both past and present, provide many ways to alleviate computational burdens (McClamrock 1995). Symmetries and rhythms allow us to organize our physical activities with simpler control mechanisms (Raibert 1986). Direct observation can often replace elaborate deductions (Kirsh 1995). Inspection of one's materials can often indicate what has to be done next, eliminating the need for complex control structures (Hammond, Converse, and Grass 1995; Norman 1988). Cultures provide representational artifacts that support complex forms of cognition (Goody 1986; Latour 1986). Activities are arranged in space in meaningful ways (Harré 1978; Lansky and Fogelsong 1987). Improvisation often eliminates the need for detailed plans (Agre and Horswill 1992). Complex tasks are distributed among participants in group activities (Cole and Engeström 1993; Hutchins 1995). Incremental learning can substitute for mechanisms whose great generality is rendered useless by their great expense (Berwick 1985). The social world organizes activity and learning in numerous helpful ways (Lave and Wenger 1991). Past experience provides abundant precedents for the decisions we must make in our activities (Schank 1982). By investigating the full range of these dynamic phenomena, designers can gain experience with the "fit" between particular types of machinery and particular kinds of environments.

Making technical sense of these intuitions is, of course, hard work. The hardest part is accounting for all of the phenomena which interactionism understands as marginal. The long-cultivated habits of mentalism place enormous emphasis on these phenomena: on innovation, cogitation, problem solving, and so forth. The point is that these phenomena are considerably more complex – heterogeneous, error-prone, socially organized, culturally specific, and so forth – than mentalism makes out. They will be the last phenomena to receive satisfactory computational explanation, not the first, and interactionist research will assign them a lower priority than mentalist research will. Chapter 14 will return to them. Meanwhile, the empirical and theoretical investigations in other fields can serve as rough guides to the *sort* of theory one might hope to obtain.

Interactionist and mentalist methodologies also differ in the way that they individuate "problems" for research. The interactionist seeks principles about the relationship between machinery and dynamics, isolating particular dynamic phenomena and asking about their implications for machinery. The mentalist, by contrast, generally studies something that (as I mentioned in Chapter 1) mathematical theories of computation since Turing have misleadingly called a "problem": a mathematical function from a finite input set to a finite output set. An algorithm "solves" a problem if it computes this function. The difficulty with this sort of problem and solution is that they make strong assumptions about the kind of interaction a given device will have with its world: a single input, a spell of abstract computing, and a single output. While this is conceivably a reasonable description of some human activities, it is surely a poor description of most of them. This conventional notion of "problem–solution" covertly prescribes the use of mentalist metaphors in the very specification of research topics. It follows that both the theory of Turing computability (Turing 1936) and the theory of computational complexity (Tarjan 1987) introduce some strong and rarely acknowledged assumptions into AI. This is a serious situation, since these theories provide the principal technical means of answering questions like "What is a computer?" and "What can computers do?"

In summary, I am suggesting a reversal of AI research values. Faced with an empirical phenomenon to explain, our first explanatory recourse should be to dynamics, not to machinery. Likewise, faced with a technical problem to solve, our design methods should begin with dynamics, not with machinery. Heretofore, AI has placed a high value on new machinery. Instead, I would like to suggest that we place our highest value on *getting rid of* machinery. We should aspire to invent novel dynamic effects and experience regret when forced to invent novel devices. As a result, the models I will present employ simple, largely familiar forms of machinery. This machinery, however, is not the critical contribution of the work. My contributions are my case studies of a new perspective, my description of certain dynamic aspects of activity, and my tentative suggestions about how a particular type of simple machinery is capable of participating in these dynamics.

4 Abstraction and implementation

Structures of computation

All engineering disciplines employ mathematics to represent the physical artifacts they create. The discipline of computing, however, has a distinctive understanding of the role of mathematics in design. Mathematical models can provide a civil engineer with some grounds for confidence that a bridge will stand while the structure is still on paper, but the bridge itself only approximates the math. The computer, by contrast, conforms precisely to a mathematically defined relationship between its inputs and its outputs. Moreover, a civil engineer is intricately constrained by the laws of physics: only certain structures will stand up, and it is far from obvious exactly which ones. The computer engineer, by contrast, can be assured of realizing any mathematical structure at all, as long as it is finite and enough money can be raised to purchase the necessary circuits.

The key to this remarkable state of affairs is the *digital abstraction:* the discrete 0s and 1s out of which computational structures are built. This chapter and the next will describe the digital abstraction and its elaborate and subtle practical logic in the history of computer engineering and cognitive science. The digital abstraction is the technical basis for the larger distinction in computer work between *abstraction* (the functional definition of artifacts) and *implementation* (their actual physical construction). Abstraction and implementation are defined reciprocally: an abstraction is abstracted *from* particular implementations and an implementation is an implementation *of* a particular abstraction. This relationship is asymmetrical: a designer can specify an abstraction in complete detail without making any commitments about its implementation.[1] The relationship is confined to the boundaries of the computer; it does not depend on anything in the outside world.

Computer engineering uses abstraction to organize its work into a hierarchy of relatively independent levels, each with its own research community. At the lowest level, device physicists can use any materials they like to build their circuitry, as long as they can support the most basic abstraction of binary arithmetic. Computer designers, likewise, can specify whatever architectures they like, as long as they can be programmed in a general way. The authors of compilers (the programs that translate high-level programming languages such as Fortran and Pascal into a form that computer hardware can interpret directly) can explore a wide variety of techniques, as long as their compilers support the semantics of some language that programmers find congenial. These programmers, finally, can implement whatever abstractions they choose, simply by writing programs. These programs may employ several levels of abstraction as well, each with a formally defined relationship to the levels below and above it.

Philosophical accounts of computation and its relationship to AI have placed much emphasis on the independence of the various levels of a computational system (e.g., Haugeland 1985: 58). And indeed, a snapshot of computational practice at any given place and time will probably resemble this picture of neatly separated levels of abstraction. The historical reality, though, is more complicated. The practical logic of computing is driven by a variety of factors, but the most pervasive factor is straightforward efficiency: the amount of useful computational work that gets done in the service of specified goals by a given amount of machinery in a given period of time. The demands of efficiency continually undermine the independence of the various levels of abstraction in conventional computer systems; more generally they undermine the separation between abstraction and implementation. At any given juncture in technical history, this tension becomes manifest through the emergence of new and often surprising design trade-offs, as well as a coevolutionary pressure for mutual adaptation among the various levels of abstraction.

In their everyday practice, few computer programmers must consciously negotiate trade-offs between matters of abstraction and implementation. Programmers are very concerned with efficiency, of course, and they have accumulated and codified a great deal of experience in choosing the most efficient algorithms for a given task. Nonetheless, their design choices are heavily constrained by the particular computer architectures with which they work. The vast majority of programmers work on computers with serial architectures. A serial processor is called "serial" because it performs only a single small operation, or "instruction," at a time.[2] It has often been observed that a serial computer attempts to defy the demands of physical implementation. Though it might consist of several million circuits distributed through a cubic foot of space, only a small proportion of these circuits will actually be doing useful work at any given time (Hillis 1985: 1–5). Serial machines are thus located at an extreme in the space of trade-offs between freedom of abstraction and efficiency of implementation: they offer a maximum of freedom with a minimum of efficiency.

As a result, many people have assumed that computers would be much more efficient if they were highly parallel, that is, if they worked by performing a large number of useful operations all the time. A highly parallel computer would distribute the useful work uniformly throughout its circuitry rather than trying to cram it into a single central processor. Computational theorists and computer designers have invested enormous effort into exploring the forms that distributed computation might take (Almasi and Gottlieb 1994; Denning and Tichy 1990; Hillis and Barnes 1987; Kitano and Hendler 1994). But parallel computers have not become prevalent, because programmers have not been willing to give up the freedom of serial computing. The performance of a serial machine might be tolerable for most tasks, and so the computer market provides manufacturers with the enormous amount of capital required to produce successive generations of serial machines. Any given parallel architecture, by contrast, will probably occupy a niche market because its performance will be extremely high on some categories of tasks and relatively poor on others. For example, a computer that consists of a dozen serial processors communicating over a network will perform well on tasks that can be decomposed into a moderate number of roughly independent subtasks; on other tasks, the processors will probably spend most of their time waiting for the network rather than doing useful work. Discovering a suitable decomposition for a task is often difficult. Specifically, in order to benefit from parallel execution, a computation must be organized so that it can be distributed in a natural way across the spatial extent of a parallel machine (cf. Hillis 1985: 137–144). On this view, *one foundation of efficient design is a natural correspondence between the structure of a computation and the structure of its physical implementation.*

An important example of this principle is found in the first several stages of visual processing (known as "early vision"), according to computational neurophysiologists such as Marr (1982). According to this theory, the visual cortex is organized, at least in part, as a set of modules, each of which computes some function from one version of the visual image to another. Although the exact roster of these computations is still uncertain, a typical proposal is that a certain module takes in a slightly blurred version of the retinal image and produces a map of where the edges in the image are located. Another module takes in these edge maps for both eyes (i.e., a stereo edge map) and produces a map of "depth" (i.e., how far away the physical edges actually are). The salient aspect of these computations for our purposes here is their physical organization. Nature might have implemented these modules in many ways, but in fact they are constructed in accord with a natural correspondence between the uniform two-dimensional structure of the images and the uniform two-dimensional structure of the neural machinery itself. That is, each successive stage of the cortical circuitry that implements early visual processing is organized as a flat sheet of computing elements. As a result, most of the wires within a given stage can be fairly short, transporting a signal from a given computing element to its neighbors.[3] The resulting visual architecture has a striking simplicity: whereas a serial computer must constantly shift its attention among different processes and different data, every element of the early visual system performs its single assigned task all the time. But not all computations will be so lucky. When computations are less naturally implemented, the work is distributed less evenly through the circuitry. Of course, a designer might have good reasons for building a computer that implements its computations in a relatively inefficient way: standardization, maintenance issues, functional flexibility, conventional divisions of design labor, and so forth. That these nonoptimal design options are available at all is testimony to the power of computation: the separation between abstraction and implementation remains perfectly valid as a conceptual matter, regardless of how natural it is as a practical matter.

What makes the conceptual separation between abstraction and implementation possible is a simple but profound invention: the wire. By connecting two physically distant points in a circuit, wires allow the designer the maximum latitude in choosing the manner in which a given abstraction is to be implemented. Given enough wires, a computer designer can physically implement an arbitrary abstract network of computing elements. This is why technical papers about computer architectures can provide functional specifications using diagrams made of boxes and arrows without providing any clue as to the actual physical geometry of the circuits that implement them. And indeed, the people who finally build these computers spend a large part of their time getting the sheer mass of wires under control. By loosening the constraints imposed on a designer by the three-dimensional geometry of space, wires allow the designer to decouple the causal patterns of an abstractly specified device from the spatially localized causality of its physical implementation.

It is thus possible, as an engineering option, to implement one's computations in perfect defiance of physical locality. But in doing so one pays a price in the efficiency of the resulting artifacts: computers with shorter wires are faster, other things being equal, because electrical signals travel along wires at a finite speed. As computers grow larger and the pure velocity of computation becomes a dominant design goal, the technology begins to break down and reorganize itself within new, more rigorous correspondences between abstraction and implementation. Hardware designers, for example, often cut wires to precise lengths, so that each signal will arrive at its destination at the exact moment when it is needed. Similarly, even though a memory chip maintains the digital abstraction in its outward behavior (if you store a 1 in a given memory location, you will get a 1 back later on), most memory chips do not actually employ the digital abstraction in their internal circuitry, which is in fact a sophisticated analog circuit that takes advantage of the physical properties of the materials from which it is made. Furthermore, one of the innovations of the IBM 801 computer, which started the trend toward RISC (reduced instruction-set computing) processor architectures, was that the processor and its compilers were codesigned, introducing peculiar instructions that permitted the processor to keep more of its circuitry usefully busy in complex situations (Hennessy and Patterson 1994; Radin 1983). In each case, the drive for efficiency has brought a more detailed understanding of the physical reality of computation to bear on the problem of implementing digital abstractions.

The moral of the story is that abstraction and implementation, far from being independent levels of technical description, are actually engaged in a dialectical relationship. Though one might wish to define each of them without any heed to the other, their fates will necessarily become intertwined as demands for efficiency intensify. As technological development explores the endlessly ramifying space of design trade-offs, implementation and abstraction will continue to be pulled into a close and potentially complicated relationship. Factors of marketing and management have largely hidden this dialectic from the consumers and philosophers of computation. Yet, I would like to argue, it is a fact with important and little-understood consequences for computational psychology.

A case study: variables

The history of computer design can be understood in large part, then, as the unfolding of dialectical interactions among various aspects of implementation and abstraction. A case study in this history that will take on particular significance in later chapters of this book concerns the development of two central abstractions of serial-computer programming, pointers and variables.

Serial computers were originally designed to automate calculations already performed in a routinized fashion by human beings – themselves often called computers – in military and business environments. Today this type of step-by-step calculation scheme is probably most familiar, at least in the United States, from income tax forms. A typical calculation step, or *instruction*, might be glossed as "Add lines 31 and 40 and enter the result on line 41." This instruction would be represented inside the machine using a sequence of four numbers, one for the "add" operation and the other three for the *addresses* of the *registers* containing the operands (31 and 40) and the result (41). This is the *register-transfer model* of computation upon which virtually all modern computers are based. Each register, or "word," of the computer's memory is implemented by an actual physical circuit that stores a sequence of binary values.[4] The first computers had only a few hundred words of memory, but memories of several million words are common today. Each register's address is itself a binary quantity. In a thousand-word memory an address can be specified with ten bits; in a million-word memory twenty bits are required. As a result, a single instruction can require nearly seventy bits: half a dozen to specify the operation ("add," "subtract," "compare," etc.), twenty each for the operands and result, and a few more for other miscellaneous purposes.

In practice this way of specifying instructions is too cumbersome. Much space is wasted by instructions that do not require the full three-operand format. As a result, computer architects began to reserve the word "register" for a small portion of the memory. In the tax-form analogy, registers corresponded to scraps of paper where individual operations were set up and performed, with the results being copied back onto the indicated line of the tax form. An instruction such as "Add words 31 and 40 and store the result in word 41" would now be broken into three parts:

1. Load word 31 into register 3.
2. Add word 40 into register 3.
3. Store register 3 into word 41.

Breaking instructions into smaller pieces permitted each instruction to be specified in a single word of memory (perhaps five bits for the operation, five bits for the register, twenty bits for the memory address, and a few miscellaneous bits). Furthermore, since the registers could be implemented in faster circuitry than the millions of general-purpose memory words, many computations could be accelerated by keeping frequently used quantities in registers as long as possible. More generally, computer memory is now organized in a hierarchy, with small numbers of faster memory elements close to the central processor and large numbers of slower elements farther away.

An important extension to the register-transfer model of computation was the invention of *indirect reference.* In the conventional model, a word of memory contains either an instruction or an item of data, and these data refer to things in the outside world, such as an air pressure or an adjusted gross income. In the extended model, a word of memory can also contain the address of *another* word of memory. An instruction can now refer indirectly to a quantity in memory, not by specifying its address but by specifying an address where its address can be found. For example, consider this instruction:

Load word @7 into register 3.

The "@" symbol indicates indirect reference. The execution of this instruction proceeds in two steps: the processor first reads the contents of register 7 and then, interpreting the resulting value as a memory address, loads the value at *that* address into register 3. Without indirect reference, any given serial-machine computation is laid out across the machine's memory in a fixed configuration, in the same way that an income-tax calculation is laid out across a piece of paper; every word of memory has a fixed meaning ("dividend income," "social security benefits," "alternative minimum tax," etc.). Indirect reference frees a computation from this fixed correspondence to the underlying hardware. In practice, programmers use indirect reference in stereotyped ways, using a set of conventions that collectively implement some useful form of abstraction. As with all abstractions, systematic use of indirect reference entails a decrease in efficiency, since an additional memory "fetch" is now required to load each value from memory. Programmers have often argued about the costs and benefits of programming innovations based on indirect reference, but the cause of increased abstraction has usually won.

One such innovation is the use of *pointers* to create dynamically linked record structures. A particularly simple form of record structure is the *list,* in which each record includes the address of the next record. This address is called a pointer. The records can be located anywhere in memory, but a program can step through them by following the pointers. In the Lisp programming language, for example, the simple list of four symbols that would be notated as `(london paris zurich prague)` would be implemented using four records, as follows:

address 704:
value = "paris"
next = 1731

address 1030:
value = "london"
next = 704

address 1366:
value = "prague"
next = 0

address 1731:
value = "zurich"
next = 1366

A register could "point" at the list simply by containing the number 1030, which is the address of the first record in the list. By convention, the pointer to address 0 in the "prague" record indicates that the list has ended (Knuth 1983).

By using enough interlinked lists, a programmer can construct enormous symbolic structures. These structures might be interpreted as computer programs (in a compiler) or as syntactic structures (in linguistics) or as a representation of the outside world (a "knowledge base" or "world model"). Given some memory and the operations for manipulating pointers, a program can build new structures, tear down old ones, revise existing ones in accord with changing conditions, and so forth. This idea is the basis of symbolic programming (Abelson and Sussman 1984; Newell 1961).

The notion of a variable is more subtle. The problem is that the term "variable" has no single definition. In fact it has a wide variety of definitions, some computational and some not. Historically, the algebraic notion of a variable arose slowly, acquiring a recognizably modern form only during the Renaissance (Klein 1968). This is the kind of variable one encounters in middle school, in exercises such as

$$3x - 4 = 5.$$

Here, at least on one conception, the variable x "stands in" for a definite but currently unknown quantity. Another mathematical use of variables is in defining functions, as in

$$f(x) = x^2 - 7x + 5.$$

In this case, the meaning is something more like "Whatever x is, $f(x)$ is this," where each occurrence of x is supposed to correspond to the same value. In set-theoretic terms, a function is simply a set of pairs, each matching a value of x to the corresponding value of $f(x)$. The word "variable" does not refer to any part of this structure; instead, it refers ambiguously to the symbol x and to the single argument position of the function f (a "function of one variable"). Yet a third use of variables is quantification in formal logic, for example:

$$\forall(x)\mathit{person}(x) \rightarrow \exists(y)\mathit{mother}(y,x).$$

"Everyone has a mother."

Even more obscure types of variables are found in higher branches of mathematics. Galois theory, for example, can be used to investigate the solutions of polynomials; one begins by taking some field, F, adding a symbol such as X, and then "closing" the resulting set under the field's equivalents of addition and multiplication to obtain a new extended field,

$F[X]$ of polynomial formulas (Goldstein 1973: 301–323). This X, however, is not just an element of notation like the x's of earlier examples; it also designates a symbol-like component of the mathematical structures that are being investigated.

In each of these cases, the common element is a syntactic indication of reference within some range of possibilities without a predetermined choice among them. But this is just a family relationship across a disparate series of phenomena, each of which is perfectly well defined in its own context. Nonetheless, it will be useful to give this vague general notion a name, so I will call it *nonspecificity*.

Programming languages employ a wide range of variable-like mechanisms, each of which resolves the tension between abstraction and implementation in its own way. In each case, the purpose is to permit an algorithm to be expressed without reference to particular numbers or symbols. Just as a million people can fill out the same tax form and come up with different correct answers, so a million computers can run the same program on different input data and return different correct outputs. Every programming language comes with its own abstract conception of computation (its *virtual machine*), which may or may not stand in any direct correspondence to the machinery that will run the programs. Programming languages are generally reckoned as "low-level" or "high-level" according to the closeness of this correspondence. At the lowest extreme, a *machine language* consists of the bit-patterns that a given computer architecture can execute directly. An *assembly language* provides the programmer with comprehensible symbolic codes that can be translated straightforwardly into a given machine language. A slightly higher-level language, like the widely used language C, maps closely onto the generic register-transfer model of serial computing without being biased toward any particular serial architecture. And a truly high-level language, like Lisp, will include a variety of abstract features whose mapping onto a conventional register-transfer architecture is not straightforward. In order to execute a program in such a language, the programmer provides the program as input to a compiler, which translates it into a given architecture's machine language.

The "height," or degree of abstraction, of a programming language can be measured in a rough way by the extent to which it uses indirect reference to loosen the correspondence in time and space between the program and the machinery that implements it. In early programming languages the correspondence was strong. A variable in such a language was implemented through a direct reference to a word of memory whose contents could actually vary: it might have the value of 23 at nine o'clock and the value of 31 a minute later. Variables seemed like mutable, physical *things* and bore little resemblance to variables in mathematics.

The spatial and temporal correspondence between the program and its implementation dissolved quickly with the development of high-level languages. These languages implement variables not in definite memory locations but rather in *stack frames* which allow several instances of the same stretch of code to be in progress at the same time. To load the value of a variable, the machine performs an indirect reference within an area of memory known as a *stack*. A given instance of some variable has a definite lifetime, during which its value is unlikely to change. In general, the trend is for variables to become less like physical things and more like the variables of mathematics. Indeed, in a language like Prolog the notion of time has almost no place, so that variables become abstractions with complex mathematical properties and a highly indirect relationship to the underlying hardware. (In practice, however, Prolog programmers are vividly aware of the temporal order in which their programs will be executed.)

The most extreme accomplishment in the general trend toward computational abstraction is Smith's (1985) 3-Lisp language, whose formal specification envisions an infinitely deep tower of abstractions that never "grounds out" in physical hardware. As a result, a 3-Lisp program can sustain the illusion of actually modifying the machinery upon which it is running. The actual running system underwrites this illusion by quickly interpolating a new layer of abstraction between the implementation and the previously existing abstractions each time the immutable hardware threatens to "show through." The 3-Lisp language encounters the trade-off between efficiency of implementation and freedom of abstraction in its most radical form.

When high-level languages were first developed, it was not obvious that they were worth the trouble. Computers were still relatively expensive to operate and the compiled programs were slow. But compiler technology has grown steadily more sophisticated, so that little is lost by taking advantage of the convenience of high-level languages. As a result, assembly languages are used only for exceptional purposes. But compilers for serial computers have a relatively easy job because they have little to gain by reasoning about the relationship between the structure of a computation and the structure of its physical implementation.[5] The compiler's only goal is minimizing the number of instructions that are executed within the extended register-transfer framework.

This issue is particularly crucial for computational theories of cognition that rely on highly abstract notions of variables. Competent agency requires a significant degree of nonspecificity: one must be equally alert on Monday and Tuesday, in conversing with this person or that, drinking from one cup or another, walking down one street or the next. Computational models have generally achieved this nonspecificity of reference to the concrete particulars of thought and action through the use of variables. In some theories these are explicitly the quantified variables of formal logic. In other theories, for example in production systems, the notion of variable is tied to a specific mechanism in which nonspecific "rules" have certain effects when the agent's database contains specific structures that they "match." (This is discussed further in Chapter 7.) In each case, the agent represents the objects in specific situations by assigning them names, or "constants," such as BLOCK-9 and CAR-37. The agent applies its nonspecific knowledge to particular individuals by "filling in" the variables with the names. Likewise, the agent formulates the general lessons it learns in a given situation by replacing the constants with variables, thus producing nonspecific knowledge that can be applied to other individuals on another day. In each case, the use of variables induces a degree of seriality in a computation that might otherwise be performed in parallel; the relatively complex mechanism that matches variables with constants is a scarce resource that can be assigned to only one purpose at a time (Newell 1980). This understanding of nonspecific knowledge will take on a deeper significance toward the end of Chapter 10. For the moment, the important point is its high degree of abstraction from conventional computer architectures. The steady increase in basic circuit speeds has long allowed computer science to take advantage of the virtues of abstraction without the costs of abstraction becoming obtrusive. This trend has reinforced, and has been reinforced by, the tendency of cognitive science to understand a wide variety of concepts in abstract terms. In each case, I will argue, the result has been the steady entrenchment in computational practice of a mentalist view of both human beings and computation.

The example of variables demonstrates how such an entrenchment might take place. Recall that specific conceptions of human beings do not inhere in machinery as such. Instead, they adhere in design practices whereby machines are built whose operations can be narrated in intentional terms. On this view, the variables of logic formalisms and programming languages provide a way of building a machine whose operation can be narrated as the application of a nonspecific competence to specific circumstances. Technical innovations might someday supersede the use of variables, of course, but only if they provide alternative discursive forms for narrating the operation of an agent's machinery. As a practical matter, any such innovations will be available to designers only once the technical community has routinized their use and integrated them into the whole support network of factories, instruction manuals, technical standards, repair facilities, programming language features, testing methods, and so on. The AI research community has historically invested enormous energy in maintaining its own largely distinctive infrastructure, including its own programming languages and computer architectures, precisely to provide itself with the technical resources to construct computer systems within its evolving repertoire of technical schemata. But this infrastructure rests upon, and in recent times has been increasingly subsumed by, the larger infrastructure of the computer industry generally.

The connectionist movement makes the practical logic of this process evident. Starting in the late 1970s, computational psychologists in the connectionist movement attempted to reinvent computation. Rejecting the model of computation implicit in symbolic programming, they began again from machinery whose form is modeled more closely on the circuitry of animal and human nervous systems. Instead of employing the register-transfer model of computing, connectionist models employ large networks of computationally simple "nodes" (loosely modeled on neurons) that are connected by a mostly fixed set of wires (loosely modeled on axons and dendrites) that are capable of transmitting only simple codes (such as binary values or firing rates). In making these alternative commitments, the connectionists lost access to the wide range of technical schemata that symbolic programmers use to design systems whose operation can be narrated in intentional terms. Dreyfus and Dreyfus (1988), among others, have hoped that connectionist research would develop an alternative repertoire of technical schemata based on different, nonsymbolic commitments about representation and cognition. One starting place is the idea of *distributed representation* (Rumelhart and McClelland 1986; van Gelder 1992), according to which the basic elements of representation are inductively derived and semantically open-ended, as opposed to the fixed categories of formal logic. This idea, though suggestive, immediately casts off most of the traditional means by which the operation of cognitive models can be narrated.

To fill this vacuum, researchers immediately set to work fashioning connectionist equivalents to conventional programming constructs – especially variable binding (Norman 1986). Touretzky and Hinton (1988), for example, built a connectionist production system, and Smolensky (1987) built a variable-binding mechanism in which each variable–constant pair was effectively represented by its own node. Though highly cumbersome, these systems provided an existence proof that inspired further refinements. Ajjanagadde and Shastri (1989) demonstrated that one could dynamically bind a handful of variables using strobe signals that synchronized the firing of variables and their values across long distances. Lange and Dyer (1989) and Sun (1992) described schemes in which relatively complex codes, representing either variable–constant pairs or the constants themselves, were passed along the wires. This research explores a space of trade-offs whose organizing principle is the tension between the frequent changes in a variable's value, which invite a high degree of abstraction, and the static connectivity of connectionist networks, which invites a low degree of abstraction.[6] This research, in short, is effectively reinventing the register-transfer model of computation – not because the register-transfer model is preferable on technical or empirical grounds, but simply because it is the only model for which a large repertoire of technical schemata has been developed. The demands of narration and the demands of architecture, in other words, are pulling connectionist research in opposite directions.

The purpose of this exercise is not to disparage connectionism but simply to describe the tides of practical logic that threaten to sweep it into the familiar channels of symbolic abstraction. In preparing the way for an interactionist alternative, Chapman (1991: 36–41) has suggested a way of distinguishing the neurophysiological facts of *essential connectionism:*

- is made up of a great many components (about 10^{11} neurons)
- each of which is connected to many other components (about 10^4)
- and each of which performs some relatively simple computation (whose nature is unclear)

- slowly (less than a kHz)
- and based mainly on the information it receives from its local connections (36)

from the empirically unsupported assumptions of many connectionist models:

- neurons are connected randomly or uniformly
- all neurons perform the same computation
- each connection has associated with it a numerical *weight*
- each neuron's output is a single numerical *activity*
- activity is computed as a monotonic function of the sum of the products of the activities of the input neurons with their corresponding connection weights. (40, emphasis in the original)

The essential connectionist commitments are firmly grounded in implementation. As Chapman observes, however, they preclude the use of pointers, and so they rule out virtually the whole tradition of symbolic programming. The models in later chapters will explore the practical logic of AI model-building within the essential connectionist commitments. But first let us consider the historical development within AI of the divide between implementation and abstraction.

Architectural and generative reasoning

In the founding documents of the cognitivist movement – the works from the early 1950s to the early 1960s by Chomsky, Lashley, McCarthy, Miller, Minsky, Newell, Simon, and others – the overwhelming intellectual concern is the refutation of behaviorism (e.g., Chomsky 1959; Lashley 1951; Newell and Simon 1972). The sharply focused nature of the struggle against behaviorism lends these documents a certain unity: they all argue for mental content as an explanatory category in human psychology. But as Chapter 3 has pointed out, the cognitivist movement shared another kind of unity as well: the background of assumptions against which the behaviorist–cognitivist debate was constituted. Among these background assumptions was the mentalist metaphor system of inside and outside: both sides spoke of stimuli and responses, or at least of input and output, debating only whether anything "mental" could be found in between. Another shared assumption was a certain conception of science: since the behaviorists had originally criticized introspectionism for its vagueness, the cognitivists would demonstrate that their theories were precise. In these ways, the cognitivist movement could win the day by presenting itself as a legitimate alternative to the stagnation of behaviorism.

Against the background of this commonality, the cognitivist movement exhibited considerable internal diversity as well. The leading figures just mentioned took their inspiration in various proportions from neurophysiology, feedback control, computer programming, and formal language theory. These various intellectual strands take up different relationships to the themes of implementation and abstraction: neurophysiology is concerned by definition with implementation and had no systematic framework for theories of abstraction; feedback control devices implemented a small set of mathematical abstractions in a wide variety of ways; research on computer programming developed abstractions that were ever more distant from issues of physical implementation on computers; and formal language theory was concerned almost entirely with abstraction, investigating whether procedures related to the formal properties of languages could be implemented without much concern for how. All of them had already been combined in other ways to form cybernetics, recursion theory, and a dozen other less readily distinguishable movements. Having come together under the common flag of cognitivism, they were now vigorously sorted out as each principal figure began training students within his own synthesis. It will be instructive to look at some of these syntheses in relation to the tension between implementation and abstraction. Though all of them have great historical significance, I will concentrate on the three that are most immediately relevant to my project: Newell and Simon (who wrote together during the 1960s), Chomsky, and Minsky. Though all of these authors have made steady progress on their respective projects over the last thirty years, their basic premises have changed little.

Chapters 1 and 3 have already mentioned Newell and Simon's theory of thought as search in a mathematically defined "problem space." To test their Logic Theorist and GPS models (1963, 1972) against the behavior of experimental subjects, they and their collaborator Shaw wrote some of the most sophisticated computer programs of that day. Newell, Shaw, and Simon, though, did not regard these programs themselves as models of human cognition, since human brains probably do not implement serial architectures like the ones on which they worked. Their theoretical claims were not for their programs as such, but simply for the abstractly defined problem-space scheme. Their programs are existence proofs for the *possibility* of implementing their abstractions, as well as ways of mechanically generating predictions from their theory, and they have only gradually been working toward proposals about physical implementation (Newell 1990).

Chomsky, similarly, defined his project in linguistics in terms of a distinction between "competence" and "performance." Linguistic competence is a mathematical abstraction expressing in ideal terms the grammar of a given language. Linguistic performance, by contrast, concerns the actual situated employment of language by real people. The relationship between competence and performance is different from the software–hardware distinction in computer programming, since it is not necessary for linguistic performance to implement linguistic competence in precise detail. (Actual speakers can experience slips of the tongue, misunderstand complex sentences, suffer brain injuries, make jokes by deliberately mangling grammar, etc.)

Newell and Simon's problem-space theory of thinking and Chomsky's formal theory of grammatical competence are both *generative* theories, meaning that they involve mathematical operations capable of generating an infinite number of mental structures by the repeated application of a small number of basic rules. Chapter 3 has introduced Newell and Simon's theory, in which one engages in a search of a formally defined space of states, each of which provides a choice among several operators. In Chomsky's theory, each sentence has a *derivation* consisting of several steps each of which permits one to apply several different rules. It is a characteristic of generative theories that they envision an exponentially expanding number of possible courses of reasoning. Given four operators and a single state, Newell and Simon will predict 4 possible outcomes from the application of a single operator, 16 possible outcomes from the application of two operators in sequence, 64 from the application of three, 256 from four, and so forth (although some of these multitudinous states might duplicate others). Likewise, the iterative application of the rules of linguistic competence can rapidly produce an enormous variety of sentences. Newell and Simon regarded the generative application of operators as a psychologically real phenomenon, whereas Chomsky was agnostic about whether people perform grammatical derivations in their heads or whether they generate and parse sentences by some other method whose results are simply consistent with the grammar. For Newell and Simon the empirical question was whether their model applied the same operators as experimental subjects. For Chomsky the empirical question was whether the iterative application of a grammar's rules produced exactly the set of grammatical sentences, no more and no less. Newell and Simon's project was subject to experimental falsification in a way different from Chomsky's. But in each case, the possibility of generating an infinity of possible mental structures was an emphatic point of contrast with the relatively impoverished scope of human possibility that seemed implicit in behaviorist theories. Chomsky in particular described grammars as making "infinite use of finite means" and invoked the thought of Descartes as precedent for his rigorous distinction between competence, *qua* abstract forms of thought, and performance, *qua* implemented action (Chomsky 1966).

In contrast to the generative theorists, Minsky (1985) has consistently been concerned with issues of implementation. His work represents the deepest and most sustained inquiry into the practicalities and consequences of the physical implementation of the various aspects of human intelligence. Where the generativists portray human reasoning as generating and exploring infinite abstract spaces, Minsky's theories have been resolutely finite: he has constantly reformulated his understandings of human intelligence in accord with the constraints of physical implementation. Though intended as a foundation for neurophysiology, his models do not invoke in any detail the claimed behavior of human neurons. Instead, he starts from the fundamental nature of physical implementation as such: the locality of causal interactions, the nearly fixed structure of physical artifacts, the logistical difficulties of centralized control, and the inevitability of conflict and inconsistency in a large system of any sort.

Minsky's theories employ an *architectural* style of reasoning. Architectural reasoning attempts not merely to implement a given abstraction but to discover abstractions that both do the required work and admit of natural implementations. Engineers who design computer hardware engage in architectural reasoning all the time, since it is their job to mediate between the abstractions that programmers presuppose and the currently available circuit fabrication and packaging technologies. But, as I have explained, today's computer designers are highly constrained by the expensive freedoms of serial architectures. The evolutionary processes that produced the human brain were probably not under any comparable constraints. Minsky views the human brain as a decentralized and massively parallel computational system, with an enormous number of specialized functionalities each occupying a localized region of neural circuitry.

The generative and architectural styles of research have led to different views about the relationship between abstraction and implementation in human cognition.[7] Generative theories posit large, consistent abstractions that operate by their own formally specifiable laws. McCarthy and the other proponents of the reconstruction of human knowledge in formal logic, for example, take as their starting point the generative power of logical formalisms and the formal consistency presupposed by their semantics (McCarthy 1958; Genesereth and Nilsson 1987). Some generativists have begun constructing large abstract theories that they refer to as *architectures.* Foremost among these theorists is Newell (1990), who presents his SOAR architecture as a "unified theory of cognition."[8] Yet such theories, valuable as they are, are concerned primarily with abstraction. They appeal to the practicalities of physical implementation only in a general way, for example in the parallelism that ought to be obtainable in the mechanism that selects rules to run.[9]

Architectural theories, by contrast, hold that the demands of physical implementation have profound consequences for the formulation of abstract theories of cognition. Minsky in particular emphasizes physical locality so strongly that cognition becomes a large, fragmentary collection of mutually inconsistent abstractions, each bearing a different relationship to its physical implementation. He formulates his theory not as a single unified mechanism, or even as a single list of axioms of design, but as a constellation of mini-theories, each examining some feature of human intelligence – whether language, reasoning, decision-making, memory, emotion, imagination, or learning – on the level of engineering intuition. The emphasis on physical implementation and its consequences for theories of cognition is also present in the connectionist movement. Feldman (1985), for example, has offered architectures for the human visual system that are compatible with both the phenomena of human vision and the practicalities of architectural design. Likewise, research in cognitive neuroscience has made a principle of shaping its abstractions in accord with the empirically discovered structures of the brain (Churchland and Sejnowski 1992). But whereas these projects hypothesize mechanisms to explain detailed empirical data, Minsky uses general principles of physical implementation to drive the formulation of *explanatory* theories of cognitive architecture.

The dialectical relationship between abstraction and implementation is evident in the patterns of dispute between generative and architectural theorists. Whereas the generative theorists promote the virtues of abstract rigor and generality, the architectural theorists emphasize physical realizability. The literature on these disputes has focused principally on the difficulty of reconciling the generative view of cognition with connectionist implementation (Clark 1992; Fodor and Pylyshyn 1988; Pinker and Prince 1988; van Gelder 1992). The problem, already apparent in the case of variables, is a conflict of metaphors: whereas a connectionist network has a fixed pattern of connectivity, general-purpose symbolic processing requires the continual reconfiguration of the symbolic structures that implement the agent's mental states. It seems safe to predict that the dispute will continue, and that the respective demands of abstraction and implementation will continue to influence the models of both the generative and architectural theorists. I believe that this impasse cannot be resolved without considerable rethinking of the generative view of cognition. My own project, however, is not to participate in the debate but to explore alternatives to the hidden assumptions that have determined its structure.

Generative reasoning and mentalism

The versions of computational research that I have surveyed differ in their details, but all of them define their problems within the framework of abstraction and implementation. The dialectical relationship between abstraction and implementation defines a space within which various projects trace particular trajectories. Though this history is complicated, a fundamental determinant of its overall pattern is the metaphor system of mentalism within which the distinction between abstraction and implementation makes sense.

Cognitivism and behaviorism, as we have seen, shared the mentalist vocabulary of inside and outside, stimulus and response, contents and behavior. At issue was the question of whether scientific sense could be made of the notion of abstract mental structures and processes: thoughts and thinking, memories and remembering, plans and planning, and a boundless repertoire of other nouns and verbs that shuttle easily between the vernacular and scientific vocabularies of psychology. The conception of computation as implemented mathematics provided the license that cognitivism needed. Make it mathematical, or make it seem likely to admit finite formalization, and it becomes a valid psychological category. As the project of AI accelerated, the mind became a space for the free exercise of the theoretical imagination.

The multiplication of mental constructs reached its point of greatest fertility in the AI research of the 1970s: mental structures that remain largely fixed or that continually change; mental languages based on English, formal logic, programming languages, or some composite of these; reasoning that proceeds through a detailed simulation of the outside world or through the calculation of obscure numerical functions; processing organized in a centralized or a decentralized manner; processes that cooperate or compete; and so forth. The metaphor system of mentalism has lent itself to this explosive unity in diversity for several reasons. The first is that everyone in Western culture, computationalists included, inherits a substantial degree of mentalist self-understanding from the philosophy that lies sedimented in the vernacular language. The second is that the computer itself arose within the discursive environment of a cultural enthusiasm for the symbols of cognition, mentioned in Chapter 1, that has waxed and waned from the manifestos of George Boole to the popular images of Einstein and rocket scientists and "giant brains" (Berkeley 1961) of the Cold War (Edwards 1996). The third and most important reason concerns the specific metaphors of mentalism. The Cartesian mind is both inaccessible and transcendent: on the one hand, it is the private domain of a unique individual who has incorrigible access to its states; on the other hand, it is also a perfectly generic cognitive essence found within every human being. This ambiguity is found as well in the phrase "the mind": in announcing its intention to study the mind, with its singular form and definite article, psychology does not assert that only one person in the world has a mind, but rather that all minds are, in the relevant sense, identical. Laboratory scientists studied the mind with great difficulty, but the practitioners of AI generated numerous personal intuitions about their own minds and made them real, on a small scale, in computational demonstrations (Turkle 1984: 265–267).

Abstraction, then, took on a special meaning within cognitivist psychology. The conceptual independence of mental abstractions from their physical implementations was the modern computational version of the soul's distinction from the body. Computationalists emphasize that an abstractly specified computation can be implemented on any device capable of supporting the digital abstraction and the register-transfer model, be it a supercomputer or a gadget made of thread spools and rubber bands; and they identify a computation with its abstraction, not its implementation. In this way, AI has reproduced in technological form the theological and philosophical individualism that became radicalized in Descartes's method of systematic doubt. The object of inquiry was not the individual in society or the person in the world, but the self-sufficient inner realm of the mind. The conceptual autonomy and infinite generative power of mental computations has played the same role in the computational theory of mind that the transcendence of the soul played for so long in philosophy.

One might hope for an alternative conception of computation, one that is better suited to an understanding of human beings as creatures bound up in their environments. An alternative to the mentalist opposition of abstraction and implementation can perhaps be found in the interactionist concepts of intentionality and embodiment. Later chapters will discuss these terms in detail, but I will offer minimal definitions here to indicate what is at stake. Intentionality relates to the general phenomenon of an agent's taking up a relation to something in the world: picking it up, drawing it, telling a story about it, avoiding it, or trying to figure out where it is. A theory of intentionality must explain, for example, how to distinguish intentional acts from ordinary causal events such as rocks rolling down hills. Intentionality includes representation as a special case, but intentional relationships are not necessarily mediated by representations. From a computational point of view, embodiment has four crucial (and conceptually interdependent) aspects: physical realization, causal interaction with the world, the possession of sensory and motor apparatus, and the necessity of making choices about action (one cannot, for example, travel both north and south, or both raise and lower one's hand). More obvious but less fundamental issues include the contingent attributes of various specific kinds of bodies: motility, facing in a particular direction, anatomical organization, and so forth. Though abstraction and intentionality play analogous theoretical roles, they differ in that abstract computations are specifically removable from any concrete circumstances whereas intentionality is defined in relation to an agent's activities in the world. Likewise, implementation and embodiment both refer to the physical realizations of a computation, but they correspond to different understandings of *what* is being realized: an abstract process or a concrete form of activity.

The principal significance of this proposed shift from a mentalist to an interactionist conception of computation is that intentionality and embodiment do not imply the Cartesian mind–body opposition that lies latent in the relationship between abstraction and implementation. This is not to say that hard problems become easy, only that the project of resolving them ought to suffer from different contradictions. The next chapter plows some necessary ground: it reviews the ideas and techniques that underlie contemporary computational practices, with a view toward their systematic renovation within an interactionist framework.

5 The digital abstraction

Digital logic

My goal in this chapter, as in much of this book, depends on who you are. If you have little technical background, my purpose is to help prepare you for the next few chapters by familiarizing you with the building blocks from which computers are made, together with the whole style of reasoning that goes with them. If you are comfortable with this technology and style of thinking, my goal is to help *de*familiarize these things, as part of the general project of rethinking computer science in general and AI in particular (cf. Bolter 1984: 66–79).

Modern computers are made of digital logic circuits (Clements 1991). The technical term "logic" can refer either to the abstract set of logical formulas that specify a computer's function or to the physical circuitry that implements those formulas. In each case, logic is a matter of binary arithmetic. The numerical values of binary arithmetic, conventionally written with the numerals 1 and 0, are frequently glossed using the semantic notions of "true" and "false." In practice, this terminology has a shifting set of entailments. Sometimes "true" and "false" refer to nothing more than the arithmetic of 1 and 0. Sometimes they are part of the designer's metaphorical use of intentional vocabulary in describing the workings of computers. And sometimes they are part of a substantive psychological theory whose origin is Boole's nineteenth-century account of human reasoning as the calculation of the truth values of logical propositions (Boole 1854). It will be helpful to distinguish two forms of this substantive theory: a weak form, on which reasoning proceeds by assigning truth values to various propositions; and a strong form, on which these assignments are entirely governed by a definite list of rules of inference, such as those of the mathematical system known as two-valued propositional logic. As often with computational vocabulary, it can be

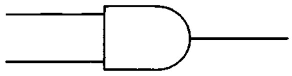


Figure 5.1. An AND gate.

the internal circuitry of memory chips does not employ the digital abstraction.) As the device physicists look for smaller and faster ways of implementing the digital abstraction, the detailed physics and chemistry define the field on which the dialectic of abstraction and implementation is played out.

Given a physical technology that supports the digital abstraction in this way, it becomes possible to build enormous abstractly specified structures out of a small number of elementary components. Digital logic – the actual physical circuitry – is made mostly of three sorts of things: wires, gates, and latches. Or so it is said, though even this statement is ambiguous because, as with "logic," it is common practice to use these terms to refer both to the various units of abstract functionality and the various discrete physical components that implement them. As such, wires, gates, and latches occupy a curiously ambiguous place at the boundary between mathematical abstraction and physical implementation. The purpose of a wire is to establish a logical equivalence among a series of physically distant points in the circuit; the purpose of a gate is to perform one of the elementary operations of binary arithmetic that is necessary to build up larger circuits. (I will return to latches later.)

The dual nature of a gate, as abstraction and as implementation, is critical to all of the architectural reasoning that goes into computational design. Consider a common type of gate, the AND gate, which is typically drawn with the symbol depicted in Figure 5.1. On an abstract level, an AND gate computes a certain mathematical function of two binary numbers. This function has a value of 1 if both of these two numbers have the value of 1; otherwise the function has the value of 0. On the level of implementation, one describes the function of AND in a way that sounds similar but has many different properties. An AND gate, on this account, takes two "inputs" and produces an "output" whose value is 1 if the two inputs are 1, and 0 otherwise. The difference between the two ways of describing the function of an AND gate is that, whereas the abstract description makes no mention of temporality, the implementation level refers to the actual physical process of computing the output given the inputs. This process takes time, albeit on the order of nanoseconds, and consists of the gate working to drive its output wire to the correct value.

The mechanism whereby a gate drives its output to the correct value operates continuously. Suppose that, at a given moment, both of the inputs to a given AND gate are 1s and the gate is happily driving its output to a 1 as well. And suppose that, in the next moment, one of those inputs, for whatever reason, falls from 1 to 0. Then, just for a brief moment, this little one-gate circuit will be in an inconsistent state, driving its output to the wrong value, until the gate manages to drive its output down to 0. Once the gate has "settled down," logical consistency will have returned and the physical circuit will properly implement its governing abstraction. A gate, then, has a directionality both in space (listening to its inputs, driving its outputs) and in time (always moving toward a logically consistent relation between these inputs and outputs).

The meaning of circuitry

A single gate is not very useful on its own. But as we begin to assemble gates and wires into a compound circuit, the relationship between talk of abstraction and talk of implementation becomes more complex. Consider the circuit diagrammed in Figure 5.2, in which the output from one AND gate serves as one of the inputs to another. It is useful and common to regard this whole assemblage of gates and wires as a functional unit, whose effect is to produce a 1 on its output if its three inputs are all 1, and otherwise to produce a 0. As with the single-gate circuit diagrammed in Figure 5.1, this two-gate circuit is a physical structure whose operation has a directionality and takes time. Suppose, by analogy, that at some moment all three of the inputs have values of 1. If one of the inputs to the first AND gate should suddenly fall to 0, a complicated sequence of events will take place. The first AND gate will, after a moment, begin driving its output to 0. The wire in the middle, joining the two gates together, will do its job of establishing a logical equivalence between its two endpoints, but a moment will pass before the new value of 0 can travel the wire's full length. If we think of the wire as a pool of electrons, the AND gate drives the wire to 0 by draining the pool as quickly as it can.[1] How long this takes will depend on the size of the pool and the diameter of the drain. We cannot know these things about our hypothetical circuit, however, since the diagram is suppressing some

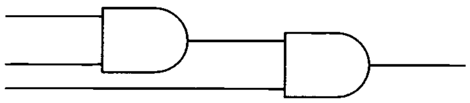


Figure 5.2. Two AND gates combined into a circuit.

important information: the distance between these two gates might be 2 microns or 2 miles. At last, however, the second AND gate's input will fall to 0, whereupon it will begin to drive its output to 0 as well, thereby restoring the logical consistency of the whole circuit. It is said that the changed input has "propagated" through the circuit, leading the circuit to settle back down to a consistent state.

In addition to AND gates, the average circuit also contains NOT gates, usually known as "inverters." An inverter takes a single input and produces a single output. Its purpose is to drive its output to the opposite value from its input. Thus, if the input to an inverter is 0 the inverter will drive its output to 1; and if the input is 1 the output will be driven to 0. An inverter often appears in a circuit directly before the input of another gate. In this case, it is customary to speak of the gate as having an "inverted input." Figure 5.3 diagrams an AND gate, one of whose inputs is inverted. This combined gate, an AND gate plus an inverter, might be referred to as an AND-NOT gate, since its output will be 1 provided that its first input is 1 and its second input is not.

Putting these ideas together, consider the circuit that is diagrammed in Figure 5.4. It has three inputs and two outputs and contains two gates, an AND gate and an AND-NOT gate. The wire leading out from the AND gate branches, taking the same value to two destinations: the first output of the whole circuit and the first input of the AND-NOT gate. This diagram, like the others, does not give us much information about how this circuit is implemented, but it does suffice to determine the circuit's abstract functionality and the inventory and connectivity of components that implement it. Unlike the other diagrams, however, this one provides some extra annotations, in the form of English words associated with the various wires: *enabled*, *tripped*, *overridden*, *light*, and *buzzer*. These words do not play any role in determining the mathematical function that the circuit computes. Instead, they represent "meanings" ascribed to the various wires by the circuit's designer. Let us say that the circuit is part of a burglar alarm. The *enabled* input is attached to a switch in the basement

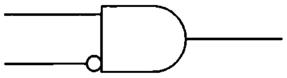


Figure 5.3. An AND-NOT gate.

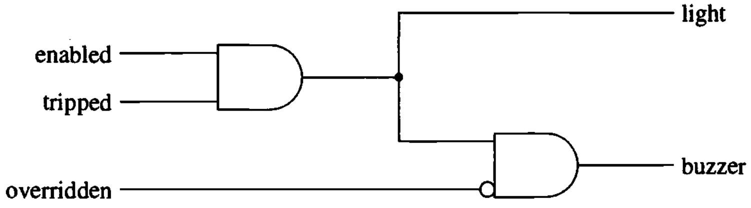


Figure 5.4. A circuit for a burglar alarm.

indicating that the alarm system should operate; the *tripped* input is attached to whatever sensors might be installed on the doors and windows; the *light* output leads to a light on the alarm unit that turns on whenever the system is enabled and the sensors are registering motion; the *overridden* input is attached to a switch on the alarm unit that prevents the alarm from sounding while the system is being tested; and the *buzzer* output is attached to a buzzer that makes a loud noise whenever *enabled* and *tripped* are 1 and *overridden* is 0. Many real digital devices are no more complex than this one.

Now, the description I just provided of this circuit's operation is different in kind from the more mathematical descriptions I provided earlier. The circuit has been endowed with a "semantics." The people who design circuits like this one routinely apply meaningful labels to the various components of a circuit. Specifications of the circuit's function that employ these labels are abstractions in their own right, but they stand in a complex relationship to the binary-arithmetic abstractions that specify the circuit's binary input–output behavior. On one hand, one has not finished specifying this circuit's functionality until some mention has been made of the switches, sensors, and indicators and the useful purpose they serve. On the other hand, it is necessary to treat all the words in special ways: if we label a certain wire *overridden* and that wire carries a value of 1, it does not necessarily follow that it was anybody's intention to override the alarm system. It is, after all, possible that the person who threw the switch was confused or was trying to determine what the switch did, or that the circuit malfunctioned or was sabotaged. Likewise, a value of 1 on a wire called *tripped* is certainly not a guarantee that the sensors have actually been tripped, much less that a burglar is present. The wires, after all, might have been accidentally connected to the main electrical system instead of to the sensors. Or perhaps they were installed poorly and do not make a proper connection. It is thus necessary to adopt a narrow construal of each of these terms.

Yet this is not satisfactory either, since one presumably purchases a burglar alarm not to assign values to wires but to detect burglars. The burglar alarm suffers from a gap between the promise implicit in the phrase "burglar alarm" and the conditions under which, according of the physics of the thing, the buzzer actually goes off. This gap is particularly obvious in the case of burglar alarms, which are notorious for their false alarms. But all artifacts have such gaps, as a result of the margins of their designers' abstract models. As Smith (1993 [1985]) points out, all abstract models exhibit margins. Sometimes, as with burglar alarms and ballistic missile detection systems, the margin is exacerbated by the adversarial nature of the activity, in which someone is actively trying to defeat the system by occupying those margins. But in any system, the margin reflects the whole network of explicit and implicit assumptions that went into the design process, as in Smith's example of the sensor-based traffic signals which can detect cars but not bicycles. An indicator lamp marked "fasten seat belts" or "missile alert" or "THC positive" is only as good as the margins of the computations that back them up.

Are the labels on the wires, then, part of the circuit? Computer people often refer to such labels as "mnemonics" because they help the designer to remember what the labeled wire is supposed to mean. The idea is that they are *only* mnemonics and do not constitute the narrowly mathematical function of the circuit. But if we define the circuit in terms of its narrative affordances – the ways in which the circuit fits into the stories that people tell about houses and robbers and flashing lights and false alarms – then they clearly *do* constitute, in some portion, the circuit's functioning. Derrida (1967: 141–152) would refer to the labels as "supplements": they are supposed to be inessential to the technical artifact and yet they are necessary to insert the artifact into the order of human meanings. The labels are signifiers, and their materiality comes to the fore on those occasions when the margins of the burglar alarm's design become problematic. Yet the designers of AI systems place enormous weight on these signifiers, since they provide the raw material for any narration of the system's operation.

Do these practices for designing and using logic circuits, then, presuppose a philosophy of mentalism? Modern formal logic emerged in the sixteenth century in an intellectual shift that Ong (1983 [1958]), in his study of Ramus, called "the decay of dialogue": the movement from reason as a form of interaction within a human relationship to reason as a formal system. Platonic and psychologistic conceptions of logic have coexisted ever since. In casting formal logic as a mathematical system, Boole (1854) believed that he was doing psychology; in elaborating Boolean logic as a theory of neurophysiology in the early days of the cognitivist movement, W. McCulloch and Pitts (1965 [1943]; cf. Edwards 1996: 187–193) believed that they were placing psychiatry on a scientific basis. On the other hand, as Newell (1983: 195) observes, "[L]ogic did not enter AI at all as the logic of thought. . . . In fact, it was precisely the split of logic from thought that set logic on the path to becoming a science of meaningless tokens manipulated according to formal rules, which, in turn, permitted the full mechanization of logic."[2] The tokens of mechanized logic are not exactly meaningless, since they are useless without the meanings that they are routinely treated as carrying. But the formal structures of logic have taken on a life of their own; they promise, in their self-sufficient simplicity, a generative principle that can be used to reconstruct and systematize the whole of reason. Thus, Leith (1987) observes that the movement to provide logical foundations for computer programming (Kowalski 1974) bears a strong resemblance to the ambitions of Ramist logic. All the while, it has becomes steadily more difficult to conceive logic as an embodied social practice; the gap between logical reason and lived experience has grown far too great (Nye 1990). Coulter's (1991) empirical investigations of logic in conversation, for example, are almost completely unintelligible within the frameworks that four centuries of research on formal logic have produced. Yet the hegemony of formalism is so complete that observations such as Coulter's cannot provoke a crisis, or even very much of a dialogue.[3] The logic circuits themselves may not encode any philosophical commitments, but the narrative practices that render them meaningful certainly do.

The temporality of computation

In describing the mechanics of digital logic circuits, I have been using the vocabulary of abstractions and implementations that I have earlier tarred with the historical connection to mentalism. And, indeed, I have been describing my example circuits in a mentalistic sort of way, or at least with an intentionalistic vocabulary. Look again at Figure 5.4: where do those input wires come from and where do those output wires go? The functional specification of the circuit is defined at its boundaries, in terms of the intended meanings of the various input and output wires. These wires, of course, are attached to the "sensory" and "motor" apparatus of a burglar alarm: switches, sensors, and indicators. In such a trivial example, of course, the philosophical burden of mentalism and its metaphors weighs lightly; little theoretical apparatus is required to wire up a burglar alarm. But when one is considering much larger artifacts or contemplating the use of digital circuitry in the construction of models of human beings and their lives, the deeper meaning of taken-for-granted design practices becomes a more difficult matter. I will return to this in the next section.

For the moment, imagine a large circuit made of gates and wires. It might have several hundred gates and a few dozen inputs and outputs. If built in a compact way, it might occupy a cubic millimeter of space and possibly much less, though the contacts by which the circuit is attached through wires to other devices may be many times larger. This hypothetical device is a marvelous thing in many ways. While it is operating, it continuously enforces a certain logical relationship between its inputs and outputs. When one of its inputs changes value, changes propagate quickly throughout the circuit, until the outputs settle at their new values. How exactly this propagation unfolds will depend on the structure of the circuit. In particular, the amount of time it will take for the whole circuit to settle down to a new consistent set of values can be determined by finding the longest path through the circuit from a newly changed input to an output. If the circuit is relatively "shallow," then the propagation will occur quickly. But if the circuit is relatively "deep," even if it has only one long chain of gates between its inputs and outputs, then the propagation will occur relatively slowly, though still in a minute quantity of time. All of the hustle and bustle in the circuit is highly parallel, in the sense that every gate will react instantly to any change in its inputs. The whole circuit is alive, constantly monitoring its inputs and adjusting its outputs as changes propagate through it. This situation is the very ideal of parallelism that I described in Chapter 4: a collection of simple, uniform computing elements distributed in three-dimensional space, each of them doing useful work all the time. A computation that can be implemented this way is, other things being equal, highly preferable to one that cannot. Circuitry built within this framework is called *combinational logic.*

In the actual design of computers, things are not this simple, because of a phenomenon known as a *race condition.* Recall that a change to the input of a combinational logic circuit will cause the circuit to enter a logically inconsistent state until it finally manages to drive all of its outputs to their correct values. A small window of inconsistency opens up between the moment an input changes and the moment the last output assumes its correct value. While this window is open, it is important that nobody trust the output values, which are unreliable as long as the circuit is still settling down. Consider, for example, a circuit that has two paths from its inputs to its outputs, where one of the outputs controls some device that takes an irreversible action, such as a spot welder. One path is a simple circuit that decides that it is time to take some action. The second path, however, is a long and complicated chain of reasoning that detects certain subtle exceptional conditions that make this action inadvisable. An AND-NOT gate provides for the possibility that these exceptional conditions can override the decision to take the action. But, as Figure 5.5 demonstrates, these two lines of reasoning are engaged in a sort of race. If an exceptional condition actually arises, the arrival of the propagation on the shorter path will cause the output to rise briefly to 1 before the other line of reasoning arrives and forces the output back down to 0. As a result, it is important for a digital circuit to have settled down and attained a consistent state before any irreversible actions are taken on the basis of its outputs.

A race condition is the electrical manifestation of the difference between the timeless abstract specification of a combinational logic circuit and the causal workings of its implementation. To restore the accurate implementation of the abstraction, it becomes necessary to impose the same kind of artificially discrete structure on time that the digital abstraction imposes on numerical values. This is the purpose of latches, which I mentioned in the preceding section, and a regime of *clocking.* The purpose of a latch is to hold a digital value in a definite place until it is needed for a future calculation.[4] A latch sits on a wire, serving as a kind of valve. In the most usual case, the latch will divide the wire into two segments, one of which is being driven to some value while the other serves as the input to another circuit. When the latch is closed (i.e., when

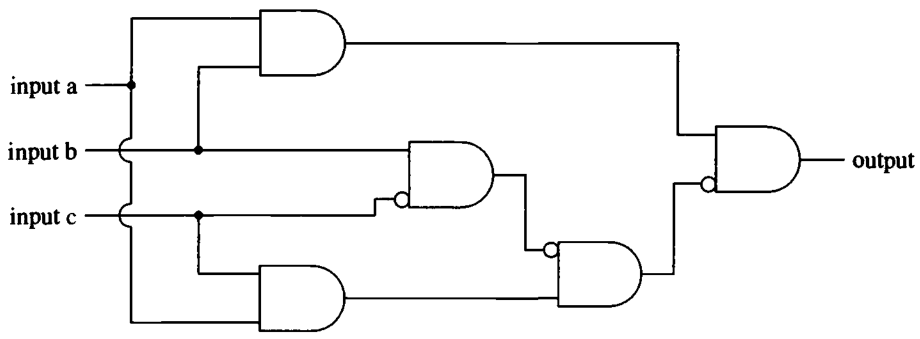


Figure 5.5. A circuit with a race condition.

it is not blocking the wire), the whole wire will be driven to the same value. When the latch is open, it effectively severs the wire in two. The half of the wire that is being driven will continue being driven as before. The other half of the wire, the one that is not driven to any particular value, will now become an isolated pool of electrical charge. This isolated segment of wire will retain its logical value, either 0 or 1, until the latch is closed again or until its charge dissipates. As long as this logical value remains in effect, the other circuits that look at it will continue to see its stored value rather than whatever values might be coming and going on the far, driven side of the latch. Although designers play it safe by assuming that the isolated charge dissipates quickly, in most cases it will retain its value for much longer, depending on the properties of the implementation technology.

A *clocking regime* employs latches within a rigorous discipline for synchronizing complicated computations by means of a *clock*. The purpose of a clock is to establish periodic moments at which all of the combinational logic circuitry in an entire machine is in a consistent state. Clocking schemes can become arbitrarily complicated, but the simplest one is called a "two-phase nonoverlapping clock"; it signals *tick, tock, tick, tock, tick, tock*, . . . for as long as the circuit is running, where *tick* means "Let us give the circuits some new values on their inputs" and *tock* means "Let us assume that the circuits have had sufficient time to settle down and that, consequently, their outputs are now consistent with their inputs." In the race-condition circuit diagrammed in Figure 5.5, one might avoid tragedies by installing latches on both the inputs and the outputs, with the input latches keyed to *tick* and the output latches keyed to *tock*. With this two-phase clocking regime in place, the circuit will appear from the outside to implement a perfect realization of its abstraction, since its outputs will be "visible" only at moments when they stand in the correct logical relationship to the inputs.

Clocking is an excellent example of the trade-offs required to implement the digital abstraction. In order to make sure that no mistaken values slip out of a circuit, the designer must establish a pause between *tick* and *tock* that is long enough to allow all of the circuitry to settle down. Since an entire computer will generally run on the same clock, the clock speed is determined by the slowest circuit in the whole machine. Moreover, the designer must find some way to make sure that the *tick* signals (and, likewise, the *tock* signals) arrive at all parts of the machine at nearly the same time. This requires careful attention to the lengths of the wires that distribute the clock signals.

The combinational logic circuits I have described are intimately involved with their "environments," continually watching their inputs and updating their outputs in response to any changes. Latches and clocks, though, make it possible to endow a computer with its own abstractly specified internal space and time, separate from that of the outside world. With latches and clocks, one can assemble a circuit in such a way that some of its inputs derive from its own outputs. Without a proper clocking regime, such a circuit might thrash back and forth between 1s and 0s without even reaching a consistent state. But *with* a clock, one can ensure that the circuit will inspect its own outputs only when they are consistent with its inputs. Thus, the outputs take on a series of consistent sets of values, one for each *tock*. Such a device is called a *state machine* and a designated subset of each of these successive consistent sets of outputs is called a *state* (Figure 5.6).[5] A modern computer is made of a large number of small state machines. A bit of memory, for example, is a tiny state machine whose state is simply 1 or 0. As the clock runs, the computer moves from state to state, constantly feeding the outputs of each operation back to form the inputs for the next. Here, in the techniques for endowing computers with memory (also known more technically as "internal state"), lies the compelling analogy between the abstract internal space of the computer and the abstract internal space of the mind, and thus the metaphor of computation as thought.

The introduction of clocking endows a computer's operation with a

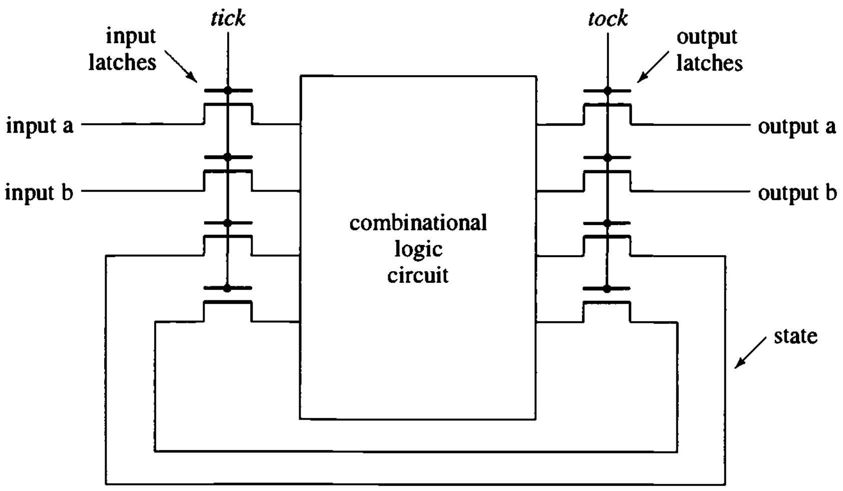


Figure 5.6. A state machine.

curiously unreal temporality. The *tick tock tick tock* of the clock defines an entirely abstract time course that has no necessary relationship to the passage of time outside, provided only that the time between *tick* and *tock* is long enough for the circuits to settle (anywhere above perhaps 20 nanoseconds) and that the time between *tock* and *tick* is short enough that the latched values do not dissipate (anywhere below perhaps 2 seconds). In a serial computer, the execution of a given instruction might require a dozen clock cycles (*tick–tock* pairs). In describing the workings of a computer program, the narrative ("First it does this, then it does this, then it does that") is set in abstract time, not the "real time" of the outside world. Thus, a computer might be executing an enormously complex program that employs a vast amount of data, yet it is possible to vary the speed of its clock across a wide range without affecting the logical correctness of the computation.

Despite this decoupling between abstract time and real time, real time still proceeds outside, possibly including a human being waiting for an answer or a mechanical device waiting for its parameters to be adjusted. Considerable ingenuity goes into maintaining the right kinds of synchronization between the abstract time of a computation and the real time of the outside world. For the most part, the outside world must simply wait.

When the outside world cannot wait, it is necessary to engage in *real-time programming* (Joseph 1996). This means nothing more fancy than discovering enough about the real-time properties of a computation (i.e., a particular abstract process implemented on a particular model of machine) to ensure that its answers will be available when they are needed. The problem is that the whole development of computers has assumed a strong temporal distinction between abstraction and implementation. For example, computers regularly interrupt the running of a given program in order to attend to other business. Time stops, in the sense that real time passes while abstract time stands still, and then the program is allowed to carry on as if nothing happened. If the program in question is controlling a robot whose motors need their torques adjusted every hundredth of a second, this kind of interruption must obviously be suppressed. Serious real-time programming often means circumventing or disabling most of the advanced features of a given computer, resisting the historical trend toward abstraction to restore a rough temporal correspondence between the abstract process and its implementation.

I have emphasized the difficulty of real-time programming because it is an especially clear symptom of the mentalist framework that governs the conceptual distinction between abstraction and implementation. Computation, on this account, occurs in its own space (i.e., the internal state of the computer) and its own time (i.e., the abstract temporality of clocked circuitry). The space and time of computation are entirely abstract: a set of metaphors made into mathematics that is capable of being implemented in an infinite variety of ways. Indeed, since abstract time is simply a mathematical construct, it is only a matter of convenience if this "time" stands in any consistent proportional relationship to "real time," that is, the time of implementation. It is entirely possible to embed the mathematics of a given abstractly specified computation within another, more complicated layer of mathematics, one that permits abstract time to break into pieces, run backward, repeat itself, or be shuffled into some alternative ordering (Abelson and Sussman 1984; Bolter 1984: 100–123). Computational abstraction thus gives engineers the freedom to create phenomena that have little necessary relationship to the space and time of ordinary corporeal things. It is in this (very specific) sense that the inside of a computer, like the inside of the mind, figures in the worldview of mentalism as a realm of absolute imaginative freedom.

Embodied computation

The internal space of computation, then, like the Cartesian soul, is a not simply partitioned off from the outside world but actually different *in kind.* As the mediating term between the mind and the world, the body has played a curious role in the history of mentalist philosophy. The three pivotal figures in this history, Augustine, Descartes, and Turing, lived in different times and wrote in different philosophical registers, but each of them developed in his own way the theme of opposition between the transcendence of the soul and the finitude of the body. Each man's cultural milieu provided fresh meaning for this opposition: Augustine struggled to maintain his ideals of Christian asceticism, Descartes described the soldier's soul overcoming his body's fear as the Thirty Years' War raged, and Turing idealized disembodied thought as he suffered homophobic oppression in modern England. As physics changed and explanatory metaphors multiplied, the soul's identity as a realm of pure thought evolved and grew sharper. Yet the lines of descent are broad and clearly drawn.[6] Contemporary computational practice, with its dialectical tension between the imperatives of abstraction and implementation, is the inheritor of this tradition, the modern incarnation of a philosophy so confident that it has forgotten its intellectual contingency and recast itself as pure technique. The first step in getting beyond mentalism is simply to restore, by means of critical and historical inquiry, a sense that it might be possible to hold other views.

Exhibiting the contingency of mentalism does not mean throwing everything away and starting over. Mentalism is a system of metaphors, not a deductive premise. Changing the metaphors will change everything, but in a coherent way. Mentalist rhetoric has absorbed a wide variety of intellectual and technical materials, assimilating them to mentalistic themes in ways that can be teased apart only with great effort. Must combinational logic be understood within the mentalistic framework of abstraction and implementation? The happy answer of no can be sustained only by cultivating an awareness of the choices that lie implicit within day-to-day technical practice.

Conventional technical practice is not a seamless whole to be accepted or rejected en masse. Yet its internal fault lines are far from obvious. They reside in aspects of technical practice that are not normally counted among its defining features. One of these is the figurative nature of its language, for example in the philosophically laden metaphors of internal space to which the rhetoric of state machines lends itself. In order to formulate an alternative, let us start by speaking about digital circuitry in a different way, rescuing it from the discourse of abstraction and implementation and reappropriating it within the alternative discourse of intentionality and embodiment.

Recall from Figure 5.6, then, that a state machine has two classes of inputs and outputs: some that travel to and from the circuit's environment (which may include various other circuits and ultimately the environment of the entire device) and others that pass immediately from the circuit's outputs back to its inputs. Both sets of wires close causal loops; the only difference in these loops is, so to speak, their diameters. Mentalist discourse draws a firm distinction among the various causal loops. In particular it places primary emphasis on the smallest ones, which it defines as the state of the circuit. The larger loops, the ones that pass through the outside of the circuit, are often not visible on the diagrams, either because they are obscured in a maze of circuitry or because the lines simply dangle off the edge of the page. (Figure 5.6 itself is a case in point.) Interactionist technical practice would treat all causal loops – whether in memory circuits, thermostat controllers, cats chasing rabbits, or drivers steering cars – in the same way, according no privilege to the special case in which the outputs of a device connect directly back to the inputs.

This redirection of rhetorical emphasis is, in its own small way, a different fork in the road. Simply talking about the same machinery using different language will not cause any revolutions, of course. The idea, however, is not simply to relabel things but to aim the research process in a different direction. If the traditional machinery of AI is inherently mentalist, someone trying to do interactionist AI with that machinery is likely to reach some instructive technical impasses. And it is only through impasses that a critical technical practice can learn anything really new. The next few chapters will try to put this alternative, critical design project into practice.

6 Dependency maintenance

Critical technical practice

Having prepared some background, let us now consider a technical exercise. Since readers from different disciplinary backgrounds will bring contrasting expectations to an account of technical work, I will begin by reviewing the critical spirit in which the technical exercises in this book are intended.

Reflexively, the point is not to start over from scratch, throwing out the whole history of technical work and replacing it with new mechanisms and methods. Such a clean break would be impossible. The inherited practices of computational work form a massive network in which each practice tends to reinforce the others. Moreover, a designer who wishes to break with these practices must first become conscious of them, and nobody can expect to become conscious of a whole network of inherited habits and customs without considerable effort and many false starts. A primary goal of critical technical work, then, is to cultivate awareness of the assumptions that lie implicit in inherited technical practices. To this end, it is best to start by applying the most fundamental and familiar technical methods to substantively new ends. Such an effort is bound to encounter a world of difficulties, and the most valuable intellectual work consists in critical reflection upon the reductio ad absurdum of conventional methods. Ideally this reflexive work will make previously unreflected aspects of the practices visible, thus raising the question of what alternatives might be available.

Substantively, the goal is to see what happens in the course of designing a device that interacts with its surroundings. Following the tenets of interactionist methodology, the focus is not on complex new machinery but on the dynamics of a relatively simple architecture's engagement with an environment. Those accustomed to reckoning research contributions in terms of new mechanisms may be disappointed. Instead, the principal substantive contribution is a way of thinking about the relationship between machinery and dynamics for certain purposes. To get the process started, I will describe the observations about the dynamics of ordinary activities that originally motivated this research. This description will not provide the kind of proof that laboratory psychology promises or the kind of descriptive robustness that ethnographic fieldwork seeks. Rather, it is intended as a stimulus to the esoteric practice of technical reflection.

Technically, I want to explore the automatic construction of complex combinational logic circuits for an agent interacting with a simple world. At the outset, the specifically technical questions concern the suitability and limits of logic circuits for this purpose, as well as the means by which these circuits might be efficiently and incrementally constructed by an automated device negotiating set tasks. By the end of the story, though, deeper technical issues about representation and learning will have arisen through reflection on difficulties encountered along the way.

In addition to the perhaps unfamiliar expectations I hope to have created here, the reader will observe that the second half of this chapter, as well as the remaining chapters of technical exposition in the book (Chapters 7, 10, and 13, and the second half of Chapter 9), are written in a voice distinct from that of the rest of the book. The voice in these chapters is approximately that of conventional AI, in which the primary goal is to describe the operation of machinery in intentional terms. In presenting and motivating the various mechanisms I have built, I will present a series of cartoon stories. Scraps of computer code, likewise, will employ cartoon examples of various degrees of silliness. This practice can seem bizarre to readers without substantial technical background, and especially to readers with strong views about suitable methods for describing human experience. Inasmuch as the various features of the machinery must be explained individually before being assembled into larger wholes, cartoon examples are an expedient means for providing a rough, tentative orientation to the technical function of a given mechanism and the narrative structures that go with it. At the same time, the whole point of a critical technical practice is to work from the inside, driving the customary premises and procedures of technical work toward their logical conclusions. Some of those conclusions will be problematic, and the whole complex process of reaching them will then provide the horizon against which critical reflection can identify fallacies and motivate alternatives.

About routines

The theoretical work reported in this book was originally motivated by an empirical interest in the organization of ordinary routine activities in everyday life; a brief account of this work may convey some of the intuitions behind the technical proposals to follow. As Heidegger argues in *Being and Time* (1961 [1927]), Western cultural and philosophical traditions have left us ill equipped to observe, describe, or pay attention to the most ordinary aspects of our lives. Activities such as making breakfast, driving to work, and writing a letter are largely terra incognita – except, perhaps, when one is first learning them, when something exceptional happens to render them problematic, or when they are turned into objects of rational manipulation. Dreyfus (1972) has pointed out that this is a particularly serious situation for AI, whose practitioners must constantly tell stories about ordinary human activities. The planning view of human action, outlined in Chapter 1, is one such story. If computational explanation has any value, then planning is either a good way to talk about both people and computers or else a bad way to talk about both. In either case the reasons ought to be instructive. Therefore I wanted to look closely at ordinary activities to determine if planning was a reasonable story.

It would take another book to fully explain and defend my methods in studying ordinary activities.[1] Briefly, I kept a notebook of detailed descriptions of my own activities. When I noticed that a particular episode of activity had features that were relevant to the questions that interested me, I would write out the episode from memory in as much detail as possible.[2] Later I would try to check my memory in various ways, sometimes through videotape or through the various evidences that the activity left behind, but most often by simply being aware of the same sort of activity again on future occasions. (Having written detailed descriptions in my notebook would cause me to spontaneously notice similiar phenomena afterward.) Of course, this cultivated awareness of an activity would often cause the activity itself to change, but I regarded this as simply one more dynamic phenomenon to investigate. This method of investigation, of course, is prone to selection bias, the rational reordering of memory (Bartlett 1932), and the difficulty of drawing definite conclusions from single cases (March, Sproull, and Tamuz 1991). But its purpose is not to compel assent but to help stimulate consciousness of the usually overlooked mundane features of daily life, with the aim of defamiliarizing the conventional assumptions that lie behind computational design reasoning.

I was particularly interested in *routines.* A routine is a frequently repeated pattern of interaction between an agent and its familiar environment. A routine is a dynamic. The difference between the nouns "routine" and "dynamic" is that a routine involves a particular individual (it is, for example, "my routine" or "your routine") whereas a given dynamic might occur in the lives of many individuals. For example, one might have a routine for picking up a fork, preparing a bowl of cereal, measuring out two cups of flour, putting on one's heavy backpack, selecting a toll booth at the San Francisco Bay Bridge, putting one's watch on, washing the dishes in one's kitchen after a large dinner, tossing a wad of paper into the trash can in the far corner of the office, or writing the word "the." One might speak of a routine "for" some task, but a routine is defined by what happens rather than by any endpoint or overall intention. One need not have a set routine "for" any given purpose, since the relevant circumstances of the task and its environment may differ sufficiently from occasion to occasion that no stable routine emerges. Likewise, one may have a dozen different routines "for" some purpose if that purpose arises in a dozen different contexts.

Routines have all of the properties of dynamics that Chapter 2 has discussed. A routine, like any dynamic phenomenon, is purely a descriptive construct, not a thing in the head, not a plan or procedure. It need not correspond to an ingrained habit or rigid custom. No specific knowledge or competence or cognitive machinery is required to engage in routines.[3] Doing something the same way every time, in other words, need not result from a specific intention to do it the same way every time. An agent can engage in routines without knowing it. A routine might involve a series of actions, each a response to the situation resulting from the previous action, without a specific prior intention to perform *that* series of actions. One version of this idea is the behaviorist notion of stimulus chaining, whereby a stimulus provokes a response that causes another stimulus and so forth. But the general point is that, except in the simplest or most heavily controlled environments, agents necessarily act without complete foreknowledge of the consequences of their actions. The world is a complex terrain, and each step may change the terrain or reveal further information about it. The agent's path will inevitably be a joint product of its own intentions and the topography of the terrain. When the same agent traces the same path repeatedly, it is a routine. A driver, for example, might weave down a potholed street in the same way every day, without any specific intention to do so, just because the same potholes are always there.

The actions comprising a routine are not dictated by the routine; they are simply the individual's chosen actions in particular situations. For example, an agent might simply improvise the same response to a recurring situation. Perhaps that response is the only sensible one. You might switch your umbrella from your right hand to your left hand so you can use your right hand to get your house keys out of your pocket on every rainy day without ever having made a deliberate policy of it. Furthermore, a routine is not a law of nature; you might have poured your morning coffee the same way a thousand mornings straight, but tomorrow morning your routine might be altered by any of an endless variety of contingencies small or large, such as a momentary shift in your posture to glance at something out the window, a knock at the door, a worn-out coffee pot, or the onset of an ulcer.

Different individuals might engage in different routines for the same task. Not everyone has a routine for every task. It happens that I make an omelette in my kitchen in pretty much the same way every time. This routine varies so little not because I am deliberately unvarying but because my kitchen has a fairly stable order and because I have had no occasion to change any of the opinions that motivate the choices that go into making an omelette. In fact, I have thought many of these opinions through by exploring alternatives, reading cookbooks, and comparing notes with other cooks. But thinking things through is not a necessary condition for an activity becoming routine. It would *really* need explaining if I made a series of omelettes and did it differently every time. I can think of five ways this might happen:

1. I am new to making omelettes and am still making mistakes.
2. I am deliberately exploring different ways of making omelettes.
3. I am thinking up gratuitous variations for the sake of variation.

4. I am subject to uncoordinated patterns of distraction from other sources. Perhaps I have other concurrent obligations (like minding a two-year-old) or am persecuted by interferences (like ringing telephones).
5. I am always making omelettes in different circumstances. Perhaps I am always moving to new quarters, or someone or something keeps rearranging my kitchen, or the omelette-making activity itself always leaves the kitchen arranged in some relevantly different way.

In short, the existence of routines requires no more explanation than physical determinism. Put the exact same individual in the exact same situation on several occasions and the exact same things will happen. An appeal to determinism, of course, is a rough sort of explanation. No routine for pouring coffee is going to come off the same way every morning down to the last muscle twitch and molecule. The larger point is that routines exist because people bring a relatively stable set of practices to a relatively stable set of circumstances, and because the routines themselves tend to reproduce the relevant aspects of those circumstances.[4]

Everyday life exhibits routines at all scales. A routine for driving to work will probably have a hundred smaller routines as parts – buckling up, signaling a left turn, looking for speed traps, keeping distance behind the cars in front – many of which arise within other routines as well. Even the most innovative improvised activity will be composed of already-routine parts. Aside from being practically inevitable, this property of routines is fortunate as a computational matter, since nobody could improvise adaptive responses to unprecedented situations on all levels of organization at once.

In practice, routines vary from occasion to occasion in numerous small ways. Traffic patterns will vary slightly, utensils will find different resting places on countertops between steps of a recipe, source materials will vary in small ways, and so on. More important, routines evolve (Agre 1985a; Agre 1985b; Agre and Shrager 1990). In observing the complexity of any given episode of real activity, no matter how small, it helps to think of an agent's actions as the result of a long process of routine evolution. A new routine might arise in the course of ordinary activity, but then it generally evolves to more complex forms. Just as one can engage in routines without knowing it, one's routines can – and regularly do – evolve without one's knowing it. As long as the relationship between individual and environment is relatively stable, most of this undeliberate evolution takes the form of a series of discrete mutations to the routine. Routines can change because the individual changes (perhaps by learning something) or because the environment changes (perhaps by getting rearranged in a more convenient fashion). In either case, the fundamental reason for routine evolution is that the *relationship* between the individual and the environment changes, and the study of routine evolution is precisely the study of these changing relationships.[5] This theory should be distinguished from theories, such as Van Lehn's repair theory (1990), that explain changes in action through changes in procedures that generate that action. A procedure, like a plan, is a mental entity that directly and systematically determines an individual's actions; a routine, by contrast, is an emergent pattern of interaction whose evolution might be influenced either by changes in an individual or by changes in the individual's environment. A change in a routine *might* be caused by a change in a plan or procedure, but it need not be.

An evolving routine, as a general matter, will tend to take account of more and more detailed aspects of the environment. Actions and processes that had occurred serially may begin occurring in parallel (e.g., you might get out the tea bag while the water is heating rather than before putting the kettle on the stove). Warning signs become noticed as if expected (e.g., when the bread starts to snag in the toaster as you push it down), and precautions are taken without missing a beat. Sometimes a routine will evolve in several distinct directions in response to variations of circumstance, whether new or previously ignored (e.g., when you back the car out of the driveway differently at home, where children or their toys might be around). Sometimes a routine's evolution will stall in some comfortable pattern, only to resume at some provocation or chance discovery (e.g., when driving on a road for an errand reveals a new route to work). Actions that began as improvisations or afterthoughts become regular occurrences, and the boundaries among artificial phases of activity (like preparation, execution, and cleaning up) fade as actions rearrange and recombine. Workplaces and dwellings begin to bear the marks of one's routines, through both accumulated side effects (trails [Batali 1993], wear patterns [W. Hill and Hollan 1994], built-up piles of stuff, tools always left in the same places) and deliberate conventions (getting a clothes hamper to reduce clutter, dedicating a certain space to meditation); and the marks left by each routine prod mutations of the others.

Watching routines evolve, I have been impressed by the steady background of small changes, each of which makes sense singly but whose accumulated effects can be formidably complicated. A particularly striking pattern of mutations to routines is *backward transfer.*[6] In a cyclical activity, or an activity that takes place in the same circumstances repeatedly, elements of the routine will often move countercyclically – that is, toward the beginning of the routine. This will often cause the cycle to change in subtle or complex ways. The effect is evident in the story about my coat that I told in Chapter 3, and in these two stories:

> I had a stack of records propped up against a box and I was alphabetizing it according to the artist's name, forming a second, sorted stack propped up next to the first. I would take a record from the top of the first stack with my left hand, find and hold open the right place for it in the second stack with my right hand, place the record in its space, let the stack close over it, and repeat the cycle. After a while I found I was doing something different: whereas before my eyes stayed on the new record until I had picked it up, now I would read the artist's name as soon as I was done with the last record. Then as I picked it up with my left hand, my eyes were already helping my right hand find the right place in the second stack.
>
> I was trying to get a long C program to compile. I was working on a Sun workstation and had divided the screen between two Unix shell windows so I would not have to exit the editor to run the compiler. I would run the compiler and it would produce error messages, for example "syntax error near { on line 173," so I would go back to the editor window. The only way I knew to get to line 173 was to go to the top of the buffer and go down 172 lines. This got to be a cycle, fixing errors and recompiling. After a while, I found that I would move the editor to the top line before the compiler had even starting generating error messages. (Finally the compiler completed without errors and half of me had to skid to a confused halt.)

In each case, fragments of the routine (reading the artist's name, moving my eyes to the second record stack, moving the cursor to the top of the editor) drifted to an earlier point in the cycle. Various actions took place as soon as they could instead of in the more logical order in which they first assembled themselves as I improvised my way through my dealings with each record or syntax error.

What is happening, intuitively, is that an improvised form of activity is rearranging itself by increments, with various elements of my own thinking moving from the place in the cycle where I first formulated them to the place where they were first applicable. Chapters 8 and 9 will return to the larger issue of how to think computationally about routine activity. But for the moment, I would like to introduce some technical ideas for thinking about this kind of incremental evolution of routine forms of activity.

Main ideas of dependency maintenance

Dependency maintenance is a technical method from AI research for keeping track of the logical structure of an agent's symbolic reasoning.[7] It makes several presuppositions about the agent's architecture:

- Some symbolic mechanism, call it the *reasoner,* constructs discrete *thoughts* that the agent decides to believe or disbelieve.
- With the exception of logical axioms and sense data, which are treated as veridical, the agent's reasons to believe or disbelieve a thought can be represented as a small, finite list of other thoughts.
- These thoughts are the means by which the agent decides what is happening and what actions to take.
- Occasions to believe or disbelieve thoughts come packaged into discrete *situations.*
- The agent frequently employs the same thoughts on different occasions.

Although these presuppositions obviously carry a great deal of philosophical baggage, virtually all symbolic models of action share most of them. In the spirit of reductio ad absurdum, I wish to adopt them here as leading assumptions; Chapters 8 through 12 will reexamine some of them. The most highly developed example of these presuppositions is formal logic, for which the thoughts are logical propositions and the reasons to believe them are dictated by syllogisms. But many schemes for modeling thoughts do not employ formal logic, and I will take no position here about the virtues of formal logic for modeling human action.

As an intuitive design matter, dependency maintenance starts from two premises: first, that thinking new thoughts is difficult; and second, that if a thought has been useful once it is likely to be useful again. New thoughts might arise through analogy, induction, storytelling, advertising, or exhaustive testing of hypotheses; for present purposes it does not matter where they come from, it being taken for granted that novel thoughts are relatively unusual. By performing the bookkeeping work that is required to reapply the old thoughts when they are needed, the dependency system makes it unnecessary for the reasoner to reinvent old thoughts on new occasions.[8] The dependency system has a simple interface that makes no presuppositions about what might possibly count as a "thought."

First a blizzard of technical definitions, then some examples. On any moment the reasoner can hand the dependency system a *conclusion* and a set of *reasons.* The conclusion and the reasons are all *propositions.* There are two kinds of reasons, *positive* and *negative.* Every proposition has, at any given time, a *value* of either IN or OUT, meaning roughly "believed" or "not believed." (If a proposition has an OUT value, that does not mean that the agent regards it as false; it only means that the agent does not have enough reason to regard it as true.) Given a conclusion and some reasons, the dependency system constructs and stores a *justification.* This new justification declares that henceforth the conclusion is to be IN whenever all the positive reasons are IN and all the negative reasons are OUT. (Sometimes justifications are also called "dependencies," since the conclusion is said to depend on its reasons.) A proposition might have several justifications; it is IN if any one of them satisfies this condition and OUT otherwise. A proposition with no justifications is called a *premise;* propositions are thus divided into premises and conclusions. The value of a premise might be wired IN. Or it might be determined by some other piece of machinery, like a sensor. If so, it is called an *input.* A proposition might also directly drive some other piece of machinery, such as a motor command. If so, it is called an *output* (Figure 6.1).

The entire collection of propositions and justifications is called a *dependency network.* Think of a dependency network as a binary logic circuit (regardless of how it happens to be implemented in a given instance). Each proposition is a wire – technically, a *node* – which carries a value of 1 if it is IN and 0 if it is OUT. (The values of 1 and 0, then, do not mean "true" and "false" but rather IN and OUT.) Each justification is an *n*-input AND-NOT gate joining some reasons to some conclusions. A

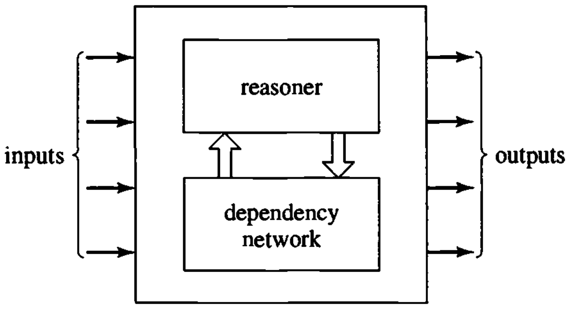


Figure 6.1. A dependency system maintains a dependency network that consists of propositions, modeled as electrical nodes, which are joined by justifications, modeled as logic gates. Some of the propositions correspond to inputs and others to outputs. The system occasionally adds new circuitry when its reasoner does something new.

network is said to be *settled* if all the IN conclusions are justified and vice versa, and *consistent* if there exists some valid assignment of IN and OUT to conclusions to which the network can settle. Whenever a premise changes value or a new justification is added to the network, the dependency system somehow finds a new assignment of IN and OUT to the conclusions that settles the network. Whether settling the network is easy and whether the outcome is unique depend on the network, as I will explain later.

The vocabulary of dependency maintenance is suggestive, but few of its connotations are actually intended. Propositions have no internal structure (nouns and verbs, connectives and parentheses, or whatever) as far as the dependency system is concerned. The reasoner may construct propositions that contain quantifiers, negations, probabilities, or magic words, but the dependency system only knows whether two propositions are exactly the same or not. Likewise, the reasoner may justify a given conclusion deductively, heuristically, or numerologically, but the dependency system only has access to the positive and negative reasons it has been given to justify a conclusion. If new conclusions might be drawn among the existing premises and conclusions, the dependency system will not draw them automatically.

I will illustrate the idea of dependency maintenance using some cartoon examples. Let a justification be specified in this way, using a standard sort of notation scheme derived from the Lisp programming language:

```
(<= conclusion
    (in positive-reason1 positive-reason2 . . .)
    (out negative-reason1 negative-reason2 . . .))
```

The italics describe what sort of thing is found in each "slot."

To record an ordinary monotonic deduction, use only positive reasons:

```
[Example 1]
(<= (mortal Socrates)
    (in (for-all x (implies (human x) (mortal x)))
        (human Socrates))
    (out))
```

"As long as all humans are mortal and Socrates is human, Socrates is mortal."

This justification is modeled after the classical syllogism. It will be implemented by a single 2-input AND gate, joining the two reasons – namely `(for-all x . . .)` and `(human Socrates)` – to the conclusion – namely `(mortal Socrates)` (Figure 6.2).

For the reasoner, the three propositions in Example 1 have an internal structure – evidently one modeled on first-order logic and implemented using linked lists. The dependency system sees none of this. As far as it is concerned, the reasoner said,

```
[Example 2]
(<= mortal-Socrates
    (in for-all-x-implies-human-x-mortal-x
        human-Socrates)
    (out))
```

"As long as allhumansaremortal and Socratesishuman, Socratesismortal."

where the propositions have no internal structure at all, so that they appear simply as arbitrary distinct symbols. (The hyphens are another notational convention drawn from Lisp. Symbols such as `human` and `for-all` and `mortal-Socrates` have no internal structure as far as the programming language *qua* formal system is concerned.) For that matter,

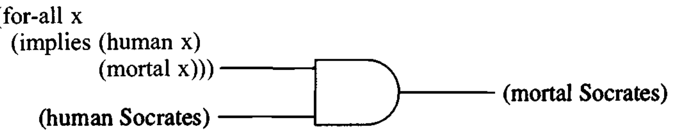


Figure 6.2. A justification with only positive reasons creates an AND gate that ensures that its conclusion will be IN whenever both its reasons are IN.

the dependency system would be indifferent if the reasoner delivered it a justification that makes no sense on logical grounds, for example,

```
[Example 3]
(<= (immortal Plato)
    (in (for-all x (implies (human x) (mortal x)))
        (human Socrates))
    (out))
```

"As long as all humans are mortal and Socrates is human, Plato is immortal."

The dependency system, in short, makes few assumptions about the workings of the reasoner. In particular, it does not require the reasoner to be correct or optimal or logical or efficient, as long as it delivers a series of syntactically well-formed justifications. Thus, for present purposes, the words "thinking" and "reasoning" are inherently imprecise. They will attain a more satisfactory formal definition in Chapter 7, but I will not adopt any definite philosophical explanation of them.

A proposition with an empty justification will always be IN:

```
[Example 4]
(<= (life is short)
    (in)
    (out))
```

"Life is short, no matter what."

The propositions in a dependency network do not have to be examples from formal logic texts. They are, in fact, more likely to be conclusions about what to do.

```
[Example 5]
(<= (intend (become philosopher))
    (in (want truth))
    (out (want money)))
```

"As long as I want truth and do not want money, I intend to become a philosopher."

The system records its dependencies using digital logic gates, but this does not restrict its reasoning to any particular rules of inference. Heuristic justifications, for example, are easily accommodated:

```
[Example 6]
(<= (incomprehensible Derrida)
    (in (philosopher Derrida)
        (French Derrida))
    (out))
```

"As long as Derrida is French and a philosopher, he is incomprehensible."

A proposition might have several justifications. If the reasoner also says,

```
[Example 7]
(<= (incomprehensible Derrida)
    (in (thinks Derrida (too-easy Heidegger))
        (writes-in Derrida French-puns))
    (out))
```

"As long as Derrida thinks that Heidegger is too easy, and as long as he writes in French puns, he is incomprehensible."

then the resulting network will comprise two AND gates with their outputs wired together, so that the node corresponding to `(incomprehensible Derrida)` will be IN if *either* of its two justifications is satisfied (Figure 6.3).

A justification cannot contain variables. That is, the dependency system does not recognize or reason with any variables that the reasoner might include in a proposition. If the reasoner, mimicking the syntax of quantification in first-order logic, issues a justification that reads

```
[Example 8a]
(<= (incomprehensible x)
    (in (philosopher x)
        (French x))
    (out))
```

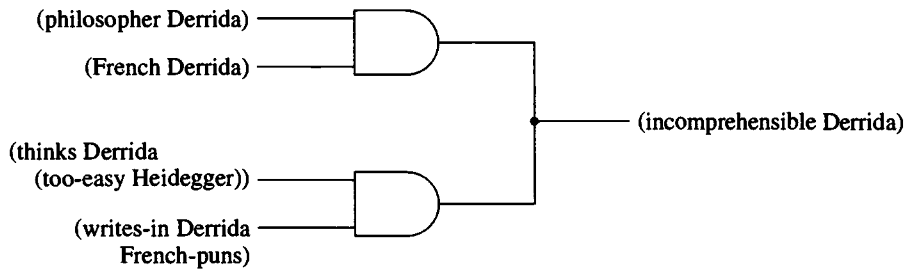

(philosopher Derrida)
(French Derrida)
(thinks Derrida (too-easy Heidegger))
(writes-in Derrida French-puns)
(incomprehensible Derrida)

Figure 6.3. When a proposition has more than one justification, any of its justifications can support it.

"As long as X is a philosopher and French, X is incomprehensible."

not

"For all x, if x is a philosopher and French, x is incomprehensible."

then the dependency system will not interpret this as a universal statement that all French philosophers are incomprehensible. If the reasoner surmises that several French philosophers are incomprehensible, it needs to issue a separate justification for each one:

```
[Example 8b]
(<= (incomprehensible Barthes)
    (in (philosopher Barthes)
        (French Barthes))
    (out))
(<= (incomprehensible Foucault)
    (in (philosopher Foucault)
        (French Foucault))
    (out))
(<= (incomprehensible Deleuze)
    (in (philosopher Deleuze)
        (French Deleuze))
    (out))
```

"As long as Barthes is a philosopher and French, Barthes is incomprehensible."

> "As long as Foucault is a philosopher and French, Foucault is incomprehensible."
> "As long as Deleuze is a philosopher and French, Deleuze is incomprehensible."

I will refer to a connected subnetwork as a *patch* of the whole network. As this example makes clear, traditional sorts of representation lead to replicated structure in dependency networks; each French philosopher gets his own patch of network. It would be nice if the propositions that are connected by dependencies could include variables, so that one small circuit could deduce the incomprehensibility of all French philosophers, without the necessity of naming them all individually. But this is not how combinational logic circuits work; moreover, it appears that no simple scheme can make them work that way. Chapters 10 through 13 will dwell on this fact.

Negative reasons provide a way to express heuristic lines of reasoning whose assumptions are to be made explicit. One technique is to establish a default:

```
[Example 9a]
(<= daytime
    (in at-work)
    (out nighttime))
```

That is, if the agent is at work, it should assume it is daytime unless convinced that it is nighttime. This justification is said to be *non-monotonic.* It would be implemented as an AND-NOT gate (Figure 6.4).

Let us elaborate the example:

```
[Example 9b]
(<= daytime
    (in out-of-doors bright)
    (out))
(<= nighttime
    (in out-of-doors dark)
    (out))
```

Joining 9a and 9b produces the network depicted in Figure 6.5. This network contains six propositions and three justifications. Four of the propositions – that is, `at-work`, `out-of-doors`, `bright`, and `dark` – are premises because they have no justifications. Imagine that the values

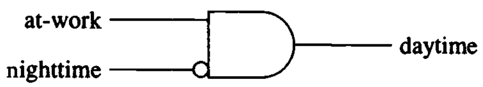


Figure 6.4. Sometimes a justification will have negative reasons, indicating that it should support its conclusion only if all its positive reasons are IN and there is no reason to believe in any of its negative reasons.

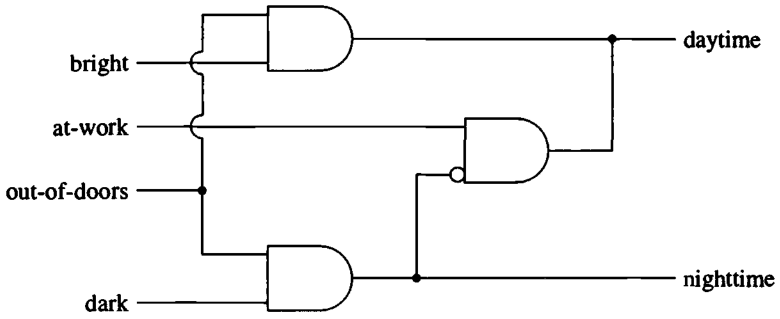


Figure 6.5. In this diagram, the agent will accept certain information as evidence of its being daytime and other evidence of its being nighttime. If no relevant evidence is available, the agent will assume that it is daytime.

of `at-work` and `out-of-doors` are determined by other justifications not shown. Imagine as well that the whole network is located in the head of an agent named Thomas and that `bright` and `dark` are inputs connected to Thomas's visual system. Their values are continually updated according to how bright it is.

Now suppose that Thomas is at work and indoors. `At-work` is IN and `out-of-doors` is OUT. Probably `bright` will be IN, but since `out-of-doors` is OUT neither `bright` nor `dark` will influence the conclusions at all. `Nighttime` is OUT because `out-of-doors` is OUT. And `daytime` is IN because `at-work` is IN and `nighttime` is OUT.

Now suppose Thomas gets off work and walks outside. When Thomas notices the clock hitting 5:00, `at-work` will go OUT and so `daytime` will go OUT too. At this point both `daytime` and `nighttime` will be OUT, assuming that Thomas has no other justifications for these propositions. Once he gets out the door, though, it being winter in Boston, `dark` will be IN. Thus `nighttime` will be IN and `daytime` will be OUT.

Thomas's reasoning thus far has some holes. Starting at about 4:30 p.m. it will be dark outside. But he, still hard at work, will assume it is daytime. Suppose that Thomas, longing for a beer as 5:00 approaches, looks out the window toward the parking lot and is startled to find it dark. He might then formulate a new justification for `nighttime`:

```
[Example 9c]
(<= nighttime
    (in looking-out-window dark)
    (out))
```

This new insight, while not spectacular, was no doubt hard work, just because thinking anything new is hard work. Fortunately, dependencies never go away. As soon as tomorrow's approaching beer leads Thomas to look out the window again, this bit of thinking (if one is willing to call it that) will happen automatically. The same thing will happen with next week's approaching beers, and next year's. With successive new insights, his network will grow larger and larger.

Now suppose Thomas also thought something like

```
[Example 9d]
(<= nighttime
    (in winter-time late-afternoon)
    (out))
```

The values of propositions like `late-afternoon` do not get updated by magic. If Thomas has no sixth sense for wall-clock time, `late-afternoon` will stay OUT, regardless of the time, until some other circuitry drives it IN. But one has many occasions to try guessing a rough time of day, and these ways of guessing will accumulate in one's dependency network, applying themselves whenever they get a chance.

Not all lapses of reasoning can be repaired by adding new justifications. A solar eclipse might make it dark in the daytime and a baseball stadium might make it light in the nighttime. But it would have taken impossible foresight to have phrased Example 9b as

```
[Example 9e]
(<= daytime
    (in out-of-doors bright)
    (out at-baseball-game))
(<= nighttime
    (in out-of-doors dark)
    (out solar-eclipse))
```

Any conclusion about a real-life situation is true only ceteris paribus. One never stops discovering exceptions to general rules. Consequently, it must be possible to add new negative reasons to some of the existing justifications. When Chapter 7 describes how a particular program uses dependencies, these justifications will be the ones created by "UNLESS rules."

A dependency network can in principle be circular. This happens, for instance, when propositions entail one another. To take a standard philosophical example:

```
[Example 10]
(<= looking-at-the-Morning-Star
    (in looking-at-the-Evening-Star)
    (out))
(<= looking-at-the-Evening-Star
    (in looking-at-the-Morning-Star)
    (out))
```

Thus, the agent concludes it is looking at the Morning Star whenever it believes it is looking at the Evening Star, and vice versa. Unfortunately, when the Morning/Evening Star goes away, the propositions will continue to justify each other. In general, circularities can arise when there are several propositions, any few of which justify the rest.

```
[Example 11]
(<= games=7 (in wins=3 losses=4) (out))
(<= wins=3 (in games=7 losses=4) (out))
(<= losses=4 (in games=7 wins=3) (out))
```

Circular dependencies cause immense technical problems because it is impossible in general to determine a consistent assignment of IN and OUT values to the various propositions. They will, however, play no role in the mechanisms I will describe.

By now I have enumerated everything that can happen in a dependency network: gates propagate binary values and new justifications and connections are made. Dependency networks, in other words, are not general-purpose data structures that algorithms can inspect, search through, rearrange, measure, or prove things about. Instead, they resemble the neural networks described by Feldman (1982) and Marr (1970), in which new elements are added through gradual, incremental recruitment over a long period.

7 Rule system

Using dependencies in a rule system

This chapter discusses one use of dependencies, a programming language called Life. Although the demonstrations of Chapter 9 and 10 will use Life to make some points about improvised activity, this chapter describes Life programming as a technical matter with little reference to theoretical context. Readers who find these descriptions too involved ought to be able to skip ahead without coming to harm.

Life is a *rule language,* a simplified version of the Amord language (de Kleer, Doyle, Rich, Steele, and Sussman 1978). This means that a Life "program" consists of a set of *rules.* Each rule continually monitors the contents of a database of propositions, and sometimes the rules place new propositions in the database. The program that does all of the bookkeeping for this process is called the *rule system.* The rule system functions as the reasoner for a dependency system. The rule system and dependency system both employ the same database, and most of the propositions in the database have a value of IN or OUT. Roughly speaking, when an IN proposition (the *trigger*) matches the *pattern* of an IN rule, the rule *fires* and the appropriate *consequence* is assigned the value of IN. If necessary, the system first builds the consequent proposition and inserts it in the database. This might cause other rules to fire in turn, until the whole system settles down. In computer science terms, this is a *forward-chaining* rule system. The role of dependencies is to accelerate this settling down without changing its outcome. The technical challenge is to get the rule system to mesh smoothly with the dependency maintenance system underneath.

One might take two views of the Life rule system in operation. On one view, dependencies are accelerating the operation of rules. But on another view, a dependency network is being incrementally built from convenient but theoretically uninteresting parameterized specifications. It is the second view that will become important in the next two chapters. I do not believe that people have rule systems in their heads. Instead, the purpose of the rule system is to provide *some* source of new thoughts so that the dependency system can build a network. The question of theoretical interest is how an accumulated network will lead its owner to interact with the world.

The machinery underlying the Life rule language is designed on the assumption that most rule firings have happened before. Though many applications exist in which this assumption would not hold, it seems plausible for the purpose of simulating human activity because life is more or less routine. Intuitively, most things that happen have happened before, so that the dependency system can do most of the work. Whenever a rule fires, the rule system constructs a justification declaring that the rule and the trigger are reasons for believing the conclusion. That rule need not fire again on that trigger. Consequently, unlike advanced production systems such as OPS5 (see later), the Life rule system has not been optimized for the speed of running a large number of rules. Instead, it has been optimized for the speed of determining which rules, if any, should run when a proposition in the database changes from IN to OUT or OUT to IN.

A cartoon example may convey some idea of what the rule system does. Here is a rule:

```
R13: (if (sees the-shepherd the-wolf)
         (rings the-shepherd warning-bells))
```

One sunny Monday, the first wolf appears (either as the consequence of a rule or as a premise):

```
A27: (sees the-shepherd the-wolf)
```

Then rule R13 fires on proposition A27. The rule has not fired on this trigger before, so the system builds a new proposition,

```
A28: (rings the-shepherd warning-bells)
```

and a new justification,

```
J41: (<= A28 (in R13 A27) (out))
```

which would be drawn as an AND gate connecting R13 and A27 to A28. Once the wolf goes away, A27 will go OUT. Assuming A28 has acquired no other justifications, A28 itself will go OUT as well. Now Tuesday comes and the wolf appears again, so that A27 comes IN again. Assuming that rule R13 is still IN as well, A28 will also come IN, due to the justification J41. Whereas on Monday the system had to do some pattern matching and assemble a new proposition and a new justification, on Tuesday it only had to propagate a changed value through a gate. If the wolf visits the shepherd's flock every day, A27 and A28 will cycle between IN and OUT. This effect is impressive on a large scale; the system accelerates as everything that happens often, happens once.

Rule language semantics

The Life rule language is neither procedural nor declarative (Fodor 1981a; Johnson-Laird 1977). That is, a set of Life rules is neither a step-by-step computer program nor a catalog of knowledge. Each proposition in the database is a Lisp list structure composed of list cells, symbols, and variables. Variables are notated with question marks, for example `?x`. Most of the propositions, again, have a value, either IN or OUT, meaning roughly "currently believed" or "not currently believed." A proposition is always OUT unless the rule system discovers some definite reason for it to be IN. Here are some possible (though hardly exemplary) propositions:

```
(forgets moose (flies squirrel))
(language is the house of being)
(for-all ?x (implies (human ?x) (mortal ?x)))
((((?x mortal) (?x human) implies) ?x for-all)
(plan put-on (?x ?y)
  (preconditions (clear-top ?x) (clear-top ?y))
  (actions (pick-up ?x) (move-to ?y) (put-down))
  (results (cleartop ?x) (on ?x ?y)))
```

For technical reasons, two propositions are considered identical if they are the same except for the names of variables. Thus, `(loves ?x ?x)` is the same as `(loves ?y ?y)` but different from `(loves ?x ?y)`.

Some of the propositions are rules. There exist two kinds of rules, IF rules and UNLESS rules. Rules are the only propositions with a prescribed syntax. Their syntax and informal semantics are as follows:

`(if` ***`pattern consequence-1 . . . consequence-n`***`)`

"As long as *pattern* is IN, make each *consequence-i* IN as well."

`(unless` ***`pattern consequence-1 . . . consequence-n`***`)`

"As long as *pattern* is OUT, make each *consequence-i* IN."

Each pattern and consequence is a proposition. These propositions are likely to contain variables. The critical phrase is "as long as." Let us consider some examples.

The example considered earlier,

```
(if (sees the-shepherd the-wolf)
  (rings the-shepherd warning-bells))
```

should be read

> "As long as the shepherd sees the wolf, the shepherd rings warning bells."

When the rule's pattern comes IN, the consequences come IN as well. (This rule, like most, has only one consequence.) When the pattern goes back OUT, each consequence goes OUT as well, assuming it is not being justified by another rule.

Rules can include variables. Variables are interpreted differently in the two kinds of rules. We might write

```
(if (sees ?anyone the-wolf)
  (rings ?anyone warning-bells))
```

"Whoever sees the wolf rings warning bells."

In general, an IF rule fires when (a) the rule is IN and (b) some proposition matching the rule's pattern is IN.

Thus, if we assert (meaning, establish as a premise),

```
(sees the-farmer the-wolf)
```

then the new proposition

```
(rings the-farmer warning-bells)
```

will come IN as well. If forty people see the wolf, the rule will fire forty times and all forty people will ring warning bells. If the wolf passes out of sight of ten of those people, those ten will stop ringing their bells even if the other thirty continue.

In general, an UNLESS rule fires when (a) the rule is IN and (b) no proposition matching the rule's pattern is IN. We might write

```
(unless (rings ?someone warning-bells)
  (sneaks-toward the-wolf the-sheep))
```

"As long as nobody rings warning bells, the wolf sneaks toward the sheep."

If this rule is asserted and nobody is ringing warning bells, the rule's consequence will be IN. As soon as someone starts ringing warning bells, it will go OUT again. As soon as everyone stops ringing warning bells, it will come IN again.

Since rules are propositions, a rule's consequences might include other rules. We might write

```
(if (owns-sheep ?person)
  (if (hears ?person warning-bells)
    (runs-to ?person the-meadow)))
```

"Anyone who owns sheep and hears the warning bells runs to the meadow."

```
(if (sneaks-toward the-wolf ?sheep)
  (if (is-a ?sheep sheep)
    (unless (shoots-at ?person the-wolf)
      (grabs the-wolf ?sheep))))
```

"If the wolf is sneaking toward a sheep, then unless somebody shoots at the wolf it will grab the sheep."

When rules fire, they create circuitry in the dependency network. An IF rule defines an AND gate each time it fires. If an IF rule's pattern has no variables, it will define a single gate. If it does have variables, it will define a separate AND gate for each matching proposition (Figure 7.1). An UNLESS rule defines a single AND-NOT gate; each proposition matching its pattern will be assigned an inverted input to that gate (Figure 7.2). As a result, a complex rule defines a sort of template; every binding of its variables will produce a new patch of dependency network.

The Life rule system differs from a production system (Forgy 1981, 1982; Newell and Simon 1972) in several ways:

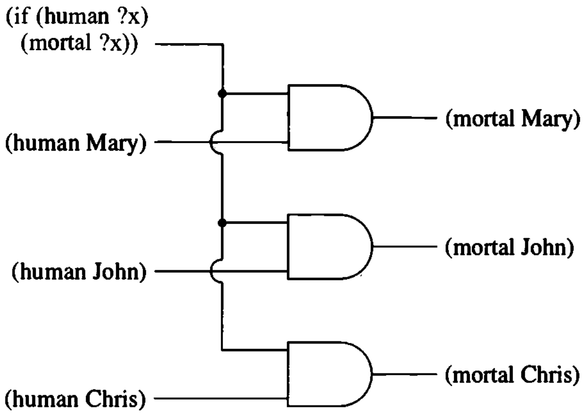


Figure 7.1. If an IF rule has a variable in its pattern, many propositions might match it. Each one of the resulting rule firings will generate its own AND gate.

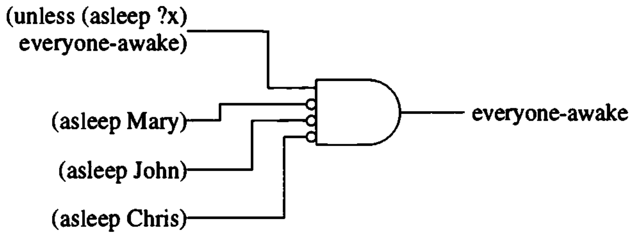


Figure 7.2. If an UNLESS rule has a variable in its pattern, many propositions might match it. Each one of those propositions will get its own inverted input into the rule's AND-NOT gate.

- All rules fire whenever they can. The architecture neither defines a notion of conflict between rules nor provides mechanisms for conflict resolution.
- A rule is not an imperative. It does not say "when" but "as long as." It does not simply make its consequence IN; it also arranges (through the justification it creates) for the consequence to go OUT again when it is no longer justified.
- There is no working memory as distinguished from the database as a whole. All IN rules can fire and all IN propositions can trigger rules. (This is true in practice for many users of production systems.)

- The speed of the system does not depend on the speed of the rule-firing machinery. Most of the work is done in the dependency network, which is easy to implement in parallel machinery.
- Life's rule-firing machinery is not part of any proposed cognitive architecture. Where a production-system theory might propose that human brains implement production systems, a dependency theory would propose that human brains implement combinational logic circuits without taking a position on the nature of the reasoner.

The Life rule system shares much of the motivation and spirit of production systems, and I will return to the relation between them.

Life programming is more similar to logic programming languages (Kowalski 1974; van Caneghem and Warren 1986), particularly in the semantics of variables. Also, UNLESS rules differ from logical negation in the same way as negation in logic programming; each means "not (yet) derived" instead of "not derivable" or "not true." However, there are some differences:

- The Life language defines no notion of predicates or functions. Propositions do not need to follow any syntactic rules unless they begin with `if` or `unless`.
- Patterns are not matched by unification; in order for a proposition to trigger a rule it must match the rule's pattern, that is, there must be some assignment to the pattern's variables that produces the triggering proposition.
- Whereas standard logic programming languages are backward-chaining, the Life machinery is forward-chaining. Also, whereas a logic program would not normally derive all its logical consequences, Life continues firing rules as long as there are rules that can fire.
- The efficiency of a Life rule set does not depend on the speed of pattern matching. Once the system gets going, the dependency network does most of the work.

How it works

The rule system operates inside a loop. The loop is driven by the type of clock described in Chapter 5. On every *tick* of the clock, the rule system begins firing rules and the dependency system begins assessing justifications and moving propositions IN and OUT, and this process proceeds until no more rules can fire and the dependency system has settled. On every *tock* of the clock, any outside programs inspect the values of any propositions that concern them and change the values of any premises that they wish to change. (The nature of these outside programs will become clear in Chapters 9 and 10.) On the first few cycles of the clock, the rule system is slow to settle because many rules are firing for the first time. If no premises change their values on a given *tock*, nothing need happen in the rule system or dependency system. If the exact set of premises has been encountered before, only the dependency system should have any work to do. And with time, novel sets of premises should become less frequent.

As a design matter, the rule system's interface to the dependency system should have the following properties:

1. A rule fires only once on a given trigger.
2. It takes almost no time to determine which, if any, rules need to fire.

The first condition is easy enough. Every rule has a list of triggers on which it has fired in the past. Given a candidate rule and trigger, the rule system first checks this list and proceeds only if the trigger is not on it. The second condition is harder, and the rest of the section explains how it is achieved.[1] The database is organized as a *lattice* (Birkhoff 1967) of propositions under generalization – a *subsumption lattice.* Given two distinct propositions P1 and P2, P1 generalizes P2 if there is some assignment to P1's variables that produces P2. Here are some examples:

```
(rings ?x warning-bells)
```
  generalizes
```
(rings the-farmer warning-bells)
```

```
(eats ?x ?y)
```
  generalizes
```
(eats wolf sheep)
```

```
(eats ?x ?x)
```
  generalizes
```
(eats fish fish)
```
  but does not generalize
```
(eats cat fish)
```

```
(eats ?x ?y)
  generalizes
(eats ?x ?x)

(knows ?p ?x)
  generalizes
(knows ?p (sees ?q ?p))
  which in turn generalizes
(knows the-wolf (sees ?q the-wolf))
  which in turn generalizes
(knows the-wolf (sees the-shepherd the-wolf))
```

A proposition without variables generalizes nothing and the proposition `?x` generalizes everything. A proposition does not generalize itself. Generalization is a partial order because it has no cycles. But it is not a total order because it is common for a proposition P1 to generalize both P2 and P3, neither of which generalizes the other. For example,

```
P1: (knows ?p ?x)
P2: (knows the-wolf ?x)
P3: (knows ?p (sees ?q ?p))
```

Often the propositions P2 and P3 will both generalize some further proposition P4:

```
P4: (knows the-wolf (sees ?q the-wolf))
```

This is called *reconvergence* and it causes difficulties for parallel implementation of a large class of symbolic indexing schemes, including the lattice technique.

Figure 7.3 depicts a sample lattice. The rule system's data structures record only immediate generalization relationships – in technical terms the *minimal generalizations* or the *cover* of the relation. The algorithm that *indexes* new propositions into the lattice is subtle and tolerably fast. (Unfortunately, propositions cannot be indexed in parallel.) In the demonstrations of Chapter 10, a typical lattice has a few thousand elements.

The rule system's algorithm uses the proposition lattice. The lattice includes every proposition that has been made a premise or derived by firing a rule. In particular, it includes the rules themselves. It also includes some propositions that have no values. Among these are the patterns of all the rules. Thus, if the rule

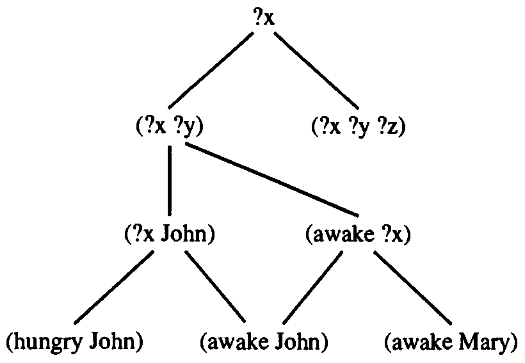


Figure 7.3. The patterns in the rule system's database are stored in a lattice.

```
(if (sees ?anyone the-wolf)
  (rings ?anyone warning-bells))
```

is in the lattice, so is the proposition

```
(sees ?anyone the-wolf)
```

even though this proposition makes no sense by itself and is presumably not IN. As Figure 7.4 illustrates, the proposition lattice has a useful property. If P is the pattern of rule R and T is another proposition, T is a potential trigger for R just in case P is above T in the lattice. This suggests a simple algorithm for the rule system. (I will describe the actual algorithm afterward.) For IF rules, the algorithm is

1. Whenever a proposition T comes IN, climb up the lattice from it. Whenever you encounter a proposition P that is an IF rule R's pattern, if R is IN then fire R on T (unless R has already fired on T).
2. Whenever an IF rule R comes IN, climb down the lattice from its pattern P. Whenever you encounter a proposition T that is IN, fire R on T (unless R has already fired on T).

To fire an IF rule R on proposition T:

1. Match R's pattern to T, obtaining a list of variable assignments.
2. Perform these assignments on each of R's consequences, producing a list of new propositions $C_1 \ldots C_n$ (typically there is only one of them).

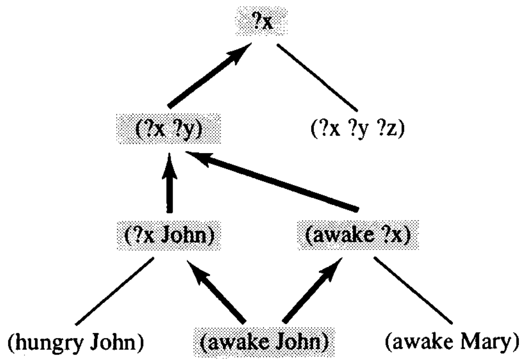


Figure 7.4 To find the patterns of all the rules that a given proposition might possibly fire, one need only move upward in the lattice starting from that proposition.

3. Index each of the new propositions into the lattice. (Some of them might already be there.)
4. For each C_i, construct a new justification: (<= C_i `(in` R T`)` `(out`)).

The algorithm for UNLESS rules is not as intuitive. An UNLESS rule "fires" as soon as it first comes IN. From then on, whenever a new proposition matches the rule's pattern, it "unfires," meaning that the new trigger is added to the OUT list of the justification the rule created for its consequence. The algorithm for an UNLESS rule of the form `(unless` P $C_1 \ldots C_n$`)` is

1. Whenever a proposition T comes IN, climb up the lattice from it. Whenever you encounter a proposition P that is an UNLESS rule pattern, unfire R on T (unless R has already unfired on T).
2. Whenever an UNLESS rule R comes IN for the first time, climb down the lattice from its pattern P. Along the way, accumulate a list of every proposition T_i that has ever been IN. Finally, make a justification for each of the rules' consequences: `(<=` C_i `(in` R`)` `(out` $T_1 \ldots T_k$`))`.

To unfire an UNLESS rule R on proposition T:

1. For each C_i, find the justification that mentions R. (In practice they will all share a single AND-NOT gate.)
2. Add T to its OUT list so that it reads `(<=` C_i `(in` R`)` `(out` $T_1 \ldots T_k$, T`))`.

Note that no lattice climbing occurs when a proposition goes OUT. If the OUT-going proposition enters into the justification of any other propositions, the dependency system will take them OUT too if necessary.

If it seems expensive to climb around in the lattice, remember that every proposition one encounters while climbing down is a potential trigger and every proposition one encounters while climbing up is a potential rule pattern. Little effort is wasted.

In reality, although the scheme just described would work correctly, the system is actually more complicated. Recall that the goal is to permit the system to decide rapidly which, if any, rules need to be fired. The system just described does not achieve this goal, because the system must perform a search in the lattice every time a proposition goes IN. I will describe the true algorithm only for IF rules because the algorithm for UNLESS rules cannot be stated simply but is still easy enough to rederive.

At a small expense of memory space, the system can avoid searching the lattice except the first time a given proposition comes IN. Every proposition maintains a bit indicating whether it has ever been IN. When a proposition comes IN for the first time, it searches upward in the lattice; when a rule comes IN for the first time, its left-hand side searches downward in the lattice. While searching, it looks for rules that might be able to fire now. But it also looks for potential rule firings. When a trigger is searching upward for patterns, it looks for rules that are not currently IN but have been IN at some point in the past. Likewise, when a rule's left-hand side is searching downward for triggers, it looks for propositions that are not currently IN but have been IN at some point in the past. All of this information is stored with the proposition doing the searching.

Thus, every proposition has a list of potential rules and every rule has a list of potential triggers. Whenever a proposition comes IN or goes OUT, it checks all of its potential rules to see if they are IN. If so, the rules fire and are removed from the proposition's list of potential rules. Whenever a rule comes IN or goes OUT, it checks all of its potential triggers to see if they are OUT. If any are, the rule fires on them and they are removed from the rule's list of potential triggers.

This algorithm works well in practice, because the lists of potential rules and triggers are always short. One could write pathological rule sets in which the lists were choked with rules and triggers that could lead to useful work in principle but never will in practice, but this has never happened with any actual rule set.

Finally, note that although the generalization relation does form a lattice once we go through technicalities like defining a unique minimal proposition, the algorithm requires generalization to be only a partial order. Not all partial orders form lattices; in particular, the generalization relation probably does not form a lattice when restricted to the set of propositions that has actually been indexed at a given moment.

Incremental updating

We can now leave the microscopic details of the Life rule system machinery and begin considering what the system does on a large scale in practice. Rather than present a full-scale example of the system in action, I will demonstrate the benefits of dependency maintenance indirectly by discussing the process of writing and debugging Life rule sets. Normally when fixing a bug in a program, a programmer must run the program over from the beginning to the point when the problem arose. With dependency maintenance, though, the program is rerun incrementally. The system does only the work that needs to be done differently. If you rewrite a rule, take the old version OUT and make the new version IN. If a rule set starts going awry, poke around in the dependencies.

Suppose we wished to implement a set of rules to generate the action in the story of "The Boy Who Cried Wolf." Here are some cartoon premises:

```
R12: (if (sees ?anyone the-wolf)
       (rings ?anyone warning-bells))
R37: (if (in-town ?person)
       (if (rings ?anyone warning-bells)
         (hears ?person warning-bells)))
A46: (owns-sheep Katya)
R47: (if (owns-sheep ?person)
       (if (hears ?person warning-bells)
         (runs-to ?person the-meadow)))
```

(The system assigns labels like R12 and A46 to provide short names for the propositions.)

Now the system is running. The story reaches its climax. The wolf appears and sneaks up on the sheep. The shepherd sees the wolf and rings the warning bells. The townspeople hear the bells and go running, even though the story requires them to ignore the bells, having been inconvenienced by a series of false alarms. Something is wrong, so stop the system. Among the IN propositions is

```
A63: (runs-to Katya the-meadow)
```

We can inspect the dependencies behind A63:

```
(why? A63)
A63 is in because R51 ran on trigger A39:
   R51: (if (hears Katya warning-bells)
          (runs-to Katya the-meadow))
     R51 is in because R47 ran on trigger A46:
        R47: (if (owns-sheep ?person)
                (if (hears ?person warning-bells)
                  (runs-to ?person the-meadow)))
          R47 is a premise.
        A46: (owns-sheep Katya)
          A46 is a premise.
  A39: (hears Katya warning-bells)
    A39 is in because R49 ran on trigger A28:
       R49: (if (rings ?anyone warning-bells)
               (hears Katya warning-bells))
    R49 is in because R37 ran on A40
       R37: (if (in-town ?person)
               (if (rings ?anyone warning-bells)
                 (hears ?person warning-bells)))
         R37 is a premise.
       A40: (in-town Katya)
         A40 is a premise.
         A28: (rings the-shepherd warning-bells)
           A28 is in because R12 ran on A27:
             R12: (if (sees ?anyone the-wolf)
                     (rings ?anyone warning-bells))
               R12 is a premise.
           A27: (sees the-shepherd the-wolf)
               A27 is in because . . .
```

and so on through the reasons for the wolf being in the meadow and the shepherd seeing it. Even if the premises include a hundred rules, this display presents only the ones that entered into the troublesome conclusion. Reading them, we find that the townspeople are too gullible. We have written rules that lead them to conclude that the shepherd is a liar, but we have not made them act on that conclusion. So let us rewrite R47 to make it more general:

```
R68: (if (owns-sheep ?person)
       (if (believes ?person (at the-wolf the-meadow))
         (runs-to ?person the-meadow)))
R69: (if (rings ?anyone warning-bells)
       (if (hears ?person warning-bells)
         (unless (believes ?person (liar ?anyone))
           (believes ?person (at the-wolf the-meadow)))))
```

(Those with experience in formalizing such things will recognize the deficiencies of these rules, as well as the instructive difficulty of writing completely general rules for such purposes. When I gave up trying to write rules to completely capture the "real reasons" behind the events in the wolf story, I was long past a hundred rules with no end in sight.)

When we retract R47, both R51 and A63 go OUT but everything else stays unchanged. When we now assert R68 and R69, they both run, deriving

```
R70: (if (believes Katya (at the-wolf the-meadow))
       (runs-to Katya the-meadow))
R71: (if (hears ?person warning-bells)
       (unless (believes ?person (liar the-shepherd))
         (believes ?person (at the-wolf the-meadow))))
R72: (unless (believes Katya (liar the-shepherd))
       (believes Katya (at the-wolf the-meadow)))
```

The system has already derived

```
A54: (believes Katya (liar the-shepherd))
```

so rule R72 does *not* license the conclusion that Katya believes the wolf to be in the meadow. Finally we let the simulation proceed. Katya does not go to the meadow (nor does anybody else), the wolf eats the sheep, and the shepherd feels bad.

Advanced rule writing

This section contains technical details that will help explain some of the code fragments in Chapter 9. It can be safely skipped by most readers.

Life is a simple language, but it is still perfectly general. It is Turing universal, for whatever that is worth, in two senses. One can write a Lisp interpreter in Life rules alone, though the result is terribly slow. On the other hand, if a Life system is connected to a world that behaves like the tape of a Turing machine, one can write Life rules for a universal Turing machine. These rules can be written without variables, so the size of the resulting dependency network has a definite upper limit.

Unlike many rule-based programming languages, Life makes no explicit provision for Boolean combinations of rule patterns. For example, one might like to rewrite rule R37:

```
old: (if (in-town ?person)
       (if (rings ?anyone warning-bells)
         (hears ?person warning-bells)))
new: (if (and (in-town ?person)
              (rings ?anyone warning-bells))
       (hears ?person warning-bells))
```

To allow such rules, one can write rules that convert rules with Boolean triggers into equivalent forms that use only the facilities provided directly by the language:

```
(if (if (and ?p ?q) ?c)
  (if ?p (if ?q ?c)))
(if (if (or ?p ?q) ?c)
  (if ?p ?c)
  (if ?q ?c))
(if (if (not ?p) ?c)
  (unless ?p ?c))
```

(These are simplified versions of the rules. The actual rules are considerably more complicated for purposes of generality and efficiency.)

One could even write rules that support backward chaining:

```
(if (if-shown ?p . ?q)
  (try-to-show ?p)
  (if ?p . ?q))
(if (unless-shown ?p . ?q)
  (try-to-show ?p)
  (unless ?p . ?q))
(if (can-show ?p)
  (if (try-to-show ?p)
    ?p))
```

(These rules descend from de Kleer, Doyle, Steele, and Sussman 1977.) For example, it is often necessary to constrain two variables to have different bindings. To express the idea of two different people seeing the wolf, one might say,

```
(if (and (sees ?a the-wolf) (sees ?b the-wolf))
  (if-shown (neq ?a ?b)
    . . .))
```

(The predicate neq means "not equal.") We can now write rules for demonstrating equalities and inequalities:

```
(can-show (eq ?x ?x))
(if (try-to-show (neq ?x ?y))
  (try-to-show (eq ?x ?y))
  (unless (eq ?x ?y)
    (neq ?x ?y)))
```

Given these rules, if both Katya and Anna see the wolf, the following series of propositions would get asserted by these rules:

```
(if-shown (neq Katya Anna)
  . . .)
(try-to-show (neq Katya Anna))
(if (neq Katya Anna)
  . . .)
(try-to-show (eq Katya Anna))
(unless (eq Katya Anna)
  (neq Katya Anna))
(neq Katya Anna)
. . .
```

In practice the system does not use backward chaining in more complex ways than this.

Rules like these raise a serious question of what is fair in Life programming. If my psychological theory allows me to write arbitrarily sophisticated Life programs, it has little content. The general constraint is that the rules I write must lead to the construction of dependency networks that could plausibly be acquired through learning in the real world. This constraint is hard to make operational, but a few principles are clear enough. One should not write rules that must fire regularly (i.e., with different variable bindings) in situations that ought to be routine. It is also a good policy not to write rules that fire on other rules except as a notational convenience. Above all, keep in mind that our topics are routine activity and the interactions between an already existing dependency network and its world. Life rule sets are not psychological theories but specifications for networks.

8 Planning and improvisation

The idea of planning

For the past thirty years or so, computational theorizing about action has generally been conducted under the rubric of "planning." Whereas other computational terms such as "knowledge" and "action" and "truth" come to us burdened with complex intellectual histories, the provenance of "plan" and "planning" as technical terms is easy to trace. Doing so will not provide a clear definition of the word "planning" as it is used in AI discourse, for none exists. It will, however, permit us to sort the issues and prepare the ground for new ideas. My exposition will not follow a simple chronological path, because the technical history itself contains significant contradictions; these derive from tensions within the notion of planning.

In reconstructing the history of "plan" and "planning" as computational terms, the most important road passes through Lashley's "serial order" paper (1951) and then through Newell and Simon's earliest papers about GPS (e.g., 1963). Lashley argued, in the face of behaviorist orthodoxy, that the chaining of stimuli and responses could not account for complex human behavioral phenomena such as fluent speech. Instead, he argued, it was necessary to postulate some kind of centralized processing, which he pictured as a holistic combination of analog signals in a tightly interconnected network of neurons. The seeds of the subsequent computational idea of plans lay in Lashley's contention that the serial order of complex behavioral sequences was predetermined by this centralized neural activity and not by the triggering effects of successive stimuli.[1] At a deeper level, Lashley's paper established a pattern for later cognitivist research in its tendency to resist behaviorism by shifting to an opposite extreme. Whereas the behaviorists portrayed behavior as driven entirely by successive stimuli, Lashley placed his principal emphasis on the predetermination of action by mental processing. While certainly not denying that this processing had inputs, Lashley gave these inputs no clear role in his story.

Newell and Simon kept the notion of centralized mental processing but offered a different account of its workings. As Chapter 3 has explained, they characterized human thinking as a matter of mental search in a problem space through the application of "operators" to "objects." Solving a problem meant discovering a series of operators that could be applied to the initial state to yield the goal state. GPS was a theory of cognition and not a theory of action. It did employ a technique that Newell and Simon referred to as "planning," but this was a way of reducing the effective size of large search spaces and was only tangentially connected with more recent uses of the term. Yet GPS has had an enormous influence in subsequent computational theorizing about action. Why? Recall that the mentalist tradition, while founded in a firm separation between the mind and the world, tends to conflate them in practice. One aspect of this phenomenon is that mentalism generally blurs the difference between thought (in the head) and action (in the world). This is not because mentalist theorists consciously believe that thought and action can reasonably be conflated; instead, the conflation is a largely covert yet powerfully driven consequence of the practical logic of technical model-building within a mentalist discourse. As Chapter 3 demonstrated, the theories of Newell and Simon tended to elide the phenomenon of action by formulating instances of problem solving activity in mental terms. Thus, they interpreted activities that actually take place through complex interactions with scratch paper and chalkboards as manipulations of working memory (Agre 1993a, 1993b).

In *Plans and the Structure of Behavior,* Miller, Galanter, and Pribram interpreted the implications of GPS differently.[2] They made three proposals: that the behavior of organisms has a structure, that this structure is hierarchical, and that structured behavior results from the execution of Plans that have the same structure. The idea that structured action could be predetermined by mental processing, as we have seen, had been proposed by Lashley. The notion that mental structures could be hierarchical came from Newell and Simon's discussion of the organization of large problem spaces. Miller, Galanter, and Pribram combined these two ideas and developed a systematic set of speculations based on the resulting theory of Plans.[3]

Central to Miller, Galanter, and Pribram's procedure was the presumed interchangeability of two different structured descriptions of behavior. The first was the retrospective, external description that a theorist might use to record an organism's behavior. The second was the prospective, internal description that an organism might execute to give rise to that behavior. Without saying so explicitly, Miller, Galanter, and Pribram constantly move back and forth between these two perspectives. The Plan (the organism's internal description) gave rise to the observable behavior (as recorded in the theorist's external description) through execution, yet the concept of execution remains largely unexplicated. The assumption throughout is that execution is a simple, unproblematic matter, so that to exhibit some behavior one need simply *decide* to exhibit it. The actual performance of the action is, as it were, an afterthought.[4] Nor do Miller, Galanter, and Pribram offer a complete account of how the Plans themselves arise.

In the course of their mobile robot project of the early 1970s, Fikes, Hart, and Nilsson (1972a) drew together the GPS conception of problem solving and the Miller, Galanter, and Pribram conception of Plans into a powerful synthesis that defined planning research until the late 1980s.[5] In a technical tour de force, Fikes, Hart, and Nilsson programmed their robot to predict the consequences of various potential sequences of actions using automatic symbolic theorem proving. Using the vocabulary of GPS, they viewed this process as a matter of problem-space search: the problem space corresponded to the possible states of affairs in the robot's world; every operator corresponded to an action the robot might take; an operator had its effect by transforming one state into another state; and a problem was solved when a series of operators had been found that would transform the initial state to the goal state.[6] Then, once the problem solver found a sequence of operators that led from the initial state to the goal state, it gathered up these operators and, interpreting each as an action it could perform in the real physical environment, assembled them into a plan. A separate program could now "execute" this plan, thus carrying the problem solver's envisioned series of actions into effect.

For Fikes, Hart, and Nilsson, thought and action were closely related: not identical but isomorphic. Every structure and process had a foot in each realm: operators in the realm of thought corresponded to possible plan steps in the realm of action; logical deduction in the realm of thought corresponded to causal entailment in the realm of action; and so forth. The processes of thought and action followed parallel courses and produced analogous outcomes.

Thought and action were parallel in structure, but they were not equal in status. The device that did the thinking, the problem solver (called STRIPS), had a great deal more latitude to explore options and work out their consequences. The device that did the acting, the *executor* (called PLANEX), could omit steps that proved unnecessary or repeat steps that proved unsuccessful, but it could go beyond the plan itself only by giving up and restoring control to the problem solver, which would then begin its reasoning again from scratch.[7] (This mechanism will be discussed in the next section.) Though the executor could accommodate certain kinds of trouble, the assumption was that thought could be converted to action in fairly large hunks, large enough to encompass the solution of an entire problem posed to the robot. The role of the plan was to mediate this conversion.

Troubles with planning

The picture that emerges from this history portrays thought as simulated action and action as realized thought. On such a view, the idea that action results from the execution of plans is almost a tautology. An ambiguity has consequently grown up among the uses of the word "planning," which can refer either broadly to any reasoning about action or narrowly to the construction of a plan with the intent of executing it. In practice, the word (both the gerund "planning" and the verb "to plan") shifts freely between these two meanings. Of course, it is possible in principle that an agent might pursue goals, anticipate the future, learn from experience, and engage in complex forms of symbolic reasoning without constructing and executing plans. Yet within current AI discourse it is impossible to discuss alternatives to planning (i.e., narrowly construed as a doctrine) without keeping track of some subtle distinctions. The verb "to plan" has become so ambiguous that it is probably unsalvageable. And I will use the word "planning" in the most specific fashion, to refer to the process of constructing a plan with the intention of executing it. Used this way, the word is meaningful only within a large set of assumptions that it will be my purpose to question and replace.[8]

A further difficulty is that the noun "plan" itself takes on a number of meanings. In everyday life it refers to business plans and dinner plans, as part of a culture's ways of using representations of action as aids in organizing and coordinating activity. But uses of the term "plan" in AI discourse have been constrained by the necessity of providing some technical specification for the notion of execution. It is thus that "plan" has come by default to mean "computer program" and "execution" has come to be modeled on the operation of a programmed computer, even though few authors have explicitly embraced this understanding of the terms. Miller, Galanter, and Pribram offered the assimilation of plans to computer programs only as a conjecture. But they also denied that their version of the word "plan" was meant to correspond to the word's vernacular meaning, even though theirs was (like Newell and Simon's) a psychological theory and not (like Fikes, Hart, and Nilsson's) an engineering proposal. At the same time, other authors have asserted that recipes and other everyday plans are best viewed as defective computer programs.[9]

Critical discussion of the issues related to planning as a view of human action, then, demands a certain tolerance of ambiguity. Nonetheless, we should not overlook one point of considerable agreement among these authors and their successors: that the purpose of a plan is that its execution should achieve some preset goal. A particular instrumentalism is central to the project: the adoption of goals is distinguished from the construction and execution of plans to achieve them. The selection of goals has been a topic of AI research only insofar as those goals are instrumental to the achievement of some previously existing goal.[10] Newell and Simon establish goals for their experimental subjects in their capacity as scientists; Miller, Galanter, and Pribram, who emphasize goals less than the other authors, posit goals ad hoc in the course of their fictional scenarios; and Fikes, Hart, and Nilsson assign goals to the robots they have built.

The simplest interpretation of the distinction between planning and execution is to think of them as subserved by different mechanisms: a planner that takes a goal and produces a plan and an executor that carries that plan into effect, thereby achieving the goal. The division of labor is such that the planner is a relatively cerebral device, whereas the executor is capable only of managing the details of the execution process. The most straightforward elaboration of this story occurs when the executor runs into trouble or when another goal comes along, in which case the planner and executor alternate, with each sleeping while the other runs. This idea is called *interleaved planning* (Chien and Weissman 1975; Giralt, Chatila, and Vaisset 1984; McDermott 1978; Wilkins 1988).

An agent executing a plan runs obvious risks because of the likelihood that reality will not turn out exactly the way the planner imagined it would. The environment might change, actions might have unintended effects, margins for error might be exceeded, or other agents might fail to cooperate. Actions that seemed reasonable at planning time, in short, may not turn out to be advisable at execution time. The executor, therefore, must detect such conditions and return control to the planner. The only scheme in general use for this purpose is to abort execution if one of the plan's prescribed actions turns out to be detectably inapplicable, an idea called *execution monitoring* (Fikes 1971, 1982; Ghallab 1985; Munson 1971). But monitoring is only a mechanism for failing safely when things go wrong; it does not anticipate these contingencies or prepare for them. Indeed, such advance preparation will frequently be impracticable. When future states of the world are genuinely uncertain, detailed plan construction is probably a waste of time.[11]

The original exposition of planning by Miller, Galanter, and Pribram anticipated these issues, after a fashion. I have described these authors as asserting that behavior derives its structure from the execution of previously constructed Plans that have that same structure. But they also present a second account of the origins of action. This second account is conflated with the first, but the two accounts are different and represent distinct partial resolutions of an internal tension in their theory. On the second account there exists, instead of or in addition to the stored Plans, a mental structure called "the Plan." It can be difficult to distinguish the two notions, "Plans" and "the Plan." Both of them are hierarchical formal representations of action that possess the same structure as the behavior they cause. The difference is that *a* Plan is drawn from a library of Plans and executed as a whole, whereas *the* Plan is constructed as one goes along. For example, some of the higher levels of the hierarchy might be sketched out for days in advance, with the details at the lower levels left to be filled in as the time approaches. The Plan might be constructed as a sort of patchwork, with fragments of Plans being drawn from the library and stitched together into the Plan as it becomes possible to commit to executing them. The whole process is subject to the constraint that, at any given moment, the Plan must specify in concrete detail what the agent is to do next. If we assume that the agent does not throw away or recycle the portions of the Plan that it has already executed, we can imagine the agent trailing behind itself a complete hierarchical specification of all the behavior it has ever exhibited. As a retrospective document of the agent's behavior, the hierarchical Plan will bear no traces of the incremental process by which it had been assembled.[12]

Whether Miller, Galanter, and Pribram's two accounts of planning are compatible depends on what the theory is supposed to explain. Observe that the second account (incremental assembly of the Plan) is consistent with a mechanism that is never certain what it is going to do until the very moment on which it acts; this is the extreme case on which none of the hierarchy gets filled in until the last second. To the extent that the Plan is assembled incrementally and not selected from the canned Plans in a library, the explanation for the structured nature of the organism's behavior lies in the assembly process, which Miller, Galanter, and Pribram left almost entirely unexplicated. Yet throughout their book they offer hypotheses, following the first account, that the structure of a given form of behavior is explained by the execution of Plans that are plotted out in advance. Despite this logical instability, their work has been a powerful inspiration for planning research – exactly because, in its attempt to have things both ways, it provides the materials to assemble a plausible account of any particular phenomenon considered in isolation. Once one has begun a technical project along the lines of one account or the other, the text has many suggestions to offer, some of which can be assimilated immediately and others of which cannot. The result is a long tradition of projects that intelligently work out certain facets of Miller, Galanter, and Pribram's research program while also inadvertently displaying its internal tensions.

Let us briefly consider an example. In a justly influential paper, Hayes-Roth and Hayes-Roth (1979) report an empirical study of plan construction. They begin with the premise that activity is organized through the construction and execution of plans, and they propose to investigate the first of these two stages of processing. They provided experimental subjects with a map of a town and a list of chores to be performed, and they asked the subjects to explain their thinking as they assembled a plan for running errands around town. They observed that the people constructed their plans in an *opportunistic* and *incremental* way: their work was systematic in part, but it was also informed by chance observations that such-and-such an errand could be fitted into the schedule at such-and-such a place. Hayes-Roth and Hayes-Roth contrast their observations with the more common view that plan construction proceeds by filling out successive hierarchical levels, and they suggest some factors that might influence plan-construction strategies, with the opportunistic and hierarchical approaches regarded as extreme cases. What is striking about this study is the contradiction between its opening premise, that people act by constructing and executing plans, and the analysis they provide of the particular activity they investigate, namely talking through a hypothetical shopping plan when presented with a list of chores and a map of an unfamiliar town. The people are engaged in a complex activity, but this activity's organization is not well accounted for in terms of the planning model. What is more, Hayes-Roth and Hayes-Roth repeatedly describe their subjects as performing "mental simulation" even though the subjects are interacting with the instructions and the map – and even, at one point, receiving unsolicited help from the experimenter. As with Newell and Simon's early studies of theorem proving and cryptarithmetic, it is as though the stereotypically "cognitive" nature of this activity prevents it from being recognized as an embodied activity in the world. In reading the paper, it is hard to get any concrete sense of this activity; although the authors provide a detailed protocol of one subject constructing a complicated itinerary that he calculates down to five-minute intervals, they never say whether he writes this plan anywhere – much less where his gaze is directed at each step, whether he uses his hands to keep track of where he is, or how he relates the list of chores to the map. Instead, the whole protocol is narrated in the theoretical vocabulary (e.g., "level of abstraction") that will shortly be embodied in a computer model. And in that model, everything that the subject encountered as a paper artifact becomes an internal data structure.

The ambivalence within Miller, Galanter, and Pribram's theory of action reflects their failure to address adequately a central question: How is it that human activity can take account of the boundless variety of large and small contingencies that affect our everyday undertakings while still exhibiting an overall orderliness and coherence and remaining generally routine? In other words, how can flexible adaptation to specific situations be reconciled with the routine organization of activity? Each of Miller, Galanter, and Pribram's two theories addressed one term of this question: the incremental assembly of the Plan accounted for flexibility in the face of contingencies and the execution of preconstructed Plans accounted for routine organization. Neither theory accounts for both.

Planning and execution

The "planning view" of action, then, is not so much a definite doctrine as a discursive formation: a network of figurative associations and subtle ambiguities whose dissection has required sustained effort. Its practical logic lies below the surface of the page and is best understood in historical perspective. The work of Fikes, Hart, and Nilsson makes an excellent case study in the practical logic of technical work because they took the best original ideas of the cognitivist movement, tried to build robots that instantiated them, and employed good sense in interpreting their experiences.

The practical logic of Fikes, Hart, and Nilsson's experiences, as I have remarked, has its roots in an overt partition and a covert conflation between thought and action. The distinction between thought and action, of course, has deep historical roots. The scientific management movement, for example, projected these distinctions onto the structure of organizations: planning was the responsibility of an engineering department and execution was the responsibility of the line employees, who were provided with detailed instructions specifying every movement necessary for the performance of their jobs (Gilbreth 1921; Holmes 1938; F. Taylor 1911; cf. Montgomery 1984). Though the cognitivists inherited aspects of their technical orientation toward action from this tradition, their goal was not to attack or defend social ideas as such but rather to make psychological models or design robots that are capable of performing actual tasks. Although the strengths and weaknesses of the mentalist framework manifested themselves in their attempts to build things, these authors exhibited only a partial understanding of the problems they encountered and did not manage to transcend the intellectual framework within which these problems arose. One particularly striking moment in this history is found in one of the papers on PLANEX:

One of the novel elements introduced into artificial intelligence research by work on robots is the study of execution strategies and how they interact with planning activities. Since robot plans must ultimately be executed in the real world by a mechanical device, as opposed to being carried out in a mathematical space or by a simulator, consideration must be given by the executor to the possibility that operations in the plan may not accomplish what they were intended to, that data obtained from sensory devices may be inaccurate, and that mechanical tolerances may introduce errors as the plan is executed.

Many of these problems of plan execution would disappear if our system generated a whole new plan after each execution step. Obviously, such a strategy would be too costly, so we instead seek a plan execution scheme with the following properties:

1. When new information obtained during plan execution implies that some remaining portion of the plan need not be executed, the executor should recognize such information and omit the unneeded plan steps.
2. When execution of some portion of the plan fails to achieve the intended results, the executor should recognize the failure and either direct reexecution of some portion of the plan or, as a default, call for a replanning activity. (Fikes, Hart, and Nilsson 1972a: 268; emphasis added)

The untenability of mentalism emerges at this point in the STRIPS project as a technical difficulty. It is as if thought and action wished to intertwine themselves and were attempting to tear down the barriers that keep them apart – not to merge into one another, but to engage in a dance of give-and-take. The anthropomorphism of this analysis may seem overblown, yet its logic is clear: mentalism is not simply an inadequate description of people, but an untenable way of life for any creature in a world of any complexity. As the quotation demonstrates, these authors were aware of the form that an alternative might take, but they did not have any technology that was capable of implementing it. As a technical problem it is not a trivial matter. The STRIPS plan-construction algorithm involves a great deal of complicated symbolic reasoning. Yet an agent in the real world must take action frequently, ideally many times a second – even supposing that action should be conceived as a series of discrete "actions." Consequently, the designers of STRIPS (which built the plans) and PLANEX (which executed them) partitioned the necessary computational work between the two programs in a special way, maximizing the flexibility with which PLANEX acted while also permitting actions to be issued with sufficient frequency. In other words, PLANEX was the most intelligent mechanism its authors could devise that could choose its actions at a rapid pace.

The precise way in which the authors of STRIPS and PLANEX tried to reconcile the competing demands of intelligence and flexibility was extremely clever. In addition to supplying PLANEX with a plan, STRIPS also supplied it with a summary rationale for the plan, in the form of a *triangle table.* The triangle table records certain information about the relationships between the steps of the plan. Suppose the plan in question consists of three steps: A, B, and C. Each of these three steps has both *preconditions* and *effects.* The various actions produce their intended effects only when their preconditions are satisfied. In prescribing steps A, B, and C, STRIPS has performed some subtle analysis, satisfying itself that each action's preconditions will obtain when its turn comes to be executed. Some of these preconditions – the *initial conditions* – will simply be properties of the world as STRIPS has found it. Others will arise through the effects of previous actions. Since step A comes first, all of its preconditions must be initial conditions. Step B's preconditions can have two sources: the initial conditions and the effects of step A. Step C's preconditions, analogously, can have three sources: the initial conditions and the effects of both step A and step B. The robot's overall goal, finally, can be composed of conditions from *four* sources: the initial conditions and the effects of all three steps of the plan. Following J. S. Anderson and Farley (1988), we can think of each condition as having a *producer* and *consumers:* the producer of a condition is the plan step that causes that condition to become true (more precisely, the step whose execution allows us to be certain that the condition is true, regardless of whether it was true before); and the consumers of a condition are the plan steps whose execution requires that condition to be true.[13] (If the condition is true in the initial state, the producer is defined to be that initial state. Likewise, if a given condition is part of the goal state, that goal state is defined as one of its consumers.)

The producer–consumer relations behind a plan contain enough information to construct a formal proof that the plan will work, provided that the world really corresponds to the initial conditions and nothing goes wrong in executing the various steps. The purpose of the triangle table is to summarize these producer–consumer relations in a handy form. Let us consider an example. If one is in London and one's goal is to be in San Francisco, the plan might involve taking a train to Heathrow, a plane to Oakland, and a bus to San Francisco. Several conditions figure among the producer–consumer relations within this plan. For example, the condition of being at Heathrow is produced by the train trip and consumed by the plane flight; the condition of being in San Francisco is

CONSUMERS				
train to Heathrow	in London money			
plane to Oakland	passport ticket	in Heathrow		
bus to San Francisco	money		in Oakland	
end				in San Francisco
	start	train to Heathrow	plane to Oakland	bus to San Francisco

PRODUCERS

Figure 8.1. A triangle table for a trip from London to San Francisco.

produced by the bus trip and consumed by the goal state; and the condition of having one's wallet along is produced by the initial state and consumed by all three plan steps but not by the goal state. The plan's triangle table would record these facts and several more, as Figure 8.1 shows.

In a perfectly well-behaved world, the triangle table would be unnecessary; PLANEX could simply execute each plan step in sequence without paying any attention to the outside world. The world, though, has a tendency to depart from the ideal course that STRIPS predicts as it constructs its plans. PLANEX thus uses the triangle table to choose its next action in a way that is sensitive to the actual state of the world, as opposed to the state that STRIPS predicted. As the earlier quote indicates, this scheme permits PLANEX to omit some unnecessary actions (ones whose intended effects already happen to obtain) or repeat actions that did not manage to achieve their intended effects. PLANEX in effect rapidly recapitulates some of the reasoning that led STRIPS to construct the plan in the first place. But PLANEX does not recapitulate *all* of STRIPS's reasoning, since STRIPS has explored and discarded many possible courses of action that do not appear in the triangle table. If reality fails to correspond to STRIPS's predictions in any crucial way, PLANEX will find itself unable to justify *any* of the steps in its plan. In this case, it will simply give up and return control to STRIPS, which will start over again and make an entirely new plan.

The uncertainty and unpredictability of the robot's environment encouraged Fikes, Hart, and Nilsson to move as much of the robot's reasoning as possible into the executor. As a result, their understanding of the planner–executor relationship resembled in some interesting ways Descartes's understanding of the soul–body relationship. Descartes, like the Aristotelian philosophers before him, held that it was the soul that distinguished human beings from animals. What Descartes added was a radical distinction between body and soul wherein the body is governed by deterministic physical law and the soul possesses an acausal free will. Human bodies, roughly speaking, could do the things that animals could: breathe, digest, perceive (though not in a conscious way), exhibit fear and desire, engage in instinctual or reflex action, and so forth.[14] Both PLANEX and the Cartesian body, then, had a certain degree of autonomy, in whatever sense clockwork can have autonomy. The difference between PLANEX and the Cartesian body is that PLANEX is much more pliable: whereas the Cartesian body is a more or less fixed structure and even engages in sharp struggles against the commands of the soul, STRIPS simply programs a new triangle table into PLANEX every time a new policy seems indicated. Perhaps a more appropriate Cartesian metaphor would be to think of STRIPS as a toy designer and PLANEX as an assembly kit for windup animals. The important point is that the logic of mentalist technical practice led Fikes, Hart, and Nilsson, for technical reasons, to reproduce a partition of responsibilities broadly congruent with that envisioned by Descartes.

In calling for its plans to be executed, then, STRIPS entrusts the robot's well-being to a device that is capable of performing only a fraction of the reasoning that went into creating the plan. In particular it trusts this device to determine when STRIPS itself ought to resume operation. As I mentioned, this occurs when PLANEX encounters a situation for which its triangle table makes no provision. The ideal policy would be to cease execution any time the next prescribed action turns out to be irrational or suboptimal (as opposed to merely inapplicable) when the time comes. In practice, this might mean that the action in question is simply different from the action the planner would suggest were it to be run again. It follows that the ideal executor would be qualitatively as smart as its planner. Or, as we have seen, one might interleave planning and execution very rapidly, so that the executor returns control to the planner after every single action. It seems likely that many activities would require such extreme measures. Schemes that rely on the construction of plans for execution will operate poorly in a complicated or unpredictable world such as the world of everyday life. In such a world it will not be feasible to construct plans very far in advance; moreover, it will routinely be necessary to abort the execution of plans that begin to go awry. If contingency really is a central feature of the world of everyday life, computational ideas about action will need to be rethought.

The ultimate difficulty here is the mentalistic metaphor system of inside and outside. The exact nature of the difficulties with mentalist technical ideas provides evidence for this diagnosis. These difficulties cluster around the issue of contingency: a rigorous separation between inside and outside will be natural only if inside and outside have a strong tendency to remain coordinated. Whether inside and outside remain coordinated depends on the particular world and the particular activities within it: certain highly controlled worlds, like a factory floor or the simulated *microworlds* of most research on planning, are designed to facilitate strong forms of coordination between model and reality. The world of everyday life is not an utter chaos, of course, but neither does it resemble the simple mathematical ideals that mentalistically designed agents require. Contingency, in other words, is a ubiquitous phenomenon of everyday activities, even routine ones.

The contingency that prevails in ordinary environments does not defeat classical planning schemes in a straightforward, formally conclusive sense. Since the problem is one of inappropriate metaphors, trouble manifests not through specific technical impossibilities but rather through *patterns* of trouble. Further technical work can probably patch any given instance of trouble, but further troubles will always arise, perhaps unrelated formally but unified by the underlying system of metaphors.

As a result, the theorist faces a trade-off: an agent will be able to construct plans to the extent that someone (the agent itself or its designer) imposes constraints on the world itself. Chapman (1987) has formalized this trade-off in a theorem that relates the degree of expressive power in an agent's models of its world to the inherent computational difficulty of constructing plans to achieve goals in that world. The technical details of the theorem do not matter here, but the upshot is that plan construction is technically tractable in simple, deterministic worlds, but any nontrivial form of complexity or uncertainty in the world will require an impractical search. Results like this one are valuable, provided that they are interpreted at the level of underlying metaphors. The theorem itself presupposes the mentalist distinction between the construction and execution of plans, and it does not exhaust the enormous range of specific trade-offs that might be possible in worlds with particular kinds of beneficial structure.[15] Like any theorem, Chapman's result does not logically require the field of AI to doubt its metaphors or choose new ones.[16] It does, however, articulate a pattern of difficulty that is intrinsic to any mentalist project.

Improvisation

For these reasons, I propose that activity in worlds of realistic complexity is inherently a matter of *improvisation.* By "inherently" I mean that this is a necessary result, a property of the universe and not simply of a particular species of organism or a particular type of device. In particular, it is a *computational* result, one inherent in the physical realization of complex things. In vernacular usage, the word "improvisation" tends to imply a lack of concern for the future, but I mean to draw out a different implication of the word, namely, the continual dependence of action upon its circumstances. My contribution is not this idea itself, which is common enough in the social sciences,[17] but rather some suggestions about how it might be rendered in computational terms.

The cognitivist tradition tends to associate the theme of interaction with behaviorism. This is in part because, as we have seen, the cognitivist movement's founding documents argued against the complete determination of behavior by stimuli, shifting instead toward the opposite extreme of behavior as the execution of mental plans. If the behaviorist and cognitivist tendencies are viewed as thesis and antithesis, an opposing pair defined within the mentalist logic of inside and outside, the interactionist view might be viewed as their synthesis. As I have already explained, this synthesis arises by a particular reversal of priorities, transcending the mentalistic metaphors by taking as central what they treat as marginal (i.e., contingency and interaction) and taking as problematic what they treat as given (i.e., the idea that people have insides). The agent and the world, on this view, are understood not as radically separate but as dialectically interrelated and therefore difficult to understand except in terms of their interactions with one another.

Several authors in the social sciences have described important features of improvised action.[18] The idea that activity is improvised is implicit, for example, in the notion of a dialectical relationship between people and their environments. Lave (1988) has elaborated this idea in a study of everyday arithmetic practices. Observing ordinary people as they shopped in supermarkets or cooked dinner according to diet plans, Lave was led to oppose the view that cognition proceeds through the cycle envisioned by Newell and Simon: formalization of an abstract problem, solving the problem by reasoning within formal models, and interpreting the resulting solution in concrete terms. Instead, Lave found that the people in her study leaned heavily on their environments in conducting their reasoning. Moreover, both the problem and its solution were subject to continual reinterpretation as the process unfolded. The conventional distinctions did not bear any natural correspondence to these data.

Since the activity she observed consisted of the interaction of people with their environments, and since the people and their environments had both been deeply influenced by the surrounding culture, Lave argued that it was necessary to understand the activity on several levels. On the most microscopic level, each episode of activity arose through the mutual shaping of particular people and a particular environment.[19] On an intermediate level, grocery shopping and cooking take place in particular *arenas* which arise through large-scale social processes and serve in turn to shape individuals' actions. And on the highest level, a particular social structure both manifests and perpetuates itself through these organized interactions. Lave's dialectical view of activity is an instance of the overall theme of interactionism. In particular, a dialectical theory of activity will take care to distinguish between the theorist's objective view of a society's workings from individuals' subjective view.[20]

Suchman (1987) has investigated empirically the nature of plans and their role in situated action. Following the Wittgensteinian and ethnomethodological critique of representation (Garfinkel 1984 [1967]; Wittgenstein 1968 [1953]), Suchman observes that plans and actions are different sorts of things, that plans require interpretation in situ, and that the use of plans is a form of improvised action like any other. In particular, she insists that plans are not control structures that generate action; rather, they are resources for the fashioning and recounting of actions.

This view can seem only paradoxical from within the planning tradition, given the terminological ambiguities and slippages that I have outlined. If the execution of plans is a simple, mechanical process, plans and action correspond in a simple, systematic way. But in reality, the use of plans entails a considerable degree of preunderstanding of the activities and situations in question. The notion of following a plan, Suchman argues, is itself embedded in the local social setting. For example, when a plan actually regulates an activity, as in the case of formal office procedures, the role of the plan is not to *generate* the activity but rather to aid in organizing an *account* of the work as having been conducted according to plan. Anyone who actually followed the plan literalistically would not be doing their job (Suchman 1983).

Though these theoretical projects differ on many points, they are united in their goal of describing human activity in a post-Cartesian way. They give priority not to thought but to action; and they attempt to locate its meaning not solely in individuals but also in an encompassing yet contingent social order that individual episodes of action both presuppose and (usually) reproduce. In particular, these projects are critical of the attempt to mark out matters like planning and execution as modular faculties. This critique has a moral component, which arose in large part from the conflicts in the United States over scientific management. But this moral critique converges with the technical critique of the practical logic of planning research. The immense effort that STRIPS and PLANEX invested in manipulating plans provides a clue to the deeper shortcoming of the planning–execution distinction. A plan-construction device will necessarily be complicated, since it must engage in reasoning that anticipates a large space of possible futures and construct a plan that succeeds (or fails safely) in all of them. The plan itself, though, bears few marks of the process by which it was constructed. Even a fairly subtle change of circumstances (say, relative prices of commodities or the willingness of collaborators) can change a planner's output in ways that are difficult if not impossible to read off the plan itself. The plan is thus a precipitate of something much more complex.[21] A module boundary, such as the boundary between planning and execution, is always defined by the small amount of standardized information that passes across it, relative to the mass of possibly more heterogeneous processing that goes on within the modules. As an engineering practice, modularity has a robust practical logic whose workings are readily observed in the planning literature. Simply put, modularity trades off against efficiency: as a system is pressed to become more efficient, its modularity boundaries begin to break down; it will seem necessary for greater amounts of information to pass over the boundaries between modules, so that each module can be informed by more of the reasoning behind the information produced by the other. STRIPS and PLANEX reflected this tendency by recording certain components of the planner's reasoning, namely a summary of the producer–consumer relations that provides a correctness proof for the plan itself and for a certain space of variations (those involving repeated and omitted steps, as explained earlier). Any change in the world whose consequences are not captured by this summary will pose a specific danger to the agent: the plan will seem perfectly applicable even though it does not correspond to what the planner would produce, given the chance. The boundary between the construction and execution of plans is thus a hazard in environments in which contingency is the rule. A plan, on the conventional account, is nothing *but* the structure that is exchanged across this modularity boundary in a conventional planning system. Yet, as Suchman has demonstrated in her case studies, the plans that people use in everyday life play an entirely different role in organizing human activities.

A theory of activity, then, should give a central place to contingency and interaction. The role of computational theorizing is to account for the individual as a physically realized agent in contingently organized interactions with an environment, but conventional notions of computation make such an analysis difficult. The Cartesian roots of contemporary computational ideas obstruct the project of reconceiving "computation" in interactionist terms. Even more formidable is the disciplinary requirement that computational ideas actually be *realized* in working artifacts – computational models of activity. From this point of view, Cartesianism has the methodological virtue that individuals, be they people or insects or robots, are conceived in isolation. An artifact can exemplify Cartesian computational ideas without leaving the laboratory, without being socialized into a position in a social order, and indeed without taking on a bodily form at all.[22] But if activity is organized through engagement with a suitably structured environment, it will be necessary to supply computational models with such an environment, or at least with an adequate simulacrum.

9 Running arguments

From plans to arguments

Critical analysis is necessary and valuable, but the progress of intellectual work always turns out to be underlain by deep continuities. Technical work in particular will always pick up again where it left off, hopefully the wiser but nonetheless constrained by the great mass of established technique. Critics interrogating the existing techniques may discover a whole maze of questionable assumptions underneath them, but that discovery in itself does not make the techniques any easier to replace. I will not try to throw the existing techniques of AI out the window and start over; that would be impossible. Instead, I want to work through the practical logic of planning research, continuing to force its internal tensions to the surface as a means of clearing space for alternatives. My starting place is Fikes, Hart, and Nilsson's suggestion (quoted in Chapter 8) that the construction and execution of plans occur in rapid alternation. This suggestion is the reductio ad absurdum of the view that activity is organized through the construction and execution of plans. The absurdity has two levels. On a substantive level, the distinction between planning and execution becomes problematic; "planning" and "execution" become fancy names for "thinking" and "doing," which in turn become two dynamically interrelated aspects of the same process. On a technical level, the immense costs involved in constructing new plans are no longer amortized across a relatively long period of execution. Even without going to the extreme of constant alternation between planning and execution, Fikes, Hart, and Nilsson still felt the necessity of heroic measures for amortizing the costs of plan construction. These took the form of complex "editing" procedures that annotated and generalized plans, stored them in libraries, and facilitated their retrieval in future situations. Fikes, Hart, and Nilsson came to the brink of a discovery: thought and action do not occur in alternating stretches of time; instead, each is continual and intertwined with the other.

As thinking and acting intertwine, improvisation becomes a matter of *continually redeciding what to do.* This is the formulation of improvisation that I will assume here. The intuition behind it derives from Sartre, who holds in *Being and Nothingness* (1956) that the entirety of one's self must be viewed as a continual positive choice. Any other view, Sartre argues, would imply an innate, unchangeable human nature and would thus constitute an ethical abdication. This account is still Cartesian in the sense that each moment's action is brought about by an individual's discrete, deliberate choice, but this is still the only principled account of the relation between thought and action of which anyone can currently make any computational sense.[1] At the same time, it is an interactionist view in one important respect: individuals continually choose among options presented by the world around them. Action is not realized fantasy but engagement with reality. In particular, thought and action are not alternated in great dollops as on the planning view but are bound into a single, continuous phenomenon.

Further, I propose to understand improvisation as a *running argument* in which an agent decides what to do by conducting a continually updated argument among various alternatives. This is an engineering proposal in the case of robots and a scientific proposal in the case of human beings. It is surely not a final answer; instead, its value will lie in the process of pursuing its practical logic to a new and instructive reductio ad absurdum. Subsequent sections will explain running arguments in more detail. The argument that agents conduct with themselves will consider issues and options on several levels, from strategic matters in relationships and careers to the minute details of motor control. The argument might make reference to plans, maps, mnemonic devices, precedents from the actions of others, or anything else. Unanticipated issues can arise at any time, leading to new patterns of argument and possibly to changed courses of action. At any given moment, the complex of arguments leading to an agent's current actions is called its *argument structure.* As the agent interacts with its world, the argument structure will evolve. Its most fundamental aspects will remain relatively stable: wholesale shifts in one's personal identity and long-term strategies do occur, and sometimes abruptly, but they are rare. On the other hand, the detailed arguments that drive moment-to-moment interactions with the world will generally undergo constant change as a course of activity unfolds. The patterns of stability and change are not driven from some explicit principle, though, but are dynamic phenomena, that is, epiphenomena of the organized interactions between the agent (which is continually conducting a fresh argument about what it should be doing) and its world (whose states and relations to the agent may continually change).

Running arguments are altogether more flexible than conventional planning techniques. The handoff from a planner to its executor reduces the agent to a simple automaton, cognizant of only a few issues of local and literal relevance to the particular plan being executed. For a running argument, by contrast, everything is at issue all the time. An agent using running arguments improvises, taking advantage of opportunities and responding sensibly to contingencies. Fikes, Hart, and Nilsson rejected this approach on computational grounds. But the continual reassessment of the reasoning behind one's actions might indeed be computationally practical, provided that the agent maintains dependencies on all of its reasoning. If the agent and its environment are such that everyday life is nearly routine, new arguments will spontaneously assemble themselves from moment to moment, built up from fragments of reasoning that were novel at one time or another.

The value of improvisation becomes clearest in those situations when a system like Fikes, Hart, and Nilsson's pulls up short, unable to carry on executing its constructed plan. When trouble arises, it is critical to know what you are doing. When an executor runs into trouble, it can only hand control back to the planner. Imagine, though, the plight of a planner suddenly awakened to discover spilled milk, skidding tires, burned fingers, or angry people. Having little idea how any of this came about, it must reconstruct what it was trying to do, interpret the newfound damage or danger, assess its consequences for the ongoing project, and make a new plan that resolves the trouble and gets the project back under way. This would be much easier if it had been awake all along. When the executor carried out the first step in the plan, the planner's rationale was fairly likely to have corresponded to reality. But as the executor did its work, simulation and reality drifted apart, until finally some symptom of this divergence came to the executor's attention. If every single action could have been produced by its own fresh reasoning-through of the issues and options, the agent would have been less likely to lose its synchronization with reality. Perfect correspondence is impossible, of course, but an agent constantly engaged with its world will be able to make a continual reassessment of its course of action, steering whenever possible by reality as experienced and not just as imagined.

On this account, the dynamics of an agent's interactions with its environment will have two intertwined components: the evolving configuration of the agent's argument structure and the evolving relationship between the agent and its world. Though the metaphors of mentalism might suggest investigating each of these dynamic phenomena in isolation from the other, little can be gained in this way. One might, for example, record patterns of activity in a dependency network, as if through a brain scan, thereby yielding a temporal record of "active regions." Likewise, one might record the serial order of an agent's actions, thereby yielding a temporal record of "behavior." Both of these records would be interesting, and of some value, but neither of them is of much use on its own. The dialectical relation between agents and their worlds does not require us to stop speaking of "agents" and "environments," but it does require us to consider them in terms of their interactions with one another. Furthermore, when the agents and worlds in question are at all complex, it will probably be equally useless for empirical study to "control" one term or the other (e.g., by placing subjects in laboratory environments), inasmuch as these agents and worlds have arisen through a complex process of mutual adaptation. A fish out of water makes little sense.

It is important to distinguish two kinds of reasons, negative and positive, for preferring my own proposal to any based on the construction and execution of plans. The negative reasons concern all of the negatively defined properties of the world of everyday life – uncertainty, unpredictability, complexity, change, resource limitations, and so on – that frustrate the design of algorithms for constructing correct plans. During the 1980s, AI research attempted to frame such negative phenomena as problems seeking technical solutions – for example, "planning in uncertain, unpredictable, or changing environments" (Hendler 1990).[2] It turns out, however, that these negative concepts are hard to analyze in the abstract; they are broad families of phenomena that become manifest in different ways within each technical project. Reifying such negative phenomena as technical problems in their own right tends to reinforce the Cartesian standoff between a tractable mental Inside and an untrustworthy Outside. The world of everyday life does, of course, include dangers.

But a technical characterization of the world in negative terms has the effect of giving danger an ontological status that it does not deserve. To make explanatory accounts of human psychology, or principled methodologies for the design of robots, we need *positive* characterizations of why everyday life ought to be *possible*. Some of these might include our socialization into cultures, the tendency of physical things to sit still when nobody is disturbing them, and the useful properties of artifacts. In short, an account of machinery must be correlated with an account of the dynamics of everyday life.

The point is worth recapitulating in relation to the theme of anticipation. Plans are constructed, on the classical account, by searching through a space of possible futures, and the correctness of the resulting plans is founded on the accuracy with which this search anticipates the future. If the future unfolds along a line that mistakenly went unexplored, or if something was amiss with one's whole model of the world, then the nearly blind execution of the resulting plan may result in disaster. It is often observed, though, that such thorough anticipation of the future is quite impossible. The world, after all, is a complicated place; all manner of unanticipated contingencies can intervene in our activities at any moment. The problem is particularly severe when things happen in the world that are not entirely under our own control.[3] An ordinary conversation, for instance, has a branching factor beyond calculation. What is the proper response to this negative observation? One might maintain the planning framework and accept the inevitability of some errors. Or one might continue to explore the space of trade-offs that arises as various sources of knowledge are taken into account. Perhaps the right balance can be found after all, at least for particular applications. But the approach I would suggest is investigating why the success of action is not entirely conditional on the accuracy of anticipation. Let the world simulate itself, and view the organization and coherence of activity as an emergent property of interactions between individuals and worlds that are (for various reasons, from heredity to culture to idiosyncratic habit) adapted to one another.

Some examples might make the point more intuitive. Here are some things it would rarely be worth trying to anticipate:

- Where the chalk is located on the blackboard's chalk tray
- How the aspirin tablets are arranged in the aspirin bottle

- Whether any mail has come today
- Where a free seat can be found in the subway
- How all the dirty dishes in the kitchen will be arranged when it is time to wash up
- How many bottles will accumulate in the recycling bin this month
- Which side of a record will be up when you remove it from its sleeve
- Whether the record will need cleaning before you play it
- How the spatula will be tangled among the gadgets in the gadget drawer
- In what order the shirts are hung in your closet
- Whether you will have to open another box of cereal this morning
- How many pennies you have in your pocket
- How the other pedestrians will be distributed along the sidewalk
- Which slots in a half-full egg carton have the eggs in them
- When your watch battery will start running down

Here are some observations on this list, including some dynamic phenomena deserving further description and explanation:

- These things might be worth anticipating in another culture: perhaps a penny is a great deal of money, family consensus is needed to open another food package, or shirts are worn in a certain order.
- For every entry on this list, one can plausibly imagine circumstances under which it would be worth trying to anticipating it in any culture.
- In particular, whens and how-manys are often worth anticipating if they are going to be grossly outside their usual range. We have a good sense of normal ranges even though they are often hard to define.
- Some of these can be anticipated by taking the effort to "keep track." Keeping track of things is usually difficult.
- Things that are not worth anticipating are often hard to remember afterward. Most of them matter so briefly and are accommodated so easily that they make little impression.

- We are rarely aware of explicitly declining to anticipate something. More likely we try to anticipate only what we think we have to.
- Equipment that nobody else uses will be easier to anticipate. People who live alone often know exactly how the cereal boxes in their cupboards are arranged.
- Sometimes you will find yourself anticipating things like these after using the same item of equipment for the same purpose under the same conditions many times.
- Little is lost in trying to anticipate these things. Many regular subway riders have superstitious beliefs about where free seats will be found.

In short, "anticipation" is not a unitary phenomenon. Difficulties of anticipation are always difficulties of anticipating *something,* some particular complicated circumstance of everyday life.

Argument and centralization

In speaking of a "running argument," I am obviously using the word "argument" in some special way. My use of the word has only a loose and metaphorical relationship to its vernacular uses. I do not intend these arguments to model everyday quarrels or philosophical disputes. I will provide no formal definition of arguments.[4] Instead, I will work out the ideas in the context of the system I described in Chapters 6 and 7, in the form of a discipline for writing rules to build systems that fit with certain dynamic ideas.

The preceding section has already sketched a few ideas about arguments. From moment to moment, it suggested, an agent conducts an argument with itself. On each next moment, this arguing arrives at some conclusion about what actions to take. The structure of this agent's argument evolves by incremental changes. These changes ought to be small when the agent's situation is changing slowly; they should be large only in the relatively rare moments when the situation changes drastically in its implications for the agent's reasoning, whether through the agent's own actions or for some other reason. Argument structures might look like the diagrams logicians draw to trace how some conclusion might be justified from certain premises, but nothing about the agent's dependency network itself will enforce any of the local, formal standards of admissible inference found in systems of formal logic.

My ideas about arguments descend from those of Doyle (1980). Doyle's concern, unlike mine, was to make a model of reflective thought, that is, the sort of thing you do when you struggle with a big decision. Doyle views thought as a species of action, so that deciding what to think is like deciding what to do.[5] Decisions about action arise through arguments. Within one of these arguments, any active component of the agent's machinery can offer suggestions about what to do and adduce arguments about why. When disagreements arise, choosing among the arguments is itself a decision to be made by argumentation, recursively: once again, each component may adduce arguments as to why its arguments are better than the others.[6] Concretely, "argumentation" names a style of rule-language programming. This style of programming depends on the other-things-being-equal control structure found in the Life rule language described in Chapter 7. Recall that this scheme contrasts with the deductive semantics of logic programming languages like Prolog, in which the right-hand side of a rule is a proposition to be unconditionally believed as soon as the rule manages to fire. It also contrasts with an imperative semantics, whereby the right-hand side is an action to be unconditionally taken as soon as the rule manages to fire. Before I go into detail, consider the following cartoon example:

```
        (propose (hold-up main-st-bank))
```

Contradictory arguments are put forward:

```
(propose (support (hold-up main-st-bank) money-in-it))
(propose (object (hold-up main-st-bank) it-would-be-wrong))
```

The first argument encounters no objections, so it is accepted:

```
        (take (support (hold-up main-st-bank) money-in-it))
```

But some part of the system considers the second argument inferior and proposes that it be considered so:

```
(propose (object (object (hold-up main-st-bank)
                         it-would-be-wrong)
                 prefer-practical-to-moral-arguments))
```

There being no objections to that line of argument, it is accepted:

```
(take (object (object (hold-up main-st-bank)
                      it-would-be-wrong)
              prefer-practical-to-moral-arguments))
```

Consequently, the moral argument against robbing the bank is rejected:

```
        (blocked (object (hold-up main-st-bank)
                         it-would-be-wrong))
```

Since no other objections are outstanding, the motion stands:

```
        (take (hold-up main-st-bank))
```

Argumentation, then, involves the adducing of arguments and counterarguments concerning a proposal for action. Because each argument and counterargument is itself an action, the argumentation process itself is a topic for argumentation. The system approaches the problem of deciding between conflicting arguments in just the same way that it approaches any problem in the world. The method has some useful properties:

- Any reason for action is *defeasible*, meaning that it might be overridden if there are good reasons to do so.
- The decision process is *additive*, meaning that all rules can contribute to the reasoning in a uniform way, by contributing arguments.
- Individual decision processes are automatically converted into dependency networks. Thus a complex argument structure can be used many times a second. In particular, an argument's conclusions will stay IN as long as its premises stay IN.
- Because an argument recorded in the dependency network will be recapitulated whenever its premises are satisfied, it will automatically be carried over to analogous future situations.
- An agent's patterns of reasoning can be modified without rearranging any of its existing dependency-network circuitry. Given a new argument explaining why a given action is a mistake in a given circumstance, objections will be raised to that action in appropriate situations forever afterward.

All of the subsequent rules will refer to a version of *blocks world* in which a simulated agent moves blocks around with its hand. Blocks world originated in early AI projects at MIT (Winston 1975). At first it was intended for computational vision research – the blocks were actually made of wood and digital video cameras were aimed at them – but the wooden blocks soon gave way to formal abstractions. Among the earliest blocks-world projects was Sussman's ambitious model of procedural learning (1975). The one project that attempted to simulate in software the physical properties of actual wooden blocks was that of Fahlman (1974). Through these projects, the blocks world became for many years the benchmark for research on automatic plan construction. Although this line of research has led to some valuable understandings of the process of plan construction, Chapter 10 will conclude with some critical comments about the lingering influence of blocks world on computational research on action.

The rules in a running argument do not prescribe actions. Instead, they simply submit proposals for action, subject to the subsequent arguing back and forth. One rule might propose that the agent lift its hand. Another might propose placing its hand on block B. Another might propose moving left. As all of this arguing gets converted to circuitry, we can imagine various regions of the network arguing with one another. One region might have proposed moving left and another might have proposed moving right. Since these proposals conflict, the one region might raise an objection to moving left, perhaps on the grounds that doing so would knock over the tower. The other might respond, objecting that it would be unwise to move right because getting to the objective by moving right would take too long. A third region of the network might then step in and object to the latter objection on the grounds that it is weaker than the former. In general, rules can propose objections to objections to objections to an arbitrary depth. It is up to the programmer to ensure that the system neither deadlocks (by rejecting all possible actions) nor spins out of control (by generating an infinite regress of arguments about arguments) nor attempts to perform conflicting actions. (It is also up to the programmer to formulate a consistent ontology of actions, reasons, etc., and to express these in a real representation. All of the examples in this chapter use cartoon representations for expository purposes, though none of the representations in the system I have actually built are very convincing either.)

Despite the anthropomorphism, all of this proposing and objecting is implemented by ordinary rules. The rule system itself does not have any special knowledge of proposals and objections; they are just list structures in a database. To propose an action, assert

```
(propose action)
```

To object to an action, propose objecting to the action on some grounds:

```
(propose (object action reason))
```

An objection, then, is actually the *proposal* of an objection. The whole process pivots around a rule that says that any proposed action is taken unless some objection is sustained against it:

```
R1: (if (propose ?action)
        (unless (take (object ?action ?reason))
          (take ?action)))
```

Observe that the UNLESS rule here has an unbound variable, namely `?reason`. Even if R1 fires once for a given binding of `?action`, and even if that action actually gets taken, someday some new objection might come along to defeat the proposal. If this happens, the action will no longer be taken. If the action is currently under way, it will stop. If the proposal to perform the action is asserted again and the objection is still in force, the proposal will not be adopted. The new objection will amend the AND-NOT gate in the dependency network corresponding to this UNLESS rule, adding a new inverted input to the gate.

Here are some examples of argument in the blocks world. Suppose a rule proposes that the agent move its hand to the left:

```
A2: (propose (move hand left))
```

When this proposition comes IN, R1 will fire, producing the following:

```
R3: (unless (take (object (move hand left) ?reason))
        (take (move hand left)))
```

In other words, move the hand left unless some argument to the contrary is accepted. The UNLESS rule, R3, will generate a nonmonotonic dependency, an AND-NOT gate. For the moment, this gate will have no inverted inputs since no objections have ever been raised to that proposal. The output of that gate will be

```
A4: (take (move hand left))
```

But suppose that some rule files an objection:

```
A5: (propose (object (move hand left)
                     (would bump tower)))
```

Now something more complicated happens. As it is a proposal like any other, rule R1 fires a second time, producing a rule that will accept this objection unless some objection is sustained against it in turn:

```
R6: (unless
        (take (object (object (move hand left)
                              (would bump tower))
                      ?reason))
      (take (object (move hand left)
                    (would bump tower))))
```

This new UNLESS rule, R6, now checks for second-order objections. If none are present, then the UNLESS rule will license its conclusion:

```
A7: (take (object (move hand left)
                  (would bump tower)))
```

This sustained objection will now attract the attention of the original UNLESS rule of a moment ago, R3. Thus the proposal of moving the hand left will not be adopted. Once again, R3 has most likely already fired, creating an AND-NOT gate whose output is A4. If so, this gate will now receive an additional inverted input, namely A7. And the agent will not move its hand to the left.

The proposed action in this example, moving the hand left, happens to be one of the system's primitive actions. At the end of every clock cycle, the motor system decides whether to move left by checking whether A4 is IN or OUT. These conventions about proposals and objections apply equally well to compound or abstract actions. In each case, when the system adopts some proposal, that adoption is good only for the current clock cycle. If some compound action covers many clock cycles, it must be proposed, argued for, and adopted on every one of those cycles. This is not a great computational burden as long as all of this arguing, at least after the first cycle, takes place through the propagation of binary values through the dependency network and not through the firing of new rules. In other words, the system takes its actions only so long as they are supported by argument.

Rule R1 is special in that it explicitly relates proposals and objections to actions. All the other rules propose actions, raise objections, and adduce reasons pro and con. In practice one programs with a collection of rules for expressing complex ideas about priorities, weighing of arguments, decomposing compound actions, and proposing alternative and interpolated actions. The details of these methods of argumentation are not especially original or general, nor do they bear on the theoretical issues at hand.

The notion of argumentation could potentially be construed in two different ways, according to who has the burden of proof in putting forward their argument. A system could have an outer loop that says, "Decide what to do then do it." That method poses the positive task of choosing some particular action to take, one action at a time. The running argument system takes a different, decentralized view. Any patch of dependency network can make proposals and the proposed actions are taken by default. Rather than having to argue positively for an action, a rule might just propose jumping off a cliff, and if no other rule offers any objections then the conclusion will simply be to jump. All through the network, proposals are being made, arguments are being conducted, and objections and supporting evidence are being offered.

The point of this decentralized style of programming is that rules do not have to offer guarantees. If a rule proposes an action without ironclad guarantees that it is the best thing to do, that information can be embodied in separate rules that raise objections in appropriate situations (Minsky 1980, 1985). As a result, the rules can address prototypical cases and leave the endless enumeration of exceptional cases for later. As a practical engineering matter, new rules get written when the designer observes the system making a mistake, going into a loop, floating off into space, or seizing up. After looking at its arguments, one can formulate a new rule objecting to the erroneous proposal and perhaps proposing an alternative. The particular system I implemented grew to employ several dozen forms of argument about blocks world over a period of several months. The system is still clumsy about circumventing obstacles and working in tight spaces. But it does engage in interactions of some complexity, as the next chapter demonstrates.

Here is an example rule from this system:

```
(if (and (propose (try (grasp ?x)))
         (on hand ?x)
         (on hand ?y))
```

```
        (if-shown (neq ?x ?y)
          (suggest (prefer-option
                       (try (center-on hand ?x))
                       (try (grasp ?x)))
                     (avoid-unnecessary-grabbing ?y)))))
```

This rule uses the and and `if-shown` constructs described in Chapter 7. In English it says, "If we have proposed grasping x and the hand is on two different objects x and y then propose postponing the grasping operation until we have had a chance to center the hand on x." The `suggest` form combines making a proposal with proposing a reason to support it:

```
      (if (suggest ?action ?reason)
      (propose ?action)
      (propose (support ?action ?reason)))
```

If nobody raises any objections to preferring centering to grabbing, the system will adopt that preference. Taking this action will then offer support for centering and raise a concomitant objection to grabbing:

```
(if (take (prefer-option ?better-action ?worse-action))
  (propose (object ?worse-action
                   (preferable ?better-action)))
  (suggest ?better-action
           (preferable-to ?worse-action)))
```

This rule is part of the system's domain-independent knowledge about arguments. Observe how the argument about whether to take the action of grabbing x has spawned another argument about whether to prefer another action instead. Only once the system has resolved this second, inner argument can it resolve the first, outer argument.

Centering the hand is sometimes a bad idea, though. If block x already has a block on it, the system should not push it all the way off of x without a good reason:

```
      (if (propose (support
                     (prefer-option
                        (try (center-on hand ?x))
                        (try (grasp ?x)))
                     (avoid-unnecessary-grabbing ?y)))
```

```
(if-shown (and (on ?z ?x) (neq ?z hand))
  (propose (object (prefer-option
                     (try (center-on hand ?x))
                     (try (grasp ?x)))
                   (centering-would-shove ?z)))))
```

This rule is simpler than it looks. It reacts to the proposal made by the `suggest` form a couple rules back, checks whether something besides the hand is resting on `x`, and if so, it objects to the proposal of preferring centering the hand on `x` to grasping it immediately. If no other arguments are put forward, this objection will cause the system to decline to prefer centering to grabbing. Lacking any other arguments, the system will go ahead with its original proposal of grabbing `x`.

Both Doyle's argumentation scheme and the one I have described have much in common with Laird and Newell's notion of *universal subgoaling* (1983). Laird, Rosenbloom, and Newell have demonstrated universal subgoaling in the context of the SOAR architecture (1986). Universal subgoaling is roughly the idea that any decision an agent makes can become the topic of general reasoning. In the context of the SOAR architecture, this general reasoning takes the form of search in a problem space; uncertainties about where the search should proceed lead to the creation of a subgoal. This new subgoal itself becomes the object of a problem space, as if the system had called itself recursively. Indeed, the system maintains a stack of active goals and only considers a single goal at a time. This is not a severe restriction since SOAR has a scheme analogous to dependency maintenance, called chunking, for summarizing problem solutions (see Chapter 10). One difference between SOAR's universal subgoaling and the running argument system's argumentation scheme is that the former is an explicit part of the architecture whereas the latter is a programming convention, instances of which are converted into dependency network structure.

How running arguments work

Having described how arguments work, it is finally possible to describe the architecture of running arguments. I have built a computer program, RA, that employs this architecture. On its coarsest level, the RA agent architecture is wholly consistent with conventional proposals from cognitive science. (Subsequent chapters will complicate this picture.) Using the vocabulary of Fodor (1983), this architecture consists of a *central system* and a set of modular *peripheral systems* (collectively called the *periphery*), comprising *input systems* such as the visual modules described by Marr (1982) and *output systems* for low-level motor control.[7] A rule system and dependency system working together in the manner described in Chapter 7 form RA's central system. RA interacts with a blocks world. This interaction is simulated: some of the propositions in the database serve as inputs *from* the periphery and others serve as outputs *to* the periphery. RA proceeds through a cycle, as follows:

1. The world simulation updates itself.
2. The periphery computes new values for the central system's inputs, which represent perceptual information.
3. The periphery compares the new input values with the old ones. Any newly IN input is declared a premise; any newly OUT input is declared no longer a premise.
4. The central system propagates dependency values and runs rules. Both the dependency system and rule system continue to run until they have both settled.
5. The periphery inspects the values of the central system's outputs, which represent motor commands.
6. The periphery and world simulation together arrive at a set of proprioceptive propositions (judgments about the success or failure of the agent's primitive actions) and a set of motor effects (the immediate physical consequences of the agent's actions).
7. The world simulation updates itself again, and so on ad infinitum.

Since this "world" is simulated, the peripheral systems do not manipulate actual images or mechanical limbs. Instead, the various modules work together to simulate the effect of an agent getting about in a world.[8] This is a common practice in computational modeling because of the great difficulty of building real visual and motor control systems.

The overall picture, then, is that an agent is interacting moment by moment with an environment, and its responses to successive situations are determined by a large, complex symbolic reasoning system. At the beginning of its operation, and as it encounters any qualitatively new situation, it must perform a considerable amount of novel reasoning. But as it encounters a range of situations, it builds up a combinational logic circuit that obtains the same effect as the symbolic reasoner but with much greater efficiency. Eventually, this circuit will do the vast majority of the work, with the reasoner being invoked only for those bits and pieces that are truly novel.

This picture has some obvious shortcomings. It says nothing about memory, for example, inasmuch as a combinational logic circuit has no internal state. (The peripheral systems may have internal state, though.) Nor does it include language, social interaction, or a variety of other things. Instead, as I explained at the outset of this chapter, it is a means for discovering what happens when some conventional technology is applied to a computational model of improvised activity.

Before proceeding to some demonstrations, let me summarize, trying to make the picture intuitive. A running argument, once again, is a continually evolving argument structure. Proposals, objections, and reasons come and go over time. The argument structure "evolves" in the sense that little of it typically changes from one clock cycle to the next. A new objection might arise as the agent encounters an unexpected condition, but then it might be overruled, leaving the agent to carry on as before. An opportunity might arise to pursue some task in a straightforward way, leading to a proposal that is adopted and acted upon and that then evaporates once the task has been completed. Every once in a while there comes a big change, such as in an emergency or when the agent finishes with one large task and moves on to another. But usually nothing at all changes past the finest details of perception and motor control. In general, the argument structure changes to the extent that something meaningfully different is happening. That, at least, is the idea.

The combinational logic that eventually implements these arguments is not a glamorous technology, and its virtues are easily forgotten. Imagine a large sheet of circuitry, with inputs on the left and outputs on the right. All of this circuitry will operate continually, working to maintain some relationship between its inputs and its outputs. Every gate is always ready to change its state if its output ought to have a different value. An extraordinary amount of computation is effectively happening all the time, and the agent's actions are always based on a fresh analysis of what it ought to be doing. An executor running a plan, by contrast, takes a given action because it has reached step *n* in its current plan. Suppose, for example, the executor is moving the agent's hand to the left. The agent once understood why it should be moving its hand left just now, back when it was constructing the plan, but that understanding is located in another module if it was saved at all. If those reasons no longer apply, the executor will proceed anyway unless it is impossible to do so. But a system based on combinational logic will take unexpected conditions in stride, meeting the world's contingency halfway. If the agent has adopted some subgoal, it knows why. If a given subgoal is no longer a useful means to an end, the reasoning behind it will become invalid and the agent will take some other tack instead. The use of dependencies means that only the obsolete portion of the agent's reasoning will have to be redone. Simmons (1992) has described the use of dependencies in reconstructing plans to accommodate new information about the environment. But running arguments dispense with the distinction between planning and execution altogether. The agent reasons about what to do *now* and then does it. This reasoning might involve the construction of plans or it might not. The important point is that it is this reasoning itself that is directly driving the agent's actions, not a plan structure that summarizes some of the conclusions of that reasoning.

The RA architecture offers a technical account of the notion of continually redeciding what to do. The system accepts a new set of perceptual inputs and delivers a new set of perceptual outputs on a rapid clock. A rapid clock is not literally "continual," but it is a reasonable approximation. The system sometimes needs to run some rules and thereby create some new circuitry, but only what is both novel and necessary for the argument of the moment. If the agent's activity, once it has settled into a pattern, is almost entirely routine, then it should require few new rules to be run. The scheme has its faults, which the next chapter will discuss, but at least it provides one technical rendering of the notion of improvisation.

Running arguments are part of an account of routine activity. They are not an account of the origins of new lines of reasoning. From the point of view of existing computational research on action, this focus on routine activity can be hard to accept because of the emphasis on novelty in AI, whether through plan construction, problem solving, or learning. This emphasis on novelty is ingrained in technical language. If, for example, one proposes an AI research project on "making breakfast," one will be understood to mean something like, "making breakfast, never having made breakfast before, doing it without any help, and getting it right the first time," rather than "making breakfast, given that you have done it enough that you can do it routinely." Given the customary distinction between (elaborate, interesting) plan construction and (simple, mechanical) plan execution, it is hard to imagine how this can be an interesting topic for study. If no new cognition is going on, it must simply be executing a compiled plan. This argument offers a choice between equally unworkable alternatives. The point here, however, is not to choose among the existing positions but to get beyond them.

10 Experiments with running arguments

Motivation

This chapter demonstrates RA, the computer program introduced in Chapter 9 that illustrates the concept of running arguments. RA has three motivations, which might be reduced to slogans as follows:

1. *It is best to know what you're doing.* Plan execution – in the conventional sense, where plans are similar to computer programs and execution is a simple, mechanical process – is inflexible because individual actions are derived from the symbols in a plan, not from an understanding of the current situation. The device that constructed the plan once had a hypothetical understanding of why the prescribed action might turn out to be the right one, but that understanding is long gone. Flexible action in a world of contingency relies on an understanding of the current situation and its consequences.
2. *You're continually redeciding what to do.* Decisions about action typically depend on a large number of implicit or explicit premises about both the world and yourself. Since any one of those premises might change, it is important to keep your reasoning up to date. Each moment's actions should be based, to the greatest extent possible, on a fresh reasoning-through of the current situation.
3. *All activity is mostly routine.* Almost everything you do during the day is something you have done before. This is not to say that you switch back and forth between two modes, one for routine situations and one for the occasional novel situation. Even when something novel is happening, the vast majority of what you are doing is routine.

As a matter of computational modeling, all of this is more easily said than done. This chapter explains how RA instantiates these three ideals. In what way does the system know what it is doing? The phrase "deciding what to do" usually suggests a process that takes time, so how can the system be redoing it continually? How can the system determine which small portion of its reasoning needs to be conducted by nonroutine mechanisms?

Let us briefly review RA's design. RA engages in a rapid interaction with its simulated world. From the outside, RA is a fairly standard kind of rule system, already described in detail in Chapter 7. When the left-hand side of a rule matches something in the database, the right-hand side is instantiated with the appropriate variable bindings and the resulting proposition is asserted in the database, perhaps causing other rules to fire in turn. The rules themselves are specified by perhaps forty pages of code. Some of the rules effectively extend the Life rule language in various ways. Some encode domain-independent ideas about actions, plans, evidence, issues, and arguments. Some encode stereotyped forms of domain-specific inference, such as the transitivity of "above" (if x is above y and y is above z then x is above z). The most interesting rules concern domain-specific strategies and tactics and the arguments by which the system selects the applicable ones and chooses among them in particular situations.

The system behaves as if the entire set of rules ran to completion, forward-chaining until nothing was left to run, on every clock cycle. After the rules run, certain propositions in the database indicate which actions the agent intends to take on this cycle. Running all of those rules on every cycle, however, would be far too slow. Consequently, the system accelerates its operation by accumulating dependencies. Dependencies are helpful because life is mostly routine. Storing dependencies is a good investment because so much of what you do is something you are likely to do again. Furthermore, dependencies permit even a fairly general decision mechanism to operate in real time because so much of what you are doing at any given time is something you have done before. When a rule fires, the system builds a dependency record stating that a certain rule and a certain trigger lead to a certain conclusion. That rule need never fire on that trigger again. As Chapters 6 and 7 have explained, a dependency network can be realized in a combinational logic circuit. Each proposition and rule corresponds to an electrical node in this circuit, and each dependency record corresponds to a logic gate: an AND gate for IF rules and an AND-NOT gate for UNLESS rules.

This scheme offers technical accounts of the three slogans presented earlier. As the system runs, it accumulates a dependency network. If the system gets into a routine way of life, it should be able to run almost entirely out of this network. The system knows what it is doing in the sense that it effectively reasons from scratch rather than following a prespecified plan. Combinational circuitry is parallel and fast, so this reasoning takes little effort. The system is continually redeciding what to do in the sense that it produces a fresh set of decisions about action many times a second. Finally, due to the algorithms described in Chapter 7, the system can rapidly identify the novel aspects of each situation, so that the necessary rules can begin running immediately. Even if a given cycle's decision is effectively based on hundreds of rules, the system pays the price only of these few novel rule firings.

Whether RA actually does justice to its motivating slogans is an empirical question. Since no two situations in life are identical, the system's success will turn on whether the dependency system really transfers the agent's lines of reasoning to appropriate future situations. Intuitively, if the current situation is 95 percent similar to something we have seen before, the system ought to be reusing 95 percent of what it figured out before. As a detailed technical matter, does the system satisfy this ideal? That is what this chapter's demonstrations are about.

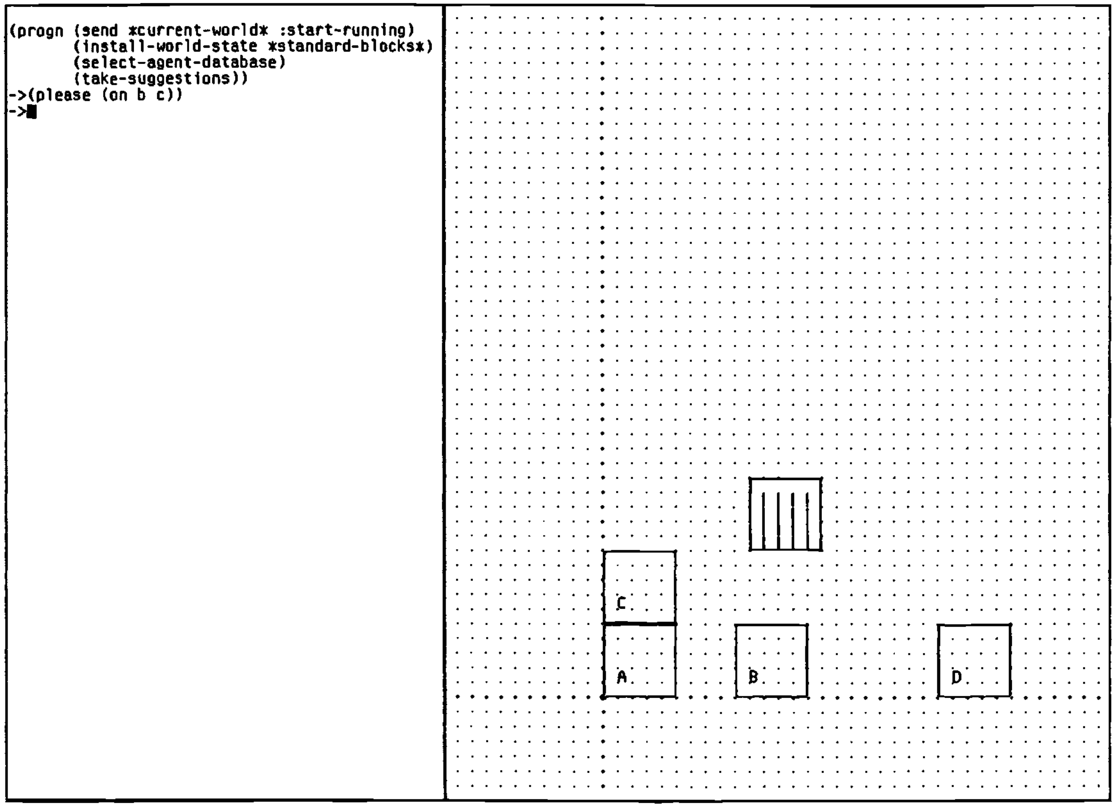
Successive sections present these demonstrations in three groups of two. The first group illustrates dependency maintenance in action and shows the dependency network growing through experience. As an unsurprising result, the dependencies can accelerate the system in a situation that is precisely identical to one it has already been through.

The next section's demonstrations concern whether and when the dependencies will transfer to other tasks. In this the system achieves a mixed success that offers clues for later analyses.

The final section's demonstrations involve a more complex task involving a compound goal. The system goes through some instructive gyrations to perform the task. Many issues arise along the way. The system then repeats the task with the benefit of the additional dependency network circuitry.

Demonstration

Figure 10.1(a) is a snapshot of the system before it has been asked to do anything. The horizontal line of bold dots is a table. The vertical line simply indicates the y axis and has no physical significance.



```
(progn (send *current-world* :start-running)
       (install-world-state *standard-blocks*)
       (select-agent-database)
       (take-suggestions))
->(please (on b c))
->
```

Figure 10.1(a). The user asks the system to put block B on block C.

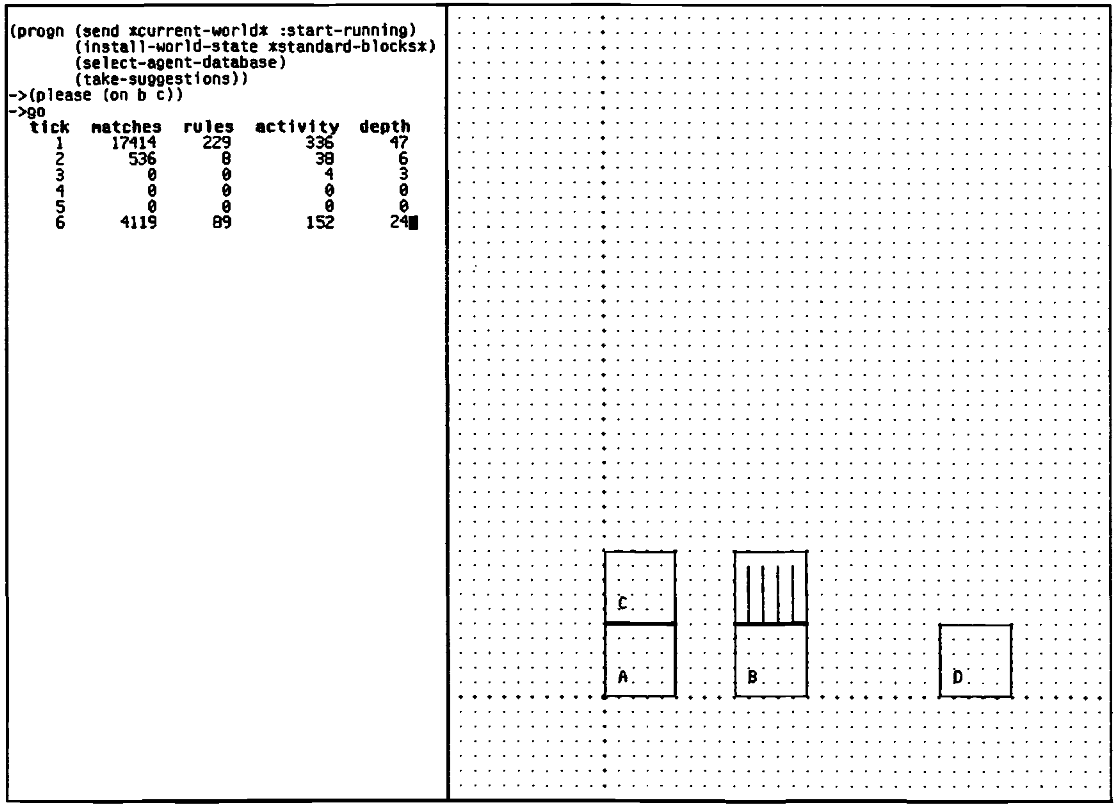
The squares with letters A–B–C–D in them are blocks. The square with stylized fingers is the hand. The "physics" of this blocks world is very simple. On a given clock cycle, the agent can move the hand one unit horizontally or one unit vertically (or both). Thus, single-unit diagonal motions are allowed. If the bottom surface of the hand is touching the top surface of a block, the agent can grasp the block. The hand has no state, so if the agent wishes to keep grasping the block, it must continue asserting the grasping action. The world has gravity, so unsupported blocks fall. It has no momentum, however, so a moving hand stops immediately when the agent stops telling it to move. Blocks cannot rotate. They obey velcro physics: one block will stay stacked on another as long as the bottom of the upper block is in contact with the top of the lower block, regardless of where the upper block's center of gravity is located. All of this will become clearer as the demonstrations proceed.

In the first demonstration, the user has asked the hand to put block B on block C. The hand starts above and a little to the right of center of B. After five cycles it has moved down and landed on B (Figure 10.1(b)).

On the left side of the frame are some statistics to measure the system's effort. Each line presents the statistics for one clock cycle. Only the rough comparative magnitudes of these numbers are significant. On the first cycle the number in the "rules" column is 229, meaning that the system performed 229 rule firings. This is the largest number of rules the system will run on any one cycle during these demonstrations. The number is so large because this is the first time the system had ever been asked to do anything, and so a large number of basic, functional rules are running for the first time.

On the first cycle, the system made two decisions: one to go left to get the hand centered on B and another to go down. As a result, it went left and down on that first *tick*. The reasoning that led to this step involved successively decomposing the goal into subgoals and thinking ahead. Its first step was to get the hand on B; then it was going to have to grab B. All of the arguing this required took 229 rules.

The particular rule set employed in these demonstrations implements a simple plan-construction scheme using ordinary ideas about subgoal decomposition, preconditions, and the like. Despite this, the system has no executor: its reasoning leads it to take a single incremental action, whereupon it effectively throws out all of its reasoning and starts over on the next cycle. The decision to move the hand down and to the left, then,



```
(progn (send *current-world* :start-running)
       (install-world-state *standard-blocks*)
       (select-agent-database)
       (take-suggestions))
->(please (on b c))
->go
  tick  matches  rules  activity  depth
    1    17414    229      336      47
    2      536      8       38       6
    3        0      0        4       3
    4        0      0        0       0
    5        0      0        0       0
    6     4119     89      152      24
```

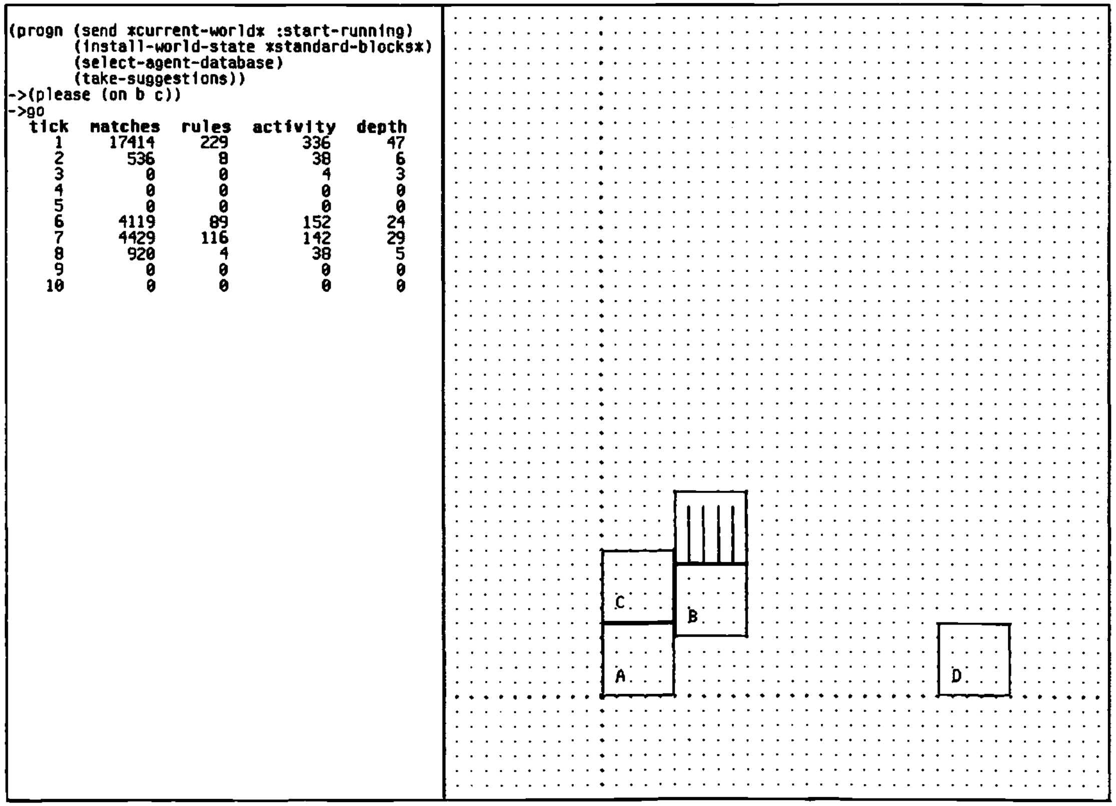
Figure 10.1(b). Never having picked up a block, the system has to run some rules to figure out how.

was the result of a fairly complex chain of reasoning. This chain of reasoning led to two results: an incremental hand motion and a tangle of newly constructed combinational logic gates. On the subsequent cycle, the system effectively starts over, conducting an equally elaborate process of reasoning to decide on its next action. The system will decompose goals, evaluate preconditions of actions, and so forth, just as it did during the first cycle. But this time, the newly constructed dependency network will carry most of the burden.

The outcome of the second cycle is slightly different from that of the first: since the hand is now directly over block B, the hand moves straight down. But no new arguments apply, so the second cycle's reasoning is conducted almost entirely through the dependencies accumulated on the first cycle. (The system runs some rules on the second cycle because it has just received proprioceptive feedback from the hand for the first time.) The hand moves down in this way for five cycles. Each cycle effectively recapitulates the 200-odd rule firings that would be required to make the decision from scratch. Yet no qualitative changes occur in the system's relationship to its "world," so no actual rule firings are required past the second cycle. (A qualitative change is one that causes the system to engage in a different pattern of reasoning, whether through rules or dependencies or both.) Figure 10.1(b) depicts the situation just before the system has noticed that the hand has arrived on the top of block B.

Figure 10.1(c) shows the scene exactly one cycle later. The system's senses have observed the hand touching block B, and this has caused 89 rules to run, a large number. But other things happen on this cycle as well. As the number for cycle 6 in the "activity" column indicates, 152 nodes in the dependency network changed their state.[1] Much of that change reflects the newly run rules; the system has never before found its hand on something it is trying to pick up. But much of it also reflects the lines of reasoning that have gone OUT now that one of the reasons to keep moving downward, namely that the hand did not touch B, is now false. A wire that once carried a 0 now carries a 1. The effects of that 1 have propagated through the network, removing the support formerly enjoyed by the arguments for moving down. As these arguments went OUT, 0s propagated through large parts of the network. The system stopped moving its hand because the argument leading it to keep moving was no longer justified.

On cycle 7 the system requires 116 rule firings to decide to move B toward its destination. As before, the system must run a large number of



```
(progn (send *current-world* :start-running)
       (install-world-state *standard-blocks*)
       (select-agent-database)
       (take-suggestions))
->(please (on b c))
->go
  tick  matches  rules  activity  depth
    1    17414    229      336      47
    2      536      8       38       6
    3        0      0        4       3
    4        0      0        0       0
    5        0      0        0       0
    6     4119     89      152      24
    7     4429    116      142      29
    8      920      4       38       5
    9        0      0        0       0
   10        0      0        0       0
```

Figure 10.1(c). Never having moved a block anywhere, the system has to run some rules to figure out how, after which it runs smoothly until it strikes an obstacle.

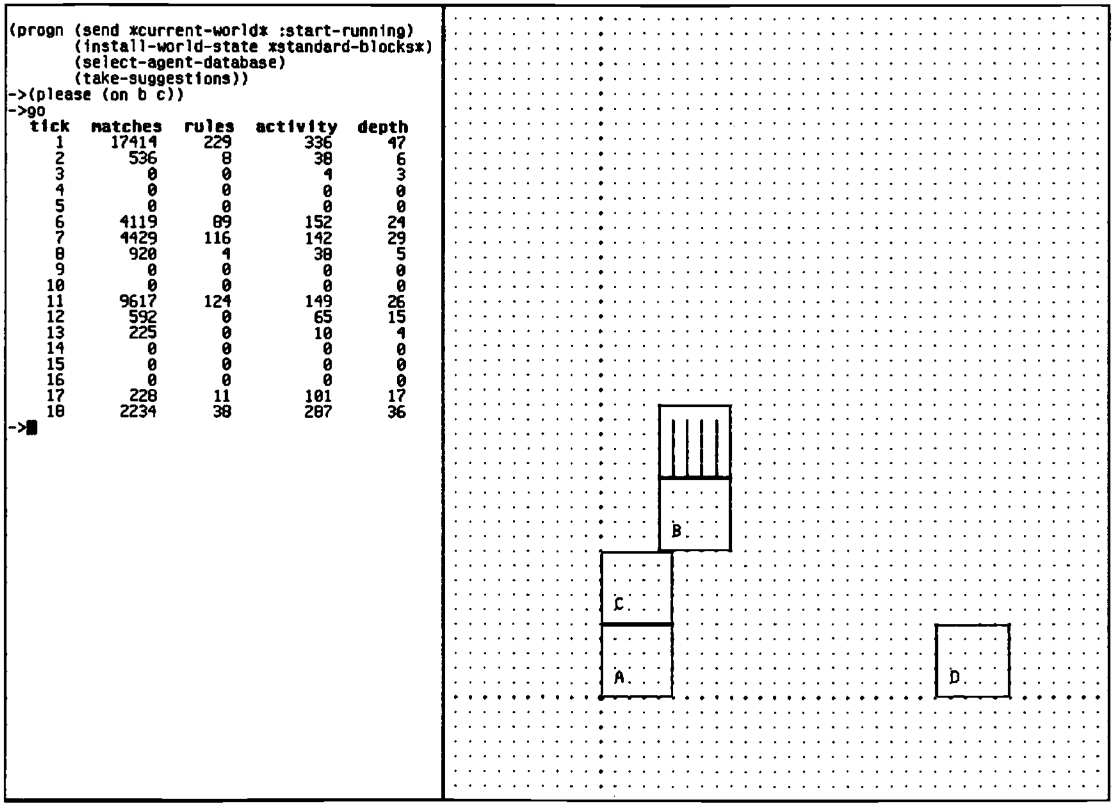
rules since it has never moved a block toward its destination before. Once it gets moving, however, all the arguments that take place in those 116 rules continue to apply, so the hand moves along without the system having to run any more rules.

Figure 10.1(d) shows the system after four more cycles. The hand has picked up B and has moved off toward C. The system knows little about trajectories. It has chosen its motions serviceably, but nonetheless it has bumped into C. What happens now, on cycle 11? There had been an argument for going up, which was that B was below C and needs to be above C and the hand is holding B. There was also an argument for moving left, which is that B needs to be overlapping horizontally with C, and B is all the way to the right of C. This argument for moving left is still good since B is still to the right of C. But now the side of B is touching C. That fact is causing some rules to fire that had never been involved before. These rules object to moving left on the grounds that it is not a good idea to push the destination block without cause. This objection, like all objections, is offered ceteris paribus. But since no rule objects to it in turn, it is accepted and the proposal to move left is defeated. All this action requires 124 rules to fire. But the argument for going up is still uncontroversial, so the hand continues moving upward.

When it gets up far enough that B clears C, the argument against moving left no longer holds, so there is a great deal of activity in the network as that objection goes OUT. Henceforth the hand is free to move left. The proposal of moving left has been active all the while, but it has not been adopted because of the objection overriding it.

On the final cycle, B is now on top of C. The system runs 38 rules that recognize its achievement of a goal; these rules have never run before, since this is the first time it has ever finished a job. The network is extremely active on this final cycle, with 287 nodes changing state, a large portion of the whole network. Those 287 nodes represent the whole apparatus of inference and argument behind the top-level goal, its decomposition into subgoals, and the idea of having the hand on something and moving it along. Now that B is on C, the goal has been achieved, so the support for all of that apparatus has gone OUT. Where once there was a 1, there now is a 0; so 0s propagate through the network, which then finally settles into its rest state. Large sections of the network that turned on during the first cycle now turn off during the last cycle.

Patterns in these numbers indicate some of the important dynamics of the system's interaction with its world. Most of the patterns concern the



```
(progn (send *current-world* :start-running)
       (install-world-state *standard-blocks*)
       (select-agent-database)
       (take-suggestions))
->(please (on b c))
->go
  tick  matches  rules  activity  depth
     1    17414    229       336     47
     2      536      8        38      6
     3        0      0         4      3
     4        0      0         0      0
     5        0      0         0      0
     6     4119     89       152     24
     7     4429    116       142     29
     8      920      4        38      5
     9        0      0         0      0
    10        0      0         0      0
    11     9617    124       149     26
    12      592      0        65     15
    13      225      0        10      4
    14        0      0         0      0
    15        0      0         0      0
    16        0      0         0      0
    17      228     11       101     17
    18     2234     38       287     36
->█
```

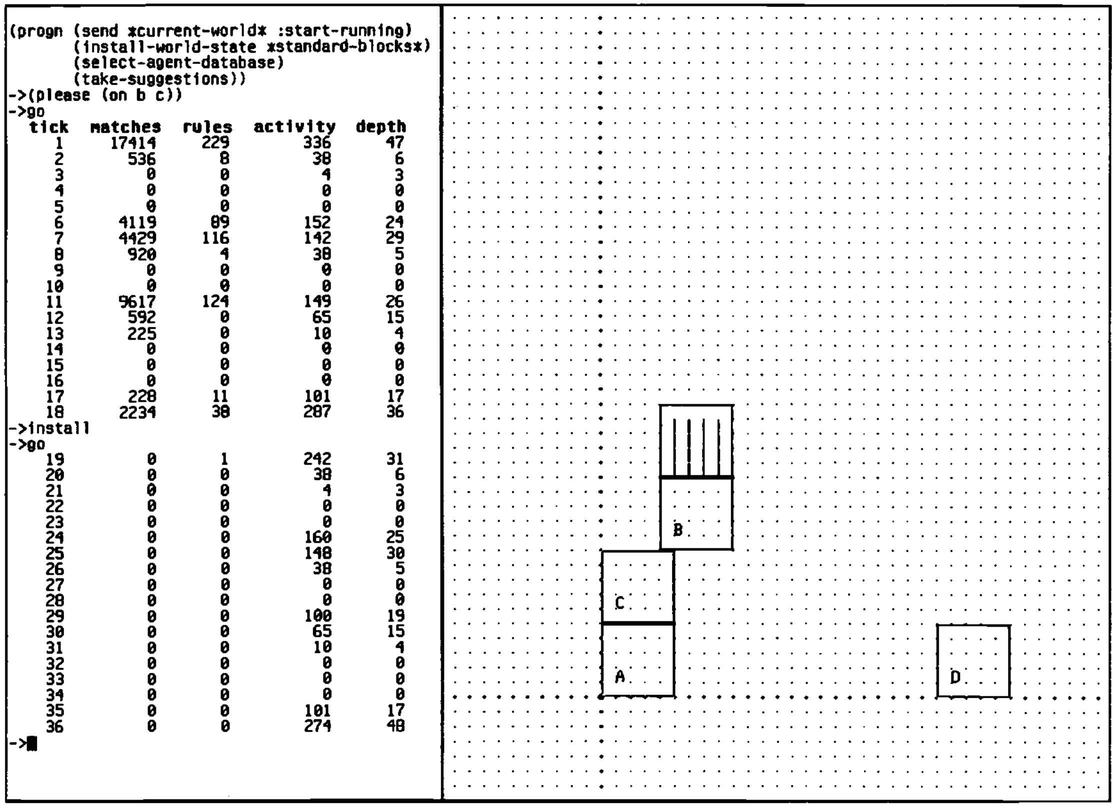
Figure 10.1(d). Never having hit an obstacle, the system has to run some rules to figure out how to circumvent it. Some more rules are required right at the end.

levels of activity in the network. Observe that on the first three cycles an initial burst trails off to zero: 336 to 38 to 4 to 0. Then a second burst starts on cycle 6. That burst lasts for two cycles, one for grabbing B and one for getting moving, but then as with the initial burst it fades to zero. Then a third burst starts on cycle 11 as B hits C and begins sliding up along it. At the end are two bursts on adjacent cycles, for clearing C and then for finishing the job. This burst–decay pattern will be ubiquitous throughout the demonstrations.

To understand the burst–decay pattern, think of the dependency network as moving through an enormous phase space with a dimension for each node. As the system works on its task, it describes a trajectory through this space. When nothing is qualitatively changing, the network's state remains unchanged. When the network crosses a boundary into a qualitatively different region, such as when it encounters an unexpected situation or moves from one subgoal to another, the network will change its configuration to reflect the change in the agent's relationship to its world. Patches of the network that had been active will turn off and other patches that had been quiet will become active in their place. Each burst of activity reflects this sort of change. The burst takes an extra cycle to decay to zero because of proprioception; the system receives a signal from its "body" indicating that an attempt to move the hand or to grab something is actually succeeding. Observe that the burst–decay pattern concerns only the amount of activity in the network, not the number of rules that fire. In later demonstrations, the activity number will repeatedly burst and decay without any rules at all being fired.

A second pattern concerns the relative magnitudes of the activity numbers. The network is most active at the very beginning and the very end. At the beginning it decomposes a top-level goal into subgoals. That top-level goal and its associated structure remain IN during the whole process. In other words, a region of network configures itself for putting B on C and it stays in that configuration until the end. The other peaks of activity occur when the system switches from one major subgoal to the next. Another peak occurs when the system must pull up short and reason about an unexpected condition, B's hitting C, and then again to retract that reasoning when the unexpected condition goes away.

This pattern is even clearer in the "depth" numbers, which indicate the longest chain of nodes in the network that changed state – that is, the longest causal chain within the network on this cycle. This number tends



```
(progn (send *current-world* :start-running)
       (install-world-state *standard-blocks*)
       (select-agent-database)
       (take-suggestions))
->(please (on b c))
->go
  tick  matches  rules  activity  depth
     1    17414    229       336     47
     2      536      8        38      6
     3        0      0         4      3
     4        0      0         0      0
     5        0      0         0      0
     6     4119     89       152     24
     7     4429    116       142     29
     8      920      4        38      5
     9        0      0         0      0
    10        0      0         0      0
    11     9617    124       149     26
    12      592      0        65     15
    13      225      0        10      4
    14        0      0         0      0
    15        0      0         0      0
    16        0      0         0      0
    17      228     11       101     17
    18     2234     38       287     36
->install
->go
    19        0      1       242     31
    20        0      0        38      6
    21        0      0         4      3
    22        0      0         0      0
    23        0      0         0      0
    24        0      0       160     25
    25        0      0       148     30
    26        0      0        38      5
    27        0      0         0      0
    28        0      0         0      0
    29        0      0       100     19
    30        0      0        65     15
    31        0      0        10      4
    32        0      0         0      0
    33        0      0         0      0
    34        0      0         0      0
    35        0      0       101     17
    36        0      0       274     48
->█
```

Figure 10.2. The user has restored the blocks to their original positions and set the system running again with the same goal. After running a single rule to recover from the discontinuity, the system can run entirely out of its dependency network.

to measure the depth of the system's patterns of reasoning. The depth is very high when the system is decomposing its top-level goal on the first cycle and when it retracts all the reasoning behind the decomposition on the last cycle. In more complex examples that pattern will appear recursively.

The system has now finished performing its first task. In doing so, it has had to run several hundred rules, but it has also built up several hundred gates worth of dependency network. If the system is working well, those dependencies ought to permit the system to run fewer rules in the future. Later sections will assess whether the system's experience putting B on C transfers to other activities. For the moment, though, let us consider the simplest case. Do the dependencies permit the system to perform exactly the same task in exactly the same situation without running a significant number of rules?

The second demonstration begins after the blocks and the hand have been restored to their original positions. (This will slightly confuse the system for a moment because the discontinuous change will disrupt the proprioceptive information.) The same goal, putting B on C, is still in effect. The system is then set running. It runs a single rule on the first cycle (to recover from the discontinuous change), but after that it runs no rules at all (Figure 10.2). It makes the same moves it made before, including bumping into the side of C. The system is not learning in the sense of adaptively changing its behavior; it is just doing the same thing more efficiently. The system is firing no rules and performing no pattern matches, but the numbers for the activity in the network are qualitatively the same. As in the first time through, the activity is very high at the beginning and end, with peaks at significant transitions and a recurring pattern of bursts and decays. Likewise, the depths are the largest at the beginning and end, with intermediate values when the system changes subgoals or when something exceptional happens. From the outside, the system appears to be running all of the hundreds of rules it ran before. But from the inside, no rules are firing because the dependency network covers every case that comes up.

Patterns of transfer

The remaining demonstrations address the question of whether and when dependency maintenance accelerates the system. Each demonstration except the last picks up where the original demonstration left off, with the system just having been asked to put block B on block C. In having performed this task, the system has built several hundred gates worth of dependency network. The question is, what can the system now do automatically, running out of its dependencies instead of rules, in virtue of having had that experience? The preceding section demonstrated that the system can now do precisely the same thing without running any new rules. But what can it do that is *not* precisely the same? What makes a new situation sufficiently similar for the old reasoning to transfer?

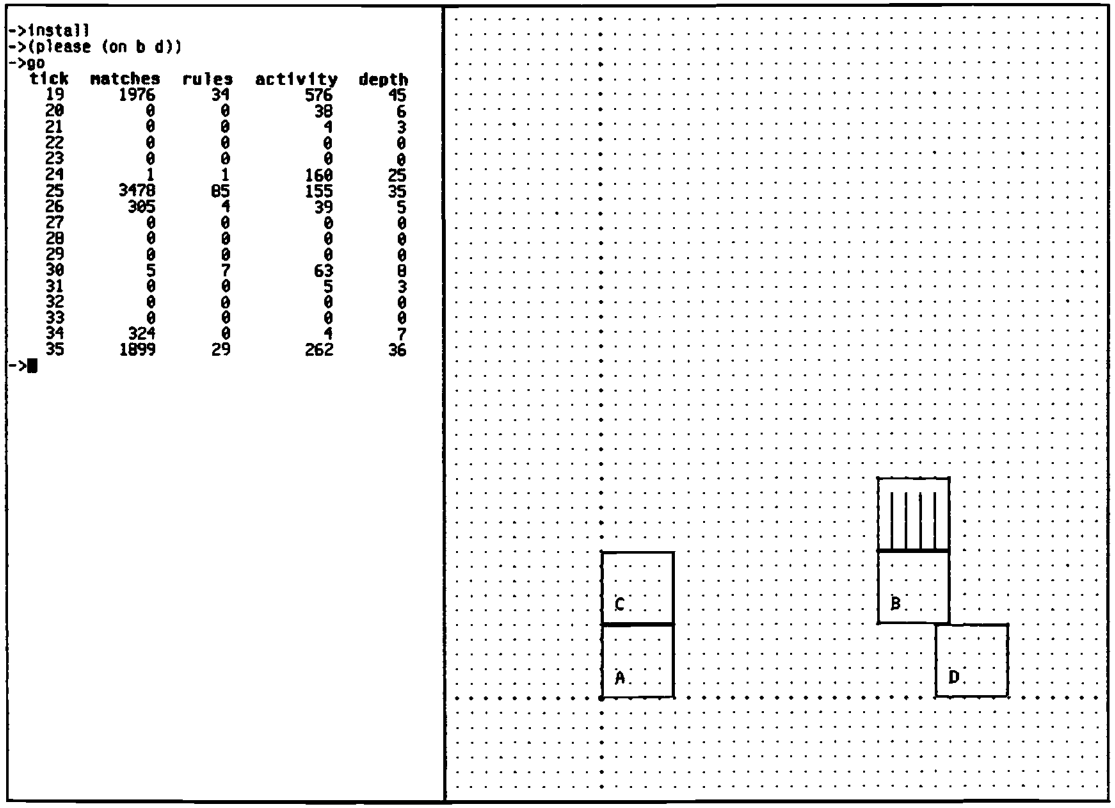
The third demonstration is a success story. The user has backed the system up to just after the completion of the first demonstration; the four blocks and the hand are back in their original positions. This time, instead of asking it to put B on C, the user asks it to put B on D. It is a different task, but both tasks involve picking up B.

On the first cycle, number 19, the system runs 34 rules (Figure 10.3). That is not bad given the 229 rules it ran on the first cycle during its first task; many of those rules must have carried over to this second task. Those 34 rules concern the different destination.

On cycle 24 the hand stops upon reaching B and the program decides it should grab B. The first time out, during the task of putting B on D, that took 89 rules. This time it takes one rule. The activity number is still high, 152 as compared with 160 before, reflecting the network's change in configuration as one region of the network goes OUT and another comes IN.

On the next cycle, number 25, the system decides to pick up B and move it to the right toward D. That takes 85 rules, as compared with the 116 rules it took to decide to move B to the left toward C during the first demonstration. Evidently little has transferred from the first task to the second, but it is hard to say how much ought to have transferred. Certainly moving left toward C and right toward D are different activities.

The system completes its task without further incident on cycle 35. The final cycle, on which it realizes it is done, takes 29 rules, as compared with the 38 rules it took to complete the first task. The activity number is high and approximately the same in both cases, 262 as compared with 287 before, reflecting all of the apparatus that has gone OUT once the system has achieved its goal. Again, little appears to have transferred from the completion of the first task to the completion of the second.



```
->install
->(please (on b d))
->go
  tick  matches  rules  activity  depth
   19     1976     34       576     45
   20        0      0        38      6
   21        0      0         4      3
   22        0      0         0      0
   23        0      0         0      0
   24        1      1       160     25
   25     3478     85       155     35
   26      305      4        39      5
   27        0      0         0      0
   28        0      0         0      0
   29        0      0         0      0
   30        5      7        63      8
   31        0      0         5      3
   32        0      0         0      0
   33        0      0         0      0
   34      324      0         4      7
   35     1899     29       262     36
->█
```

Figure 10.3. The system has already picked up B in the previous task, so that portion of the dependency network transfers to the new task. The rest of the task requires somewhat fewer rules than before, but the dependencies do not transfer to putting something on D.

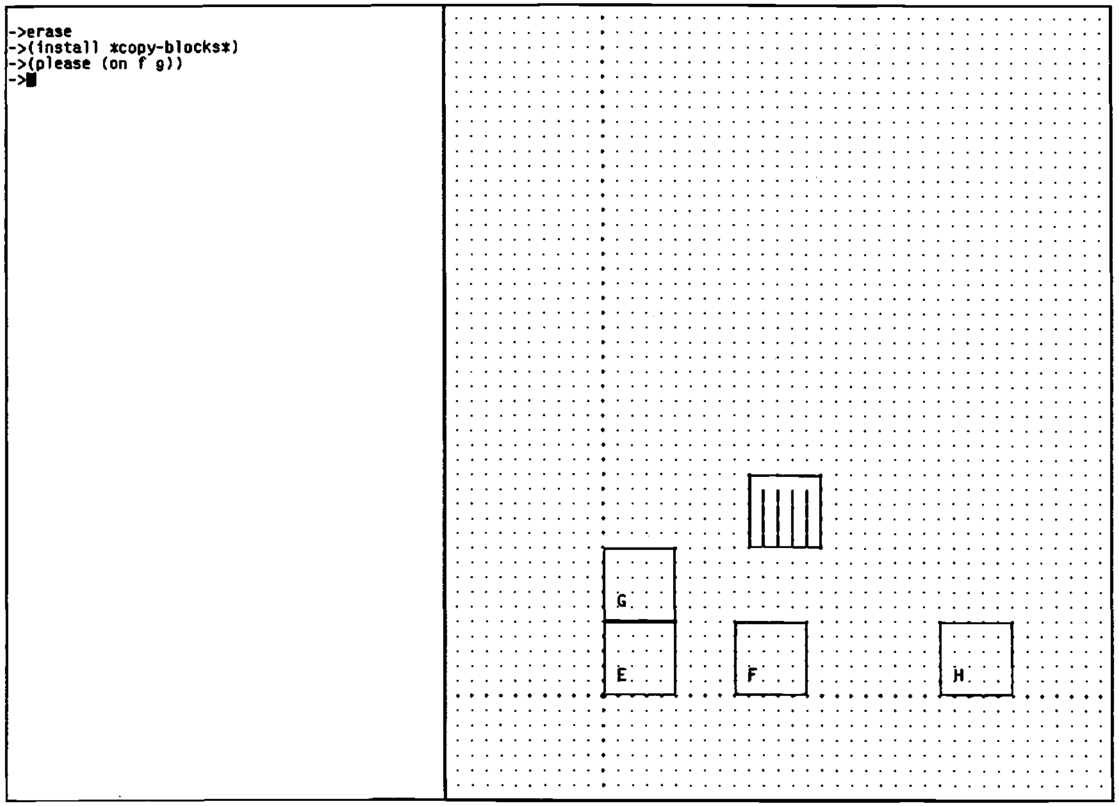
Despite the uncertainties, this demonstration offers two clear instances of dependencies constructed on the first task carrying over to the second. On the very first cycle, much of the apparatus of decomposing the goal and initiating action appears to have been independent of the particulars of the two tasks. And when the hand found itself on B and decided to grab it, the apparatus associated with that subgoal was independent of the larger goals of which it was a part.

The fourth demonstration also begins after the system has completed its first task of putting B on C. The initial situation is the same except that all of the blocks have different names now, E–F–G–H instead of A–B–C–D (Figure 10.4(a)). The task now is putting F on G. Thus, the situation and goal are identical to the originals except for the names of the blocks. Are the situation and task the "same" as before? Not precisely, but ideally most of the dependency structure from the first task ought to carry over here. The system's actions and the reasoning behind them, after all, will have exactly the same form as before.

Unfortunately the degree of transfer is only moderate (Figure 10.4(b)). The number of rule firings at major transitions ranges from half to three-quarters of the corresponding number in the first demonstration. It takes 159 rules to get moving. When the hand lands on F, the system takes 45 rules to grab it and 64 rules to get it moving. Hitting an obstacle causes 96 rules to run. The decay portion of the burst–decay pattern, corresponding to the system's reactions to proprioceptive information, has transferred. In general, those parts of the reasoning that were independent of particular individuals have transferred. For example, the system has already unfolded some relatively domain-independent apparatus having to do with arguments and plans. Yet nothing that relates to these particular blocks has carried over. If the system runs a rule like

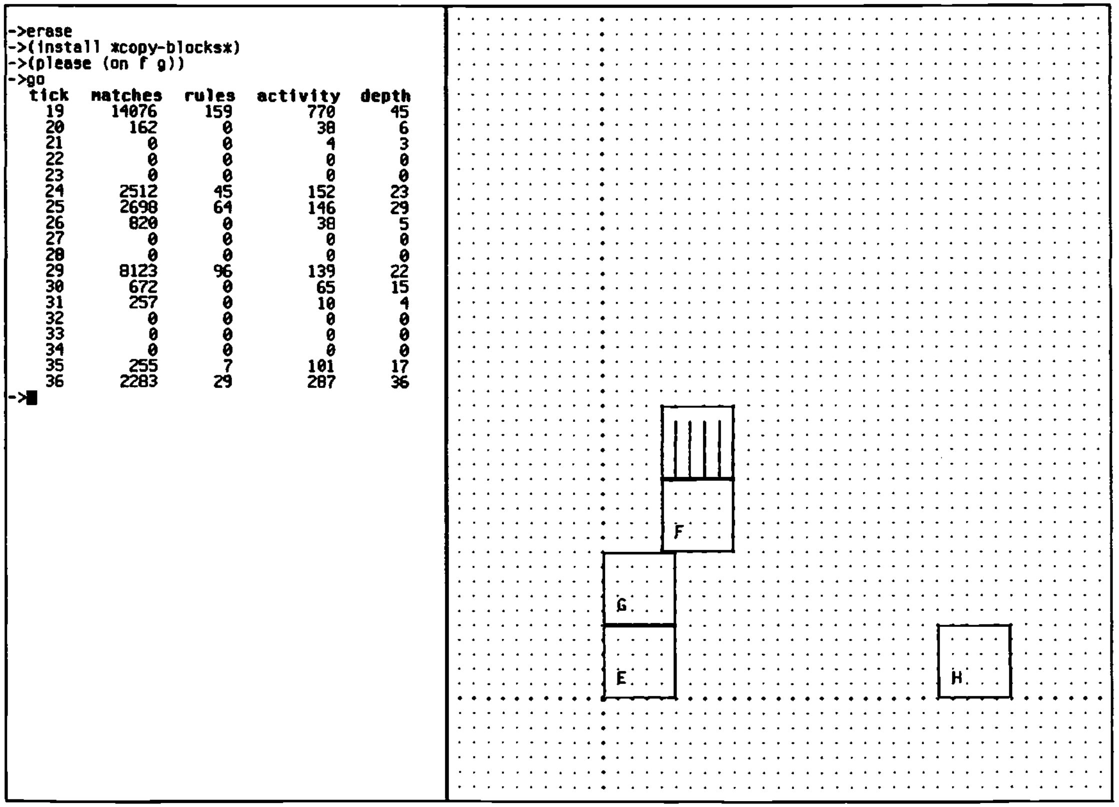
```
(if (and (trying (grasp ?x))
         (on hand ?x))
    (propose (grasp)))
```

then this rule will have to run again for every block that the system ever tries to grasp. During the first demonstration the rule ran with **x** bound to B. During the second and third demonstrations the rule did not have to run because the system did not ever try to grasp any block except B. During this fourth demonstration, however, it tried to grab F. As far as the system's representation scheme is concerned, F is a wholly different



```
->erase
->(install *copy-blocks*)
->(please (on f g))
->█
```

Figure 10.4(a). Here the original blocks have been removed and replaced with identical blocks that occupy the identical locations but have different names.



```
->erase
->(install *copy-blocks*)
->(please (on f g))
->go
  tick   matches   rules   activity   depth
   19     14076     159       770       45
   20       162       0        38        6
   21         0       0         4        3
   22         0       0         0        0
   23         0       0         0        0
   24      2512      45       152       23
   25      2698      64       146       29
   26       820       0        38        5
   27         0       0         0        0
   28         0       0         0        0
   29      8123      96       139       22
   30       672       0        65       15
   31       257       0        10        4
   32         0       0         0        0
   33         0       0         0        0
   34         0       0         0        0
   35       255       7       101       17
   36      2283      29       287       36
->█
```

Figure 10.4(b). Having been set to a task that is perfectly analogous to one it has performed before, the system must run many rules.

block from B, so all the rules that originally mentioned B must now be fired again with **x** bound to F, thus creating a duplicate, analogous dependency structure. None of this makes the dependency network any deeper, but it does make it bulkier. This proliferation of circuitry has no limits. So long as new blocks appear, the system will have to build more circuitry.

The situation is even worse in the case of a rule that mentions two blocks. Consider the following rule:

```
(if (and (propose (move hand ?direction))
         (horizontal ?direction)
         (grasping ?x)
         (sides-touch ?x ?y)
         (in-direction ?direction ?y ?x))
  (propose (object (move hand ?direction)
                   (dont-push ?pushed))))
```

This rule, or actually a more general version of it, posted an objection in the first demonstration when B accidentally bumped into C. It also posted an objection on the fourth demonstration when F accidentally bumped into G. Each time, the system had to build a new patch of network structure. The system might have to build an amount of circuitry proportional to the square of the number of blocks it encounters. This is physically possible, but it is not very satisfying.

RA might be at its worst on an assembly line. Asked to perform analogous tasks in an endless stream of analogous situations comprising different individuals, the system will generate a vast number of analogous circuits, none of which it will ever use again. Imagine passing a thousand cars on a long car trip, turning the knobs on a thousand doors over the course of a year, turning a thousand pages while reading comic books on the beach, or eating a thousand spoonfuls of cereal over successive breakfasts. Each series involves interactions, each perhaps subtly different, with a thousand different individuals whose individuality per se has little effect on what actions are indicated. The knowledge embodied in RA's rules is abstracted away from the particulars of a thousand situations with the use of variables. But dependencies do not support variables, nor can dependencies be generalized to support variables without losing most of the virtues of dependencies.

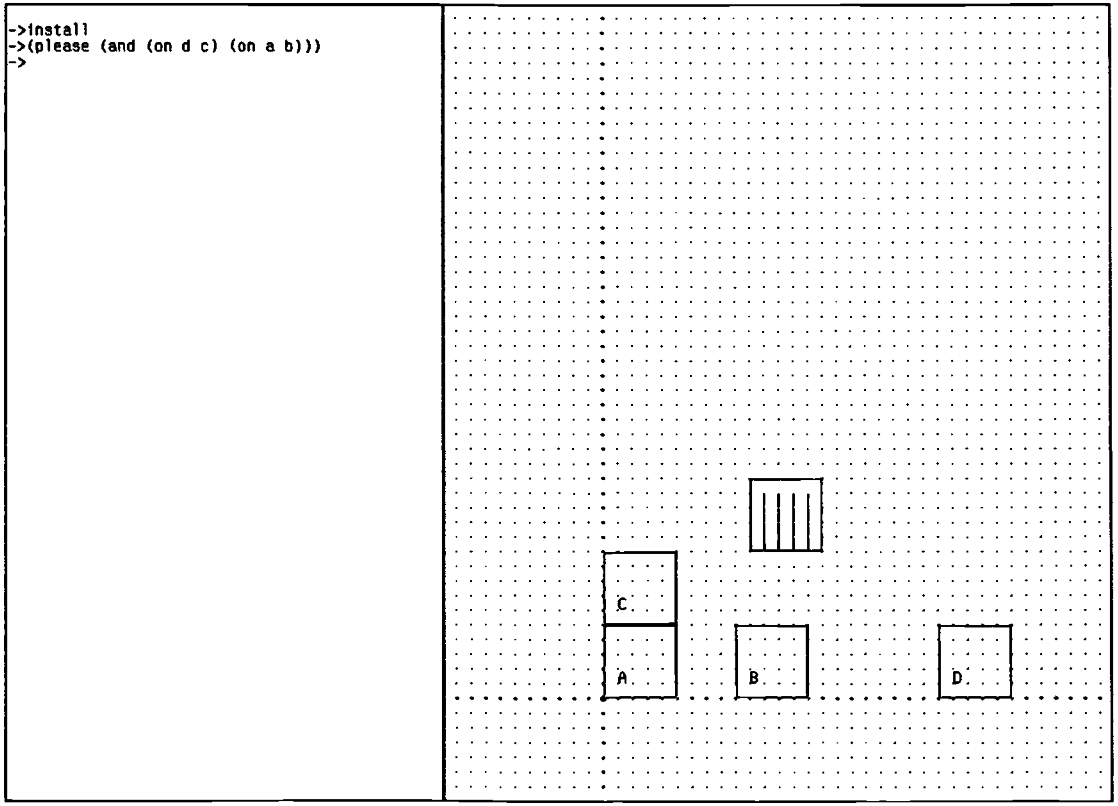
In short, reasoning that involves the same individuals will transfer, even into different contexts, but no transfer will take place between one set of individuals and another. The system decomposes its reasoning automatically but it does not make analogies automatically.

Transfer and goal structure

The fifth demonstration is longer and, having been chosen for its twists and turns, illustrates some additional dynamic effects. Once again the system has just put B on C and the blocks and hand have been restored to their original positions. The task now is a sequence of two subtasks, putting D on C and then putting A on B (Figure 10.5(a)).

The and in the task description is not a logical conjunction but rather an instruction to perform the subtasks in sequence. Originally and meant "do them all in any order," but I was unable to implement the conjunctive semantics in a satisfactory way. When the system faced two tasks, it needed rules to determine which one to perform first. As with all its decisions, the system conducted an argument with itself, putting forward proposals and weighing the arguments for and against them. In many situations the decision was straightforward. For example, the hand might already be perched on the block that would have to be moved first in order to perform one of the subtasks. Or the blocks might need to be stacked in a particular order. Rules for cases like these were easy enough to write. The rules were harder to write when there existed no reason at all for choosing one task rather than another. The rule language, unfortunately, had no way of expressing an arbitrary choice. I considered extending the rule language with a mechanism for performing arbitrary choices, but none of the obvious schemes were sufficiently principled and general. I tried writing rules that implemented utterly arbitrary decision schemes, but no one criterion sufficed to discriminate in all cases, so the criteria had to argue among themselves. This did work, but nothing was gained in waiting for all the necessary rules to fire. The issue is deeper than it seems; I will return to it after the demonstrations.

On the first cycle, number 19, the system has a good amount of work to do. It has never been asked to perform a sequence of tasks, so it must run many rules to decompose the compound task, assign itself the subtask of putting D on C, and figure out how to head toward D. All of this action takes 251 rules. Once it gets moving, it puts D on C without incident. The burst–decay pattern occurs throughout. Deciding to grab D takes



```
->install
->(please (and (on d c) (on a b)))
->
```

Figure 10.5(a). The blocks have been restored to their original arrangement and the system has been assigned a complex task.

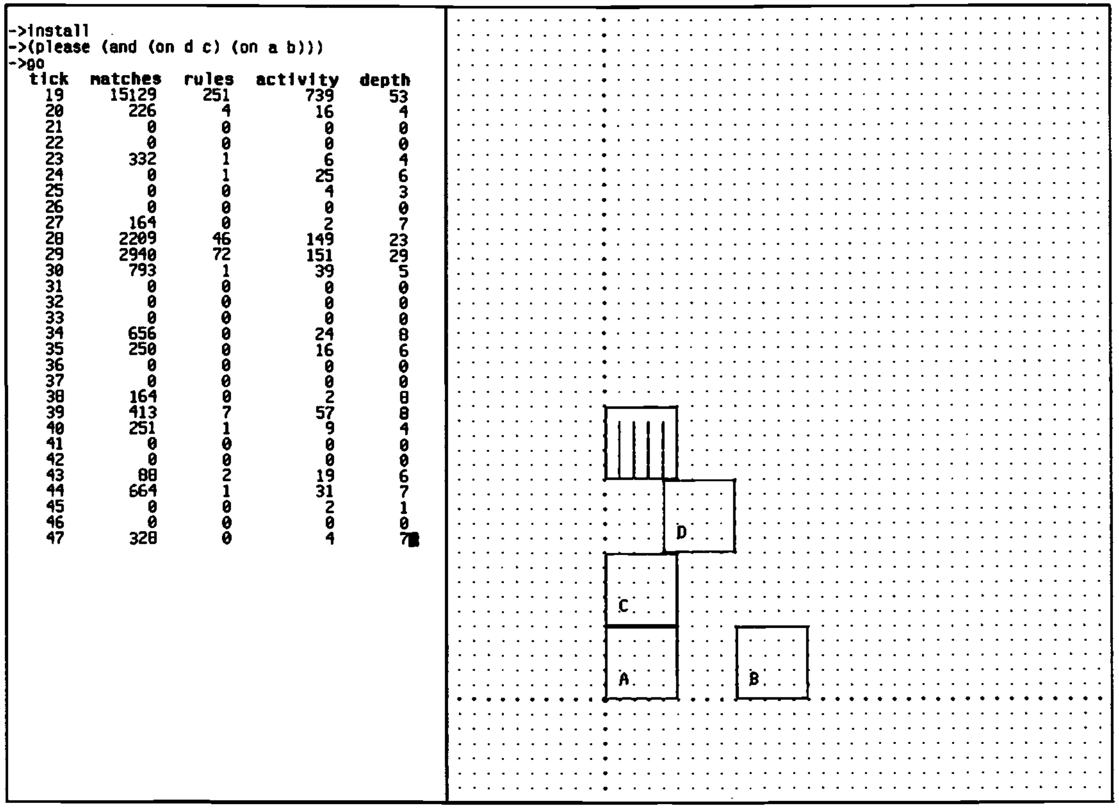
46 rules; deciding to move D upward and leftward toward C takes 72 rules. Both of these numbers are excessive, given that the system has already put B on C, a fairly analogous task.

Figure 10.5(b) shows the system after cycle 47, immediately after the system has put D on C and is about to discover that it has finished with the first subgoal, so that it can get started on the second. A great deal takes place in the next few cycles.

Something complicated happens on cycle 48 (Figure 10.5(c)). Having finished with its first subtask, the system is pursuing the task of putting A on B. Its first subsubtask is to get its hand on A. A is below the hand so the system proposes moving downward. But then the objection arises that the hand cannot move down because a stack of blocks is directly below it. Another rule then offers the alternative of going around and proposes moving left. There are no objections, so the hand moves left. This required 250 rules – quite a complex process. The level of activity in the network is high, 799, reflecting both this reasoning coming IN and the reasoning behind the first subtask going OUT.

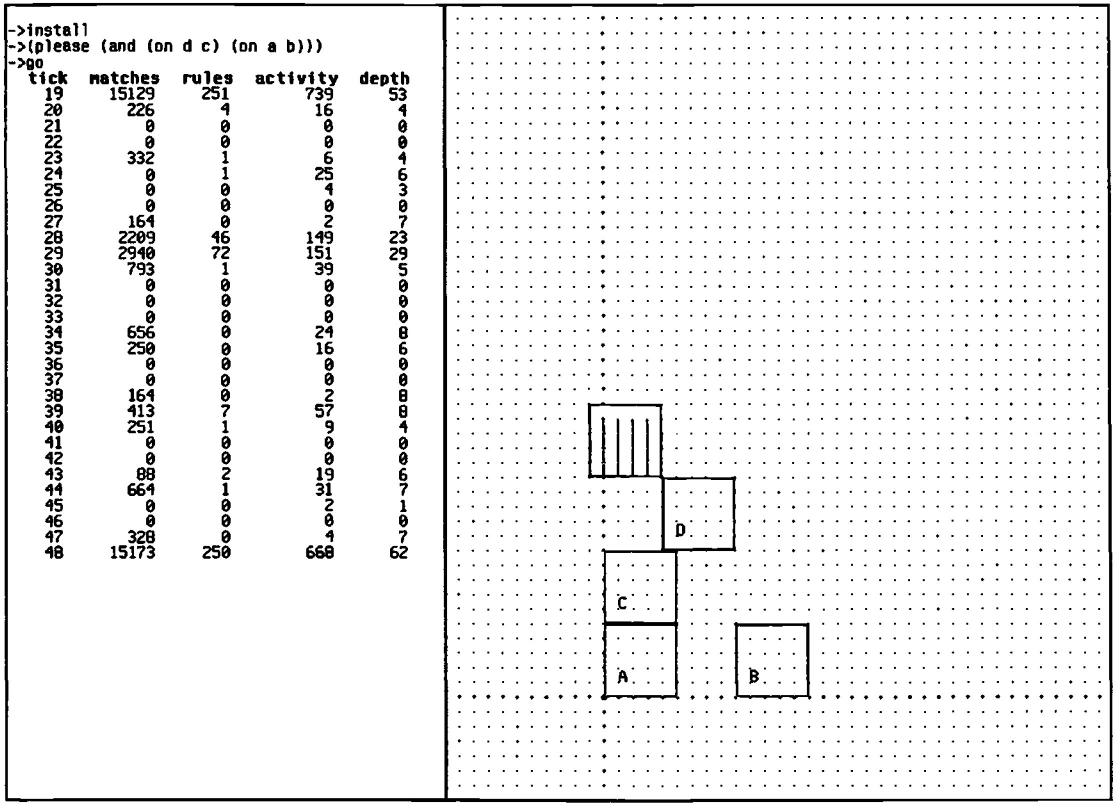
Something unfortunate happens on cycle 49 (Figure 10.5(d)). The proposal of moving down to get on top of A, having been made on the previous cycle, still applies. And the objection against it, that the stack of blocks is in the way, no longer applies, so the hand moves down. Another rule points out the virtue, other things being equal, of centering the hand over the block, and thus proposes moving to the right. No rule objects to the combination of proposals, so the system adopts both and the hand moves downward and to the right, pushing D to the right off of C. All of this took only 5 rules, since almost all of this reasoning had taken place in other contexts. It did involve an activity of 165 in the dependency network, most of which was due to objections from the previous cycle that no longer applied and so went OUT.

Cycle 50 is the most interesting. Since D is no longer on C, the first subgoal does not hold true, so one of the justifications for pursuing the second subtask has been removed. The activity is very high, 582, as the network changes configuration, sending the second subtask's apparatus OUT and bringing the first subtask's apparatus back IN. The process takes 61 rules, many of which reflect D's top surface being above the bottom surface of the hand, leading the system to conclude that it must move the hand upward. Some of them also reflect the hand being in side-to-side contact with D, leading the system to defeat the proposal of moving to the right.



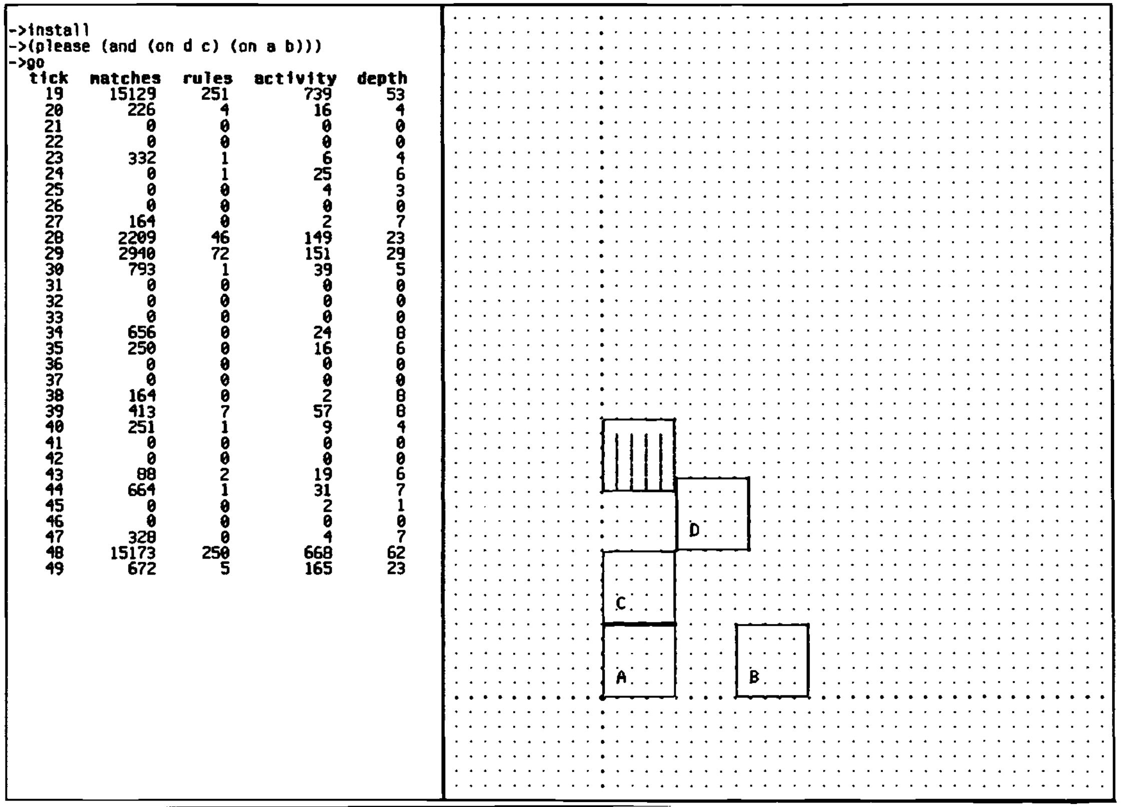
```
->install
->(please (and (on d c) (on a b)))
->go
  tick   matches   rules   activity   depth
   19     15129     251       739       53
   20       226       4        16        4
   21         0       0         0        0
   22         0       0         0        0
   23       332       1         6        4
   24         0       1        25        6
   25         0       0         4        3
   26         0       0         0        0
   27       164       0         2        7
   28      2209      46       149       23
   29      2940      72       151       29
   30       793       1        39        5
   31         0       0         0        0
   32         0       0         0        0
   33         0       0         0        0
   34       656       0        24        8
   35       250       0        16        6
   36         0       0         0        0
   37         0       0         0        0
   38       164       0         2        8
   39       413       7        57        8
   40       251       1         9        4
   41         0       0         0        0
   42         0       0         0        0
   43        88       2        19        6
   44       664       1        31        7
   45         0       0         2        1
   46         0       0         0        0
   47       328       0         4        7
```

Figure 10.5(b). The system places D on C without incident.



```
->install
->(please (and (on d c) (on a b)))
->go
  tick   matches   rules   activity   depth
   19     15129     251       739       53
   20       226       4        16        4
   21         0       0         0        0
   22         0       0         0        0
   23       332       1         6        4
   24         0       1        25        6
   25         0       0         4        3
   26         0       0         0        0
   27       164       0         2        7
   28      2209      46       149       23
   29      2940      72       151       29
   30       793       1        39        5
   31         0       0         0        0
   32         0       0         0        0
   33         0       0         0        0
   34       656       0        24        8
   35       250       0        16        6
   36         0       0         0        0
   37         0       0         0        0
   38       164       0         2        8
   39       413       7        57        8
   40       251       1         9        4
   41         0       0         0        0
   42         0       0         0        0
   43        88       2        19        6
   44       664       1        31        7
   45         0       0         2        1
   46         0       0         0        0
   47       328       0         4        7
   48     15173     250       668       62
```

Figure 10.5(c). The system prepares to move the hand around D on its way to pick up A.



```
->install
->(please (and (on d c) (on a b)))
->go
  tick   matches   rules   activity   depth
   19     15129     251       739       53
   20       226       4        16        4
   21         0       0         0        0
   22         0       0         0        0
   23       332       1         6        4
   24         0       1        25        6
   25         0       0         4        3
   26         0       0         0        0
   27       164       0         2        7
   28      2209      46       149       23
   29      2940      72       151       29
   30       793       1        39        5
   31         0       0         0        0
   32         0       0         0        0
   33         0       0         0        0
   34       656       0        24        8
   35       250       0        16        6
   36         0       0         0        0
   37         0       0         0        0
   38       164       0         2        8
   39       413       7        57        8
   40       251       1         9        4
   41         0       0         0        0
   42         0       0         0        0
   43        88       2        19        6
   44       664       1        31        7
   45         0       0         2        1
   46         0       0         0        0
   47       328       0         4        7
   48     15173     250       668       62
   49       672       5       165       23
```

Figure 10.5(d). Centering the hand over A causes it to bump D off of C, thus undoing the first subgoal.

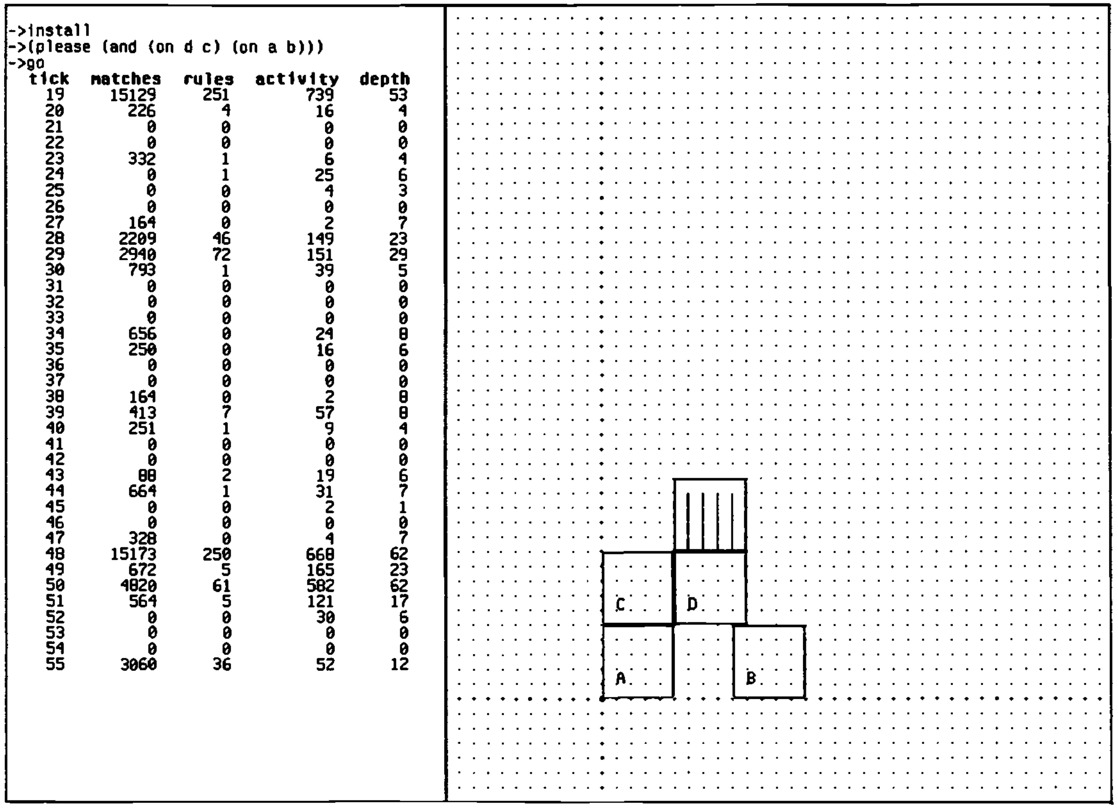
Starting on cycle 51, D falls and the hand begins chasing it rightward and downward. The hand's transition from upward to downward motion occurs mostly through the network, having transferred from the first time it fetched D. Finally the hand lands on D on cycle 55 (Figure 10.5(e)). On cycle 56 it decides to grasp D, which reasoning carries over entirely from the first time it grasped D, except for one stray rule. On cycle 57 the hand lifts D straight up, having run some rules to defeat the proposal of moving leftward as well. When B bumped into C during the first demonstration, it took 124 rule firings, many of which have transferred over to this case despite the different individuals. Much of this transfer does not concern bumping-into per se; cycle 11 was the first argument of any difficulty that the system had conducted with itself on any topic. From cycles 58 to 62 the system moves D back up onto C with no rule firings at all (Figure 10.5(f)).

On cycle 63 the activity number is very high again, 715, reflecting another large swap as the first subtask's apparatus goes back OUT and the second subtask's apparatus comes back IN (Figure 10.5(g)). The system does run 49 rules on cycle 63; these concern the old problem of getting the hand around the stack of blocks and onto A. This time, the hand is right of center instead of left, so a rule proposes moving to the right around the blocks. The system adopts this proposal and moves right. As before, objections defeat a proposal to move downward.

The rest of the trip is comparatively uneventful. The hand clears the right edge of D on cycle 67. On cycle 68 it begins moving downward and to the left toward A, inadvertently pushing D to the left as it goes. Fortunately it does not push D off of C again. (If it had, it would probably have gone into an endless cycle.) Instead it lands on C on cycle 72 and repeats the same sequence, moving to the right one step and then changing course again to move downward and to the left. This motion pushes D to the left as well (Figure 10.5(h)).

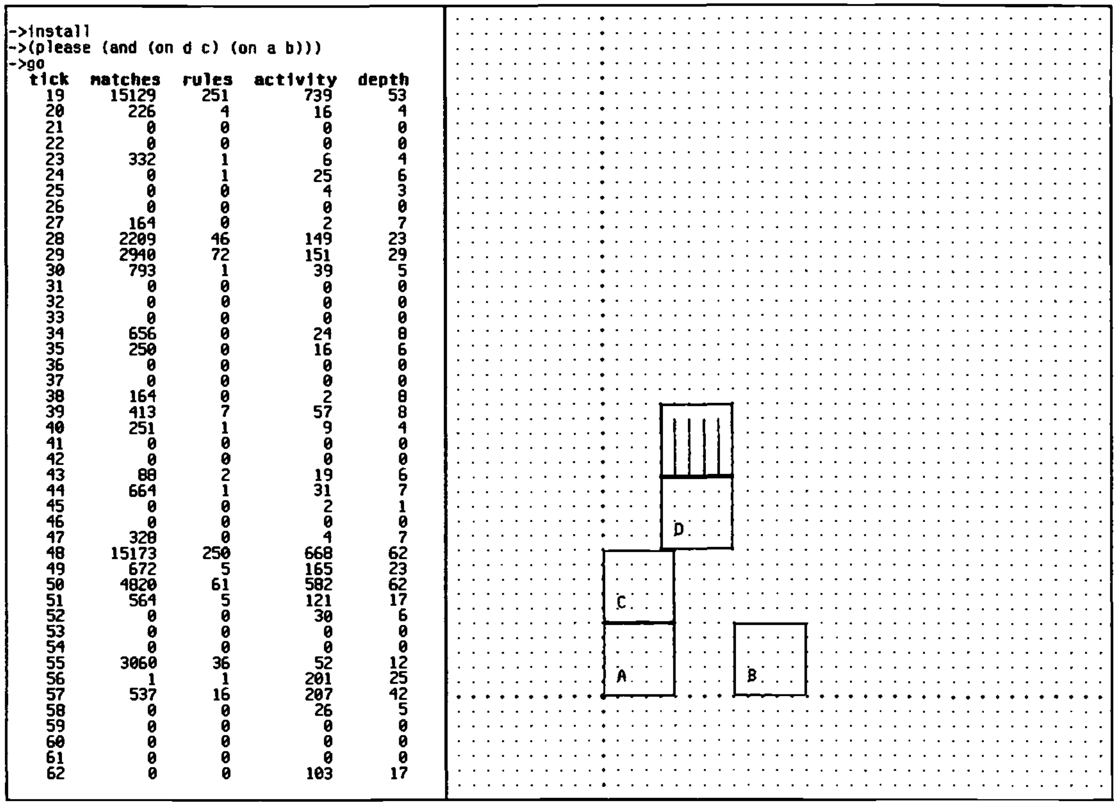
The hand moves to the left all the while in order to get itself centered on A. An extended argument concludes that it is all right to push the blocks out of the way. Although the line of reasoning for circumventing C was closely analogous to that for D, little transfer occurs because D and C are different individuals. Some general-purpose reasoning about getting around things did transfer, but not very much.

Finally the hand lands on A on cycle 78, just in time to push D and C completely off of A (Figure 10.5(i)). Some rules fire on cycles 79 and 80



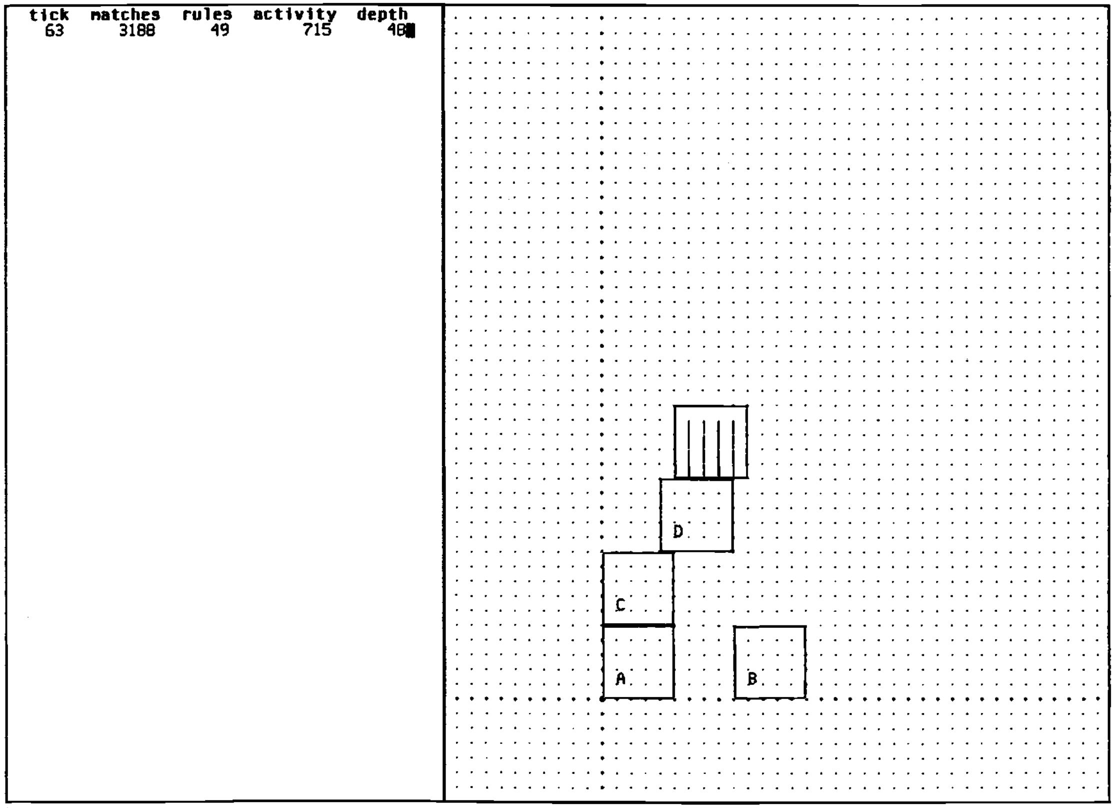
```
->install
->(please (and (on d c) (on a b)))
->go
 tick  matches  rules  activity  depth
   19    15129    251       739     53
   20      226      4        16      4
   21        0      0         0      0
   22        0      0         0      0
   23      332      1         6      4
   24        0      1        25      6
   25        0      0         4      3
   26        0      0         0      0
   27      164      0         2      7
   28     2209     46       149     23
   29     2940     72       151     29
   30      793      1        39      5
   31        0      0         0      0
   32        0      0         0      0
   33        0      0         0      0
   34      656      0        24      8
   35      250      0        16      6
   36        0      0         0      0
   37        0      0         0      0
   38      164      0         2      8
   39      413      7        57      8
   40      251      1         9      4
   41        0      0         0      0
   42        0      0         0      0
   43       88      2        19      6
   44      664      1        31      7
   45        0      0         2      1
   46        0      0         0      0
   47      328      0         4      7
   48    15173    250       668     62
   49      672      5       165     23
   50     4820     61       582     62
   51      564      5       121     17
   52        0      0        30      6
   53        0      0         0      0
   54        0      0         0      0
   55     3060     36        52     12
```

Figure 10.5(e). Shifting back to its pursuit of the first subgoal, the hand chases D as it falls.



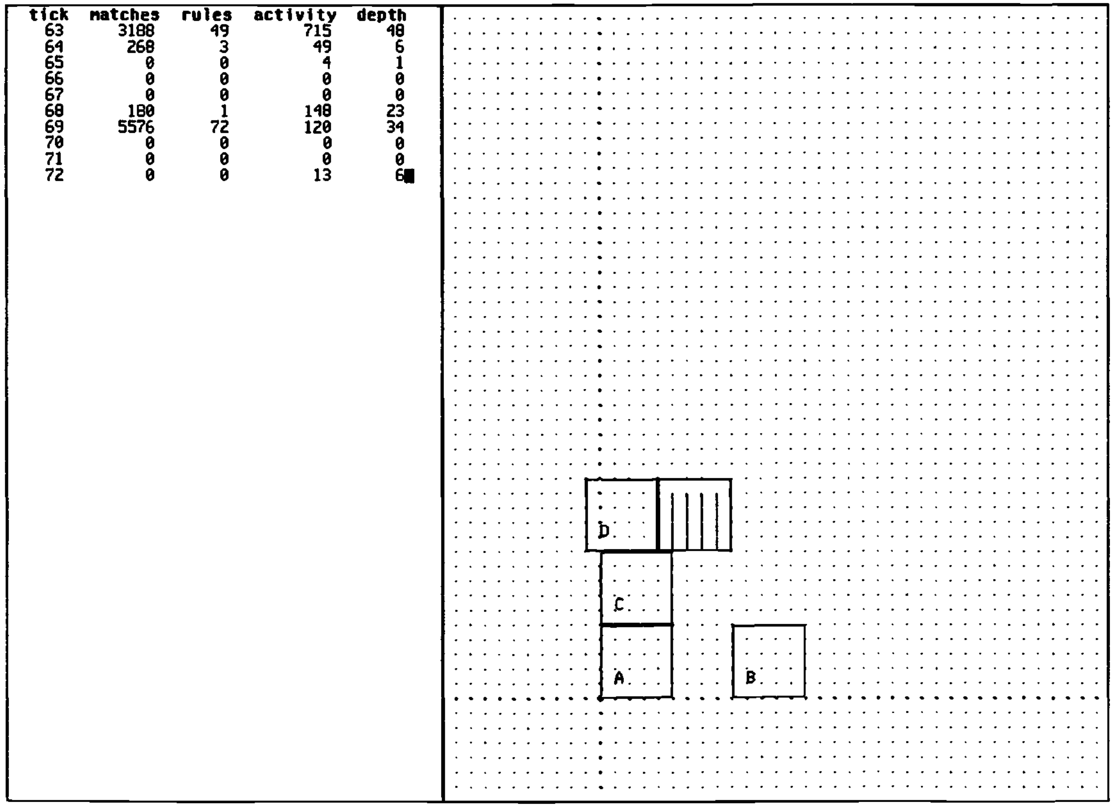
```
->install
->(please (and (on d c) (on a b)))
->go
  tick   matches   rules   activity   depth
   19     15129     251       739       53
   20       226       4        16        4
   21         0       0         0        0
   22         0       0         0        0
   23       332       1         6        4
   24         0       1        25        6
   25         0       0         4        3
   26         0       0         0        0
   27       164       0         2        7
   28      2209      46       149       23
   29      2940      72       151       29
   30       793       1        39        5
   31         0       0         0        0
   32         0       0         0        0
   33         0       0         0        0
   34       656       0        24        8
   35       250       0        16        6
   36         0       0         0        0
   37         0       0         0        0
   38       164       0         2        8
   39       413       7        57        8
   40       251       1         9        4
   41         0       0         0        0
   42         0       0         0        0
   43        88       2        19        6
   44       664       1        31        7
   45         0       0         2        1
   46         0       0         0        0
   47       328       0         4        7
   48     15173     250       668       62
   49       672       5       165       23
   50      4820      61       582       62
   51       564       5       121       17
   52         0       0        30        6
   53         0       0         0        0
   54         0       0         0        0
   55      3060      36        52       12
   56         1       1       201       25
   57       537      16       207       42
   58         0       0        26        5
   59         0       0         0        0
   60         0       0         0        0
   61         0       0         0        0
   62         0       0       103       17
```

Figure 10.5(f). Having caught D again, the hand puts it back on C.



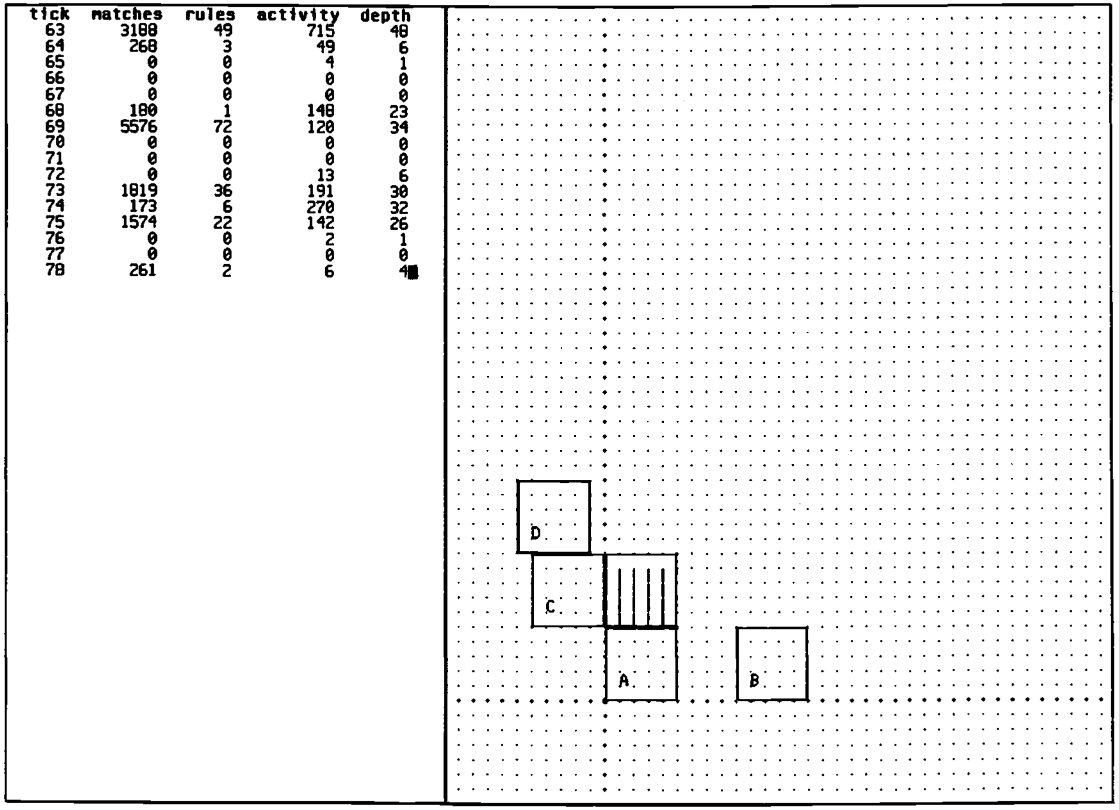
```
tick  matches  rules  activity  depth
 63     3188     49       715     48
```

Figure 10.5(g). This time the hand decides to go around to the right.



```
 tick  matches  rules  activity  depth
   63     3188     49       715     48
   64      268      3        49      6
   65        0      0         4      1
   66        0      0         0      0
   67        0      0         0      0
   68      180      1       148     23
   69     5576     72       120     34
   70        0      0         0      0
   71        0      0         0      0
   72        0      0        13      6
```

Figure 10.5(h). Heading for A, the hand strikes C.



```
tick  matches  rules  activity  depth
 63    3188     49      715      48
 64     268      3       49       6
 65       0      0        4       1
 66       0      0        0       0
 67       0      0        0       0
 68     180      1      148      23
 69    5576     72      120      34
 70       0      0        0       0
 71       0      0        0       0
 72       0      0       13       6
 73    1819     36      191      30
 74     173      6      270      32
 75    1574     22      142      26
 76       0      0        2       1
 77       0      0        0       0
 78     261      2        6       4
```

Figure 10.5(i). Working its way around C, the hand arrives on A and grabs it, pushing C and D off along the way.

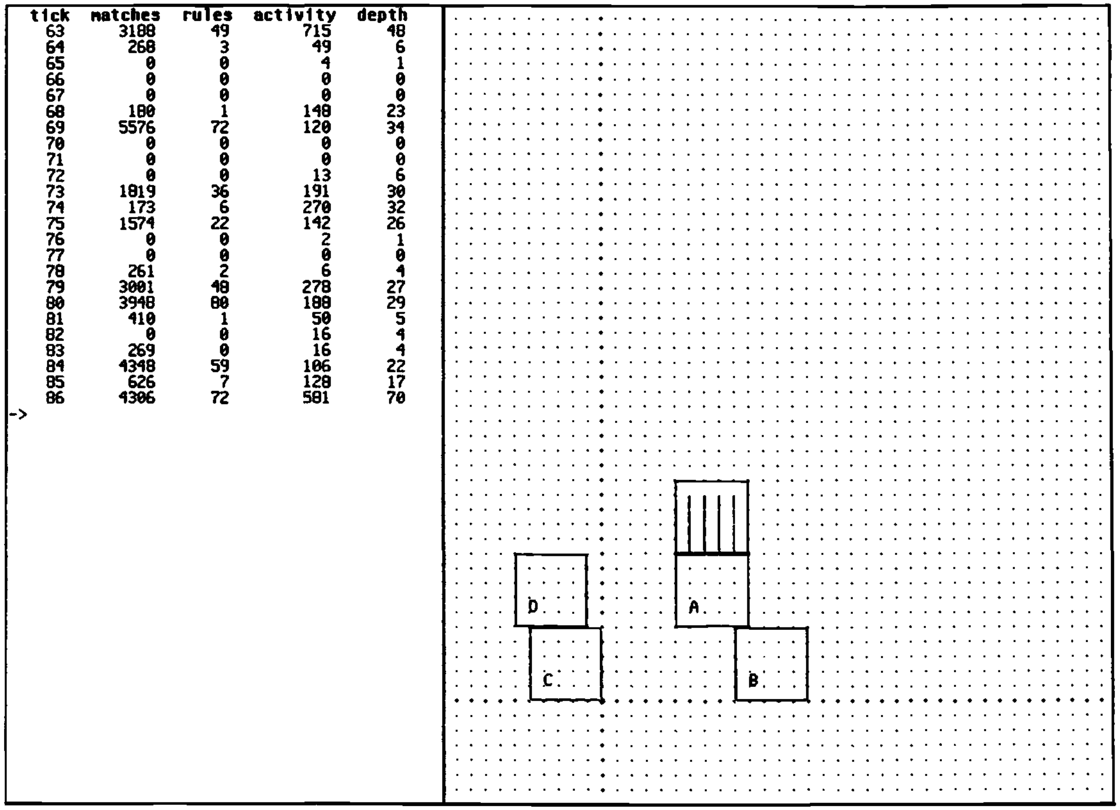
because the system has never picked up A or moved a block to the right before. (Left and right, like the block names, are individuals across which transfer also fails.) Some additional rules navigate A over top of B; this reasoning exhibits little transfer from previous such cases. Finally the system completes its task on cycle 86 (Figure 10.5(j)).

Before looking at the numbers in more detail, let us consider a sixth and final demonstration. This demonstration picks up from the fifth. The system has just finished with the compound task of putting D on C and A on B. The user has restored the blocks to their original positions and set the system running again. The same compound goal is in effect, so the system marches through precisely the same actions as before (Figures 10.6(a) and 10.6(b)). It takes a couple of rules to get moving. The activity numbers exhibit the usual burst–decay pattern. The system has not gotten any smarter about performing this task but it has gotten faster.

In both the second and sixth demonstrations, the system was running through a task it had already performed. In each case, the system went through the same reasoning and the same motions and ended up with the same results. Yet in each demonstration some stray rules were run at various points. In no case did the newly run rules change the system's behavior on that cycle. Instead, these rule firings resulted from a peculiarity of the relationship between the perceptual system, the dependency network, and the rule firing mechanism. On each new cycle the perceptual system assigns new values, either IN or OUT, to the proposition corresponding to each of the primitive percepts. The blocks do not move very much, so the new cycle's value is usually the same as the old cycle's value. When the value changes, the consequences of that change ripple through the dependency network. This network activity sometimes causes new rules to fire.

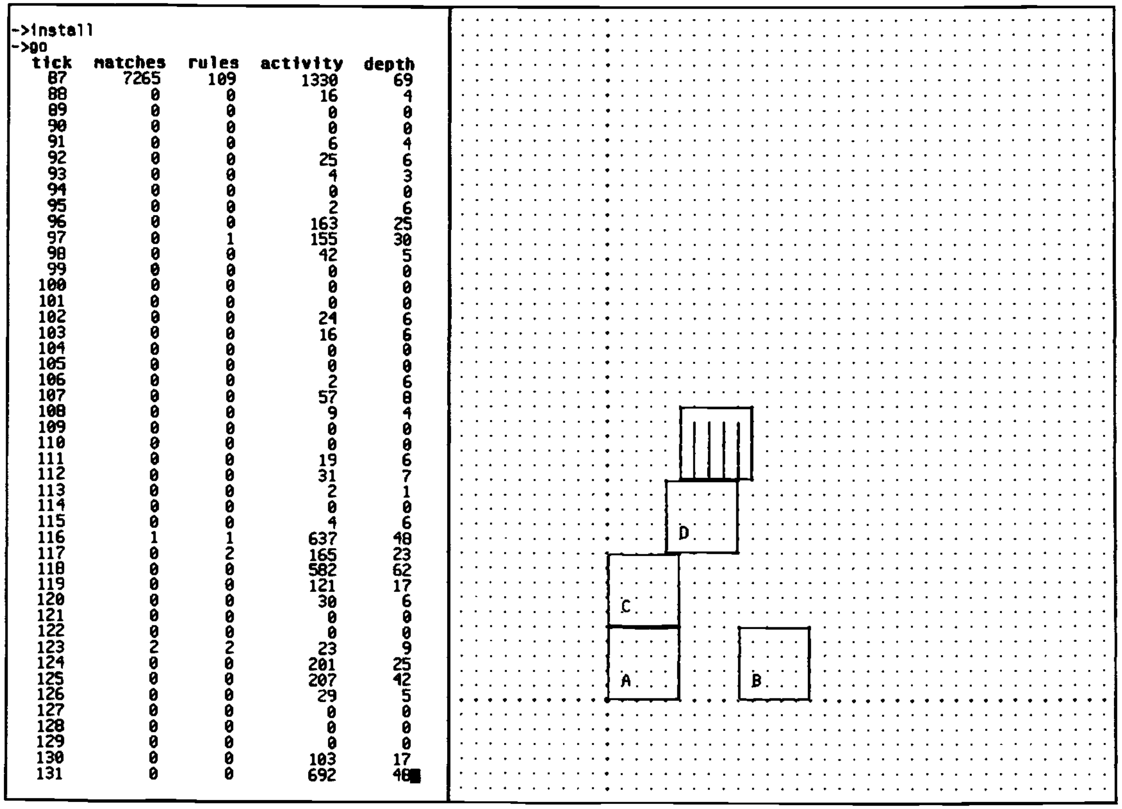
While the perceptual propositions are still being updated, the central system is effectively being told a false, or even inconsistent, story about the state of the outside world. As long as some of the perceptual propositions have been updated and others have not, rules will fire in an attempt to pursue the current goal in the improperly specified world state. The central system will restore itself to a consistent state once the perceptual propositions have all been updated, but only after a certain amount of extraneous activity.

It is impossible to avoid this problem completely. Since I do not believe



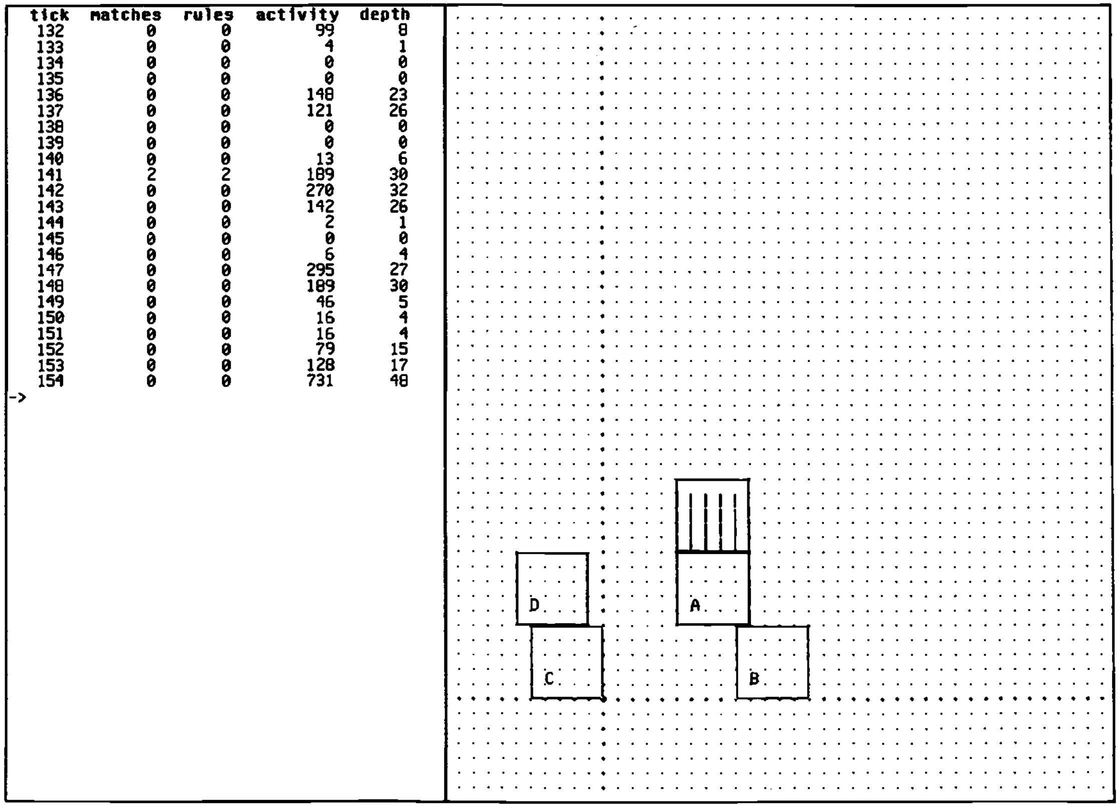
```
tick  matches  rules  activity  depth
  63     3188     49       715     48
  64      268      3        49      6
  65        0      0         4      1
  66        0      0         0      0
  67        0      0         0      0
  68      180      1       148     23
  69     5576     72       120     34
  70        0      0         0      0
  71        0      0         0      0
  72        0      0        13      6
  73     1819     36       191     30
  74      173      6       270     32
  75     1574     22       142     26
  76        0      0         2      1
  77        0      0         0      0
  78      261      2         6      4
  79     3001     48       278     27
  80     3948     80       188     29
  81      410      1        50      5
  82        0      0        16      4
  83      269      0        16      4
  84     4348     59       106     22
  85      626      7       128     17
  86     4306     72       581     70
->
```

Figure 10.5(j). Finally the system places A on B without incident.



```
->install
->go
  tick   matches   rules   activity   depth
   87     7265      109      1330       69
   88        0        0        16        4
   89        0        0         0        0
   90        0        0         0        0
   91        0        0         6        4
   92        0        0        25        6
   93        0        0         4        3
   94        0        0         0        0
   95        0        0         2        6
   96        0        0       163       25
   97        0        1       155       30
   98        0        0        42        5
   99        0        0         0        0
  100        0        0         0        0
  101        0        0         0        0
  102        0        0        24        6
  103        0        0        16        6
  104        0        0         0        0
  105        0        0         0        0
  106        0        0         2        6
  107        0        0        57        8
  108        0        0         9        4
  109        0        0         0        0
  110        0        0         0        0
  111        0        0        19        6
  112        0        0        31        7
  113        0        0         2        1
  114        0        0         0        0
  115        0        0         4        6
  116        1        1       637       48
  117        0        2       165       23
  118        0        0       582       62
  119        0        0       121       17
  120        0        0        30        6
  121        0        0         0        0
  122        0        0         0        0
  123        2        2        23        9
  124        0        0       201       25
  125        0        0       207       42
  126        0        0        29        5
  127        0        0         0        0
  128        0        0         0        0
  129        0        0         0        0
  130        0        0       103       17
  131        0        0       692       48█
```

Figure 10.6(a). After the system completes its two-part task, the blocks are restored to their original positions and the system is set running on the same task.



```
 tick  matches  rules  activity  depth
  132        0      0        99      8
  133        0      0         4      1
  134        0      0         0      0
  135        0      0         0      0
  136        0      0       148     23
  137        0      0       121     26
  138        0      0         0      0
  139        0      0         0      0
  140        0      0        13      6
  141        2      2       189     30
  142        0      0       270     32
  143        0      0       142     26
  144        0      0         2      1
  145        0      0         0      0
  146        0      0         6      4
  147        0      0       295     27
  148        0      0       189     30
  149        0      0        46      5
  150        0      0        16      4
  151        0      0        16      4
  152        0      0        79     15
  153        0      0       128     17
  154        0      0       731     48
->
```

Figure 10.6(b). This time the task requires only a trivial number of rules. The levels of activity in the dependency network still have a complex structure.

that people have rule systems in their heads, any attempt at alleviating the problem would be only an engineering curiosity. Some important issues would, however, arise in such a project. One is that it is unrealistic to hold the system still on cycles when hundreds of rules need to run. The whole system would be more honest if the dependency system ran at a fixed real-time rate, occasionally letting the rule system fall behind. Unless the goal is plain engineering, however, research into such matters should wait until their connection to dynamic issues is clarified.

Let us look once again at the depth numbers for the fifth and sixth demonstrations. Recall that the "depth" measures the longest causal chain in the just-completed cycle's modifications to the dependency network. These numbers are particularly large when the system is either switching from one subgoal to another or performing a complicated argument in a difficult situation. The deeper the change in the goal hierarchy, the deeper the network activity is likely to be. The reason for this is simple. The top-level reasoning about breaking the compound goal `(and (on d c) (on a b))` into two subgoals `(on d c)` and `(on a b)`, comes IN on the first cycle and remains IN through the entire process, finally going OUT on the final cycle once the compound goal is finally achieved. The reasoning for each goal in the hierarchy forms part of the support for the reasoning for its subgoals. The depth numbers are particularly high on the first and last cycles because the reasoning for the entire goal hierarchy is going IN and OUT. In general, when the system moves from one subgoal to another, the depth numbers will be proportional to the depth of those subgoals in the system's current goal hierarchy. Thus, on cycles 48, 50, and 63, when the system switches back and forth between its first and second major subtasks, the depths are 62, 62, and 48. (The two 62s are a coincidence.) As the intermediate goals switch back and forth the depth numbers assume intermediate values.

Analysis

As I explained at the beginning of the chapter, the goal of RA is to do technical justice to three slogans: that activity necessitates "knowing what you're doing," that it is a matter of "continually redeciding what to do," and that it is "mostly routine." For the system to approximate these ideals, it somehow had to produce the effect of a complex decision process on every cycle of a fairly rapid clock. The system proposed to achieve this effect by maintaining dependencies on all of its novel items of reasoning (rule firings). As a narrow technical matter, this technique is certainly a success.

Maintaining dependencies, though, is sufficient only if two conditions hold. First, it must turn out that almost everything you do is something you have done before. Although it is difficult to evaluate such a broad proposition in such a narrow and artificial domain as blocks world, I have posited that this first condition is a property of the everyday activity of human beings.

Second, dependency records must transfer to a sufficiently broad range of future situations. This second condition has been the principal focus of the analyses in this chapter. The results have been equivocal. On one hand, a dependency record generated in one situation cannot help but transfer to a broad class of other situations, for all the reasons discussed in Chapter 6. The exact patterns of transfer will depend on the kinds of arguments and goals that drive the system and on the properties of particular domains. Unfortunately, the system I have described does not exhibit nearly enough transfer among situations involving different individuals. It may not be clear exactly which analogies the system ought to exhibit, but it does fail in some clear-cut cases.

Although I have not dwelled on the point, the system also fails to exhibit any transfer dynamics that involve genuine learning. New insights about block stacking or cooking or driving depend on fortuitous circumstances, but once one has encapsulated an insight into a dependency record, it will be available whenever it is observed to apply. An insight might transfer to a different activity or to a different place in subsequent instances of the same activity. If a system's performance on a task has room for improvement, the dependencies ought to pick up any insights into the difficulty and transfer them to the moment when the system can use them to act differently. RA was intended only as a model of routine activity and was not designed to explain anything about learning. Still, it is striking that, even though the dependencies are working correctly, the system always performs the same actions no matter how many times it is assigned a given task.

Yet another failure, already noted in the context of the fifth demonstration of this chapter, is that the system requires too much laborious rule writing to make arbitrary choices. I changed the meaning of conjunctive goals in order to avoid writing these rules and I expect that such constrictions would plague a rule writer in any domain where the system was not so clearly focused on a single task as in blocks world.

I see these three problems – failure of transfer by analogy, failure of transfer learning, and the difficulty of making arbitrary choices – as symptoms of a mistake I made in building the system. Something is wrong with the system's perceptual apparatus and the way goals are specified. Perception works in this system in the traditional AI fashion. On every cycle the system is given a full specification of the current world situation in terms of a set of propositions such as

```
(on a table)
(on b a)
(left-of b c)
(above hand table)
(touching hand c)
(moving hand left)
(moving c left)
```

In particular, the system is given a situation (i.e., a situation description) in which all the individuals have names. But it is unrealistic to think that actual perceptual systems should be able to provide such information. For one thing, there will be a large amount of it. All the individuals and their spatial relationships in the room in front of me right now, for example, add up to a great deal of information. And it is odd that the situation description contains such constant symbols as A, B, C, HAND, and TABLE. What perceptual system knows what the objects it senses are called? Yet it has been standard AI practice for the input and output representations of plan-construction systems to employ such names.[2]

What really must be explained is why this practice has never seemed odd. One big part of the problem, I believe, can be discovered in the nature of blocks world. The use of blocks world has also been a common practice in the planning literature. A goal presented to a computational agent might read, `(put-on B C)`. One does not present the goal by aiming a TV camera at some blocks on the table, reaching one's hand into the scene, pointing, and saying, "Put this block on that block." As implausible as the `(put-on B C)` style of goal specification might be for much of everyday life, in blocks world it does not seem so intolerable, since children's toy blocks very often have large letters written on them. The diagrams in the literature depict the blocks in just this way. The block's name is not drawn outside the block and connected to it with an arrow. Instead, the diagrams put the name inside the block, as if one could read the names of things off of them. The names are effaced, and the effort of connecting names to things is elided. This convention would seem to suppress some hard work.

One might argue, however, as follows: "All of that is someone else's problem. We are simplifying the problem for ourselves by assuming that we are being given all this information. We aren't doing perception research; we are doing planning research." This line of argument, I believe, is critically misleading. Referring to the objects in an input representation through names is a simplification that actually makes things harder. One symptom of this trap is the failure of dependencies to make some reasonable transfers. Dependencies support transfer in a very simple way, whereas many programs have used much more complex machinery, such as pattern matchers and subgraph isomorphism algorithms, either to actually construct an analogy between two situations or to abstract away from the individuals of one situation to get a structure with variables in it that can then be instantiated in future situations.

Another approach is the mechanism of *chunking* (Rosenbloom 1983) used in the SOAR production system architecture (Laird, Newell, and Rosenbloom 1986; Laird, Newell, and Rosenbloom 1987).[3] Chunking is a technique for summarizing the consequences of a collection of production firings that have collaborated to solve some problem by locating a successful path through a problem space. The summary takes the form of a single large production rule, or "chunk," that collects the productions' initial premises and asserts their final consequences. Chunking is not intended to enable the system to solve any new problems, only to save it effort – often a considerable amount of effort – when it faces similar problems in the future.

Chunking and dependency maintenance are related concepts. The process of building the chunk is similar to tracing the dependencies of the productions' consequences. Like dependency maintenance, it is a learning scheme that works in the background without requiring the agent to deliberately go out and learn things. The goal in each case is to produce transfer effects (Laird, Newell, and Rosenbloom 1984). Whereas a simple dependency system such as that of RA need only record the dependencies in the form of digital circuitry (whether actual or simulated), the chunking mechanism actually inspects the dependency network, traversing a region of it to collect its premises. A chunk effectively connects these premises directly to their consequences, skipping the intermediate layers of network structure.

A system that relies on its dependency network to recapitulate old lines of reasoning, by contrast, need not short-circuit these intermediate layers, because the processing in each layer requires only a single gate delay. Chunking is necessary in a production-system architecture because the system must invest a great deal of effort to fire the productions: matching the patterns, selecting the correct productions, calculating their consequences, and asserting them. Since the productions in SOAR guide a symbolic search, summarizing their operation is all the more important since the productions may have been numerous and because many of them may have had no useful consequences as the system explored false paths in the search space.

Chunking faces a number of difficulties. One is that it is not always safe to collapse the internal structure of nonmonotonic reasoning, lest some novel assertion come along to invalidate an intermediate deduction. The system must somehow recover from the resulting overgeneralization. A second problem is that the chunks themselves tend to be too large. A chunk may summarize dozens of productions into one, but the savings will not be proportional, since that one may be expensive to use (Tambe and Newell 1988). Because it can be computationally complex to match productions with many clauses against a database, the SOAR project has explored schemes for restricting the expressive power of the production language. The most prominent approach is to require that a variable only appear once in the left-hand side of any given production. Although this constraint is restrictive for programmers, it appears difficult to improve upon (Tambe and Rosenbloom 1994). A third problem is a tension between chunking and the agent's real-time interaction with the world; the agent's reasoning may take place over a stretch of time, but the resulting chunk presupposes that all of its premises must hold true simultaneously (Agre 1993b: 443–447).

Chapter 11 will argue that a better solution lies in an account of representation that is more suitable for an embodied, situated agent. Instead of starting from a notion of objective individuals, it starts from a notion of focus and of the causal relationships between an agent and objects in its environment. The agent represents the objects not in terms of their objective identities but in terms of their indexical and functional relationships to the agent's body and ongoing projects. Such an account of representation has many virtues compared with conventional objective accounts. In particular, it does not involve variables and is thus far more compatible with simple central system machinery and with dependency maintenance.

RA keeps track of the objects in its environment by maintaining an exhaustive representation of them – a world model of the traditional sort. World models have advantages and disadvantages. Their principal advantage is that they are complete. Their most obvious disadvantage is that this completeness is difficult to obtain, both epistemologically and computationally, in an environment of any real complexity. But a more subtle disadvantage is that a world model suppresses the natural relationship of focus between an agent and its environment. Whereas a person can face in only one direction and focalize only one part of the resulting visual display at a time, a world model gives equal billing to everything that exists in the known world. The world model supports a kind of homogeneity in which everything is represented independently of the agent's current goals, its current practicalities of perception and motion, and indeed even of its existence. As far as the process of maintaining the model is concerned, the system as a whole is always involved equally with every individual in the world, every property of these objects, and every relation among them. Inasmuch as computational effort is a resource to be distributed, however, the agent's architecture must provide mechanisms that focus computational resources on the parts of the world model that correspond to the currently relevant parts of the world itself. In standard technical practice, this internal focusing mechanism is implemented with the aid of variables and constants. An alternative, I will argue, is to deny world models a central architectural role and to give greater weight to the natural relationship of focus that the world-model scheme suppresses.

The necessity of focus has substantial consequences for architecture. In particular, it suggests a resolution to the problem of arbitration. There are usually many things you could be doing. Often a major reason you choose one of them is that it is the first one you laid eyes on. Or perhaps it was the one that was ready to hand. When it does not matter very much what you choose, these arbitration schemes are as good as any. Usually you need not even become aware of the existence of a choice.

The theme of focus also helps in understanding RA's failure to alter its behavior through the learning. The point is subtle. People sometimes arrive at understandings on Tuesday that would have changed their actions on Monday, if only they had known of them. For example, someone might water a garden on Monday by running a hose from a spigot a hundred feet away, only to discover a much closer spigot during a stroll on Tuesday. When it is time to water the garden again, the new information might lead to different and simpler methods. Alternatively, the routine for watering the garden might evolve in a more complex fashion, along the lines I described in Chapter 6. Such things, however, never happen to RA. They *could* happen – nothing about RA's architecture prevents them from happening. One problem is that RA, courtesy of its always-updated world model, has complete knowledge of its world. It cannot discover anything because it already magically knows everything.[4]

Blocks world also tends to reinforce mistaken views of representation because all blocks look alike. The blocks on a blocks-world table, unlike the tools and materials of most concrete activities, do not lend themselves to readily perceptible functional distinctions. A spatula plainly affords pancake flipping, a pancake plainly affords being flipped with spatulas, a hammer plainly affords nail pounding, a nail plainly affords being pounded with a hammer, and your hand plainly affords all manner of things once it is suitably cupped or flattened or clenched. Each of these things carries more than enough information on it to deduce its relevance to a given activity. Blocks-world blocks, by contrast, are simple, bleak squares with letters in them. All blocks look alike, regardless of the functional roles they might play in particular activities. Blocks do afford grabbing and stacking. But if someone says, "Please stack block A on block B," nothing about either block will signal its role as the-block-to-stack or the-block-to-stack-it-onto. Lacking meaningful cues, one has no choice but to memorize arbitrary names.

There is a valuable lesson here. AI people normally choose to demonstrate their new technological ideas in those domains that made the underlying ideas look most obvious. This practice is reasonable enough, whether for its heuristic value, or for ease of exposition, or because one's intention is to solve engineering problems one at a time rather than produce a theory that explains everything all at once. The people who invented the blocks world did so because it was a simple place to demonstrate some complex and poorly understood forms of reasoning about subtask ordering.[5] Once blocks world was written into textbooks and taught to students, however, many of its assumptions became invisible. One of these invisible assumptions was the availability of a world model in which objects are automatically labeled with their names. It is helpful to "read" a domain critically and ask what biases it has and how it is atypical of human activities. It also helps to experience the domain yourself and compare it with the task being posed to programs. All too often, a program runs into trouble in precisely those areas in which the program's task fails to correspond to any real task. I interpret the technical complexities of pattern matching, for example, as a symptom of ways in which domains have been accidentally falsified. Perhaps the skill of making and testing such interpretations can help get at the essence of activity in the world.

11 Representation and indexicality

World models

As an agent gets along in the world, its actions are plainly *about* the world in some sense. In picking up a cup, I am not just extending my forearm and adjusting my fingers: those movements can be parsimoniously described only in relation to the cup and the ways that cups are used. A conversation about a malfunctioning refrigerator, likewise, really is *about* that refrigerator; it is not just a series of noises or grammatical constructions. When someone is studying maps and contemplating which road to take, it is probably impossible to provide any coherent account of what that person is doing except in relation to those roads.

AI researchers have understood these phenomena in terms of representations: actions, discussions, and thoughts are held to relate to particular things in the world because they involve mental representations of those things.[1] It can hardly be denied that people do employ representations of various sorts, from cookbooks and billboards to internalized speech and the retinotopic maps of early vision. But the mentalist computational theory of representation has been simultaneously broader and more specific. In this chapter I will discuss the nature and origins of this theory, as well as some reasons to doubt its utility as part of a theory of activity. Chapter 12 will suggest that the primordial forms of representation are best understood as facets of particular time-extended patterns of interaction with the physical and social world. Later sections of the present chapter will prepare some background for this idea by discussing indexicality (the dependence of reference on time and place) and the more fundamental phenomenon of intentionality (the "aboutness" of actions, discussions, and thoughts). A detailed survey of AI theories of representation would be impossible in a small space, since these theories do not constitute a unified doctrine but a family relationship among numerous variants. By painting the big picture, I hope to make this extended family of theories comprehensible as a series of attempts to work through the practical logic of mentalist AI research.

AI vocabulary mixes visual and linguistic metaphors for representation; in linguistic terms, the purpose of representation is to "express" states of affairs. This conception of representation has a long history and entails some strong assumptions about the broader phenomena of which representation is a part. Important assumptions are encoded in the notions of *world models* and *expressive power,* both of which reflect a view of knowledge as a picture, copy, reflection, linguistic translation, or physical simulacrum of the world. This conception of knowledge is found in many contexts (G. McCulloch 1995; Rorty 1979). Discourses and practices of representation in AI have developed through two main lines of intellectual descent. One of these, which later sections of this chapter will discuss in detail, is modern formal logic, particularly model theory.

In areas such as planning, automatic diagnosis, and machine vision, however, the elaborate mathematical and rhetorical machinery of model theory is generally replaced by, or overlaid with, the looser and less systematic notion of a world model. The notion of a world model has already been discussed in Chapter 10. It has no consistent definition, but in each case it refers to some structure within the mind or machine that represents the outside world by standing in a systematic correspondence with it. Rough though it is, the notion of a world model has a clear history whose roots lie in the metaphor of knowledge as an inner reflection of the outer world.[2] According to the modern computational notion of the idea, reasoning is a matter of using one's world model to perform, by means of mental inference, a simulation of the outside world. This form of the world-model theory first appeared in its full-blown form in Kenneth Craik's 1945 book, *The Nature of Explanation,* and his explanation is worth quoting at length:

One of the most fundamental properties of thought is its power of predicting events. This gives it immense adaptive and constructive significance as noted by Dewey and other pragmatists. It enables us, for instance, to design bridges with a sufficient factor of safety instead of building them haphazard and waiting to see whether they collapse, and to predict consequences of recondite physical or chemical processes whose value may often be more theoretical than practical. In all these cases the process of thought, reduced to its simplest terms, is as follows: a man observes some external event or process and arrives at some "conclusion" or "prediction" expressed in words or numbers that "mean" or refer to or describe some external event or process which comes to pass if the man's reasoning was correct. During the process of reasoning, he may also have availed himself of words or numbers. Here there are three essential processes:

1. "Translation" of external processes into words, numbers or other symbols.
2. Arrival at other symbols by a process of "reasoning," deduction, inference, etc., and
3. "Retranslation" of these symbols into external processes (as in building a bridge to a design) or at least recognition of the correspondence between these symbols and external events (as in realizing that a prediction is fulfilled).

One other point is clear; this process of reasoning has produced a final result similar to that which might have been reached by causing the actual physical processes to occur (e.g. building the bridge haphazard and measuring its strength or compounding certain chemicals and seeing what happened); but it is also clear that this is not what has happened; the man's mind does not contain a material bridge or the required chemicals. Surely, however, this process of prediction is not unique to minds, though no doubt it is hard to imitate the flexibility and versatility of mental prediction. A calculating machine, an anti-aircraft "predictor," and Kelvin's tidal predictor all show the same ability. In all these latter cases, the physical process which it is desired to predict is *imitated* by some mechanical device or model which is cheaper, or quicker, or more convenient in operation. Here we have a very close parallel to our three stages of reasoning – the "translation" of the external processes into their representatives (positions of gears, etc.) in the model; the arrival of other positions of gears, etc., by mechanical processes in the instrument; and finally, the retranslation of these into physical processes of the original type.

By a model we thus mean any physical or chemical system which has a similar relation-structure to that of the process it imitates. By "relation-structure" I do not mean some obscure non-physical entity which attends the model, but the fact that it is a physical working model which works in the same way as the process it parallels, in the aspects under consideration at any moment. Thus, the model need not resemble the real object pictorially; Kelvin's tide-predictor, which consists of a number of pulleys on levers, does not resemble a tide in appearance, but it works in the same way in certain essential respects – it combines oscillations of various frequencies so as to produce an oscillation which closely resembles in amplitude at each moment the variation in tide level at any place. (50–52; emphasis in the original)

For Craik, then, reasoning meant internal manipulation of models of the external world. The prototype for this manipulation is the simulation of a physical system (in Craik's terms, predicting it through imitation), though other kinds of manipulation are possible. Perception supports this process by translating outside reality into internal representations.

Although the vocabulary has changed, this conception has organized nearly all subsequent work. For Newell (1990), for example, it takes the form of the "Great Move":

> Instead of moving toward more and more specialized materials with specialized dynamics to support an increasingly great variety and intricacy of representational demands, an entirely different turn is possible. This is the move to using a neutral, stable medium that is capable of registering variety and then *composing* whatever transformations are needed to satisfy the requisite representational laws. (61, emphasis in original)

Newell describes the use of this generalized medium for building world models in terms analogous to those of Craik. Instead of "translation" and "retranslation," Newell uses "encoding" and "decoding." The transformations that these representations can undergo need not amount to simulations of the world, and nothing about Newell's architecture requires that they be treated as simulations (Agre 1993b: 421–423), but in practice the paradigm cases are simulations nonetheless: inferences about what the world will be like if a given action is taken, or a given operation is applied, in a given situation.

Likewise, in computer vision research it is often said that the purpose of human visual machinery is to reconstruct a model of the visual world from the information available in retinal images (e.g., Marr 1982: 295–328). As mentioned in Chapter 3, this process is known metaphorically as "inverse optics" because the computations effectively invert the physical processes that produced the image in the first place. The intuition behind this approach is clear enough: real images include extraordinarily complex patterns of light intensities, and their interpretation will inevitably be ad hoc without an understanding of the processes that created them (Horn 1986).

World models are the epitome of mentalism. On its face, the idea seems implausible: a model of the whole world inside your head. The technical difficulties that arise are obvious enough on an intuitive level. First and most obviously, world models are cumbersome. They require a great deal of storage, regardless of what implementation technology might be used. Second, it is necessary to keep the model up to date as the world changes; this takes effort and, it would seem, requires the agent to be everywhere at once so that changes do not go undetected. Third, it is computationally expensive to perform computations with world models. This computational expense can sometimes be quantified; when the model represents three-dimensional geometry, for example, the necessary calculations can be provably intractable (Hopcroft and Kraft 1987).

The problems with world models have taken particular forms in planning research. World models have been intimately connected with planning ever since Miller, Galanter, and Pribram (1960) introduced Plans as a complement to Boulding's (1956) concept of an Image. Boulding's concept was not at all technical; it relied heavily on the vernacular meaning of the word "image" to suggest that we organize our lives by building and inspecting mental images of relevant parts of the world. In the planning literature, the world model permits an agent to anticipate the effects of its actions – in other words, again, to simulate the world. This task immediately gives rise to the *frame problem:* the problem of determining which (few) parts of the world will be affected by an action and which (many) parts will not (Ford and Hayes 1991; Pylyshyn 1987; Toth 1995).

Planning researchers, of course, are well aware of the difficulty of building and maintaining world models. The issues are rarely addressed in a general way, and most theories simply assume that the world model is automatically kept up to date.[3] Although the issues are not secret, neither are they regarded as a crisis for the research program. As long as underlying metaphors of mentalism remain unquestioned, the technical difficulties that arise in building world models can only be interpreted as inescapable. Human planning, after all, seems to provide an existence proof, and therefore, it seems, future technological progress must be capable of closing the gap between the systems' current performance and their potential. The difficulties, moreover, do not become manifest as brick walls that stop the research in its tracks. Rather, they produce an endless maze of trade-offs that new research can explore and characterize. A given research project might have its agent create plans in a simplified microworld that can be easily represented. It can explore techniques for storing only those aspects of the world model that are relevant. Or it can employ representation schemes that ensure that the necessary inferences follow easily. The resulting technical methods might find practical application in suitably structured environments, and it is impossible to demonstrate that further research can never find its way through the maze of trade-offs to a satisfactory theory of human activity. It *is* possible, though, to cultivate alternative intuitions about activity that make these trade-offs seem more explicable and more daunting – not as limits of computation per se, but as limits of mentalism.

Knowledge representation

Computational theories of representation, as I mentioned earlier, descend historically along two lines. One of these lines is mentalistic; it is concerned with world models. The other line, which begins with Frege, is Platonic. Frege's (1960 [1892], 1968 [1918]) central concept is *Sinn,* normally translated as "sense." Frege's theory of sense is peculiar, and its peculiarity explains a great deal about subsequent intellectual history.

A sense is an abstract entity that, roughly speaking, captures those aspects of a representation that determine whether it is true. A sense is thus not itself a representation or a symbol or a component of a psychological mechanism; it is, rather, the content that those things might express. For example, if I notice that my car is idling badly one morning, the sense of that observation includes the day and time, the car itself, and the condition of idling badly. In other words, in Frege's famous dictum, sense determines reference. If I notice that my car is idling badly again tomorrow, that will be a different sense. If several people simultaneously notice that their respective cars are idling badly, those will be different senses as well. But if several people simultaneously notice that *my* car is idling badly, each of those observations will have the same sense. And if I happen to remember tomorrow that my car had been idling badly today, that memory will have the same sense as my observation about my car today. Even though they encode particular objects and times, senses are eternal; they are never created or destroyed or modified, and they have no location. That is why many different people can say things that express the same sense, and the same person can say many different things that express the same sense.

In this way, a sense seems like a Platonic object, completely independent of any person's life. But it is a strange Platonic object, since it incorporates – or at least is capable of uniquely determining – specific concrete objects, such as my car, in specific places and times. And on top of all this, Frege also wishes to say that senses are essentially the same as thoughts (*Gedanken*); the theory of sense is also a theory of cognitive content (Burge 1979; Perry 1977). Frege would not wish to locate my thoughts inside my head, so his theory is not mentalistic. But he does wish to say that the thoughts are *my* thoughts, and not necessarily yours.

By this point, we should be able to discern the underlying drama: the frantic and contradictory attempt to close a gap between the world of abstractions and the world of concrete activity (Nye 1990). Yet Frege's theory has had tremendous influence because it provides a precise foundation for formal logic. The central idea here is *compositional semantics.* As a representation, a sentence of logic differs from a photograph in being assembled from a standardized set of discrete elements according to fixed rules. But a sentence of logic and a photograph are analogous in that each represents its object through a point-by-point correspondence. In logic, each discrete element of the logical sentence corresponds to an element of reality, with the sentence's logical structure corresponding to the relationships among these elements. In the philosopher's standard example of a sentence, "ON(CAT,MAT)," the symbol CAT corresponds to some actual cat, the symbol MAT corresponds to some actual mat, and the predication of ON corresponds to a physical relationship between the cat and the mat. The logical expression "ON(CAT,MAT)" is intended as a precise reformulation of an English sentence such as "The cat is on the mat," although Frege insisted that natural language semantics was simply one application of his theory of sense.

This kind of semantic scheme, which also grew from the work of Russell and Wittgenstein, is called a compositional semantics because the sense (or, for other theorists, the meaning) of the whole sentence is composed from the senses (or meanings) of the individual symbols. Model theory (Chang and Keisler 1973) is a mathematical formalization of compositional semantics, starting with Tarski in the 1940s. A *model* of a given set of logical sentences is a mathematical structure within which every one of the sentences (and every one of their deductive consequences) can be assigned an interpretation that makes it true. The theorems of model theory provide a precise account of the intuition that the more one knows, the less uncertainty remains as to the structure and properties of the world. If a set of logical sentences has no models at all, it is logically inconsistent; if it has numerous models, model theorists investigate the relationships among them. For model theory, the model is neither a mental representation nor the everyday world of cats and mats, but rather a mathematical "world" constructed using set theory. More complex forms of model theory, based on the semantics of modal logics proposed by Kripke (1963), envision not a single such model but a whole network of models. Once one has accepted the premises underlying the logical theory of representation, model theory is a useful tool for ensuring the formal coherence of a logical scheme. Still, since model theory speaks only of mathematical structures, it has no direct bearing on the question of how an agent might interact competently with the actual, concrete world of its routine activities.

This background helps explain the criteria that guide computational research on representation. If the purpose of representation is to express states of affairs in the world, the principal criterion on representational languages is "expressive power."[4] For example, one might invent a new representation language to express temporal and causal relationships, certainty and uncertainty of beliefs, default assumptions when definite knowledge is lacking, or beliefs about the mental states of other people. One demonstrates the power of these languages by exhibiting their capacity to express certain scenarios that previous languages cannot. Having formulated a mathematical semantics for one's new language, one may proceed to realize it inside a computer, probably using conventional data structures and pointers, and presumably building computational facilities that perform deduction within the new language.

This is the paradigm for AI research on *knowledge representation.* As this research has developed, its focus has shifted steadily from implementation to abstraction. The earliest work on *semantic networks* (Quillian 1968) involved cognitive models based on associational theories of memory. In the early 1970s, this concern with mechanisms was displaced by a concern with formal semantics, largely due to Woods (1975). With this paper and the subsequent work on such ambitious representation languages as KRL (Bobrow and Winograd 1977) and KL-ONE (Brachman and Schmolze 1984), the term "network" began to refer to a notation and not an implementation scheme. For this reason and others, the architectural concerns that are central to the work of Minsky (1985) and the connectionists (e.g., Feldman 1986; Rumelhart and McClelland 1986) largely disappear from research on knowledge representation. With the publication of Hayes (1977) it became a commonplace that a semantic network is at best a notational variation of an extended first-order logic. Although issues of computational efficiency have remained current topics of representational research, efficiency has been understood not in the architectural terms of parallel realization but in terms of the computational complexity of basic problems in logical deduction (Brachman and Levesque 1981).

Throughout this history, the principal motor of technical evolution has been the demand for mathematical formalization of representational ideas. This demand, despite its virtues, has been a conservative force, favoring formally precise elaborations of conventional ideas about representation over critical examination of their premises. In retrospect, theorists such as Quillian were trying to pull a theory of disembodied representation into the causal order of a working brain, if not exactly an embodied agent. Later theorists, driven by the traditional imperatives of semantical research, moved back toward Fregean Platonism. To get beyond these imperatives, it will be necessary to trace more precisely how philosophers, linguists, and sociologists have negotiated the difficulties with the conventional view of representation. This investigation will call in turn for some reexamination of the purpose of representation, and specifically its relationship to the more fundamental phenomenon of intentionality.

Indexicality

A representation is indexical when its truth depends on the occasions of its use. The canonical examples of indexicality are the words "I," "here," and "now," since the person, place, and time referred to by an utterance of the sentence "I am here now" will usually depend on the circumstances in which it is uttered. Verb tense is indexical as well; "I am eating" might be true at one time and false at another. The great virtue of indexical language, and of indexical representations generally, is the quality Barwise and Perry (1983: 5–6) call *efficiency.* This has nothing directly to do with engineering notions of efficiency, in the sense of the quantifiable efficacy of a technical method. Instead, it refers to the capacity of an indexical representation to refer to different individuals on different occasions. For example, the utterance "Can I help you?" or a sign reading "Back in five minutes" will pick out different individuals in different circumstances. In these cases, the same representational token seems to be doing a variety of jobs, adapting itself in an agile fashion to the demands of the situation. The theoretical challenge is to account satisfactorily for the relative contribution of these linguistic forms and of the circumstances of their use to the determination of what they express.

The phenomenon of indexicality is a significant challenge to conventional theories of representation because it ties the workings of language to the circumstances of language use much more intimately than the Fregean and Tarskian theories would seem to allow. Over the past twenty years, indexicality has become a topic of extensive research. This research takes place in the fields of philosophy, linguistics, and sociology, each of which has found the phenomena of indexicality unavoidable in its particular inquiries. Briefly tracing the issues as they have emerged in these fields will illuminate the critical question of the role of representation in situated activity.

Frege was aware of the problem posed by indexicals.[5] He held that an utterance such as "I am eating" has a different sense when produced by different people, or by the same person on different occasions. (This, as Burge [1979] points out, is why sense is different from meaning; that utterance surely has the same meaning regardless of the circumstances of utterance.) Furthermore, an utterance such as "The weather is hot today," produced today, might be regarded as having the same sense as "The weather was hot yesterday," produced tomorrow. (These two utterances surely have different meanings.) But Frege's theory immediately breaks down at this point. Burge explains why:

> The problem is that it seems intuitively implausible that a person who uses proper names and indexical constructions always has or grasps abstract thought components (I shall call them "concepts") that are sufficiently complete to determine uniquely and in a context-free way the things he refers to. (1979: 425)

For example, people routinely employ the present tense (or say "now") without having any way of knowing what time it is. Likewise, people routinely say "here" without knowing where they are. That, after all, is part of the purpose of indexicals: since their reference is relative to the speaker, the speaker does not need to know what the reference is in absolute terms. Thus, Burge concludes:

> Frege appears to be caught between two objectives that he had for his notion of sense. He wanted the notion to function as conceptual representation for a thinker (though in principle accessible to various thinkers, except in special cases). And he wanted it uniquely to determine the referents of linguistic expressions and mental or linguistic acts. The problem is that people have thoughts about individuals that they do not individuate conceptually. (426)

Frege's theory fails because it attempts to blur the Platonic realm of abstract ideals with the concrete realm of epistemically situated representation use. It fails, in short, because Frege wants to eliminate the body from his theory of thought.

In fact, Frege tries to eliminate much more from his theory. Whereas sense is eternal and unlocalized, reference pertains to the concrete, historical relationships between particular representational contents and particular things. After many false starts, philosophical work on reference has established that reference is regularly achieved by means of complex, indirect, and contingent social processes by which utterances and thoughts are causally connected to their referents (Donnellan 1966; Kripke 1980; Putnam 1975a).

In their attempt to rescue compositional semantics from this impasse, Barwise and Perry (1983) retained the model-theoretic framework and broadened the notion of sense. Following Kaplan (1989 [1977]), they observe that facts about the world determine the truth value of an utterance in two ways: (1) by assigning an interpretation to the utterance (e.g., the meaning of "You got home late last night" depends on who is speaking, who is being addressed, the current time, etc.) and (2) by determining whether that interpretation is actually true (i.e., whether the addressee actually got home late). More formally, the meaning of a sentence is a relation between two *situations:* the situation in which the sentence is uttered and the situation that the utterance is *about.* This is a substantial advance over the traditional theory because it gives a central place to a range of indexical phenomena long marginalized by the philosophy of language (Evans 1982; Smith 1986). Barwise and Perry effectively reverse the hierarchical opposition in Frege's theory: whereas Frege treats context-independent meaning as central and indexicality as marginal, Barwise and Perry treat efficient expressions as the paradigm of meaningful language.

Nonetheless, the purpose of Barwise and Perry's theory is not to account for the connections between representation use and actual, concrete activities. Instead its purpose is to provide a framework that reflects in formal terms the systematic differences in meaning among various utterances. Barwise and Perry's semantics thus entails a metaphysical realism, according to which it is possible for a semantic theorist to enumerate in some objective way the ontology of a concrete situation of representation use, before the event (Winograd 1985). In this sense, their position is even stronger than that of the main Fregean tradition (Barwise and Perry 1985).

The most important limitation of Barwise and Perry's semantic theory is not their realism as such, but rather the unfinished work that their realism obscures. When a speaker uses an indexical term such as "I," "you," "here," "there," "now," or "then" to pick out a specific referent, this picking out is determined by relations between situations; it is not an *act* on the speaker's part. Consider, for example, a farmer who tells a farmhand, "No, put it there" while nodding vaguely off to the right. The farmhand will probably be quite capable of interpreting the "it" as referring to the bale that the farmhand is carrying, and "there" to refer to a region in the yard that the farmer had used for stacking bales during the preceding harvest. More generally, as W. Hanks (1990) observes, the language and customs of every culture have their own supply of potential referents for "I," "you," "here," "there," "now," and "then." When the participants in a linguistic interaction manage to refer to the same people, places, and times, this is a complex, shared achievement of the participants (Kronfeld 1990; Silverstein 1976). It is also, as Garfinkel (1984 [1967]) would insist, an achievement that is only good enough for practical purposes; individuals may well have different views of the precise boundaries of the "there," "then," or "them" being referred to, but if the discrepancy causes no trouble, it will most likely pass unremarked. A model-theoretic account of this achievement such as Barwise and Perry's can *posit* the potential and actual referents of indexical terms by constructing the appropriate situations, but it cannot explain the actions by which particular people picked out those particular referents. It is only through study of the actual practices people employ to achieve reference in situ that indexicality begins to emerge not merely as a passive phenomenon of context dependence but as an active phenomenon of context construction.

This conclusion is strikingly analogous to the development of planning research that I traced in Chapters 8 and 9. In each case, research began with a distinction between an abstract realm (the planner's simulations of the world, the world model, the Platonic realm of meaning) and a concrete realm (the executor carrying out the plans, the world being modeled, the objects being referred to). And in each case, the research evolved historically, seemingly driven against its will toward conceiving its opposed terms as intertwined. Although obvious in retrospect, this intertwining – between inside and outside, between abstract and concrete – has rarely been acknowledged. Instead, it has taken place in obscure corners of a theory, or in unreasonable assumptions that later theorists must struggle to iron out, or in technical complexities that remain on the agenda for future research. These are the margins of mentalism. More precisely, they are the margins of particular proposals within the sprawling family tree of mentalism. Mentalism deconstructs itself in these margins, and so it is understandable that mentalist researchers have kept reworking their theories until those margins have become invisible. In the end, though, the accumulated experience of the research community – the "feeling for the organism" that arises and evolves through attempts to get things working – creates the conditions in which the margins can be seen for what they are. Attempts to build minds have finally made clear, in the irreducibly intuitive way in which technical people can possibly know such things, that human beings are not minds that control bodies. It is therefore worth exploring the inverse proposition: that interaction is central, both to human life and to the life of any agent of any great complexity. Doing so, however, requires a different vocabulary – a vocabulary that does not define all issues in terms of the relationships among a mind, a body, and an outside world. It also requires that we reckon with a kind of vacuum – the vacuum that opens up when we can no longer use the technical schemata of mentalism to build new mechanisms or explain how they work.

Intentionality

The preceding section has described the tension that the phenomenon of linguistic indexicality has introduced into accounts of representation, and the resulting movement toward a view of human beings as active subjects deeply embedded in their familiar environments. My topic here is not linguistic indexicality as such but this deeper phenomenon of embedding. How are human activities structured and what role do representations play in them?

To investigate these questions we must turn to the European phenomenological tradition, which developed an account of representation as complex and derivative in relation to the originary phenomenon of intentionality. Intentionality is a broader concept than representation. It covers all types of "aboutness" or "towardness" that might be exhibited within any exercise of agency in relation to things in the world. Some examples might include talking about a house for sale, picking up a cup to drink from it, avoiding an obstacle in the street, remembering an incident at work, or playing a song on a guitar. In each case, one's actions are oriented toward certain things (a house, a cup, an obstacle, the participants in some incident, a song, a guitar), and no useful description of the events in question could fail to mention these things. That is, one might describe a cyclist's avoidance of an obstacle in terms of the maneuver's precise spatial trajectory and without any reference to the obstacle itself, but this description would not be particularly useful as an account of *what happened,* much less of *why.* In general, human activities must be described in intentional terms, as being *about* things and *toward* things, and not as meaningless displacements of matter. Physical and intentional description are not incompatible, but they *are* incommensurable.

The relationship between intentionality and representation is a subtle matter. One theory, the *representational theory of intentionality* (Cummins 1989: 14; cf. Boden 1970; Fodor 1987), holds that beneath every instance of intentionality there lies some representation, so that the avoidance of an obstacle necessarily entails the representation of an obstacle.[6] And many cases of intentionality indisputably involve the use of representations. But in asking about the nature of intentionality, it is important not to presuppose that representation exhausts the phenomena of "about" and "toward." What alternatives to the representational theory of intentionality might be possible? This question has become urgent over the past several years as phenomenologically motivated critiques of the representational theory of intentionality have been employed in disputing the premises of AI. In this section I will review some of the history of ideas about intentionality, with the goal of motivating some computational alternatives.

The term "intentionality" originates with Brentano, but for present purposes the first important treatment of intentionality was that of Husserl (1960 [1929]; cf. Dreyfus 1982). Husserl's theory of intentionality is part of his overall project of completing Descartes's attempt to place human knowledge on a firm basis by working outward from the indubitable bases of human experience. More precisely, Descartes wished to reconstruct the inferential basis of knowledge from indubitable premises, and Husserl wished to reconstruct the structure of experience from an irreducible core of what he called *acts of consciousness.* He argued that to take up an intentional relationship to anything, whether mental or worldly, was a positive act of conferral – a matter of actively placing an interpretation on one's experience. More generally, he asserted that experience has an elaborate architecture that normally goes unremarked on within the *natural attitude* of unreflectively relating to things. Through his investigation of the structures of experience, Husserl hoped to place everyday activities and scientific research alike on a firm philosophical basis. Husserl did not intend his project to derive a mechanistic account of the human perceptual and cognitive apparatus, since he regarded intentionality as an irreducible phenomenon. Instead he sought to describe in detail the structure of human experience, exhibiting the natural attitude not as a transparent whole but as a certain complex phenomenon.

Heidegger (1961 [1927]) took another approach to the phenomenological description of experience. Like Husserl, Heidegger understood his project not as the logical rederivation of difficult conclusions but as the step-by-step recovery of a primordial experience of reality that successive generations of philosophy had obfuscated in their obsession with metaphysical speculation. But Heidegger began by rejecting the Cartesian starting point of Husserl's analysis of intentionality. Husserl's wariness of the uncritical form of awareness characteristic of the everyday natural attitude had led him to begin his analysis from within an artificial form of awareness, the so-called reduction of experience to what is indubitably given in it. In particular, Husserl shared Descartes's intuition that the outside world, other people, and culture in general are far away from individual experience, entailing elaborate construction from the basic acts of consciousness. For Heidegger, though, those aspects of experience are far only from the thought of philosophers, not from the ordinary experience of everyday life. Heidegger thus was led to give concrete activity and the social world a fundamental place in his account of human experience.

Heidegger's account of intentionality in *Being and Time* reflects this suspicion of detached observation.[7] He proposed a distinction between two ways of relating to things: the routine and unreflective mode of the *ready-to-hand* and the exceptional and deliberate mode of the *present-at-hand.*[8] This distinction is not mechanistic or psychological but phenomenological: a description of the structure of everyday experience. In our everyday routine activities, according to Heidegger, we have no positive awareness of engaging in acts of interpreting pieces of matter as cups, doors, or packages. The point is not that we carry on our activities unconsciously, but that we encounter objects through the roles they play in our activities and not as objective individuals in their own right. Sometimes, as in scientific work and traditional philosophical reflection, we do relate to objects in this entirely neutral fashion, but Heidegger regards this phenomenon as exceptional, and as something that takes place against the background of ordinary practical activity and its ready-to-hand way of encountering things.

In her account of the issues involved in bringing phenomenological arguments to bear on computational issues, Preston (1988, 1993) explores the consequences of identifying Heidegger's distinction between the ready-to-hand and the present-at-hand with the distinction between nonrepresentational and representational intentionality. This view has its phenomenological basis in the observation that, in the ready-to-hand way of encountering things, we have no awareness of manipulating representations. In other words, in routine activity we encounter things within their relationship to ourselves and to our activities and not in virtue of their being the denotata of representations. Even when we do use representations, as in conversation or writing, these representations do not exhaust our experience of the things represented; instead they play a more complex and derivative role, articulating experiences that have already taken form in advance of them, or else providing a resource in forming an understanding that goes beyond them. Still, great analytical delicacy is required. As Preston points out, it is doubtful whether the whole ideology of representation entailed in the representational theory of intentionality is compatible with the secondary role that Heidegger wishes to assign to the use of representations. Heidegger, moreover, would probably not wish to rule out unconscious representation use. And the case of ordinary conversation shows that the use of representational media such as natural language is consistent with a routine mode of activity and an entirely ready-to-hand way of encountering things. Certainly representation exists and participates in at least some instances of intentionality. The issue, rather, is what kinds of representation exist and what roles they play in real activities.

Heidegger's radicalization of the question of intentionality reflects an important shift in philosophical values. Whereas the empiricist and rationalist traditions had tended to regard scientific activity and objective knowledge as the prototypes for all human existence, Heidegger insisted that these special phenomena have their basis in the very different and more fundamental phenomena of routine, concrete activity in the familiar environments of everyday life. He observed that none of us has chosen or invented the world of everyday life. Quite the contrary, we have inherited the overwhelming majority of it, piece by piece, in the course of our socialization into our culture's ways of doing things. We carry on with daily activities of extraordinary complexity without ever requiring or attaining more than a very rough explicit understanding of how any of them work. The grounds for confidence in our ways of doing things, on this view, lies not in our intellectual analyses but in our habitual and generally unreflective participation in the tried-and-true practices of our culture. As a result, everyday activities have an aspect of anonymity, in that prior to their being things that "I do," they are things that "one does" (Heidegger 1961 [1927]: 163–168).

The fundamental point concerns the intentional structure of everyday activity. Whereas Husserl and others before him had viewed everyday activities as inheriting their intentionality from creative acts of consciousness, Heidegger insisted that everyday activities had an intentional structure of their own. If I steer around a pothole or eat with a knife and fork, these actions derive their intentionality from the network of cultural practices that I embody as a result of my socialization. I might superimpose some modifications of my own against this background of cultural practices, but these must be understood *as* modifications to a much broader and deeper preexisting structure. Intentionality is thus, in this sense, "public."

Merleau-Ponty's phenomenological analysis in *Phenomenology of Perception* (1962 [1945]) also begins from an implicit critique of Husserl and takes concrete activity as its central phenomenon (cf. McClamrock 1995: 187–193). More than Heidegger, Merleau-Ponty developed the theme that our activities are those of embodied agents. He construed very broadly the traditional notion of the "body" as the residence of habit and desire. In doing so, he tried to break down the conventional opposition between consciousness and the world, replacing it with a positive account of the body as a site of convergence and overlap between them (see Leder 1990). Intentionality for Merleau-Ponty is the intentionality of one's bodily ways of relating oneself to things in the course of one's activities. Like Heidegger, Merleau-Ponty wishes to distinguish between the unreflective intentionality of routine embodied activity and the more deliberate type of intentionality that encounters objects as existing in themselves and not through their roles in one's purposive activities.

A number of authors have used these phenomenological analyses as the basis for a critique of AI. The first and most forceful of these was Hubert Dreyfus, who took AI to be the inheritor of the metaphysical tradition against which Heidegger wrote. Dreyfus drew clear sides, identifying Husserl and AI research with an uncritical objectivism and emphasizing the place in Heidegger and Wittgenstein's work of the prerepresentational realm of culturally constituted embodied activity. For example, Dreyfus (1982) argues in some detail that the troubles that Husserl encountered in working out his theory of intentionality parallel similar troubles in AI; in each case, the project starts out with a promising enumeration of rules that then cannot be brought to any satisfying closure. He suggests that, following Heidegger, we "change the subject altogether, and ask about the intelligibility, unity, and order of *public behavior* rather than *private experience*" (1982: 26, italics in the original).

The debate is difficult to resolve, due to the near incommensurability of the phenomenological and technical discourses. The philosophy that informs AI research has a distinctly impoverished phenomenological vocabulary, going no further than to distinguish between conscious and unconscious mental states. By rendering unintelligible any substantive conception of the individual's involvement in the world, this vocabulary encourages a purely representational theory of intentionality and makes it difficult to comprehend any other account. Within such a framework, and given the requirement that all concepts be given technical realization, the idea of nonrepresentational intentionality is liable to sound like mysticism, as if intentionality were an occult force connecting mental states to their referents.

Such perceptions of philosophical ideas have led many AI people to feel antipathy toward the broader intellectual world – including the intellectual background from which computational ideas derive. I have heard expressed many versions of the propositions that philosophy and technology are dichotomous, that philosophy is a matter of mere thinking whereas technology is a matter of real doing, and that philosophy consequently can be understood only as deficient in comparison and as a distraction from the real work of design and implementation (Agre 1995c). As a result of such views, computational ideas are no longer understood as ideas but as techniques, to be evaluated on the technical results they produce in practice. While this practical criterion has its place, it cannot replace critical reflection on the intellectual inheritances of the field and systemic difficulties to which these give rise. Alternative ideas will stand or fall on technical grounds, but only after a suitably critical understanding of technical work has been put in place. Technology at present is covert philosophy; the point is to make it openly philosophical.

12 Deictic representation

Deictic intentionality

As the intellectual history sketched in Chapter 11 makes clear, AI research has been based on definite but only partly articulated views about the nature and purpose of representation. Representations in an agent's mind have been understood as models that correspond to the outside world through a systematic mapping. As a result, the meanings of an agent's representations can be determined independently of its zcurrent location, attitudes, or goals. Reference has been a marginal concern within this picture, either assimilated to sense or simply posited through the operation of simulated worlds in which symbols automatically connect to their referents. One consequence of this picture is that indexicality has been almost entirely absent from AI research. And the model-theoretic understanding of representational semantics has made it unclear how we might understand the concrete relationships between a representation-owning agent and the environment in which it conducts its activities.

In making such complaints, one should not confuse the articulated conceptions that inform technical practice with the reality of that practice. As Smith (1987) has pointed out, any device that engages in any sort of interaction with its environment will exhibit some kind of indexicality.[1] For example, a thermometer's reading does not indicate abstractly "the temperature," since it is the temperature *somewhere,* nor does it indicate concretely "the temperature in room 11," since if we moved it to room 23 it would soon indicate the temperature in room 23 instead. Instead, we need to understand the thermometer as indicating "the temperature *here*" – regardless of whether the thermometer's designers thought in those terms.

As with thermometers, so it has been with the design of computational devices. Many parts of AI and computer science in general have felt no need for worked-out ideas about representation and have carried on by haphazardly adapting intentional language from the general culture.[2] As computational artifacts have grown more complex, though, the need for principled ways of talking about their intentionality has become more urgent. And as previous chapters have suggested, the representational ideas that *have* been made explicit lead to technical inefficiencies and conceptual muddles when attempts are made to put them into practice. AI research needs an account of intentionality that affords clear thinking about the ways in which artifacts can be involved in concrete activities in the world.

In reviewing the state of technical practice here, I have been speaking in the engineering register of "artifacts" and "efficiency." My own interest is in using computational ideas to help understand human beings, who are not artifacts and who do not live their lives for the sake of efficiency. And, indeed, it has long been hard to understand how it might be possible, in any principled fashion, to ascribe intentionality to an artifact. If intentional states are necessarily subserved by mental representations, it becomes difficult to say that a thermometer "measures the temperature" or that a robot "welds cars" without importing some heavy philosophical baggage. It is an interesting fact that this kind of speech assists engineers in building thermometers and robots, but that does not imply that a coherent philosophical theory could be reconstructed from it. One attractive approach is to say that intentionality is relative to a beholder's suppositions about the rationale behind the artifact's design (Dennett 1981). Many other people resist the intuition that mechanical devices can have intentional states, since this seems to confer on them a vitality or consciousness that they really do not have.[3]

The intuition that denies intentionality to current-day thermometers and robots speaks to an aspect of intentionality that Heidegger and Merleau-Ponty spell out: its basis in the individual's socialization into the everyday practices of his or her culture (Dreyfus 1972). Artifacts currently do not participate in this kind of culturally constituted activity, and it is an open question whether they ever could and whether such participation is a prerequisite of anything we might want to call intelligence. And it is hard to say which aspects of human embodiment and acculturation are necessary, either by definition or as a practical matter, for the human forms of intentionality. I will not resolve the question here. I will, however, develop an alternative to the representational theory of intentionality, beginning with the phenomenological intuition that everyday routine activities are founded in habitual, embodied ways of interacting with people, places, and things in the world. In so doing, I will speak of intentionality in a sufficiently broad way that artifacts might be said to exhibit it.[4]

Every form of intentionality implies an ontology. Let us distinguish between two kinds of intentionality and thus two kinds of ontology, *objective* and *deictic*. An objective ontology holds that individuals can be defined without reference to any agent's activities or intentional states. If an agent has an objective form of intentionality toward some individuals – that is, if it has intentional relationships to some things in virtue of their objective identities – those individuals can be considered to exist independently of the agent. A deictic ontology, by contrast, can be defined only in *indexical* and *functional* terms, that is, in relation to an agent's spatial location, social position, or current or typical goals or projects. If an agent has an intentional relationship to an *entity* then as far as the agent is concerned the latter is defined entirely in terms of the role it plays in the agent's activities.[5] In the notation to be explained shortly, some examples of deictic entities are *the-door-I-am-opening, the-stop-light-I-am-approaching, the-envelope-I-am-opening,* and *the-page-I-am-turning.* Each of these entities is indexical, because it relates specifically to me, and functional because it plays a specific role in some activity I am engaged in; they are not objective, because they refer to different doors, stop lights, envelopes, and pages on different occasions.[6]

Deictic intentionality is the predominant form of intentionality in the everyday activities of human beings. As an example of deictic intentionality, consider one's relationship to the utensils at a restaurant table. One's table generally comes equipped with a fork, a knife, a spoon, a glass of water, a napkin, and so forth, arranged in a customary pattern in relation to the table and chairs. In eating dinner, if all goes as usual, one adopts a deictic form of intentionality to these objects. That is, one treats them in just the same way in which one has treated all of the other forks, knives, spoons, glasses, napkins, tables, and chairs at all of the other restaurants at which one has eaten. Each of these entities plays its own role in a stable system of practices, and its objective identity does not normally arise as an issue. Just as the thermometer measures the temperature "here" and not "in room 11," so one eats with "this fork," or perhaps "the fork at my place at the table here," and not with "fork number 271403" in some cosmic registry of forks. Deictic intentionality is a prominent feature of everyday routine activity because everyday routine activity consists for the most part of embodied cultural practices that bring us into familiar relationships with familiar types of objects.

A deictic ontology, then, is not objective, because entities are constituted in relation to an agent's habitual forms of activity. But neither is it subjective. These forms of activity are not arbitrary; they are organized by a culture and fit together with a cultural way of organizing the material world. Individuals might have their own distinctive repertoires of artifacts and habits, but each repertoire will be assembled from, and organized against the background of, an encompassing cultural "vocabulary." A deictic ontology, then, does not entail a skeptical or solipsistic stance toward the world, nor does it assume that individuals can capriciously reorganize the world simply by confabulating a new set of entities in their minds. The contrast between objective and deictic ontologies is not invidious; neither is inherently preferable, both have their place, and transitional or borderline cases between them may well be found.

One task for research on the two types of intentionality – engineering research in the case of artifacts or psychological research in the case of human activities – is to understand their respective roles and the relationship between them. My leading principle will be the phenomenological intuition that deictic intentionality is more fundamental than objective intentionality. This intuition has two parts. Consider some activity that brings an agent into an objective relationship with something, for example a particular hammer that has its own history and quirks. First, although that hammer will be in some sense focalized as an object of contemplation or action, most of what is going on in that situation (staying balanced, keeping materials straight, figuring where the nails go, etc.) will still take the form of comportment toward deictically constituted things (Heidegger 1962 [1927]: 91–107; Merleau-Ponty 1962 [1945]: 136–147). Second, even an object whose distinctive status (in a particular situation) requires the special treatment of objective intentionality is still, for most purposes (picking up, putting down, aiming, adjusting grip – things one does with *any* hammer), a generic entity that stands as the object of one's routine cultural practices. Objective intentionality, then, is not an entirely separate phenomenon. Instead, it is built on top of deictic intentionality as a further complication or refinement. Learning a person's name, for example, is a practice that facilitates relating to him or her as a particular objective individual. Yet most of the mechanics of interpersonal interaction are generic within a given culture. They function perfectly well before someone has emerged for us as a distinctive individual, and even with close acquaintances they function in a largely generic way (Atkinson and Heritage 1984; Moerman 1988).

Further instances of deictic intentionality can be enumerated easily by considering the objects we encounter within the whole range of our everyday practices. In the normal course of things, we do not relate to particular cereal boxes, oil cans, forest paths, postage stamps, hamburgers, and test tubes in terms of their objective identities but in terms of the roles they play in our activities. The point is not that all objects playing a given role are indistinguishable but rather that they are all assimilable to customary ways of interacting with them. On the special occasion when one of these things takes on a distinctive significance for us (perhaps through a defect or sentimental association), it takes on that additional importance over and above its generic role in our normal comportment toward such things. If its distinctive quality is not readily perceptible, it will probably have to be specially marked or segregated so that it will not be confused with other objects that resemble it. In fact, examples of indistinguishable objects that are not functionally interchangeable are hard to come by. One example occurs in the practice of buying rounds in a bar, in which it becomes necessary to keep straight the visually identical beer glasses to avoid spreading illness.

Recall that indexical reference is "efficient": an indexical term can refer to different objects in different contexts. Deictic intentionality is efficient in the same way, and this points toward an important property of embodied practices: their portability. People comport themselves in their customary manner toward *whatever* circumstances they encounter, in virtue of the indexical and functional significances these circumstances might hold. What remains more or less constant across these circumstances are the cultural practices that individuals embody. Various properties of environments remain relatively invariant as well, largely as a consequence of these very practices. Nonetheless, a great deal of contingency attends the details of particular interactions between people and their environments. Whatever recurring structure we theorists might find in the dynamics of everyday life is an emergent phenomenon. This recurring structure is both the central phenomenon of human life and the principal datum for computational research.

Causal relationships

The phenomenon of deictic intentionality thus has consequences for computational research, regardless of whether we intend to build artifacts or to understand people. It is critical to distinguish between an agent's machinery and the dynamics of its interactions with its environment. An individual embodies the repertoire of habitual practices that characterize a culture, but the coherent organization of activity arises only through individuals' interactions with their familiar world. We as theorists might construct elaborate theories of the dynamics of individuals' activities within the encompassing dynamics of society and history, but it would be bad sociology and bad engineering to presuppose that the individuals themselves possess any explicit representations of these theories. As a matter of sociology, ordinary people would probably conduct their lives differently if they had a better understanding of the actual workings of society. And as a matter of engineering, it is pointless to endow an artifact with any more elaborate theory-using abilities than it requires to perform its assigned task. In each case, the leading principle is that of machinery parsimony: choosing the simplest machinery that is consistent with known dynamics. This view contrasts with the emphasis on expressive and explicit representation that is currently a central AI design value.

Instead, this alternative view suggests some different design values. The starting point, once again, is the distinction between machinery and dynamics. A designer moves back and forth between two activities: synthesis of machinery and analysis of dynamics. The question of intentionality arises when it comes time to understand how one's artifact will interact with the people, places, and things that will populate its environment. The artifact will most likely interact with a given object in virtue of the object's role in the artifact's activities, not through the object's objective identity. The designer might plot out the typical life histories of objects, assigning the artifact particular parts in the story while leaving the other parts to natural processes, other agents, and assorted factors of happenstance. It might conceivably be necessary for the artifact to represent this whole life history to itself, but closer analysis will probably reveal that some more parsimonious design strategy will suffice. This kind of analysis, of course, has analogs in the study of human life as well. Objects have life histories in the social world (Kopytoff 1986), in which particular people play only particular roles according to their locations in the overall organization of society. A complete analysis of society will map out the ways in which people and things come together in ordinary activities without necessarily attributing omniscience to anyone.

In general, of course, our relationships to objects are not merely episodic but have some time-extended structure. I need not maintain perceptual contact with the object throughout the course of my relationship with it, provided that my interactions with it have a sufficient degree of reliably invariant structure. For example, I might buy a white Oxford-cloth shirt and wear it intermittently until it becomes unpresentably worn, keeping it in my closet or hanging it over a chair in my bedroom on days when I wear other shirts. Or my wallet might trace a definite trajectory through my environment, from nightstand to pocket to gym locker and back again, being taken out momentarily on those occasions when business is being done. Or I might buy a carton of orange juice and carry it from supermarket to car to kitchen shelf to refrigerator to kitchen table to refrigerator to kitchen table to trash can to curbside. In each case, the object and I live interwoven lives for a while, tracing our respective paths through the world according to a stable set of habits and customs that a theory of activity might hope to describe.

Conventional design practice in AI understands this phenomenon of time-extended relationships between agents and objects in terms of the problem of "keeping track" of an objective individual. An agent might have in its head a symbol such as SHIRT32 that serves three purposes: (1) to name a certain shirt, (2) to participate in the logical sentences that express knowledge of that shirt, and (3) to mediate concrete interactions with that shirt on particular occasions of buying it, wearing it, washing it, ironing it, and finally packing it off to a charitable organization. In some research this process of keeping track is simply stipulated, as if symbols stayed magically connected to their referents. This assumption is actually valid in certain situations, as when the agent is a computer program that interacts only with other computer programs and not with the world outside the computer. Normally, though, simply expressing the continuing identity of an object does not suffice to keep track of the object in a causal sense. Sometimes it is imagined, somewhat more reasonably, that the agent stores a set of spatial coordinates for each such object, updating the stored values each time any fresh knowledge about the object's whereabouts becomes available, as part of the general process of maintaining an objective world model. In each case, the agent conducts all of interactions with the object via the symbol that names it and maintains a correspondence with it as time goes along.

The notion of deictic intentionality suggests a different approach to the question. First of all, it is inherently easier to deal with a deictic entity than with an objective individual, because several objects may be capable of playing a given role in an agent's activities. From this point of view, what matters is the agent's ability to get ahold of *something* that can serve the entity's defining purpose. One might have three white shirts, six dinner-table chairs, a stack of fresh envelopes, and a pound of nails, and when one is dressing for work it may matter only that one select a white shirt to put on and not a chair, envelope, or nail. As long as the objects in a given category remain indexically and functionally indistinguishable, it is not necessary to keep track of a particular individual; instead one must simply maintain access to *some* instance of a category.

Second, and more important, the theorist can usually deduce interactional invariants that make it unnecessary to keep explicit track of the locations of objects. Most of these invariants are entirely mundane: they result from the cultural practices of putting things back in their places, replenishing supplies when they are running low, carrying things around in one's pockets, getting things fixed when they break, making notes to oneself, locking doors, refrigerating perishables, and organizing one's house in pretty much the same way as everyone else in the culture.

Other interactional invariants arise in ways that are partly evolved and partly deliberate. For example, at any given time I own perhaps two dozen pens, most of them brought home from hotels or acquired in periodic binges at art supply stores, and I keep them in several places, including my desks at home and work, my computer terminal, my car, my laptop computer's carrying case, and my backpack. The ebb and flow of life frequently cause pens to move from one of these locations to another, for example when I grab a pen from my desk on my way to the library and then toss it in my car on the way home. As a result, I sometimes notice that many pens have accumulated in one place. When this happens, I put the excess pens in my backpack. And when I need a pen in one of those places and discover that I have none, I grab some pens from my backpack and put them there. Although I sometimes pay attention to the disposition of red and green pens when I am marking students' papers, I rarely attempt to keep track of specific pens. I never have a panoptic sense of which locations are well and poorly stocked with pens at a given moment, and I pay no attention to my supply of pens except when I notice a surfeit or when I need a pen and none is handy. Of course, the efficacy of these practices depends on a wide variety of factors that I did not create – such as the willingness of hotels to give away pens and the social arrangements that generally prevent other people from taking my stuff. And although this whole dynamic is particular to my own life, it is assembled from numerous elements provided by a cultural way of life that I did not choose.

My prescription, then, is a design practice that employs dynamic analysis of time-extended patterns of causal interaction between agents and objects to minimize the mechanical complexity of designed artifacts. Central to this revised design practice is a clear distinction between two points of view: that of the theorist and that of the agents being theorized about. Regardless of our ontological commitments as theorists, the question remains: what kind of ontology might be employed by the agents under study? And even if we as theorists are immensely knowledgeable about the structure of the world and the dynamics of activities, the question remains: how much of this knowledge should we ascribe to the people we study or incorporate in the artifacts we build? In each case, the principle of machinery parsimony suggests endowing agents with the minimum of knowledge required to account for the dynamics of its activity.[7] Most people get along well in the world with a relatively superficial understanding of physics and sociology, and most people can use thermostats without book learning about temperature (cf. Kempton 1986). In each case it might be preferable if people knew more than they do, but it is pointless to build artifacts that reason from first principles when tenth principles will suffice.

It might be objected, perhaps, that this is a formula for the design of special-purpose devices (called "hacks" or "toys" in the lexicon of conventional AI design values) and thus a violation of the spirit of AI, which always seeks greater explicitness of representation and broader generality of function. But the conventional conception of general-purpose functionality is misguided: the kind of generality projected by current AI practice (representation as expression, thought as search, planning as simulation) simply cannot be realized. Every artifact or organism of any complexity relies heavily, by computational necessity, on the regularity and coherence of its familiar world. Sometimes this reliance is hidden, as in the case of the simplifying and regularizing assumptions that are built into computer systems and enforced in their operating environments. Human intelligence in particular is not a matter of lone geniuses reinventing culture from scratch but of ordinary people socialized into organized forms of life. And our design practice must give a central place to this interdependency of sensible agents and orderly environments.

Entities and aspects

Previous sections have contrasted objective and deictic intentionality and have sketched the consequences for design practice of taking the latter to be the principal way in which agents relate to things. This discussion has avoided the concept of representation, for two reasons. First, the question of intentionality is logically prior to the question of representation. Second, once intentionality is understood in a suitably sophisticated way, the notion of representation must undergo painful surgery to be of continued use. It will be evident by now that the conventional AI understanding of representation involves some questionable presuppositions. But no simple alternative is available because representation is not a unitary phenomenon. Photographs, epic poems, visual images, operating-system queues, rituals, recipes, echoic memories, police records, arrangements of marching bands on football fields, and Saturday morning cartoons are all representations in various ways. Each form of representation has its own ways of referring to things, its own dynamics of creation and use, its own complex social life, and its own varieties of historical and cultural specificity.

The concept of representation often threatens to break down altogether. Must we say that thermostats employ representations? Ants? Apes? The questions are futile because no definite distinction is at stake. The matter is particularly delicate in relation to the most ordinary phenomena of human activity, the sort I have offered as examples of deictic intentionality. When I do something as routine as sitting down, changing lanes, sipping coffee, flipping pancakes, or turning a key in a lock, am I using representations? It is an empirical question, but it is an empirical question only once "representation" is defined. Rather than attempt to sort the issues, I will use the term *objective representation* to name any technical method, such as the use of first-order logic to express stored knowledge, that produces artifacts which employ an objective ontology, *whether or not* it involves constructs that would normally be called representations. The term *deictic representation* will likewise name any technical method that produces artifacts which employ a deictic ontology – again, whether or not conventional representations are involved.[8] A given agent might employ both objective and deictic ontology in various combinations, but I will not consider in any systematic way how this might work. Nor will I present a mathematical account of the "meaning" of deictic representations (cf. Subramanian and Woodfill 1989a, 1989b).

Some terminology will be useful. An agent using a deictic representation scheme might, as already mentioned, take up an indexical and functional relationship to some deictic entity. I will name deictic entities with hyphenated noun phrases such as *the-car-I-am-passing* or *the-coffee-cup-from-which-I-am-drinking* or *the-spinach-I-am-washing.* These names are conveniences for the theorist, not mental symbols of any sort. Although they seem like definite descriptions, they are not descriptions that agents represent to themselves as linguistic structures. They designate, not a particular object in the world, but rather a role that an object might play in a certain time-extended pattern of interaction between an agent and its environment. Different objects might occupy this role at different times, but the agent will treat all of them in the same way. Thus, if I am driving down Route 2 in Montana, I will pass a dozen cars an hour, each of which will be *the-car-I-am-passing* in turn. Passing cars on Route 2 is a tedious and routine matter – a recurring and stable pattern of interaction among myself, my car, the other driver, his or her car, the road, and the rest of the landscape. In each case of passing cars on the road, the name *the-car-I-am-passing* is said to *refer to* whatever car is being passed right now. The concept is not that of linguistic reference, since symbolic representations are not necessarily involved. If an entity refers to a certain object on some occasion, then that object is said to be *assigned* to that entity. This assumes, of course, that we as theorists are willing to say that the object in question actually exists and is not, for example, an optical illusion of some sort. In general, theoretical use of a name like *the-car-I-am-passing* does not guarantee that *the-car-I-am-passing* will necessarily refer to a car, since a driver might mistakenly try to pass a haystack or an optical illusion. The name designates a role, not a set of semantical conditions of satisfaction. Theorists may sometimes wish to define entities in terms of other entities, such as *the-space-bar-on-the-keyboard-on-which-I-am-typing.*

More vocabulary: an *aspect* of some entity predicates something of that entity, for example *the-car-I-am-passing-is-a-police-car* or *the-coffee-cup-I-am-drinking-from-is-empty* or *the-spinach-I-am-washing-is-clean-now.* An aspect can mention more than one entity, as in *the-cup-from-which-I-am-drinking-is-sitting-on-the-newspaper-I-am-reading.* Under suitable circumstances, an agent can be said to *register* at a given moment, the value of such an aspect. It is up to the designer, of course, to specify the actual causal events that make this registration possible, whether through perception or memory or surmise or some combination of these.

Despite this vocabulary, many of the classically difficult technical problems of representation remain. For example, the question will still arise of what entities and aspects to represent. As a technical matter, a designer must make choices as to degrees of abstraction, complexity of inferential schemes, and the like. As a psychological matter, some account must be provided of what entities and aspects people relate to and why. In practice, most entities and aspects derive from one's culture through language and through socialization into the routine practices of everyday life. Representational innovation is possible, of course, but it is not a prominent feature of ordinary activities.

It may be objected that deictic representation is just an obscure notation for concepts better expressed in conventional representation schemes, or perhaps in relatively novel schemes, such as McCarthy and Hayes's (1969; cf. Hayes 1979b; Sandewall 1994) notion of fluents, for expressing various kinds of context dependence. One might, for example, attempt to write *the-cup-from-which-I-am-drinking* with some such logical term as $\iota x.cup(x) \wedge drinking(I,x)$, where the quantifier "$\iota$" expresses the definite article (it is read "the x such that . . .") and the constant symbol "I" stands for me. For this to make sense as logic, however, one would have to provide a semantic scheme according to which I manages to pick out the appropriate individual based on the context in which it is used. (Likewise, the implicit *now* in *drinking* must pick out the correct time based on the context of use.) Even if one does provide a consistent logical semantics for this kind of expression, perhaps using situation semantics, the entire exercise is beside the point. Deictic representation is not a means of expressing states of affairs but of participating in them. Conventional AI ideas about representation presuppose that the purpose of representation is to express something, but this is not what a deictic representation does. Instead, a deictic representation underwrites a mode of relationship with things and only makes sense in connection with activity involving those things.

Every representation scheme must provide some account of nonspecificity, so that some item of knowledge might be equally applicable to the particular objects encountered in different situations. Objective representation schemes use quantified variables for this purpose. Although it is possible to imagine an objective representation scheme that does not use any constant symbols, perhaps through heavy use of the ι quantifier, a nontrivial objective representation scheme must express nonspecificity by means of quantified variables or their expressive equivalent. Quantified variables generalize across objective individuals.

A deictic representation scheme, by contrast, does not require special expressive means for implementing knowledge that is independent of situation particulars. Since a deictic entity is defined not objectively but in relation to the agent and its activities, deictic representation is, to use Perry's term, efficient. An entity like *the-car-I-am-passing* can refer to dozens of different cars at different times. As long as these cars acquire no significance beyond being cars-to-pass, the agent will not have to invent distinct means of representing them. A deictic representation scheme thus does not make a distinction between the specific and the general. As a result, any newly learned knowledge will transfer to new situations passively, without the necessity of an explicit substitution of variables for constants and back again. If you know something about what to do when *the-car-I-am-passing-is-a-police-car,* it will apply whenever you discover yourself passing a police car in the future. If two such cars have no salient differences, then the representation scheme will not distinguish them. The agent will simply be unable to tell them apart, since what matters is not their objective identities as such, but their indexical and functional relationship to the agent. These are not things that representations express but that agents participate in during their routine activities.

In previous chapters I have sketched the problematic space of technical trade-offs that ensue when variables are used routinely for nonspecific reasoning and action. I should emphasize that, in doing so, I am using the word "variable" in a particular sense, one that figures in a certain story about representation, namely, quantification over objective individuals. Though most AI representation schemes have not had any especially clear semantics, the objective approach to ontology and nonspecificity has remained clear to the extent that anything has. I do not mean to imply that absolutely any stored state entails objective representation, nor that any implementation scheme that employs a notion of "variable" in some programming language entails objective representation. I refer only to variables whose values are symbols, names, data structures, or other technical constructs that represent things by means of objective descriptions.

Certain features of deictic representation have been anticipated by earlier AI research on representation. The idea of referring to things in terms of their functions, for example, is present in Minsky's (1977) notion of frames and in Schank and Abelson's (1977) notion of a script. These two schemes were motivated by issues of architecture and reasoning about stories, respectively, and do not have any particularly systematic semantics.[9] As such, it is hard to determine whether they should be understood as objective or as deictic representations. Still, in practice they have been treated as organized sets of variables that can be "filled" by constant symbols that name individuals in the relevant (often fictional) world. The theme of functionality also resembles the concept of egocentricity in Piaget's theory of object permanence – the ability to treat objects as permanent and independent of one's own perspective (Singer and Revenson 1978; Tanz 1980). Piaget observes that this ability grows out of the child's experience with the invariant properties maintained by an object when the child interacts with it in the course of various different activities. Some other related dynamic themes appear in Drescher's (1991) computational model of the stages of sensorimotor development described by Piaget. He focuses on certain kinds of interaction that present the child with measurable correlations that reflect regularities in the dynamics of the child's interactions with things in its environment. He also describes the concept of a *canonical perspective*, a dynamic effect that facilitates the discovery of deictic regularities by modifying the child's relationship to the world in recurring ways, for example the habit of bringing interesting objects into the middle of the visual field at a standard distance.

Using deictic representation

Deictic representation is compatible with a wide variety of implementation techniques. A wide variety of artifacts employ clearly deictic forms of intentionality, since they monitor whatever surroundings they are installed in. As such, they are readily reanalyzed as using deictic representation. No such intention may have guided the design process, although the designers probably did employ definite descriptions when explaining to themselves and others what the artifacts do and how they do it.

Be this as it may, AI design practice can benefit from sustained reflection on the inevitable indexicality of situated artifacts and on the consequences of taking systematic account of the causal relationships between agents and things (Dixon 1991). When agents can relate to objects in deictic terms by maintaining suitable causal relationships to them, many design problems become easier, for several reasons. One reason is that the objective identities of things are rarely perceptible, whereas the indexical and functional relationships that things bear to agents usually are. All forks look pretty much alike, as do all pencils, carrots, door knobs, gym socks, and parking meters. The objective identities of particular forks, pencils, and carrots are usually difficult to determine unless they are physically distinctive or bear unique markings. But although it is easy to mistake one fork for another, it is much harder to mistake a fork for a pencil or a carrot or any other functionally distinct object. Functionally distinct objects usually look different.

As a general matter, objects with a shared function are likely to have important perceptual similarities. These similarities are not arbitrary but rather tend to correspond directly to the function that the objects have in common. In AI research on analogical reasoning, this idea is known as the "correspondence between structure and function" (Winston et al. 1983). Forks are readily recognized by their handles and tines, pencils by their graspable shape and point, and so forth. This is not to say that agents must routinely rederive the purposes of objects from first principles given only their visible appearance, or that the functionality of objects is inherent in the light they reflect (as in Gibson's [1986] notion of an *affordance* as an invariant quantity in an organism's optical ecology), but simply that the agents' knowledgeable interactions with things are facilitated by their functionally significant perceptual properties. How this works is largely unknown, though it would certainly be a mistake to posit any simple, context-independent mapping between functions and perceptual features, especially when the function is something as broad as "thing for sitting down on" or "thing for drinking from."

But an agent must recognize not only the functional properties of objects but also their indexical properties. In other words, the agent must detect not simply the abstract "functionality" of objects but the role they play in the particular activities in which the agent is currently engaged. A car is a car in a wide variety of circumstances, but it is *the-car-I-am-passing* only when one is actually in a certain complex relationship to it. Complex as this relation is, though, it is readily perceptible because it places the car-being-passed in a standardized location in one's visual field, where its relevant properties (relative speed, police insignia, etc.) can be registered in standardized ways. Similarly, a formal dinner table is covered with indistinguishable forks, knives, napkins, and water glasses, but one's own equipment is readily distinguished from one's neighbors once one takes up a bodily position at one's seat.

The relationships with things that we take up in concrete activity arise equally through our intentions and through our bodily involvement in a physical situation. If the kitchen looks different according to whether we are making breakfast or looking for the cat, Merleau-Ponty would say, the kitchen is displaying itself not to a disembodied intelligence but to someone who has been launched into a physical relationship with the kitchen as a meaningful space. The integrated involvements of perception and action that constitute these relationships are neither magical nor innate, but arise through socialization into the practices of one's culture. Perception and action are, in this sense, not automatic attunements but complex patterns of organized habit. The task of AI is to determine what kind of machinery can best support this kind of habitual interaction with the customarily arranged materials of familiar environments.

This conception of the role of perception in activity makes strong claims on accounts of visual architecture. The principal purpose of vision, on this view, is not to deliver a world model; nor is it to identify the objective identities of things; nor is it even to recognize and enumerate the categories into which visible objects might be classified. Instead, the principal purpose of vision is, starting from the agent's intentions and the changing visual field, to register the indexically and functionally significant aspects of the entities in the agent's environment. The work of registering these aspects frequently requires the agent to engage in complex interactions with the environment, and this work is indissociable from the rest of the agent's interactions. Vision is in this sense an *active* process, not the passive reconstruction of the world as it is projected onto the retina.[10]

The extent to which vision should be understood as an active process has been a subject of great controversy in both psychology and engineering. It has been customary to contrast two general approaches to the computational study of vision, bottom-up and top-down. At issue is the role of domain-specific knowledge and intentions in vision. A top-down approach to vision emphasizes the ways in which this domain-specific information might guide the processes of visual perception. A bottom-up approach, by contrast, portrays the visual system as an innate architecture in which information flows entirely "upward" from the retinal image to the visual system's output, a process that since the 1970s has generally been understood (but need not be in principle) through the inverse-optics metaphor of constructing a model of the visible world. It is evident that top-down processes do exist in human vision, inasmuch as people often consciously and deliberately redirect their gaze or their focal depth in order to see something. And it is probable that early vision operates in a largely bottom-up fashion, given the existence of optical illusions that are not *cognitively penetrable* (Pylyshyn 1984: 130–145; cf. Fodor 1983: 64–86). The question, technical and empirical, concerns the interrelationship between these two elements of visual perception.

The next chapter will describe one current proposal for reconciling the top-down and bottom-up theories of vision. For the moment, I simply want to specify what such a synthesis must achieve. Given the intricate dynamic relationship between perception and action, the visual system must be co-designed with the rest of the machinery with which it continually interacts in routine activity. The visual system must support the registration of numerous aspects of the entities that are found in culturally organized human activities. Finally, it must support the integration of task-specific vision with the more generic monitoring of the visual environment that a sentient agent requires. Almost all of this perceptual activity will be habitual. Some portion of the machinery underlying these habits will undoubtedly be innate, but the vast majority will have accumulated through the agent's history of interactions with its familiar environment. As a result, the visual system must combine generality and specificity in a particular way: it must be general enough to support a wide variety of specific visual processes and to deploy them in a smoothly orchestrated fashion during the agent's interactions with a contingent environment.

The system I will describe in Chapter 13 will provide an existence proof, illustrating some of the issues that arise in building an agent that uses a nontrivial visual system to engage in organized goal-directed improvisations in a contingent world. But an account of the machinery and dynamics of human life must also explain how new forms of activity arise and how deictic representations might be acquired. Let me sketch a general orientation toward the question. Learning is a highly social matter. Much learning occurs within organized interactions among people, some of which are called "teaching/learning" and others of which are not. Research in the Vygotskian tradition has identified some of the dynamics of this process (Newman, Griffin, and Cole 1989; Rogoff and Wertsch 1984; Rogoff 1989; cf. Lave and Wenger 1991). According to these theorists, interactions among people have dynamics that, often as a side effect of more directly instrumental aims, provide auspicious circumstances for learning. Much of this learning is mediated symbolically through the situated use of language. I imagine that many instances of deictic representations have their origins in language and that the linguistic appearance of such names as *the-car-I-am-passing* is not entirely misleading.[11]

The social dimensions of learning are not restricted to social interactions, since the physical settings within which people learn (houses, yards, streets, etc.) themselves result from intensive socialization. These socially organized spaces lend themselves to certain interactional patterns and not to others. They tend to channel activity in certain directions, principally those of activity in that particular culture. This social structuring of the physical environment is broadly consistent and highly redundant, so that the resulting patterns of activity tend to reinforce one another within the unarticulated framework of the society. People learn by actively appropriating the interactional resources of their environments and not by passively absorbing them, but these resources and the customary routines for using them are structured so as to give a practically inevitable shape to the habitual practices that this process generates. Such a view is common enough in developmental psychology

(Leont'ev 1981), education (J. S. Brown, Collins, and Duguid 1988), and anthropology (Bourdieu 1977 [1972]), but much work remains to give it a detailed computational interpretation (cf. Agre and Horswill 1992).

Among the virtues of this view is an alternative account of representation. Vygotsky (1978 [1934]) suggests that children learn to think by internalizing activities that involve representations. Having employed concrete symbolic forms, such as drawings or egocentric speech, to organize their activities, they learn to organize their future activities as though the concrete symbols were still present. A tremendous amount has been learned in recent years about social practices for using representations (Comaroff and Roberts 1981; Goody 1986; Hutchins 1995; John-Steiner 1985; Latour 1986; Lynch 1988; Wertsch 1985), and each of these theories can provide suggestions about the kinds of activities that are internalized (at any age) to produce symbolic thought.[12]

13 Pengi

Argument

This chapter describes a computer program that illustrates some of the themes I have been developing. Before I discuss this program in detail, let me summarize the argument so far. Recalling the scheme laid out in Chapter 2, this argument has three levels: reflexive, substantive, and technical.

The reflexive argument has prescribed an awareness of the role of metaphor in technical work. As long as an underlying metaphor system goes unrecognized, all manifestations of trouble in technical work will be interpreted as technical difficulties and not as symptoms of a deeper, substantive problem. Critical technical work continually reflects on its substantive commitments, choosing research problems that might help bring unarticulated assumptions into the open. The technical exercises in this book are intended as examples of this process, and Chapter 14 will attempt to draw some lessons from them.

The substantive argument has four steps:

1. Chapter 2 described two contrasting metaphor systems for AI. Mentalist metaphors divide individual human beings into an inside and outside, with the attendant imagery of contents, boundaries, and movement into and out of the internal mental space. Interactionist metaphors, by contrast, focus on an individual's involvement in a world of familiar activities.

2. As Chapters 1 and 4 explained, mentalist metaphors have organized the vocabularies of both philosophical and computational theories of human nature for a long time, particularly under the influence of Descartes. This bias is only natural. Our daily activities have a vast background of unproblematic routine, but this background does its job precisely by not drawing attention to itself. The phenomena that stand

out and recommend themselves for abstract theorization are the disruptions of routine interaction that show up as problems.

3. Attempts to build machines based on this view have encountered a large space of hard technical trade-offs. Chapter 8 has argued that the precise nature of this space points to the need for a different, interactionist perspective. Interactionism would restore a central place to routine activity, understanding the organization of activity not as something mapped out in advance but as an emergent phenomenon located in the interaction itself. This interaction has two parties, an agent and a world, that are fitted to one another, whether through socialization or adaptation or design.

4. This view of the relationship between agent and environment, Chapters 10 and 11 argued, calls for novel understandings of intentionality. The mentalist tradition has founded intentionality on representation; it also understands representation as some kind of systematic correspondence between inside and outside. In AI work, this view takes the form of model-theoretic semantics or the manipulation of world models. Interactionism, by contrast, suggests founding intentionality in conventional forms of activity. Agents relate to objects primarily in terms of the roles they play in activities, not in terms of their resemblance to mental models of them.

Each step of the substantive argument points at patterns of technical difficulty that arise in the attempt to work out a mentalist AI, arguing that the nature of these patterns motivates a shift to an interactionist approach. The technical argument traverses different territory, but winds up in a similar place. It too has four steps:

1. AI has understood activity as the construction and execution of plans. This proposal is unreasonable, given the extent to which our actions must be sensitive to the detailed state of our environments. An alternative proposal is that activity is improvised, and in particular that the organization of activity is an emergent phenomenon. This proposal about the dynamics of activity places constraints on the machinery of embodied agents. Combinational logic recommends itself as the basis of this machinery since it is the fundamental stuff out of which digital computers' processors are made: fast, simple, and continually sensitive to the states of both the agent and its environment.

2. Any combinational logic network that could participate in the dynamics of human activity would certainly be enormous and highly

complex. Some portions of this network might be innate, but most of it would be laid down through experience. Chapters 6 through 10 have suggested that this might occur through the incremental accumulation of lines of reasoning that have arisen in the course of ordinary activity. This proposal makes few assertions about where novel lines of reasoning come from; the particular rule system I have described is psychologically implausible in numerous respects and simply illustrates some general points. Among these points is the need for activity to be almost routine.

3. A great difficulty with this proposal is that combinational logic has difficulty supporting the conventional mechanism by which AI systems express knowledge without regard to particular individuals, namely variables. Though various schemes have been proposed to ameliorate this problem, it is better understood as reflecting something deeper. A representation scheme that expresses the world in terms of symbolic relations among namelike mental symbols offers no particular account of the actual, causal connections between those symbols and their referents.

4. An alternative proposal regards symbols as a secondary means of intentional relationship to things. The central, primary form of intentionality is the culturally organized system of interactional patterns that comprises much of everyday life. These patterns of activity constitute their objects in indexical and functional terms, as for example in *the-cup-from-which-I-am-drinking*. By not representing objects in terms of their objective identities, deictic representation supports a wholly passive form of nonspecificity, namely indexical efficiency. Taking this idea seriously, however, requires a more sophisticated understanding of perception and its role in activity.

Taken as a whole, this argument exemplifies the principle articulated in Chapter 3, that close attention to the dynamics of activity leads to simplified forms of machinery. The substantive and technical arguments approach the phenomena in different ways – the substantive argument through philosophical considerations on metaphors for describing activity, the technical argument through engineering considerations on machinery for engaging in activity – yet the two arguments converge on a common set of proposals. It remains to explain what this convergence amounts to in particular cases. This is a substantial, indeed open-ended, task. The remainder of this chapter presents a case study in the relationship between machinery and dynamics in the construction of a particular computer program. Though unsatisfactory in numerous ways, this exercise opens up new issues and suggests directions for further research.

Pengo and Pengi

Pengi is a program that David Chapman and I designed to illustrate the notion of deictic representation (Agre and Chapman 1987). Chapman worked out the detailed architecture and wrote most of the code. Pengi's domain is a reimplementation on the Lisp Machine of a commercial video game called Pengo (Figure 13.1). Pengo is a fairly difficult game. It does not call for much physical dexterity, but it does call for the complex, rapid, and artful use of one's visual system. It also calls for a great deal of goal-directed improvisation.

In this game, the player watches a video monitor that portrays a number of discrete cartoon figures: a penguin, some bees, and several dozen ice cubes. The player has a joystick and a button that control the penguin. The bees are constantly in motion, as are some portion of the ice cubes. The rules are as follows. If a bee gets too close to the penguin, the penguin suffers a fatal bee sting and the game starts over. If a bee or penguin kicks an ice cube, the ice cube slides in the direction it has been kicked, horizontally or vertically. The player causes the penguin to kick an ice cube by pressing the button; this is a large black button on the arcade game and the K key on the Lisp Machine implementation. If a sliding ice cube should happen to hit a bee, that bee dies and disappears from the board. If all of the bees die, the player wins the game. A sliding ice cube can also kill the penguin. Thus, the bees can kill the penguin in two ways, by stinging it or kicking ice cubes at it. The penguin can kill the bees in only one way, by kicking ice cubes at them. The penguin, though, is presumably more intelligent than the bees, which operate by a simple random process. The bees always move at the same speed. They tend to move in the general direction of the penguin, but they randomly change their headings every few seconds. If a bee finds itself able to kick an ice cube at the penguin then it does so, but it will not go out of its way to position itself behind an appropriate ice cube. The point of this domain is not to emphasize the element of opposition or antagonism between malevolent equals. The bees' randomness and hostility are a source of uncertainty in the domain.

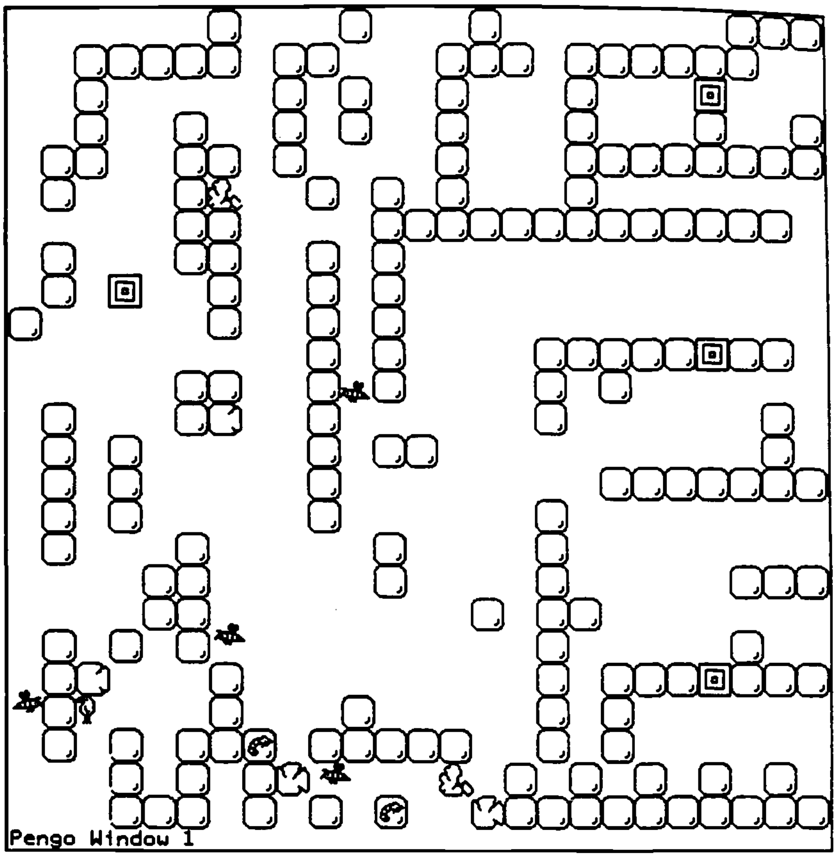


Figure 13.1. A Pengo game in progress.

As a domain, Pengo is an improvement on the blocks world. Things move, the geometry is more complicated, the arrangement of objects in space is more meaningful, and the individual tasks relate in a clear way to an overall goal (winning the game). Although it obviously fails to capture numerous elements of human activity, the combination of goal-directedness and improvisation involved in the game of Pengo is a basic aspect of routine activity about which we can hope to learn some computational lessons.

Playing Pengo is not a matter of executing plans. A plan to sneak around behind a certain ice cube and kick it at a certain bee will not usually work. The bee will fly away, other bees will fly into range, or the ice cube will be kicked out of the way. Things go wrong in these ways all the time. But playing Pengo is not simply a matter of reacting to things that go wrong. You need some notion of what you are trying to accomplish. You might want to get around behind an ice cube, or to run away between some rows of ice cubes, or to kick an ice cube out of the way. Your activity must be organized toward goals, but you have to stay on your toes.

Pengi plays a pretty decent game of Pengo. In its present state it is a little better than I am, which is to say that it wins from time to time and usually puts up a good fight. Our design was based on our understanding, as players, of the dynamics of competent Pengo-playing. Although we believe that Pengi's architecture is capable of supporting all of the Pengo-playing dynamics we feel we understand, we did not attempt to implement everything we knew about the game. In any case, my argument depends not on any qualitative measures or comparisons of Pengi's performance, but on qualitative aspects of our experience in trying to get the system working.

Pengi constantly interacts with the game, looking at the video screen and generating actions. Its architecture is divided into a periphery and a central system. The periphery is fixed, innate, domain-independent machinery for early vision and low-level motor control. The central system conducts a running argument about strategy and tactics, thereby deciding at each moment what to do next. The central system is made entirely of combinational logic: gates and wires clocked in the normal manner, as described in Chapter 5. It thus has no memory, software, pattern-matching facilities, dynamic storage allocation, or pointers. Its circuitry can be regarded as a snapshot of a process of dependency accumulation, but Pengi itself does not learn anything or augment its own circuitry (cf. Dreyfus 1992: xxxiii). Our modeling of motor control is trivial, since playing Pengo does not require elaborate motor control. Our modeling of vision, on the other hand, is fairly sophisticated. A later section will describe the architecture in more detail.

Pengi exemplifies the principle of machinery parsimony. As Chapters 8 and 11 have explained, conventional mentalist technology would address a task like Pengi's by means of a "planner," that is, a complicated set of machinery for building and maintaining world models and for constructing and executing plans. We were able to build a much simpler device to play Pengo because we could understand something about the dynamics of the activity. That is not to say that Pengi's architecture provides a complete model of human activity as a whole. Certainly not, since many important interactional phenomena do not arise in playing this game. Pengi, for example, can always see the whole game board, so it never has its back turned on anything. Nonetheless, that Pengi's central system can be built with a reasonably compact combinational logic circuit is a substantial claim. Briefly, this feat is possible for two reasons. One reason is that Pengi embodies an account of representation, namely deictic representation, that does not involve variables and can thus be readily implemented with combinational logic. The other reason is that Pengi can engage in an orderly, flexible, goal-directed interaction with its world without maintaining a plan or world model, so the central system does not need to keep any state. The next few sections explain these points in more detail.

Entities and aspects in Pengi

On just about any conventional account, an agent that played Pengo would maintain a world model of the evolving game board. This model would assign symbolic names such as BEE-34, BEE-35, ICE-CUBE-61, and ICE-CUBE-62 to all the objective individuals that the agent incorporates in its model of the world. The world model would be maintained by a process, presumably involving perception, that kept track of changes in the world and updated the world model to reflect them. Decisions about what to do next would involve, in one way or another, the simulation of various courses of action within the world model. This is an objective style of representation insofar as its elements correspond to objects in the world, each of which is identified without regard to the agent's location or goals. Maintaining a world model in Pengo would require a great deal of redundant computation, given the amount of constant change on the board and the small proportion of the board that has functional significance on any given moment.

Deictic representation is well suited to an activity such as playing Pengo. Here are some of the entities with which Pengi interacts at various times:

- *the-ice-cube-I-am-kicking.* This is simple enough. The "I" is a metonymic shorthand; a more accurate name would be *the-ice-cube-which-the-penguin-I-am-controlling-is-kicking.*

- *the-bee-I-am-attacking.* If Pengi is attacking a bee over some period, it might maneuver to get behind an ice cube to kick at the bee. The whole time, in virtue of participating in that pattern of interaction, that bee will be *the-bee-I-am-attacking.*
- *the-bee-on-the-other-side-of-this-ice-cube-next-to-me.* If the penguin is aligned with an ice cube and a bee, it is important to keep an eye on the bee since it could kick the ice cube. Or perhaps Pengi can kick the ice cube at *it.*
- *the-bee-headed-along-the-wall-that-I-am-on-the-other-side-of.* It is good to lurk behind a wall of ice cubes because the wall provides a supply of ice cubes to kick at bees. Pengi might dance around behind the wall and if a bee comes into range then Pengi can find the ice cube that lines up with it and kick it. This dynamic involves many entities: *the-wall-I-am-lurking-behind, the-bee-I-would-like-to-kill, the-ice-cube-in-the-wall-I-am-lurking-behind-that-aligns-with-the-bee-I-would-like-to-kill,* and so forth.

Pengi's entities will tend to refer to the objects in the general vicinity of the penguin. Objects and events at the other end of the board will probably have no functional significance. But it is functional significance, not proximity, that leads Pengi to focus on particular objects. If a bee is kicking an ice cube at the penguin from halfway across the board, Pengi ought to spot the cube soon enough to evade it gracefully. Likewise, most of the ice cubes near the penguin will have no special significance. (Ice cubes are rarely important obstacles since they can readily be kicked out of the way. An ice cube that cannot be moved, perhaps because it is up against the wall, will disintegrate and disappear once it is kicked a few times.)

Pengi's central system can get along without a pattern matcher or other complex variable-binding facilities because deictic representation permits it to generalize across indexically and functionally equivalent situations. If Pengi attacks BEE-34 at one moment and BEE-35 the next, an objective representation scheme would represent those two episodes differently. But all that matters to Pengi is that it is engaged in a certain routine pattern of interaction with *the-bee-I-am-attacking.* If Pengi knows what to do about *the-bee-I-am-attacking,* it will do it in each case without bothering to distinguish different objective individuals.

As Pengi actually decides what to do, it reasons in terms of aspects of the various entities. A typical aspect that Pengi registers is *the-bee-I-am-attacking-is-running-away-from-me.* (Again, "*I*" and "*me*" are metonymic shorthand. The bee is running away from the penguin, which the player is controlling.) It is an urgent situation when *the-bee-on-the-other-side-of-this-ice-cube-next-to-me-is-closer-to-the-ice-cube-than-I-am,* since the bee could fly up to the ice cube and kick it at the penguin before the penguin could run up to the ice cube and kick it at the bee. Consequently, Pengi must get the penguin out of the way. The situation is less urgent if *the-bee-on-the-other-side-of-this-ice-cube-next-to-me-is-moving-away-from-the-ice-cube,* though. The next section provides a sketch of how Pengi actually makes use of this knowledge in deciding what to do. Subsequent sections describe the program's architecture, show the program in operation, and offer some evaluation.

How Pengi decides what to do

Figure 13.2 is a diagram of Pengi's best trick. In the diagram is a bee that is not lined up with any ice cube. Lacking any direct way to kill the bee, Pengi takes an indirect approach: moving an ice cube so that it aligns with the bee. This is a common technique in Pengo-playing; it is probably impossible to win the game without it. In the example in the diagram, Pengi can kick the ice cube on the left so that it hits the ice cube on the right. (Momentum is not conserved. The moving ice cube stops dead when it strikes the stationary one.) Now that the ice cube is aligned with the bee, the penguin can move around behind it and kick it.

One might be tempted to refer to this trick as a plan. Yet Pengi neither makes nor uses plans. The two-ice-cube trick is one of the routine patterns of activity that we foresaw when building Pengi's central system circuitry. Pengi engages in the routine because it can figure out something reasonable to do in each successive type of situation, not because it refers to a plan or other representation of the routine. Let us contrast the way Pengi performs this trick with the way a conventional planner-and-executor might perform it, on three counts.

One count is, why would each type of agent kick the ice cube in the first place? How does it know that this is a good move? A plan-construction device would conduct a simulation in its world model. Through inference over some logical codification of the game board's geometry, it would deduce that this ice cube can be moved from there to

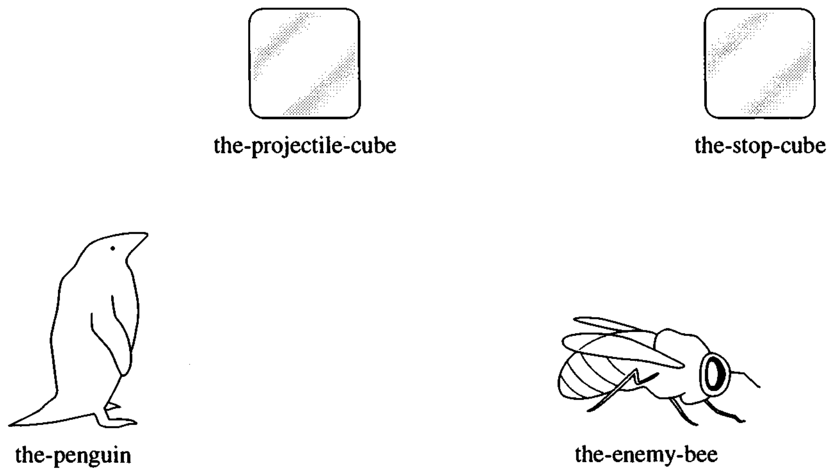


Figure 13.2. Pengi's best trick is to move one ice cube into position by kicking it up against another one.

there to there. Pengi does something different. Instead of performing inference from a world model, it visualizes potential movements. As subsequent sections will explain, Pengi's visual system provides a set of *visual operations:* "coloring" regions, finding out whether things are lined up, finding something of a given shape or color or type of motion in a given vicinity, telling whether anything is located between two designated points, marking a spot in the visual field for future reference, and several others. These visual operations are combined into *visual routines* by means of which the central system uses the visual system to register the current values of various aspects of its environment. The actual situation will contain all kinds of visual clutter not shown in the figure, but by running these visual operations Pengi will be able to visualize that these two channels are clear and that the second ice cube is in position to stop the first one.

We designed the circuitry for Pengi's visual routines ourselves. But a growing expertise at any activity, from cooking to mathematics, includes acquiring a set of visual routines and learning to use the visual operators more efficiently, more sparingly, and more accurately. Beginners use their visual operators in crude, generic, safe, redundant ways; as they become skilled, their visual routines evolve to more efficient and adaptive forms.

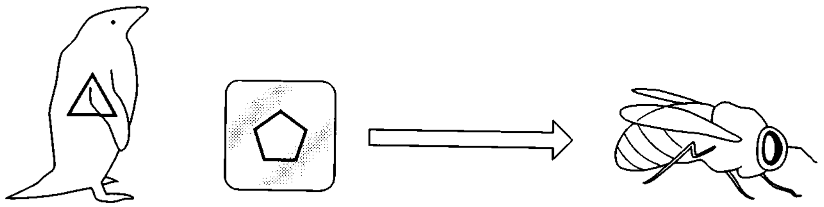


Figure 13.3. Pengi visualizes where the ice cube the penguin has just kicked will end up by using its visual operators. The polygons are visual markers.

Further work is needed to describe how visual routines evolve in the context of larger activities.[1]

A later section will explain Pengi's visual operators in detail, but here is an example. Figure 13.3 shows how Pengi finds *the-ice-cube-that-the-ice-cube-I-just-kicked-will-collide-with.* Pengi has just kicked this ice cube and has visually *marked* it. It also knows the direction the ice cube is moving. It now wants to know where the moving ice cube is going to end up. So it uses an operator that projects the ice cube's path and drops a second marker on the first thing it hits. Once this operation is finished, Pengi can use further visual operations to register particular aspects of *the-ice-cube-that-the-ice-cube-I-just-kicked-will-collide-with.*

The second count on which we might contrast Pengi with a conventional planner-and-executor concerns the reason why each of them would kick the ice cube the *second* time. An agent executing a plan would take that step because the program counter in its executor is equal to two. That is, the executor chooses its next action by following a pointer to a planlike structure such as (kick a). Pengi, by contrast, kicks *the-projectile-cube* because it is lined up with the bee. It sees a clear channel, an ice cube, a bee, and a penguin in the right spatial relationships, so it moves the penguin behind the ice cube and kicks it at the bee. Pengi does not realize that the ice cube got there through the penguin's previous actions, although these actions are liable to have left the player's visual system focused on that ice cube and therefore likely to notice its functional significance. In this sense Pengi is, at each next moment, deciding what to do in the world as it is then.

Among AI people, the proposition that Pengi does not employ plans is regularly met with incredulity. It is a common view, for example, that

Pengi's combinational logic circuit constitutes a large, highly conditionalized plan, with the operation of this circuit constituting the plan's execution. Even though I find this view odd, many intelligent people claim to find it obvious. The circuit, I would argue, has few if any of the conventional attributes of a plan. It is not a text or any other kind of symbolic structure. Even if it were, it is not "written" in anything resembling a conventional programming language. (Subsequent sections will describe a Lisp-embedded language for specifying Pengi's circuitry, but this is a language for constructing a simulation, not a language of thought whose structures can be manipulated by Pengi itself.) It does not participate in any architectural distinction between construction (or selection) and execution. It is unlikely that any plan-construction device could automatically analyze the dynamics of Pengo-playing sufficiently to construct such a "plan"; the construction of highly conditional plans is a difficult technical problem (but see Schoppers 1995). These arguments, unfortunately, do not convince many AI people. A plan for them, as for Miller, Galanter, and Pribram (1960: 16), is *whatever* is responsible for structured behavior. At this point the issue may seem to turn on nothing more profound than our chosen definition of the word "plan" and whether Miller, Galanter, and Pribram's definition is broad or simply vacuous. But in fact the issue is deeper. A plan, however defined, is responsible only for the structure of an agent's behavior in the broadest and most misleading sense. The structure of behavior – or, in my preferred idiom, the organization of activity – does not arise from the agent alone, or from its environment alone, but from the interaction between them. Even if it were granted that Pengi's individual actions are caused by planlike internal machinery, the organization of Pengi's activity is not caused, in any useful sense, by anything located inside of Pengi itself. Plans or no, the organization of activity is an emergent attribute of the player's continual interaction with the game, best understood in interactionist terms.

Pengi's ability to rederive its course of action from the evolving game situation makes it more flexible than the planner-and-executor in the face of unexpected situations. The target bee might fly out of range. Or another bee might fly dangerously close or kick *the-stop-cube* out of the way. As Chapter 8 has already argued in detail, an agent that decided what to do next by consulting a plan would be unable to recognize the relevance of these circumstances. Execution monitoring can prescribe a fixed list of relevant conditions, any of which would cause control to be returned to the planner, but applying execution monitoring to Pengo would require such a long list that new plans would still be required every few moments. Pengi's improvisatory approach, by contrast, takes continual account of the entire background of relevant circumstances – that is, all of the circumstances that would be taken into account in devising a plan. In particular, Pengi regularly abandons one of its routines when a better opportunity arises. Having pursued the opportunity, it might return to the original routine if that seems like its best option when the time comes.

The third count is machinery parsimony. Both Pengi and the conventional planner-and-executor can perform the two-ice-cube trick, but Pengi does so with simpler machinery. Pengi has no need of plans or symbolic structures. All Pengi needs is the ability to look at the world and decide what to do, something any situated agent needs. Pengi has just enough state to keep a finger on a few important aspects of the world, but it does not have a model of the world. An understanding of dynamics leads to simpler machinery.

It can help to think of Pengi's strategy as a dynamic version of the GPS technique of *difference reduction* (Newell, Shaw, and Simon 1960; see also Chapter 8). Difference reduction was originally a technique for searching problem spaces. Given a state known to be reachable from the initial state, the problem solver computes a set of *differences* between that state and the goal state. It then indexes those differences into a table to discover which ones it can reduce by applying an operator. The operator will produce a new state. If that new state is the goal state then, in Newell, Shaw, and Simon's vocabulary, the problem is considered solved. If not, difference reduction can be applied to the new state. This method can be repeated until the problem is solved. All of this happens in the mind, not in the world. To "apply" an "operator" does not involve taking an action, only simulating the action's effect.

Dynamic difference reduction, by contrast, happens in the world. The agent alternates between finding something to do and doing it. In the case of the two-ice-cube trick, each of the player's actions reduces a difference: kicking the ice cube once it is lined up with a bee, and kicking it again kills a bee. One could imagine an agent making an omelette the same way, repeatedly looking for things that need doing and then doing them (Agre and Horswill 1992). Using this method requires some knowledge of what steps will have to be taken, but it requires no explicit representation of serial order. The agent might take the actions in different orders on different occasions, according to the happenstances of noticing. But as long as it understands the preconditions of its actions – almost all of which are readily visible and furthermore enforce themselves by rendering the actions impossible by their absence – it will get the job done each time. Since these actions take place in the real world and not in a mental search space, they cannot always be retracted ("backtracked") by simply moving to a different location in the space. The agent engaging in dynamic difference reduction thus needs to be more careful, choosing difference reductions judiciously, whether through custom or experience or the benign construction of the environment.

Pengi certainly has limitations. It is hard to be precise about where the capabilities of Pengi's architecture leave off, since such machinery might have the capacity to participate in dynamics we have not yet considered. Nonetheless, Pengi was designed to participate only in a certain activity, not in the whole of everyday life. Other domains might require Pengi to have other capabilities. For example, if Pengo got harder, Pengi might sometimes have to refer to a plan. The plan might explain how to deal with some tricky situation, or perhaps what strategic issues bear on the matter of which bees to attack when. Since Pengi would still be choosing its actions from moment to moment, the plan would not be a computer program; instead, it might consist of natural language or something like it (Agre and Chapman 1990; Chapman 1991). Other activities will require a central system to maintain state, to use its visualization abilities without any domain materials being present, and so forth. Precise specification of this additional machinery should be guided by descriptions of the relevant dynamics.

Architecture

Figure 13.4 shows the top-level modularity of the system, drawn in the conventional manner with boxes and arrows. The player consists of a central system and a periphery, which in turn consists of a visual system and a motor system, each of which is connected to the game itself. The visual input from the game is updated on a fairly rapid clock, and the player generates a fresh set of commands to the game on that same clock. The various boxes and arrows must be described on two levels. One level

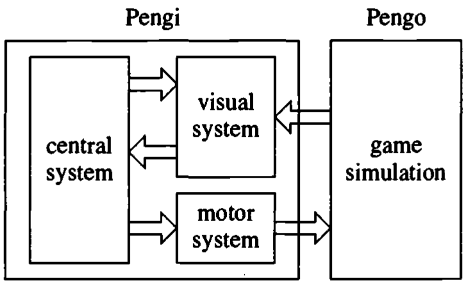


Figure 13.4. Pengi is composed of a central system made of combinational logic, a moderately realistic visual system, and a trivial motor system. It interacts with its simulated world on a fairly fast clock.

is the ideal of a real video game from which an actual video cable leads into a parallel computer, which performs early visual processing and connects to the parallel central system hardware. In reality, though, large parts of this are simulated. The player's implementation employs no early visual processing, because the technology of early vision is not yet ready for routine use. Instead, the visual system simulates its early vision by inspecting the game's internal data structures and figuring out what answers the various visual operators ought to return. Many important issues are certainly ignored in this way. The operation of the central system's circuitry is simulated as well, but fewer important issues are suppressed there, since the simulation is conducted at the wire-and-gate level. But for expository purposes, I will proceed as if the system were organized according to the ideal.

Motor control in Pengi is very simple. In the arcade, one's left hand is on a joystick that can move left–right–up–down and one's right hand is on a button for kicking. Pengi does not model the relevant motor skills in any detail. Its three action wires encode the possible values passing from the central system to the game simulation.

Computationally speaking, the visual system and the central system differ in their use of parallelism. Both systems perform a tremendous amount of computation all the time, but they distribute their parallelism differently. The visual system's computation is parallel across the image. As Chapter 4 has described, it performs locally connected parallel computation among a sequence of flat arrays of simple elements, one array per stage of visual processing. These flat arrays generate a series of transformed versions of the visual field. Certain kinds of global information might be collected or distributed, but these are relatively few; and certainly the visual system builds no elaborate symbolic structures. Once computed, these representations are all available to the central system. The central system's parallelism, by contrast, is distributed across considerations. A wide variety of matters might enter into determining the best next action. Many different details might become relevant with little warning. The player might consider taking several different actions. Time is short and it must continually decide. Thus, the central system must be able to consider its options, the reasons for and against them, and the issues that bear on its decisions – all in parallel. Ultimately, of course, at the point where motor commands are issued, all that parallelism has to result in some decision – left or right, up or down, kick or not kick – so all of the logic circuitry converges eventually. Nonetheless, a great deal of parallelism is possible.

Pengi's account of the boundary between the visual system and the central system roughly follows Ullman's (1984) theory of visual routines. Though we agree with Ullman's conception of visual routines as employing a set of primitive visual operators (Ullman refers to these as *elemental operations* or *basic operations*), we differ from Ullman in how we imagine the routines themselves to be implemented. For Ullman, a routine resembles a computer program that might be either compiled on the fly or retrieved from a library. For us, by contrast, a visual routine is merely a theorist's abstraction for a common pattern of interaction between an agent's central system, its visual system, and its environment. The player decides on each next moment what operations to invoke based on its current understanding of its situation and options.

The boundary between the visual system and the central system is like a horizontal microinstruction set.[2] The visual system presents to the central system a set of visual operators that the central system can invoke at any time. The central system uses familiar methods from digital logic design. A typical visual operator might follow a line, shade in a region, pick out the red patch and put a marker on it, or tell whether the red patch is moving. Figure 13.5 is a diagram of the general case of a visual operator's interaction with the central system. (Few of the operators are this complicated.) Operators can take arguments, all of which are simple one- to three-bit quantities, *not* pointers or symbolic names or structured information. (The arguments may or may not be necessary, as we will

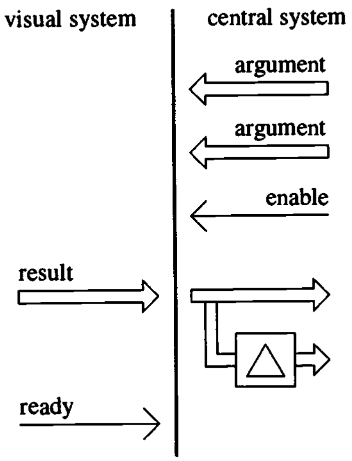


Figure 13.5. Pengi's central system interacts with its visual system through its visual operators. Many visual operators provide a constant readout of some aspect of the visual image, but others have more complex interfaces. Some of them take arguments, all of which are carried on one- to three-bit buses. Others cause side effects and are governed by a control line that indicates when they should take place.

see.) Most operators take one or two arguments, though a couple take more. Once the arguments are available, the circuitry can assert an *enable* line to actually perform the operation. A result is produced. There might be a number of results, but only a few operators have more than a single bit of result. Another bit, the mirror image of the enable bit, indicates that the result is ready. The operation might also cause side effects within the visual system.

The results that come back from a visual operator, in addition to being immediately available to the central system's circuitry, also go through delay lines. Thus, the central system can use the results from both this clock cycle and the preceding one. The reason for this, briefly, is that the central system has no state. It is combinational logic that is clocked on its inputs and its outputs. It has no place to remember things like what it was doing, or what large goal it is pursuing right now, or what use it is making of a given operator right now – whether the operator, for example, is referring to a bee or to spaces between ice cubes. Thus, the operator values that motivated the last cycle's queries are kept around long enough for the system to determine what use to make of the answers. The solution of providing delay lines on operator results may not seem very general, but it is in accord with the principle of machinery parsimony: only postulate the minimum amount of machinery necessary to explain the dynamic phenomena cleanly. One could implement the idea of "remembering what you were doing" with a full-blown state machine. But if a simpler scheme can, without unworkable complexity, limit the state to just a delay and corner it in one part of the system, it is better to adopt the simpler scheme until evidence or experience dictates something more general.

Roughly speaking, the system uses a standard two-phase clock. The central system has inputs and outputs and gates in-between. (Our convention is to reckon inputs and outputs from the point of view of the central system, not the visual system.) When the inputs change, the clock lines on the outputs must wait until the circuitry has settled down before admitting the outputs into the visual system. And once these commands to the visual system are ready, they must be driven long enough for the visual system to run. Since the wires leading from the central system to the visual system are clocked in this way, the wires going the other way – that is, the inputs to the central system from the visual system – do not have to be clocked, though in the current implementation they are anyway for clarity. Thus, one two-phase clock controls the inputs to the central system and the other two-phase clock controls the outputs from the central system. The motor outputs are clocked together with the other outputs. The input clock lines also govern the delay lines.

Visual system

Conceptually, Pengi's visual system has two components, early vision and intermediate vision. Each component is characterized by its form of computation. Early vision performs spatially uniform, bottom-up, locally based operations on the retinal image to produce a set of two-dimensional *base representations* of the visual scene. Its various modules find edges, calculate depth from stereo disparity cues where possible, infer surface normals from shading cues where possible, and so forth. (Again, in Pengi this is all simulated.) Intermediate vision includes the machinery implementing the visual operators. The visual operators perform hierarchical aggregations of the information available in the base representations in order to individuate objects, mark patches with certain specified properties, calculate spatial relationships among the marked patches, and so forth. Pengi performs all of these computations in simulation.[3]

Pengi's visual system provides about twenty visual operators. The design of these operators is governed by the principle of machinery parsimony: postulate only the minimally necessary operators and use the simplest operators that will cleanly do the job. Beyond that, the set of visual operators should have several other properties:

1. The operators should permit Pengi to play a competent game of Pengo.
2. Each operator should plausibly be computable by the sort of visual system architecture we have envisioned.
3. The operators should not strike our engineering judgment as poorly designed.
4. The operators should be domain-independent, in the sense that specifying or computing them should not invoke Pengo-specific knowledge.
5. Each operator's existence and behavior should be consistent with the results of psychophysical experimentation.
6. The set of operators should be sufficient and convenient for a broad range of activities, ideally including the whole of human life.

Later in this section I will consider Pengi's visual system against these considerations in detail. For the moment, I should mention that whereas the first three properties are nearly satisfied, some serious deficiencies remain: the operator set is very incomplete, arbitrarily specifies several underconstrained design issues, needs far more psychophysical verification, and (as we will shortly see) contains some Pengi-specific operators in areas where the current state of research did not offer us enough guidance to formulate a serious theory.

The fourth criterion is that the operators have to be domain-independent. This domain independence has two aspects. The relatively easy aspect is that the operators should not depend on domain-specific facts to perform their operations. The operators should not be defined in terms of domain-specific concepts and categories. The current operator set violates this rule in a couple of places, but the violations are not important. They could easily be repaired, but the available evidence didn't sufficiently constrain them.

The second aspect of domain independence is more subtle: the visual operator set should not be biased or restricted to some domain or some kind of domain. This condition can be hard to judge. Ideally someone should implement a set of visual operators that satisfies at least the first three criteria in some completely different domain. One could try implementing visual operators for a device that picks up industrial parts or makes breakfast or drives around a small town. One could then compare the results for the various activities, try to make them compatible, and combine them to make a set of visual operators that is sufficient for each individual activity and eventually for all human activity.

The fifth criterion is that the operators' existence be verified by psychophysical experimentation. This criterion could conceivably conflict with the third and fourth criteria. That is, people might have some operators that strike us as inelegant or domain-specific. But until we are faced with convincing evidence, the principle of machinery parsimony will recommend resisting such hypotheses.

The visual operators are defined in terms of three concepts:

> *Objects.* The visual system individuates *visual objects.* A visual object is not a three-dimensional articulated object and Pengi's visual system does not perform any kind of "object recognition." Instead, the visual system marks off as an object, purely on visual evidence, a patch of relatively uniform qualities that is relatively localized (in other words, not complicatedly distended), moves as a coherent whole, and has delineable boundaries. Several of the operators are defined in terms of these objects. The Pengo world, conveniently, has exactly three types of objects: penguins, bees, and ice cubes. The three object types are readily distinguishable, largely because the video game's designer has worked to make them so. It is up to the central system to invoke the operators that distinguish the various object types.
>
> *Indexing operations.* Pengi has operators that permit it to pick out the *odd man out* (Ullman's phrase) in a visual scene. The indexed feature might be the only red patch in the image, or the only curved patch, or the only moving patch. The properties the visual system can pick out are called *indexable properties.* Typ-

> ically only simple properties are indexable. For more complex properties, one must use a number of operators or even resort to serial scanning of the visual scene.
>
> *Markers.* Many visual operations are predicated on a focusing mechanism, suggested by Ullman, involving *markers.* Markers implement a certain sense of focus. Using a marker, one can mark a location in the visual field for future reference. Suppose you have gone to the work of locating a place on the image where something interesting is going on. If you then go off and focus somewhere else, under certain conditions you can jump quickly back to your previous location. Many operators are defined in terms of markers, especially operators for determining the spatial relationships among the locations they mark. A marker resting on a moving object moves along with it; this is called *tracking.* Pengi has six markers.

(We have defined our versions of the indexing and marker concepts broadly enough that they subsume the functionality that Ullman refers to as *shifting the processing focus.* Pengi has no operators for what Ullman calls *boundary tracing.*)

Before listing the operators, let us describe how the central system uses the visual operators. Each operator has a protocol of a sort that follows standard logic design practice. Most operators have both inputs and outputs.

> *Inputs.* Many operators have arguments, which are usually three-bit buses specifying markers, directions, or distances.[4] Each operator that has side effects has a single-bit *activation* input wire that tells the visual system to perform the operator. The operators with no side effects run on every cycle.
>
> *Outputs.* Almost all outputs are binary signals that answer specific queries. Some operators are guaranteed to produce an accurate answer on their output wires by the next clock cycle. Other operators each supply a *ready* output wire that indicates when the output wires have been assigned a new set of values.

All the output wires are clocked so that their values remain available to the central system until they are revised with new values, usually on the next clock cycle. The input wires are clocked so that the visual system only sees their values once the central system's circuitry has had a chance to settle down after the last change of the output wires' values.

Operations on markers fall into five groups.

Indexing operators

index-thing!(*m*,*t*): Cause marker *m* to move to some *t* (either bee, penguin, or cube), if one is visible. If it is already on one, choose a different one.

index-thing-near!(*m*,*n*,*t*): Cause marker *m* to move to some *t* (either bee, penguin, or cube) in the vicinity of the object marked *n*, if one exists. If it is already on one, choose a different one.

index-moving-thing-near!(*m*,*n*,*r*): Cause marker *m* to move to a moving object within distance *r* of marker *n*. *r* is one of a small number of roughly exponentially increasing distances.

Marker assignment operators

warp-marker!(*m*,*n*): Move marker *m* to the same location as marker *n*. If marker *n* is tracking an object, then marker *m* will commence tracking that object as well. (There are actually two instances of this operator, called warp-marker!1 and warp-marker!2. I will explain this and several other odd facts later.)

warp-freespace!(*m*,*n*,*d*): Move marker *m* onto whatever object you find, if any, by starting at marker *m* and moving over the empty space in direction *d*. *d* is one of north, south, east, west.

unassign-marker!(*m*): Remove marker *m*, so that it no longer has any location.

Marker inspection operators

marker-*m*-assigned?: Is marker *m* assigned? (There is one such operator for each marker that ever needs it. Each operator is always running.)

marks-bee?(*m*), marks-penguin?(*m*), marks-cube?(*m*): Is marker *m* resting on a bee/penguin/cube?

marker-*m*-moving?: Is marker *m* moving? (If so, it follows that marker *m* is tracking a moving object.) (There is one such operator for each marker that ever needs it. Each operator is always running.)

Marker comparison operators

toward?(*m*,*n*): Is marker *m* moving toward marker *n?*

near?(*m*,*n*,*r*): Is marker *m* within a distance *r* of marker *n? r* is one of a small number of roughly exponentially increasing distances.

nearer?(*n*,*p*,*q*): Is the distance between marker *n* and marker *p* greater than the distance between marker *n* and marker *q?* (Pengi has two copies of this operator, named nearer?1 and nearer?2.)

direction-from-*m*-to-*n:* Returns a code indicating the direction from marker *m* to marker *n.* This code consists of two two-bit outputs, representing the major and minor axes (out of the four compass directions) of the vector from *m* to *n.* This operator is replicated across several pairs of *m*,*n.* All these replicated operators are running all the time.

aligned?(*m*,*n*): Are markers *n* and *m* aligned (within some fixed, small tolerance) along either axis? If so, an output line encodes which axis. (Pengi has two copies of this operator, named aligned?1 and aligned?2.)

Object comparison operators

adjacent?(*m*,*n*): Are the objects under markers *n* and *m* adjacent?

wholly-beside?(*m*,*n*): Is the object under marker *n* wholly in direction *d* from the object under marker *m?*

freespace-between?(*m*,*n*): Is there nothing but empty space between the objects under markers *n* and *m?*

This list of operators reflects a series of design choices. Lacking enough empirical information, we had to make many of these choices arbitrarily.

It is an unresolved empirical issue how many markers people have – probably two or three. It is also not clear whether all markers track moving objects. (In Ullman's version of the theory they cannot.) Perhaps only one of them can, or only a few. Pengi rarely needs to track more than two, one of which is the penguin.

Most of the operators take particular markers as arguments. The aligned? operator, for example, consults two three-bit input buses, each of which encodes one of the markers whose locations it should compare. I am not sure if this is reasonable. What is the engineering trade-off? The circuitry for implementing the argument scheme, while not baroque, is a little complicated. One might choose, instead, to provide a separate operator for every pair of markers. But this alternative scheme would seem to require many more operators than the evidence currently warrants. A compromise proposal would be to provide operators only for the markers that need them. One would like to make this assignment in a principled fashion. Perhaps some heavily used markers have many operators, whereas others are used only in special circumstances. This seems like a reasonable proposal, but evaluating it will require some experience in assembling suitable operator sets for other domains.

The indexing operators take types of objects as arguments; Pengi's visual system can primitively distinguish bees from penguins from cubes. At first sight this might seem intolerably domain-specific. But in the video-arcade version of Pengo, the various types of objects are readily distinguishable by properties that might be expected to be indexable: their colors, rough shapes, and characteristic forms of motion. Pengi's visual system can distinguish the various types primitively, but we could just as easily have arbitrarily assigned distinct colors to the different types of objects and provided an index-color! operator.

We chose to make the index! operators take the object type as an argument rather than providing separate operators for each type out of simple parsimony. The question of whether the indexing property is a parameter or whether each one has a separate operator will have to be investigated in domains in which one uses a greater variety of indexable properties.

Some of the operators neither take arguments nor cause side effects. Pengi maintains a convention that such operators require no activation inputs and continually update their values. They might be thought of as providing a readout of a continually computed parameter of the image. This convention seems fairly reasonable for the particular operators in this set, but it need not be reasonable in general, depending on the practicalities of the machinery implementing the architecture. Two sets of always-running operators, marker-assigned? and marker-moving?, are replicated across all the markers.

A third set of always-running operators, direction-from, is replicated across certain pairs of markers – about half a dozen pairs. This is because Pengi often needs to keep track of the spatial relations among several moving objects at once. We have no principled reason to believe that this particular set of marker pairs should be sufficient across all tasks. But Pengo-playing, with all its incompletely predictable moving objects, requires an enormous amount of continual analysis of spatial relations compared with most activities. It is an empirical question whether Pengi can judge more spatial relations at one time than people can. Most likely it can. The machinery hypothesized for shifts in selective attention by Koch and Ullman (1985) is also more strongly focused than Pengi.

As a general matter, it seems more necessary to replicate operators across their possible arguments when different lines of reasoning have frequent, conflicting needs for the same operator. Often we find a trade-off between complexity of arbitration schemes in the central system and proliferation of operators. We have tried to call each trade-off according to our engineering judgment, but more experience with other domains will be required to make these choices in a more principled fashion.

A few operators have multiple copies in cases where it seems necessary for Pengi to use the operator in multiple ways, with different arguments, during the same cycle. We are unsure how reasonable this is, but we expect that the necessity is specific to time-pressured domains like Pengo. In any event, I suspect that much of the skill of playing Pengo lies in developing complex schemes for sharing operators across the various purposes to which they continually must be put. I also suspect that some of these schemes are more complex than we can comfortably wire up by hand in building Pengi's central system.

Some of the indexing operators start "near" some already marked location. (They are related to what Koch and Ullman call *proximity preference* in shifting operations.) These operators offer no strict contract about which object they will pick out or whether it will be the very nearest, except that if they are activated often enough the operators will eventually pick out each of the objects in turn. Experimental evidence on *return inhibition* (Klein 1988) suggests that we would be justified in making Pengi's indexing operators cycle through all the nearby indexable objects rather than always choosing new ones randomly. This would improve Pengi's performance since it would be able to focus on newly dangerous ice cubes and bees more quickly.

Likewise, the comparison operators that determine "nearness" and comparative "distance" are not guaranteed to be very accurate, and the central system should not depend on any great accuracy. Some applications would probably benefit from a more carefully specified qualitative contract, however.

Some of the operators take a "distance" argument. The argument can take only a small number of values that correspond roughly to the various scales on which the visual system operates. Pengi's visual system reflects no worked-out conception of multiple-scale vision, but I suspect that many operators are replicated across scales. Another possibility is that the whole operator set employs one scale at a time and the central system can change this scale whenever it wants. In any event, Pengi uses only one of the legal values for distance, a value corresponding to about a quarter of the width of the game board.

Some of the operators suggested by Ullman employ a notion of *bounded activation* or *coloring.* Ullman suggests that some visual operators "color in" certain distinctive regions of the visual field. (This idea is unrelated to the perception of colored light, as in red, yellow, and blue.) This can help, for example, to determine whether something is inside or outside a curve, or to sweep out certain significant regions of the scene. It is also useful in separating figure from ground. Still, although coloring must play an important role in other domains, it has not yet found a use in Pengi. I suspect that, in playing Pengo, coloring is necessary only for some advanced navigation amid mazes of ice cubes.

It is evident that the design of these visual operators is underconstrained. What is more, Pengi's visual operators fail to address many issues that do not arise in playing Pengo. Examples include texture, scale, shading, and depth. In a more realistic set of operators, the individual operators would have to be more general and there would have to be more operators for all of those concepts. Perhaps a hundred different operators are necessary, but it is far too early to tell. In the end, it is an empirical matter what operators human beings have and whether those operators would be ideal for robots. Psychophysical experiments can help resolve such questions. Ullman (1984) describes a number of such experiments.

Central system

Recall that the central system is made of combinational logic. Inputs arrive from the visual system for both the current clock cycle and the previous one; the circuitry generates the outputs. Some of the outputs are operator requests to the visual system; others are commands to the motor system. The central system consists of several hundred gates, most of them specified individually in a simple language.

The hardest part of building the central system, particularly in this domain, concerns contention for visual operators. Often Pengi will have three different uses for an operator at once. Consider, for example, the operator that determines whether two objects are close together. Pengi might want to know if the bee is close to the obstacle and if the penguin is close to the projectile at the same time. Thus arises the difficult technical question, already mentioned in the discussion of the visual operators, of how to arbitrate among competing claims on an operator. This question has a number of answers. In building Pengi's circuitry we used some conventional patterns of gates for representing conflicts and priorities among various claims on an operator. When this does not suffice, it can become necessary to replicate the visual operators. There is a trade-off between complex forms of arbitration and operator replication, but this is not the best domain to explore the trade-off. The real experts in the game clearly use their visual operators in very sophisticated ways. They may depend on properties of their circuitry and operators that Pengi does not model, such as the slow propagation time of brain circuitry.

To understand the remainder of this section and most of the next section, the reader will have to be comfortable with Lisp programming and digital logic design.

We built the circuitry with Lisp procedures; the simulator calls this code before starting the game. To build an AND gate we call `andg`. The `andg` function's arguments are the gate's input wires; its output is the gate's output wire. The `org` function builds OR gates; the `invert` function builds inverters. We can nest these calls, making an AND gate whose inputs are taken from the outputs of OR gates whose inputs are taken from the outputs of inverters. The wires can be assigned to global variables or passed around as arguments. Constants include the various directions and scales that are sent to the operators; typically they are one to three bits worth of information. Another set of functions generates simple circuitry for manipulating these buses. For example, `if-bus` takes a condition wire and two buses and returns an output bus that is driven by one of the two input buses according to the condition; this is a simple matter of logic design, two gates for every bus line. (This simple scheme for specifying circuitry is not novel. It resembles Kaelbling's Rex language [1987].)

The code that builds the central system's circuitry revolves largely around the problem of deciding which values to place on the operators' input wires. Much depends on whether a given operator has more than one use. When an operator has only a single use, one can use the `set-inputs!` form to permanently set its inputs. For example,

```
(set-inputs! marks-penguin? marker penguin-marker)
```

says to set the `marker` argument to the marks-penguin? operator to the value of the global variable `penguin-marker`, which is 0. Thus, the marks-penguin? operator's result will always indicate whether marker 0 is resting on the penguin.

Canned arbitration circuitry is necessary in the more complicated cases. This circuitry is generated by a set of functions – `condition`, `overrides-action`, and several others – that expand into calls on basic circuit-construction functions like `andg` and `invert`. These functions implement much the same argumentation scheme first discussed in Chapter 9 and used in the rules in RA. As in RA, the metaphor is that various patches of circuitry in the network conduct an argument. A patch of circuitry can propose an action, either a primitive action (such as kicking or moving a marker) or a compound action (like engaging in a certain bee-hunting tactic). Another patch of circuitry might make its own proposal or else raise an objection to the first proposal. A proposed action is taken provided no objection against it is sustained. The `condition` function declares a condition under which some objection should be posted against an action. The `overrides-action` declares that the taking of one action constitutes an objection to some other action. Each function defines appropriate circuitry in a straightforward fashion.

For example, the following code arranges for marker 0 to index the penguin as soon as the game begins:

```
(action index-to-penguin index-thing!
  marker penguin-marker
  object-type (constant 'penguin 'object-type)
  doit? *t*)
```

The `action` function defines an *action* named `index-to-penguin`. In Pengi's terminology, an action is an assignment of values to an operator's inputs. The `action` function creates circuitry that gates these values onto the operator's inputs on those cycles when Pengi decides to take that action. In this case the operator is index-thing!, which has three inputs:

the marker to be moved (i.e., `marker`), the indexable property to be employed (i.e., `object-type`), and the operator's activation line (i.e., `doit?`).

This next bit of code ensures that Pengi performs only the `index-to-penguin` action on the very first cycle of the game. Once Pengi does perform `index-to-penguin`, marker 0 will rest on the penguin for the remainder of the game. As a result, the marks-penguin? operator, whose `marker` input we set to 0 a moment ago, will return false on the first cycle and true forever afterward:

```
(condition index-to-penguin (invert *marks-penguin?-result*))
```

The `condition` function takes an action name and a wire. Here the wire is the output of an inverter whose input is the result of the marks-penguin? operator. (In general, the result of an operator is a wire or bus bound to a global variable whose name takes the form `*`*operator*`-result*`.) The index-thing! operator is also used to find bees; another bit of code ensures that the action implementing this use of the operator is suppressed while Pengi is finding the penguin:

```
        (overrides-action index-to-penguin index-to-bee)
```

The `overrides-action` function takes two actions and generates circuitry that ensures that the second one is suppressed whenever the first one is activated.

To summarize, the bits of code I have just discussed construct circuitry for three closely related purposes: (1) to ensure that the result of the marks-penguin? operator always indicates whether marker 0 is located on the penguin, (2) to use the index-thing! operator to move marker 0 onto the penguin on the first cycle of the game, and (3) to ensure that Pengi does not attempt to use the index-thing! operator to find both penguins and bees on the same cycle.

Example

Figure 13.6 presents a schematic situation from a Pengo game. The penguin finds itself in a particular situation involving an ice cube and a bee. (In a normal situation more ice cubes and bees would be found nearby.) It would probably be a good idea for the bee to fly up behind the ice cube and kick it at the penguin. How does this work? In my explana-

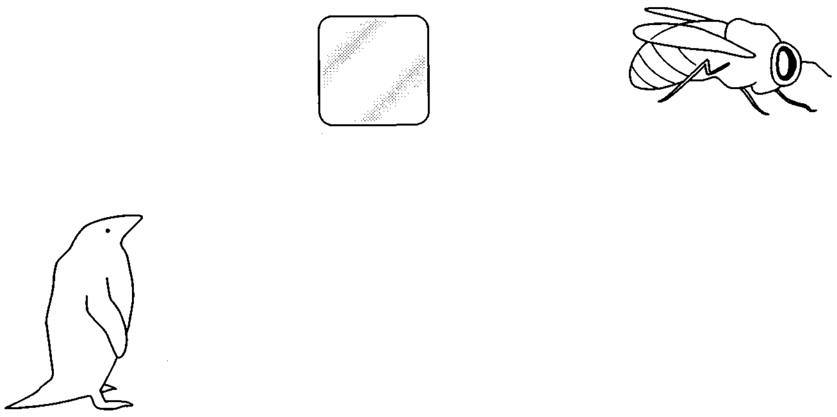


Figure 13.6. In this schematic situation from a Pengi game, the player will notice that the ice cube is aligned with the bee, move the penguin alongside the ice cube, and tell the penguin to kick it.

tion I will interpolate some of the code, simplified in various ways, that builds the circuitry responsible for these routines. I will not present all of the necessary code, because it is voluminous and very dense, so some of the terms mentioned in the code will go undefined or described only in English.

As the preceding section explained, Pengi has a convention that marker 0 tracks *the-penguin.* As the penguin moves across the screen, marker 0 moves along with it. Consequently, Pengi can now perform on marker 0 all the operators that are predicated on markers and thus find things out about the penguin. For example, it can perform visual operations to judge other objects' spatial relationships to the penguin and determine whether *the-penguin-is-moving.*

Another, more complicated convention is that marker 1 is, with specific rare exceptions, on *the-current-bee,* the bee that Pengi is either running away from or attacking. In this case Pengi will try to attack the bee. The circuitry must maintain the invariant that marker 1 stays on the current bee. If some other bee becomes more interesting, either more dangerous or more readily attackable, then somehow marker 1 has to get on that more interesting bee. Thus, marker 2 is constantly picking out moving objects and some circuitry is constantly checking if marker 2 marks a bee and if that bee is more interesting, by various criteria, than the bee under marker 1. (Dangerous attacking bees, for example, are more interesting than vulnerable bees that are running away.) If it is, Pengi moves marker 1 onto marker 2, unassigns marker 2, and then starts once again looking around at other bees with marker 2. As a consequence, Pengi will drop whatever it was doing to the preceding bee and now set about dealing with the new bee. Marker 2's scanning routine is always proceeding in the background. Whenever the arbitration circuitry lets it use the necessary operators, the circuitry implementing this routine tries to find the most interesting bee. These operations have a relatively low priority, but they are always trying to happen.

Here is the code that drops marker 2 (i.e., `moving-marker`) on moving objects near the penguin:

```
(set-inputs! index-moving-thing-near!
  tracking-marker moving-marker
  locus-marker penguin-marker
  radius (constant 200 'distance)
  doit? (andg *penguin?-result*
              (invert *index-moving-thing-near!-result*)))
```

Some care is required to make this operator run at the right times. At the very beginning of the game, it should wait until marker 0 has started to track the penguin. More important, it should run only every other cycle so that other operators can spend the odd cycles performing tests on the new object it has identified. As a result, it runs only on cycles when it is not returning a result.

Once Pengi finds a new moving object, here is the code that determines whether the moving object should be declared the currently most interesting bee:

```
(set-inputs! warp-marker!1
  marker1 bee-marker
  marker2 moving-marker
  doit? (andg *index-moving-thing-near!-result*
              *marks-bee?-result*
              (definitely *nearer?1-result*)))
```

Pengi has two warp-marker! operators because we did not have the patience to figure out how to arbitrate between their two uses. (This could be more principled. The other use involves identifying and avoid-

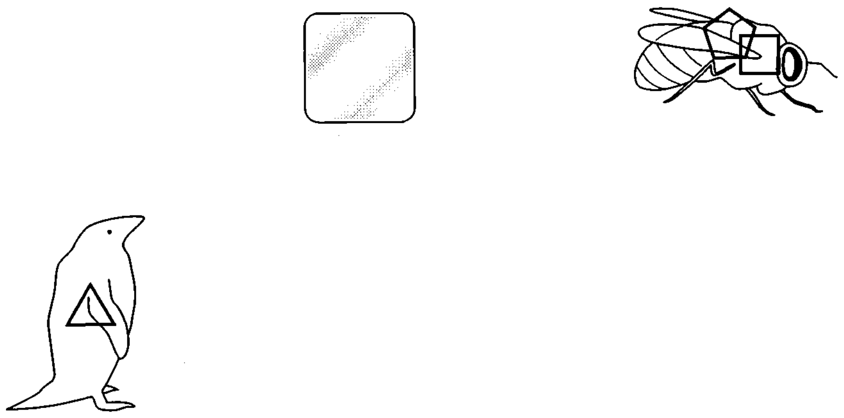


Figure 13.7. Pengi assigns marker 0 (triangle) to the penguin. It moves marker 2 (pentagon) among the nearby bees until it finds one that seems vulnerable or dangerous, whereupon it also moves marker 1 (square) onto that bee so it can focus on it.

ing ice cubes that have been kicked by bees.) This copy of warp-marker! moves marker 1 (i.e., `bee-marker`) to the object being tracked by marker 2 (i.e., `moving-marker`) provided that we have just picked out a moving object, the object is a bee, and the bee is closer to the penguin than was the preceding interesting bee. One might add many more conditions.

Figure 13.7 shows the initial scene with markers 0, 1, and 2 assigned. We draw markers with polygons. Triangle is marker 0, square is marker 1, and so forth.

Another operator picks out an ice cube near the penguin. Pengi applies this operator repeatedly, picking out all the nearby ice cubes and using some more circuitry to see if it might use the ice cube to attack the current bee. It does this by using freespace-between? to determine whether there is free space between marker 1 (the bee) and marker 3 (the ice cube). If so, and if various other conditions apply, the ice cube is a good candidate projectile. Next Pengi starts figuring out the directions among things. It uses direction-from twice, to find the axes that relate the bee and the ice cube and also to determine the direction of the penguin from the ice cube in terms of these axes. A fair amount of circuitry sorts among the various cases. For example, perhaps the penguin will have to back up and move around behind the ice cube and kick it. Perhaps it will already be lined up with the ice cube. Different circuitry covers different cases. This case at hand is relatively simple.

Here is the code that determines whether marker 3 has managed to land on an acceptable projectile:

```
(setq *direct-projectile-won?*
  (andg *freespace-between?-result*
        (invert *direct-moving?*)
        (invert (andg (possibly *nearer?2-result*)
                      *toward?-result*
                      (invert *aligned?1-result*)))))
```

The ice cube under marker 3 is an acceptable projectile under three conditions. First, there should be free space between it and the bee. Second, it should not be moving (i.e., from someone having kicked it). (Another bit of code sets the constant `*direct-moving?*` to the output wire of a circuit that determines whether the object under marker 3 is moving.) Third, it should not be the case that the bee is closer to the ice cube than the penguin and also headed toward it. The third condition is prudent lest the bee arrive at the ice cube first and kick it.

Figure 13.8 shows the same scene with the marker 3 assigned and the various directions mapped out.

The next step is to start moving. Pengi knows in what direction it needs to move now by performing a simple calculation (using logic circuitry) with the various encodings of directions and axes. In the diagram, the penguin must move upward so as to align with the projectile. Once aligned, it then moves rightward toward the projectile.

Here is the code that aligns the penguin with the projectile:

```
(action align-with-projectile go!
  direction (direction-in-other-dimension-from
              *target-to-projectile-direction-major*
              *penguin-to-projectile-direction-major*
              *penguin-to-projectile-direction-minor*)
  doit? *t*)
(condition align-with-projectile
           (invert *aligned?1-result*))
```

The global variables used in computing go!'s `direction` input are the buses that return the two components of the two direction-from operators. The call on the function `direction-in- . . .` expands into cir-

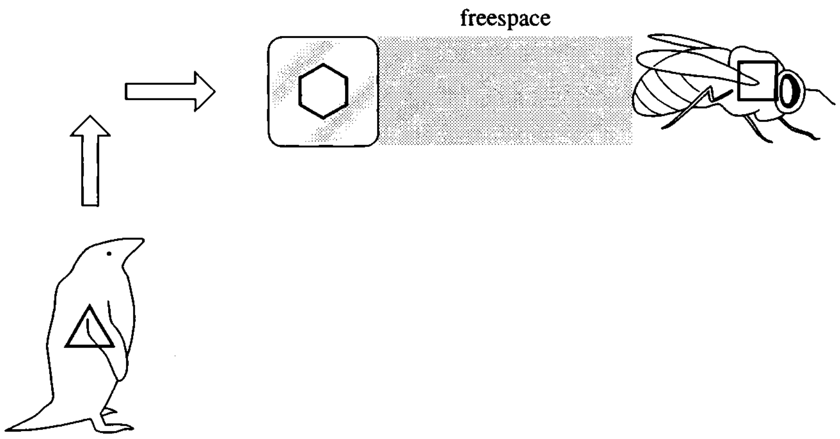


Figure 13.8. Pengi has now assigned marker 3 (hexagon) to the ice cube because it seems like a good projectile for attacking the bee.

cuitry for computing the appropriate direction. Since this action is conditionalized on the penguin not being aligned with the projectile, it will stop suggesting itself once the penguin is aligned. Once the penguin is aligned with the projectile, the following code will propose moving the penguin toward the projectile.

```
(action run-for-projectile go!
  direction *penguin-to-projectile-direction-major*
  doit? *t*)
(condition run-for-projectile *aligned?1-result*)
```

Instead of the condition on the run-for-projectile action, we could equally well have specified

```
(overrides-action align-with-projectile
                  run-for-projectile)
```

This would guarantee that Pengi would not move the penguin toward the projectile until it had finished aligning the penguin with the projectile. The two actions are incompatible because the penguin cannot be moved diagonally.

Finally, once the penguin is adjacent to the ice cube, it kicks it at the bee (Figure 13.9):

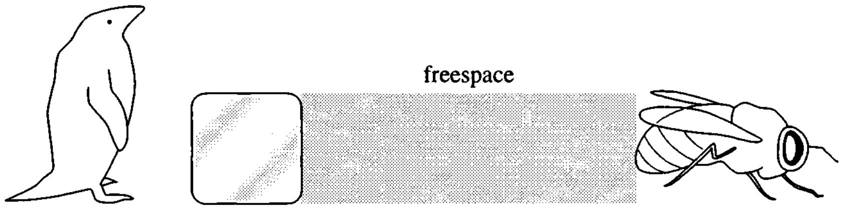


Figure 13.9. Pengi has aligned the penguin with the ice cube and is about to kick it at the bee.

```
(action kick-projectile kick! doit? *t*)
(condition kick-projectile *adjacent?-result*)
```

Remember that all of these operators have been running all the time. Pengi is constantly checking whether free space exists between the projectile and target, and that is a constant condition for the next move. It is also constantly checking whether the bee is aligned with the projectile, and that too is a constant condition for the next move. Consequently, if the bee moves out of range, Pengi will stop attacking the bee, though a different bit of circuitry could make a separate decision to proceed anyway in case the bee moves back into range. This more speculative tactic is not implemented, but a beginning player regularly does it and it would be easy to implement.

Pengi is also always checking, as much as it can, whether it has marker 1 on the most interesting bee. If some other bee is easier to attack or is too dangerous, it always wants to know that. That is why, as mentioned earlier, the routine whereby Pengi marches marker 2 among the various bees and compares them with the current bee is happening all the time as well. The penguin may take a definite path across the board to kick the ice cube, but the process by which it does so is not a four-step serial program. As a consequence, many different lines of reasoning want to make claims on the visual operators all the time. This is the origin of the problem of operator contention. Some of these routines can wait for a few cycles if necessary. Checking around for a dangerous bee, for example, cannot wait for long periods, but it *can* wait for a couple of cycles. The programmer must make many such judgments.

Although all of the visual operators can run in parallel, several constraints operate. First, Pengi must check all its conditions frequently enough. Second, Pengi must not try to do conflicting things with an operator on any given cycle (e.g., giving it two different values for the same argument). Third, Pengi must be able to keep track of the various claims on its operators despite its minimal state. In order to interpret its operators' results, it must also be able to reconstruct to some extent what it was doing on the preceding cycle. It does not, for example, have a queue into which it can put the different functions that it wants to assign to an operator. Nor would such a queue help, since the various operators making up a particular visual routine have to be coordinated closely. Pengi can reconstruct some of what it was doing by looking at the markers. A fourth constraint is that the programmer's task must be tractable. These considerations trade off in various ways.

Despite its flexibility, Pengi does have some inertia due to its visual focus. The visual markers' being on various objects focuses the system on those objects and thus to a certain extent preoccupies it. Sometimes the action moves so quickly that Pengi does not have time to move marker 2 among other interesting candidate bees, so that a bee can sneak up behind the penguin and sting it. That is how I often lose the game. Perception is necessarily selective, so that skill in the game lies largely in effective selection policies.

Seriality and focus

Pengi can have such a simple architecture because of what we learned about the dynamics of situated activity in general and Pengo-playing in particular. Central to this understanding is the fact that a situated agent, far from being a detached abstract intelligence, has a body. Having a body implies a great deal: facing in a direction, having a location, the tendency of the objects near us to be the significant ones, and the availability of various resources in the physical world. Pengi has a body only in a rudimentary sense, but its interaction with its video-game "world" does imply that, like any embodied agent, it must continually choose a single disposition for each of its parts (up or down but not both, here or there but not both). The necessity of such choices has pervasive implications for an agent's architecture. This section concentrates on two intertwined themes that figure in the dynamics of any embodied agent's interactions with its environment: seriality and focus. These themes appear on three different levels.

The first level is in the realm of dynamics. Simply having a body imposes a certain seriality on your activity. You can be in only one place at a time. You can be facing and looking in only one direction at a time. You can only do about one thing with your hands at a time. You can drink from only one cup at a time. If you have several things to do, you have to do them in some serial order.

The second level is that of the visual system. The visual system employs visual markers, in terms of which many of the visual operations are defined. Each visual marker can be located in only one place at a time, and each operator can be used for only one purpose at a time. More generally, one's visual system is focused on one thing at a time, in three senses: one's eyes are aimed in one general direction at a time, one actually focuses in a specific place, and the visual markers are allocated to certain locations and not others.

Finally, the themes of seriality and focus assert themselves on the level of representation. I can have only one *the-cup-I-am-drinking-from* at a time, and if I actually want to deal with several such objects then I must do so one at a time.

All of these kinds of seriality and focus are aligned. The representation scheme is less general than others in that it cannot represent arbitrary spaces of meaninglessly distinguished objects with identical significances all at once. But it does not need to, since an embodied agent can interact with only a small number of functionally distinguished objects at a time. In the end, this is the main way in which our understanding of dynamics has led us to simpler machinery. The machinery provides only the capacities we really need. In doing so, it takes advantage of the dynamics of interaction in relatively benign worlds for its power rather than providing cumbersome machinery motivated by a spuriously conceived generality.

Despite these lessons, again, Pengi has a body in only a rudimentary sense. It stands transfixed in front of its video screen without moving among the materials and equipment of its task. Nothing is ever behind it. It interacts continually with the world but it is not truly *in* the world. More genuinely autonomous embodied agents will require a deeper understanding of the dynamics of embodiment.

It would be instructive to try applying Pengi's technology in a different domain. Doing so will presumably reveal that we made many mistaken decisions about this architecture's underconstrained features.

Some of our mistakes will have resulted from the exigencies of having to make it programmable at all, given our rough understanding of the detailed dynamics of this particular activity. Other mistakes will be due to the biases inherent in this domain. A Pengo board is a visually busy place. Since the bees have an uncooperative random element, a Pengo player must keep a finger on an abnormally large number of details of its environment; this is why Pengi relies more heavily on markers for its causal relationships than is the case in real life. Indeed, this aspect of the domain tends to obscure the distinction between deictic and objective representation, since visual markers are much more similar to variables than most other means of maintaining causal relationships to entities. When an agent's interactions with the world have a more reliable structure than Pengi's relationship to the Pengo game, it becomes possible to focus more tightly on particular objects and tasks. Pengi, by contrast, has a fairly diffuse kind of focus that conflicts with the premises of its architecture.

Questions and answers

Pengi reliably provokes a long list of questions. Though I have tried to address most of these while developing the argument, several are best dealt with separately. I have tried to word most of them in exactly the way that AI people tend to ask them.

You say that Pengi isn't a planner. But how can it be a serious model of any kind of activity without anticipating the future?

A system can anticipate the future without constructing plans. All that is required is that one's actions take ideas about the future into account. In a planner, this takes two forms: the decomposition of goals into subgoals provided by the programmer and the simulation of possible future courses of events that the planner performs. Pengi also anticipates the future, in two senses. First, Pengi's tactics were designed using an understanding of the dynamics of the game. This understanding involves ideas about what tends to happen next and about how individual actions lead to winning the game. Second, Pengi uses its visual system to visualize possible courses of events. Pengi does not have a general facility for simulating the future, because it suffices to visualize certain patterns.

> *You say that Pengi does not use world models. But couldn't we view its retinal image of the Pengo screen as a world model and its visualizing as simulation?*

The distinction between performing a simulation and running visual routines is indeed not as clear-cut as it appears. One might choose to view the visual operations as performing some sort of logical deduction. Visual routines certainly do not constitute a general simulation or theorem-proving facility, but inference need not imply a general inference facility. The representations over which the visual operators apply are retinocentric and thus not objective world models. A retinal image is a poor world model because it encodes only the most primitive information about the world. It does not encode the identities or structural relations of the visible materials. Furthermore, a retinal image is not a world model because its relation to outside events depends on the agent's location and orientation. Pengi, it is true, always has the same location and orientation, but other systems employing the same technology need not.

> *You emphasize the way Pengi's entities support a sense of focus. Couldn't you implement focus in a much more straightforward way, perhaps by maintaining only a model of what's happening within a certain radius from the penguin?*

That wouldn't do, since a bee can kick an ice cube from an arbitrary distance. It would also be a domain-specific solution that would not be helpful in domains where it would be too complicated to model all the materials within any reasonable radius of any given point.

> *Pengi has no state in its central system. How could intelligence be possible without memory?*

People certainly do remember things. Pengi is not supposed to be a model of all human activity. Pengi does not have any state in its central system purely as a matter of machinery parsimony. It turns out that the dynamics of Pengo-playing are such that no central system state is necessary.

> *Pengo is such a specialized domain. Why should we consider that Pengi has taught us anything about everyday life as a whole?*

Pengi's ability to play Pengo certainly proves nothing about activities such as making breakfast. I do not claim to demonstrate any propositions about everyday life by generalizing from Pengi. Instead, Pengi is an illustration of some things we learned by moving back and forth between observation and technical exercises. We chose the Pengo domain because we were unable to build a breakfast simulator that did justice to the reality of breakfast-making, because we feel we understand many of the dynamics of Pengo-playing, and because the central dynamic themes of improvisation, contingency, and goal-directedness all arise in Pengo in natural ways. Pengo's only major drawback as a first experimental domain is its adversarial nature.

It seems as though playing Pengo involves reacting to a series of crises. Combinational logic is very good at this, but what about anything else? Haven't you tailored the Pengo domain to show off the strengths of your architecture and hide its weaknesses?

Playing Pengo isn't like that at all. Pengo players who do not use complex tactics and anticipate the future will indeed find themselves reacting to a series of crises. Beginners tend to alternate between periods when they attack and periods when they are in trouble and have to focus on surviving. More advanced players must often defend themselves, but the line between attack and defense is more blurred. Pengi itself is a relative beginner, but it spends at least as much time on the attack as it does dealing with crises. In particular, Pengi constantly uses its visualization abilities to see if it can use its various tactics.

You insist that the world is fundamentally a benign place, yet your domain is violent. If the efficacy of improvisation depends on the beneficence of the world, why should we believe in your analysis of why Pengi can play Pengo?

The adversarial nature of Pengo is unfortunate, but we can be precise about the way in which it disrupts our efforts to build Pengi. The principal difficulty is that Pengi, like any human player of Pengo, must continually work to keep track of the bees because of their random motion. A Pengo player cannot take advantage of many of the dynamics by which one can keep track of things in other domains, such as keeping them in a special place, carrying them in a pocket or purse, or going to find them when a need for them arises. As a result, playing Pengo places much heavier demands on one's visual system than most normal activities. The most difficult thing about writing Pengi was to arbitrate among the visual operators that Pengi uses to keep track of all the bees and their geometric relations to the penguin. That the only serious difficulty in writing Pengi corresponds so closely to the principal difficulty in playing Pengo is at least mildly encouraging.

At the same time, the video-game metaphor of killing-or-being-killed should not distract us from the benign properties that the Pengo domain does share with many routine activities in the everyday world. The materials of the player's immediate activity are readily visible. The vast majority of its objects – the ice cubes – do not move about capriciously. The domain has a strong sense of locality, since objects do not move quickly compared with the dimensions of the game board. And the player does not encounter wholly unfamiliar types of objects.

Furthermore, it is important to distinguish Pengo from activities that involve genuine violence. One simple distinction is that in Pengo it is the penguin that dies, not the player. But there is a deeper distinction as well. The world of Pengo works in a definite, unvarying way. The bees are always driven by the same simple random process. Pengo would be much harder if the bees were smarter. The bees could, for example, observe patterns in the player's actions and devise strategies to take advantage of them. If Pengo's bees changed their flight patterns, Pengi would certainly compensate to a limited extent, but if the changes were substantial, Pengi would not be able to evolve the necessary patterns of interaction. If someone is genuinely trying to kill you – if you are at war or being hunted – the way you represent the world is your Achilles' heel. All the unarticulated assumptions behind your reasoning and all the unreflective patterns in your actions create opportunities to catch you off guard. For example, as weapons systems and their logistical support grow more complex, and as the components that must work together grow more numerous, the opportunities for subversion and sabotage multiply. Little of this is at issue in Pengo, or in the routine activity of everyday life.

The lesson is that part of the benign nature of both Pengo and the everyday world is that, far from working to subvert your understanding of them, both help ensure that the representations you have developed will continue to work well enough. Pengo offers this assurance through its simple lack of change. In the everyday world, though, the story is more complicated. The unvarying nature of physical laws and the tendency of physical objects to stay where you have put them certainly help your representations to keep on working. Beyond this, considerable effort goes into keeping the physical apparatus of everyday activities tidy and in working order. But above all, the order of the everyday world is a social order that is actively maintained through the intricately organized efforts of all its participants. Far from subverting the common reality, everyone makes everyone else participate in the common task of maintaining it (Heritage 1984). The vast majority of this work is invisible to casual inspection, but careful investigation or experiments in deliberate disruption can easily reveal it. Every element of the everyday world retains its significance and its salient properties through this continual cooperative effort. Making computational sense of this process, however, remains a project for future work.

Does Pengi understand what it is doing? After all, you built its circuitry yourselves.

Pengi does not understand what it is doing. No computer has ever understood to any significant degree what it was doing. We built Pengi's circuitry ourselves; the authors of a planner provide it with the possible decompositions of its possible goals and with functions for computing the consequences of its actions. What grounds could we have for ascribing some measure of understanding to a conventional planner? One ground might be the amount of relevant material made explicit by the planner's representations. Yet the procedures needed to manipulate these representations become computationally intractable as the representations themselves become more accurate. It is too early to settle the question.

14 Conclusion

Discourse and practice

My argument throughout has turned on an analysis of certain metaphors underlying AI research. This perspective, while limited, provides one set of tools for a critical technical practice. I hope to have conveyed a concrete sense of the role of critical self-awareness in technical work: not just as a separate activity of scholars and critics, but also as an integral part of a technical practitioner's everyday work. By attending to the metaphors of a field, I have argued, it becomes possible to make greater sense of the practical logic of technical work. Metaphors are not misleading or illogical; they are simply part of life. What misleads, rather, is the misunderstanding of the role of metaphor in technical practice. Any practice that loses track of the figurative nature of its language loses consciousness of itself. As a consequence, it becomes incapable of performing the feats of self-diagnosis that become necessary as old ideas reach their limits and call out for new ones to take their place. No finite procedure can make this cycle of diagnosis and revision wholly routine, but articulated theories of discourses and practices can certainly help us to avoid some of the more straightforward impasses.

Perhaps "theories" is not the right word, though, since the effective instrument of critical work is not abstract theorization; rather it is the practitioner's own cultivated awareness of language and ways it is used. The analysis of mentalism, for example, has demonstrated how a generative metaphor can distribute itself across the whole of a discourse. The inside–outside metaphors of mentalism do not appear simply as buzzwords that can be noticed, reflected upon, and altered in isolation. The preceding chapters' studies have turned up numerous obscure and tightly interconnected manifestations of these metaphors. Some examples include

- the abstraction–implementation couple,
- the standard theoretical definition of a computational problem,
- the vocabulary of input and output,
- the distinction between planning and execution,
- the notion of thought as simulated action and action as recapitulated thought,
- the notions of world models and expressiveness,
- the idea of perception as mental model-building, and
- the theory of meaning as correspondence between knowledge-inside and reality-outside.

These mentalist ideas reinforce one another, so that – through a homeostasis of practical logic – it is hard to challenge one of them without challenging all the rest. Someone who tried to stop viewing reasoning as mental simulation without also abandoning the planning–execution distinction, for example, would be forced to formulate an alternative conception of planning that would nonetheless serve the same function. While such a conception is certainly possible in principle, it is more likely that an impasse would quickly arise as the practical logic of conventional planning research proceeded to reassert itself. Or, to take another example, someone who tried to undermine planning research by proving negative complexity results about planning problems would discover that the theorems had little effect, since the whole methodological framework of "problems" (as functions from single inputs to single outputs) does not fit the situation of an agent that alternates periodically between planning and execution. Attempts to reinvent the field within new metaphors will constantly encounter additional ways in which the earlier metaphors have gone unquestioned; putatively novel technical proposals – such as my own – will constantly turn out to combine new and old in complicated and problematic mixtures. Further historical analysis could pursue the lines of mutual reinforcement even more widely: back and forth between hardware and software, between ideas and artifacts, between science and engineering, between business and government, between popular culture and technical communities, and between the proponents and the critics of all of these things. Across this whole landscape, the analysis of generative metaphors serves as a powerful critical tool because it allows us to call entire networks of ideas into question at once by tracing the threads that unify them.

Let us review how the analysis of metaphor has helped in sketching some alternatives to mentalism. The first step, not so clear in the linear exposition here, was the long process of unraveling the metaphor system of inside and outside and discovering its pervasive effects. This process was not simple, but it would have been much simpler if suitable critical tools had been readily available in the milieus of technical work. These tools include the basic idea of generative metaphors and vocabulary of centers and margins, as well as the broader substantive critique of Cartesianism. Once a generative metaphor has been detected, the next step is to lay out its structure: the surface metaphors that obscure it, which phenomena it makes central and which it makes peripheral, how its margins become manifest, what happens when those margins are interpreted as new technical problems within the existing metaphor framework, and so forth. Next, to make clear what exactly is at stake in accepting or rejecting a given generative metaphor, one employs the strategy of reversal – founding a new, competing technical enterprise, such as interactionism, that reverses the original hierarchical relations of center and periphery. This is difficult because it is hard to become aware of the full range of ideas needing replacement. It is also difficult, obviously, because it is hard to replace them all; one will seek guidance, for example, from alien disciplines and from unsung precedents in old journals. Finally, then, one insists on the displacement of both the original discourse and its reversal. Rather than adopt the reversed theory as a new orthodoxy replacing the old, one attempts to cultivate a more sophisticated view of the role of language in technical practice. None of this will happen quickly. But even a sketchy start can throw a great deal of light on the existing practices, revealing the contingent nature of many things formerly taken for granted.

One promising aspect of this method is its capacity to predict the progress of actual technical work – in other words, to use an analysis of a discourse to anticipate certain aspects of the logic of a practice. These predictions state that a technical practice will get itself into stereotyped sorts of trouble as it encounters its margins. The practice will not simply fail, or blow up, or hit a logical wall. Indeed, the trouble may be invisible to the unaided eye: it may seem like a routine cycle of problems and solutions and new problems and new solutions. Considered uncritically, such a cycle might constitute progress or it might constitute a treadmill.

Technical fields do not die through clear-cut empirical disproof of their premises; the margins of practices lead not to sharp limitations but to endlessly ramifying spaces of technical trade-offs. At any given moment, the technical situation will be terribly complicated. Yet a great deal becomes clear once the whole space of possibilities revealed by research is mapped out through analysis of the discourses within which this research is conducted.

Of course, any such clarity is best obtained in retrospect. A critical analysis certainly does not replace practical work; nor should it be viewed as a kibbitzer invalidating this work from a comfortable seat in back. To the contrary, critical analysis is what allows the activity of research to make sense as an orderly conversation with its own practical reality. And by allowing the patterns of that conversation to become clear as soon as possible, critical analysis can prevent a great deal of futile effort. Indeed, this conception of critical insight suggests a new way of evaluating technical research: technical work is "good" when it facilitates critical reflection. Solving technical problems is a useful activity, but as an intellectual matter it is much better to get oneself into a really revealing impasse – and then to work through the internal logic of that impasse toward some fresh understanding of the research process itself. Thus, there can be bad interactionist work – when "interaction" degenerates into an indiscriminately applied stock of techniques – and good mentalist work – when, as with the work of Minsky and Newell, the internal logic of mentalism appears to rise up and compel research into genuinely complex engagements with the practical reality of computational work. Some impasses are easily resolved, of course, and reasonable people can disagree about how deep a rethinking of received ideas a given impasse requires. But the matter is not arbitrary, nor wholly a relativistic matter of opinion.

Internal analyses of metaphor systems are valuable, but they cease providing guidance as soon as it is time to choose among them. Mentalism and interactionism both have metaphors, margins, and logics. Indeed, as discourses go they are neighbors, organized by complementary sets of topological figures; what is central for one is peripheral for the other and vice versa, very neatly. It is too early to make firm decisions: arguments for each option must appeal to practical experience, and the research community has not yet accumulated enough practical experience with either option. One crucial consideration is: what understanding does each type of technical practice offer of the matters that the other type makes central? That is, what accounts can mentalist AI offer of complex interactions with the world, and what accounts can interactionist AI offer of the prototypical cases of internal mentation? I have already considered the first question – mentalist AI's accounts of interaction – at length and found some deep and serious problems. But the comparison is not fair until we can pursue the opposite inquiry: investigating the practical logic of interactionist computational research on the phenomena most closely associated with "the mind."

This inquiry has not even begun. It begins with this question: within an interactionist view of human life, in what sense(s) do people have "insides"? Obviously, any interactionist computational practice that we can currently imagine will retain the minimal, physical sense of the term: bodies and skulls have spatial interiors and exteriors. And, just as obviously, interactionism refuses to convert that minimal sense of the term into a full-fledged generative metaphor. What interactionism does suggest is that having-an-inside really is troublesome. The basic analytical categories will be defined in terms of interactions, but certain concepts of insideness might make sense against that background. Interactionism, in other words, does not say that nobody "has" anything one might wish to speak of with metaphors of "inside." It only suggests that insideness is likely to be derivative, problematic, multiple, variable, constructed, and contingent (Winnicott 1975a [1949], 1975b [1951]). That is, in the dynamics of human life are several distinct phenomena that one might want to call an inside:

- You might have an innermost self that you share only with your innermost circle of friends.
- You might describe your thoughts as occurring inside yourself, in the sense that you experience them as inside your consciousness, while still being unaware of the unconscious thoughts.
- You might hold your feelings in.
- You might have something in mind in pursuing a certain line of reasoning in a conversation.
- You might have someone in particular "in your life."

These are cultural idioms and not technical concepts. They all name perfectly real phenomena, but these phenomena must be understood within the fullness of someone's life – interactions, involvements, and relationships – and not (at least a priori) as specifying an anatomy or a cognitive architecture. In particular, they must all be understood as different varieties of "in"; each "inside" applies to a different useful way of mapping a metaphorical topology of the self.

Perhaps the most unfortunate *cultural* consequence of mentalism is its tendency to collapse all of the disparate inside phenomena into a single "mind" vaguely coextensive with the brain. At the root of this impoverishment of psychological discourse is the practice, found in Descartes and subsequently intensified by the discourse of mentalistic computationalism, of conflating a rough phenomenological description – that of inner thought – with a specification of mechanisms – processes inside the brain. In undoing this conflation, it becomes possible to explore other metaphors and other ways of relating phenomenological and mechanistic theorizing. Chapter 1 began by complaining about this confluence of technology and experience; and it is here, in this slightly more sophisticated reflexive understanding of metaphor and its role in technical work, that some alternative becomes conceivable.

Converging computational work

Several interactionist AI research projects have emerged over the past decade. These projects grow from the history of computational ideas in their own distinctive ways, and the authors appeal to a variety of research values in presenting the rationales for their work. These rationales can be usefully sorted under three headings: layered architectures, interlinked automata, and machinery parsimony.

Layered architectures

Much AI research on interaction with the physical world has been motivated by the desire to build *autonomous agents* (Maes 1990) or *artificial life* (Steels 1993; Varela and Bourgine 1992), or to use computational modeling to study biological phenomena (Meyer and Wilson 1991). An emphasis on fixed or nearly fixed circuit structures is common in this literature. Some authors attempt to replicate the properties of human or animal nervous systems, but others share a more theoretical concern with physical realization.

Brooks (1986, 1989) has explored the design of "nervous systems" for insect-like robots based on these principles. Arbib (1989) has likewise explored the *schemata* comprising the nervous systems of frogs through anatomical study and simulation. Their ideas about circuit architecture share an idea that Brooks calls *subsumption:* the circuit is divided into behaviorally specified layers, with each layer of circuitry modifying the function of the otherwise self-sufficient package of layers below it. In Brooks's robots, for example, the lowest layer of circuitry prevents the robot from running into obstacles; the next layer causes the robot to wander around, interrupted by periodic avoidance behaviors caused by the first layer. By analogy, Arbib and Liaw (1995) consider a toad's neural circuits for snapping at prey and avoiding predators; they argue that the lowest layer of circuitry will snap at anything that moves, and that the next layer overrides this reflex and substitutes avoidance behavior if the moving object is too large.

These models envision a specific relationship between machinery and dynamics. The agent's interaction with its environment is understood as a collection of discrete *behaviors* each of which is subserved by a module of machinery. Some of the behaviors subsume others, and so do the corresponding modules. This scheme is motivated by an evolutionary story: higher functions developed as additions to, and modifications of, lower ones. The research strategy is to work upward through phylogenetic history, understanding the more basic structures of brain and behavior before investigating the more advanced structures that are built on top of them (Kirsh 1991).

Interlinked automata

Several authors have tried to formalize in mathematical terms the interactions between agents and their environments. The most common approach is to interpret the agent and environment as interlinked *automata.* The theory of automata is part of the theory of computation. A given automata-theoretic formalism will provide mathematical tools for describing a class of idealized machinery: the unitary components, how these components can be assembled, and the rules that govern how a given assemblage of components evolves over time. Perhaps the most familiar of these formalisms is Conway's Game of Life, in which the components are simple binary elements (glossed as "living" or "dead") arranged on an infinite, discrete two-dimensional grid. Any given arrangement of "living" squares on this grid constitutes a particular automaton. As with most such formalisms, these games evolve step by step according to a discrete clock; simple rules determine the arrangement of living squares on step $n + 1$ given their arrangement on step n. Turing machines are also formal automata in the same way.

When an agent and its environment are interpreted as interconnected parts of one large automaton, the formalism's evolutionary rules will determine the dynamics of the interaction between them. Lyons and Arbib (1989) formalized these automata as hierarchical networks of concurrent processes; these networks implement the schemata just mentioned. Such automata might be used in a variety of ways. They might be a theorist's tool for designing and predicting the dynamics of a given agent and environment. Or representations of them might be supplied as data structures to a computer program; Lyons and Hendriks (1995) have explored how such a program might automatically identify and represent the dynamics of the agent and environment's interactions. They have applied this method to the analysis of a number of industrial robots.

Beer (1990, 1995) has taken another approach, characterizing agent–environment interactions within the formalism of dynamical systems theory. Though not commonly understood as a branch of automata theory, dynamical systems theory expresses the evolution of a formally specified system. Like Lyons and Arbib, Beer treats the agent and environment as portions of a larger system that comprises them both. Beer's formalism is continuous, so that his systems' rules of evolution are specified by differential equations. He has applied his formalism to a robotic insect whose legs are driven by simple neural networks, using genetic algorithms (Goldberg 1989; Holland 1975) to assign weights automatically to these networks by simulating various possibilities. The resulting robot insects exhibit stable, biologically realistic gaits, and Beer wishes to explain how and why. This threatens to be difficult, given that the numerical weights in the neural network, plotted in any simple way, are not obviously meaningful. The tools of dynamical systems theory, however, permit the robot's behavior to be usefully plotted as a trajectory through a mathematical space.

Rosenschein and Kaelbling (1986, 1995) take yet another approach. Their design methodology for *situated automata* is based on a mathematical logic formalism that describes the informational relationships between an agent and its environment. That is, a given element of the agent's circuitry might be held to "carry the information that" such-and-such state of affairs holds in the world: for example, that a certain door is open, a certain light is on, or a subgoal has been achieved. Since they employ the methods of model theory, they understand this "carrying" as a correspondence relationship, and they normally specify their examples in terms of objective individuals; these are named by logical constant symbols such as `robot1` and `ball37` (1995: 156). In practice, due to the demands of embodiment, they more closely resemble deictic entities (1995: 166). A designer builds a situated automaton by using a logical formalism – or a convenient specification language with logical semantics – to describe the desired goals, perceptions, actions, and inferences. A compiler similar to the Pengi designer's is used to translate these specifications into working circuitry. This circuitry is more general than Pengi's since it includes a general facility for building latches.

Machinery parsimony

A third strategy in interactionist AI research has been defined in methodological terms through the principle of machinery parsimony introduced in Chapter 3. Recall that this principle suggests that careful attention to dynamics will lead to simplifications in the design of machinery for agents that participate in those dynamics. The requisite understandings of dynamics might take many forms; some might be mathematical and others might be intuitive. In practice these understandings will typically be driven by the design of architectures. Given a partially designed architecture, a choice about how to proceed with the design can often be translated into questions about dynamics: Does this happen? Does that happen? Is this kind of information typically available? And so on.

One example of this reasoning is found in Agre and Horswill's (1992) work on a class of tasks heavily abstracted from cooking. Cooking is normally organized as a sequence of conceptually discrete steps: mixing, pouring, stirring, spreading, picking up, putting down, turning on, turning off, and so forth. Particularly with a familiar set of recipes, the question is always, which step is next? Answering this question does not normally require extensive formal analysis, because so many of the steps are either wholly independent of one another (you can beat eggs and pour cereal in either order), or else simply impossible until certain other steps have been performed (you cannot beat eggs until you break them). As a result, one can go a long way with the simple policy of repeatedly finding something that needs doing and can be done, and doing it. Applying this policy, of course, presupposes that one knows (at least) what steps must be done eventually, what enables a step to be taken, how to take the individual steps, and how to recognize that the task has been completed. The technical question is, does it require any more than that? Horswill and I analyzed this question by adapting the classical AI framework for plan construction through search. Strong constraints can be imposed on the structure of the search space by analyzing the behavior of the objects, particularly utensils, actually found in kitchens. As a formal matter, it is necessary to construct a schedule of activities only when one risks running out of stove burners or other such tools that must be committed to a fixed purpose for an extended period.

Another example is found in Hammond, Converse, and Grass's (1995) analysis of *stabilization* – the actions by which someone maintains a stable relationship to his or her surroundings. They became interested in stabilization through their development of cognitive architectures based on the storage and retrieval of "cases" in the construction and execution of plans (Hammond 1989). Case-based architectures, like dependency-based architectures, work best when the same cases tend to recur frequently; they work ideally when life is orderly enough that a manageably finite repertoire of cases covers the vast majority of the situations that tend to come up. Agents can help to minimize the diversity of situations they encounter by actively adjusting the world to normalized forms. A simple example is putting one's tools away in conventional places at the end of a task; Hammond et al. call this "stability of location." Their taxonomy of stabilization tactics also includes stability of schedule (driving to work the same time each day), stability of resource availability (keeping kitchen staples in stock so that they do not run out), and stability of cues (leaving objects that need work in locations where they will be noticed), among others. Their program FIXPOINT implements some of these tactics in the context of the task of building toy boats.

Taken together, these projects demonstrate a trend in interactionist AI research that Rosenschein and I have described as *using principled characterizations of interactions between agents and their environments to guide explanation and design* (Agre 1995a). Subsequent research can develop this theme in numerous directions. The methodological key is to maneuver between design of machinery and investigation of dynamics, searching for dynamic phenomena that constrain or support particular candidate designs.

Some next steps

The preceding section has placed this book in the context of a broader movement toward post-Cartesian computational conceptions of activity. The breadth and creativity of these projects justify a cautious optimism about the prospects for future research. But the situation is complicated. As I have continually emphasized, a critical technical practice always finds itself in the paradoxical position of digging the ground out from under its own feet. It would be all too easy to neglect this imperative and allow "post-Cartesian computational theory" and the like to become the cant of a new establishment. It is thus essential to define precisely the horizon of issues that this book defines through its limitations and silences.

First of all, the metaphor system built up around the notions of "inside" and "outside" does not exhaust the discourse of contemporary computational discourse. Further metaphors that invite analysis along similar lines, some of them already mentioned in passing, include the following:

- Information as a commodity (Agre 1995b; Buckland 1991; Schiller 1988). One has a certain amount of information; one requires sufficient information to make a given decision; information is obtained through computational expenditure (leading, for example, to a trade-off between perceiving the information and remembering it); stored information can depreciate over time.
- Society as a network. Individuals as nodes in the network; relationships as "connections" (i.e., wires); communication as signals that pass over these connections; the normative stasis of connectivity patterns in this network.
- Formalization as hygiene. "Neat" versus "scruffy" research programs; formal systems as "clean" in virtue of the alleged precision of their language; the process of formalization as a "rigorous" production of this cleanliness.

- Computation as power. The practice of comparing computers along a linear scale according to their capacity to perform computational work per unit time; the notion that computational work can be measured and thereby understood as a commodity whose expenditure is to be minimized.

In each case, the point of analyzing these metaphors is not to refute or debunk them. The point, rather, is to displace them – that is, to understand that they are metaphors. They might still be useful, but they are not the transparent reality of things. Their analysis will be far from routine; all of them are implicated in a larger network of discursive devices and practical arrangements. Nothing is simple about this; the metaphors are neither simply good nor simply bad.

Moreover, the analysis of generative metaphors is limited as a critical tool. An explicit awareness of one's metaphors will certainly be part of any critical practice. But sorting ideas by their generative metaphors is a terribly coarse procedure. In the present case such measures are surely necessary. But the next stage of analysis, within the broad mess of theoretical positions that I have called interactionism, will require more refined tools. Many of these are simply the tools of serious scholarship: clear thinking, close reading, interdisciplinary dialogue, and sustained engagement with archival and ethnographic materials. These tools may be applied to reflect on attempts to get things working, as I have done with RA and Pengi; they may also be applied to historical reconstructions of the projects that established technical traditions, as I have done with computational research on planning and representation.

Furthermore this study, like most attempts at syncretic inquiry, runs afoul of the standards of all the fields it tries to cross. Computer scientists and philosophers alike are doubtless appalled at one aspect or another of my methods and exposition. Computer people, for their part, tend to have definite ideas about reporting technical work: one documents the program thoroughly, states clearly what problem it is solving, keeps the rhetoric to a minimum, reckons the relevant literature in terms of technical similarities to other people's programs, and offers no views that are not justified by the "work ethic" of running code. Chapter 1 has already attempted to sift the legitimate from the overly inflexible in these views, but unfortunately it is not yet clear what a fully revised technical practice would be like.

Philosophers for their part will be justly upset at the rapid use I have made of some large and difficult ideas. To take one central example, I have invoked the name of Heidegger in defense of a computer program that is, contrary to Heidegger's conception of human being, in no sense a social creature. Yet Heidegger's own procedures provide the material for a reasonable reply: the renovation of technical work is necessarily a hermeneutic cycle in which each stage is a necessary preparation for the next. In taking a few steps along the way, I can be happy enough to have revealed a horizon for subsequent research. Each hermeneutic step will involve questioning everything that has gone before, without at the same time declaring the earlier work to have been a waste of time. My appeal, in short, is not to the finality of my work but to its utility as a beginning.

Nonetheless, the missing theme of sociality diagnoses the central substantive difficulty with my theory: mentalism and interactionism seem unified by an overly individualistic conception of agents in environments. Getting beyond this limitation will entail recovering the history of computational conceptions of individuality. It is perhaps a fluke of technical history that the stored-program digital computer arose far in advance of large-scale digital telecommunications, so that one could enumerate computational individuals in the same way as human individuals. With the growth of large-scale networking, though, conceptions of computational individuality are shifting rapidly toward the new topological metaphor sometimes called "cyberspace." This development deserves critical attention; it lends itself equally to the political idiom of "empowerment" and to control regimes of unprecedented scope. Far from encouraging the values of collective action, this worldview would dissolve all individualities into a boundless res cogitans. The margins of such a picture lie in the human body, in the physical realization of all computation, and in the boundaries between the res cogitans of bureaucratic rationality and the res extensa of the human world. These margins will surely become the sites of deconstructive inquiry and material contest, and the sooner the better.

Critical attention should also be devoted to the status of design arguments in computational inquiry. Early on, I insisted that science and engineering be understood as distinct activities with different claims on technical work. But the fact remains that the most powerful conceptual tool of computational work, in both psychology and engineering, starts from a sense of good design. In this sense, critical AI would benefit from historical analysis of the discourses and practices of engineering (Layton 1971; Noble 1977). How has the social organization of engineering influenced the metaphors and values of engineering practice? And how have the scientific and engineering versions of computational research interacted in the past? How have the frequent ambiguities between the two activities affected each of them?

Further directions of research are more strictly philosophical. How has the history of ideas functioned in technical work? The question is all the more interesting given that computationalists have rarely thought of themselves as engaged in an intellectual enterprise: "ideas" has long meant "techniques." Yet a book like Miller, Galanter, and Pribram's *Plans and the Structure of Behavior* can have an underground life, prefiguring whole generations of research without being extensively cited within the resulting literature. How does this work?

As the social consequences of computational machinery grow more extensive, the reciprocal influence of ideas about machines and people will presumably grow as well. Ideas about machines and people have, of course, influenced one another for a long time. But the situation is different now. When automata were curiosities, and later when computers were confined to air-conditioned rooms, one could speak of a metaphorical relation between two vaguely similar but basically independent entities. The early cognitivists, having been inspired by the new inventions of military laboratories, could return to their laboratories and invent a new psychology. This psychology's metaphors of "inside" and "outside" were a modern variation on some old themes. During the Cold War, the vocabulary of inside and outside spoke of existential loneliness in a world gone mad. Now that computational ideas and computational machinery are becoming ubiquitous, so that people find themselves growing "interfaces" to complex devices whose spirit often seems alien, it is urgent that we attend to the forms of human experience that these devices embody and help reproduce. Once we become aware of such things, they are no longer inevitable. In choosing our conceptions of computing machinery, we choose, to some small degree, the world in which we live.

Notes

1. Introduction

1. See, e.g., Miller's interview with Jonathan Miller (1983: 24), and Edwards (1996: 180–187).
2. Truesdell (1984) refers to these packages of metaphor and mathematics as *floating models* and suggests that computers encourage their use. But whereas he regards them as a mark of poor science, I will simply argue for a conscious awareness of their properties. On the ascription of intentional vocabulary see Churchland (1989), Coulter (1983), Dennett (1981), and Woolgar (1985).
3. For example, Descartes's near contempt for formal logic did not prevent McCarthy (1968 [1958]) from founding a school of AI research that translates all questions into problems of deduction in the predicate calculus.
4. Neisser (1966: 73–77) unkindly but intriguingly points out that this movement began in a period when Madison Avenue and the Cold War brought a mass-cultural preoccupation with brainwashing and other forms of human "programming."
5. For general histories of AI, see Boden (1977), Crevier (1993), Fleck (1987), Franklin (1995), H. Gardner (1985), and McCorduck (1979). For general philosophical discussion see Boden (1990); Copeland (1993); Ford, Glymour, and Hayes (1995); Haugeland (1981, 1985); and Putnam (1975b [1960]). Newell (1983) outlines the major intellectual issues that have organized debate within the field. Critical analyses of AI include Athanasiou (1985), Berman (1989), Bloomfield (1987a), Coulter (1983), Dreyfus (1972), Forsythe (1993a), K. Gardner (1991), Gilbert and Heath (1985), Lakoff (1987: 338–352), Leith (1986, 1987), Lighthill (1973), Roszak (1986), C. Taylor (1985), Winograd and Flores (1986), and Woolgar (1985, 1987).
6. It also permits the use of neuter pronouns.
7. On Heidegger (1961 [1927]) see Dreyfus (1991) and Preston (1988). On Garfinkel (1984 [1967]) see Heritage (1984) and Suchman (1987). On Vygotsky (1978 [1934]) see Leont'ev (1981) and Wertsch (1985). Leont'ev

in particular is responsible for an influential elaboration of Vygotskian psychology known as *activity theory* (Engeström 1987; Nardi 1996). My use of the term "activity," however, is less specific than Leont'ev's, in large part because I do not know how to realize many of Leont'ev's more advanced categories in computational terms.

8. For Miller, Galanter, and Pribram's influence see Forbes and Greenberg (1982); Friedman, Scholnick, and Cocking (1987); Geoghegan (1971); Greeno (1974); and Hayes-Roth and Hayes-Roth (1979). Bruner (1974) argues against the application of Miller, Galanter, and Pribram's theory to child development. Edwards (1996: 233) describes the role of *Plans and the Structure of Behavior* as a cognitivist manifesto.
9. For alternative computational theories of plans, see Agre and Chapman (1990), Alterman (1988), Chapman and Agre (1987), Grosz and Sidner (1988), Pollack (1992), and Webber et al. (1995).
10. I owe this way of looking at things to Gerald Jay Sussman. Smith (1996), extending this line of argument, suggests that "computation" is not even a meaningful category, that is, a category that distinguishes any well-defined set of objects from others.
11. Moreover, both of these methodological concepts are distinct from the substantive AI concept of "problem solving," which later chapters will discuss at length.
12. Woolgar (1994) argues that the ascription of intentional categories, whether to people or machines, is always discursively organized within a tacit order of social convention. Thus, he suggests that a status such as "knowing" or "acting" does not reflect an objective fact residing underneath an agent's surface, but is instead something assigned by the participants to a social setting based on surface displays. It follows, he suggests, that narrating the operation of machinery in intentional terms is not metaphorical, since that would presuppose that intentions are inherently attributes of people and not machines. Instead, he suggests, research should inquire into the methods and conventions of ascription in particular settings.
13. See also Holmqvist and Andersen (1991).
14. Anderson (1990) has more recently attempted to provide principled justifications for his architectural ideas by arguing that certain designs are "rational," meaning optimal in terms of some efficiency criterion. By starting out with an elaborate proposal about cognitive architecture whose details need pinning down, he is able to formulate these criteria very specifically and in quantitative terms. For the most part, the arguments depend not on fundamental issues of physical realization but on more abstract measures of efficiency.
15. In the vocabulary of AI, a *representation scheme* is a language or notation or

formal system for representing things, and a *representation* is an instance of some representation scheme. For example, the first-order predicate calculus is a representation scheme and the logical sentence *sleeping*(*Rover*) is a particular representation. It is often assumed for expository simplicity that representations represent particular things in the world, but $\forall(x)$ *snoring*(x) $\rightarrow$ *sleeping*(x) – a quantified sentence involving two predicates, *snoring* and *sleeping* – is a representation that does not. Representations are commonly held to be mental objects; representation schemes are commonly held to be structuring principles of mental life, but are not generally held to be mental objects themselves.

16. On AI's relations with its critics see Bloomfield (1987b).
17. Sengers (1995: 157) uses the phrase "immanent critique."

2. Metaphor in practice

1. Philosophical analyses of Whorf have ranged from Devitt and Sterelny's (1987: 172–184, 201–206) excoriation to Black's (1962: 244–257) carefully itemized puzzlement. More recent scholarship, however, has made it possible to situate Whorf's view in a specifically anthropological and linguistic tradition, and to employ it as a basis for empirical investigation (J. Hill and Mannheim 1992; Lucy 1992).
2. *De Anima* iii, 4: "The thinking part of the soul [i.e., the mind] must [be] capable of receiving the form of an object; that is, must be potentially identical in character with its object without being the object." See Arbib and Hesse (1986: 148–152).
3. A dialectical relationship between two entities, called *moments,* has three properties: (1) the moments are engaged in a time-extended interaction, (2) they influence each other through this interaction, and (3) this influence has grown sufficiently large that it becomes impossible to define either moment except in terms of its relationship to the other. The moments are typically, though not necessarily, thought of as being in conflict with one another; the interaction between them and their mutual influence are the products of this conflict. If this seems overly metaphysical (which is to be expected, given its origin in Hegel's *Science of Logic*), think of it in the following way. Make a list of the states or properties that the two entities possess at a given moment. Then take each of the lists in isolation from the other and ask whether it is possible to find any rhyme or reason for that set of states or properties, except by reference to the interaction and cumulative influence that the entity has gone through. If not, i.e., if the only reasonable explanation for each entity's list makes reference to its past history of interaction with the other entity, then the relationship between the two entities is

dialectical in nature. Moreover, the states and properties you have been enumerating must probably be specified in dialectical terms as well.

4. Similar arguments are found throughout the German philosophy of the mid-twentieth century. See, e.g., Heidegger's essay "The Question Concerning Technology" (1977 [1954]; cf. Zimmerman 1990) and Husserl's *The Crisis of European Sciences and Transcendental Phenomenology* (1970 [1954]), of which more later. Habermas (1987 [1981]) later formulated the Frankfurt School's version of this critique in terms of the technical colonization of the lifeworld. These authors, however, did not locate the phenomenon of technical colonization in a linguistic project. Heidegger and Husserl regarded it as, crudely speaking, a matter of worldviews, whereas Habermas viewed it as an institutional matter of the rational reorganization of daily life. Although these authors do not dwell on case materials, they formulated their arguments contemporaneously with the ascendence of systems analysis, which has been the subject of extensive critical analysis (Bloomfield 1986; Hoos 1983; Lilienfeld 1978).
5. Jordanova (1986) and Markus (1987) survey the related difficulties of analyzing the role of language in the natural sciences; see also S. Montgomery (1989).
6. "My point here is not that we *ought* to think metaphorically about social policy problems, but that we *do* already think about them in terms of certain pervasive, tacit generative metaphors" (Schön 1979: 256).
7. The notion that a generative metaphor can organize an integrated philosophical worldview is generally attributed to Pepper (1961), who asserted that four basic *root metaphors* (formism, mechanism, contextualism, and organicism) suffice to classify philosophical worldviews.
8. Perhaps the most sophisticated version of this trend is found in Turbayne (1970), who decries the tyranny of dead metaphors in both philosophy and science. Rather than argue for their elimination, Turbayne regards metaphors as inescapable and recommends that they be used consciously. This is my own view as well. But he also regards metaphors essentially as falsehoods – as constructions that can be laid over already-determined things, rather than as constitutive of our knowledge of those things. Berggren (1962) and Ricoeur (1977: 251–254) offer trenchant critiques of this view. Ricoeur believes that scientific metaphors operate not through single coined words but through the coherent deployment of connected systems of metaphors (1977: 243–244). On the other hand, he argues (1977: 289–295) that philosophical terms are not metaphorical in any important sense beyond the simple lexicalizations of metaphor found in a word like "comprehend" (Latin "grasp").
9. Compare J. Martin and Harré (1982) on the theories of metaphor of Black,

Boyd, and I. A. Richards, as well as Wheelright (1968: 102–123) on metaphorical tension.

10. See Bolter (1984: 15–42) for a general discussion of *defining technologies* such as clocks and computers. Schön (1963: 191–199) has suggested that much of the social history of ideas can be reconstructed by tracing the careers of particular metaphors which hop from one theory to another across centuries. A similar idea is found in Nietzsche's theory of metaphors as intellectual imperialists absorbing whatever they can in a given historical period (Cantor 1982).
11. See also Hayles (1990).
12. See Black's essay "Metaphors and Models" (in Black 1962). "The heart of the method consists in *talking* in a certain way" (1962: 229). Compare Hesse (1966).
13. This use of the word "logic" derives ultimately from Hegel (see Ilyenkov 1977) and is used in roughly this sense in anthropology (Bourdieu 1989; Sahlins 1981). It runs a considerable risk of being confused with the analytical and technical notions of logic (logic as the forms and notations of reasoning; logic as the basic circuitry of computers), both of which also appear extensively in this book. Yet I cannot find an alternative vocabulary that works any better. To forestall potential misunderstandings, I will take care to mark which sense of the term I have in mind.
14. A prototype of this phenomenon is Ptolemy's theory of epicycles in planetary motion, to which Dreyfus (1972: 112) likens AI's regress of rules.
15. For Derrida's own versions of them, see *Positions* (1981), *Of Grammatology* (1976), and "White Mythology" (1982a). On Derrida, Heidegger, and information technology see Coyne (1995).
16. Compare Spinosa (1992) on Derrida and Heidegger.
17. The word "case" itself must be employed with some caution. The presupposition that the world comes pre-sorted into "cases" (or "data" and "phenomena"), as opposed to being actively interpreted within discourses, is itself a tacit theory of presence. Compare Cartwright's (1983) distinction between "phenomenological" vs. "fundamental" laws in physics.
18. The term is variously translated; Derrida's word is *renversement.*
19. This notion of metaphor as establishing contact between disparate worlds is roughly Black's (1962) theory. It is also a metaphorical characterization of its own. What is really needed is a theory of metaphor as social practice. Such a theory might follow my policy of establishing articulations between the literary and practical dimensions of language use. In particular, one might start with Derrida's (1982a) refusal to establish "metaphor" as a term of critical art standing outside the field of language it describes. "Metaphor" names a wide variety of discursive practices, and its properties depend in important ways on the practical relationship between the two "worlds" in

question. In the present case, it is precisely the project of existing technical practice to privilege one of these worlds (the world of mathematics) at the expense of the other (the world of "ordinary" descriptions of everyday life) while still retaining the ability to gloss the workings of technical models in "ordinary" terms (Agre 1992). The history and dynamics of this project are still poorly understood, but they deserve further investigation.

20. One must be careful to distinguish between the structurally pathological vagueness of much technical vocabulary and the inherent imprecision of language in a functioning scientific enterprise (Boyd 1979: 403).

3. Machinery and dynamics

1. Elias (1982 [1939]: 245–263) recounts some of the history of inside–outside metaphors applied to the individual in sociology. Bachnik and Quinn (1994) and Devisch (1985) provide comparative perspectives. Rorty (1979: 32–38) describes "the diversity of mind–body problems" in philosophy.
2. Johnson's (1987) theory of "the body in the mind" describes the role in mental processing of metaphors based on the body, such as balance. Sternberg (1990) enumerates "metaphors of the mind," but these are not metaphors in the strict sense but whole disciplinary discourses. He groups these into "metaphors looking inwards" ("the biological metaphor," "the computational metaphor," etc.) and "metaphors looking outwards" ("the sociological metaphor" and "the anthropological metaphor"), recommending in the end the multifactorial approach to intelligence proposed in his earlier work. Miller's (1974) analysis of metaphors for psycholinguistics is similar. De Mey (1982: 6) suggests that "outside world" and "inside self" are conceptual models. Edwards (1996: 164) observes that "The COMPUTER metaphor . . . draws attention to the internal structure of the mind and its representational schemes." Derrida (1976) discusses the origin of inside–outside metaphors in relation to language, for example in Saussure.
3. See also Forsythe (1993a).
4. Compare Edwards (1996: 256): "In its Enlightenment-like proposals for an encyclopedic machine, AI sought to enclose and reproduce the world within the horizon and the language of systems and information. Disembodied AI, cyborg intelligence as formal model, thus constructed minds as miniature closed worlds, nested within and abstractly mirroring the larger world outside."
5. Note that AI discourse employs the word "model" for two different purposes. In its methodological meaning, already familiar from Chapter 1, it refers to a computer program whose input–output behavior is supposed to predict the behavior of some natural system such as a laboratory subject. In its substantive meaning, usually as part of the phrase "world model," it

refers to a mental representation whose structure is homologous with the parts of the outside world that it represents.

6. See, e.g., Bruner (1986); Dewey and Bentley (1949: 119–138); Garfinkel (1967); Gibson (1986); Lave (1988); Leont'ev (1981); Maturana and Varela (1988); Sullivan (1953: 102–103); Varela, Thompson, and Rosch (1991); and Vygotsky (1978). Gadlin and Rubin (1979) provide a useful critique of a weak version of interactionism. Pervin and Lewis (1978) and Pervin (1978) survey several precomputational formulations of the interaction between "internal" and "external" determinations in psychology. My use of the word "interactionism" is broader than that of the sociological movement known as "symbolic interactionism," which descends from the pragmatist movement of Dewey (1929) and Mead (1934) to present-day figures such as H. S. Becker (1986) and Glaser and Strauss (1967). Finally, my use of the term bears no relationship to the mentalist and dualist notion of "psychophysical interactionism" in Popper and Eccles (1977).
7. The distinction between task environment (environment plus task) and problem space (abstract space of paths that mental search processes might explore) is clearer in Newell and Simon's *Human Problem Solving* (1972: 55–60, 72, 79–90, 83–85), but the word "object" still floats freely between external physical objects and internal symbolic representations of those objects (1972: 21, 67, 72).

4. Abstraction and implementation

1. I am using the word "implementation" in a more restricted way than usual, to refer to the specifically *physical* realization of some abstraction. Thus, a Fortran program is not, by this definition, an implementation of anything. Indeed, *qua* software, it is a variety of abstraction, one that can be implemented by being run on a suitable physical machine. (M. Mahoney [1988: 116–117] observes that computer hardware and software have distinct intellectual histories and still retain largely separate identities. In any case, the epistemological status of the hardware–software distinction is, in my view, of little moment for AI.) Chalmers (1995: 391) suggests that "a physical system implements a computation if the causal structure of the system mirrors the formal structure of the computation." Chalmers is trying to adduce general conditions for saying that something implements something else, whereas in technical practice the question arises only for systems that have been explicitly designed to implement something.
2. Recent commercial microprocessors accelerate the basic serial-processing paradigm by performing as many as four instructions at a time. But this innovation does not change most programmers' relation to the processor in any way.

3. Despite what one might hope, however, the successive layers of early visual processing are not neatly stacked into a three-dimensional cube. Instead, they are laid out in adjacent regions of a folded sheet of cortex, with bundles of wires passing outside the sheet as they travel from one stage to the next (Maunsell and Newsome 1987).
4. On the metaphor of computer "memory" see Bolter (1984: 151–164).
5. The most significant exception is the task of "register allocation," through which the compiler generates instructions that shuffle various quantities back and forth between the millions of general-purpose memory words and the small number of registers that other instructions can invoke directly. The IBM 801 computer, mentioned earlier, was made possible in part by an elegant mathematical solution to the register-allocation problem based on graph-coloring algorithms.
6. See Sun (1994) for a brief survey.
7. On this divergence in approaches to AI see Schopman (1987).
8. Newell credits ACT* (J. R. Anderson 1983, 1993) as the first such theory.
9. See especially his analysis (Newell 1990: 121–123) of the time scales of cognition and action.

5. The digital abstraction

1. This is not actually right. The theory of electricity, unfortunately, is formulated in such a way that electrons have a negative charge, thereby making the theory almost impossible to explain with familiar analogies.
2. Mahoney (1988: 116) observes that "it is really only in von Neumann's collaboration with the ENIAC team that two quite separate historical strands come together: the effort to achieve high-speed, high-precision automatic calculation and the effort to design a logic machine capable of significant reasoning."
3. Searle (1986) is perhaps an exception.
4. The latches I will describe are "dynamic latches," so called because they depend on the electrical capacitance of the wires. "Static latches" employ gates that continually drive each wire to its correct value.
5. As the nontechnical reader has probably observed, the technical term "state" is used in several loosely interrelated senses. I hope the distinctions will be clear enough from the context.
6. For Augustine see P. Brown (1967); for Descartes see Bordo (1987) and Toulmin (1990); for Turing see Hodges (1983).

6. Dependency maintenance

1. My dissertation (Agre 1988) provides a somewhat fuller account.
2. Similarly, many linguists report that they often spontaneously notice in

ordinary conversation the grammatical constructions they have been studying.

3. Routines differ from what Schank and Abelson (1977; see also Schank 1982) call scripts. Scripts, unlike routines, are held to be mental entities with causal roles in action and understanding. A script might be considered a *representation* of a routine, but it is not the routine itself. In any event, the primary purpose of a script is not to dictate action, but to help form expectations and interpretations when engaging in the action, or in understanding stories about the action.
4. Lave (1988: 186–189), impressed by the degree of moment-to-moment contingency in the dialectically emergent interactions between people and sites of practice, argues that an activity can have a routine character only if the routineness is deliberately produced, that is, if the activity is actively shaped to give it a routine character. This sort of active shaping obviously does exist, particularly when some activity is consciously intended to be "the same" time after time, for example in the prescribed sequences of action in factory work. But as Bourdieu (1977 [1972]), Ortner (1984), and others have argued, embodied skills and the customary forms of artifacts suffice to explain a great deal of stable routine in everyday life. It is important to distinguish between an industrial conception of routine, which is carried over into AI's conventional ways of representing action, and emergent routines, which have been considerably harder to conceptualize in computational terms. And it is crucial to distinguish between a theorist's description of activity as routine and participants' own obligations to exhibit their activities as having been conducted according to some routine. Lave's argument has implications for the latter phenomenon, but not for the former. (This is discussed further in Chapter 8.) On routine work generally see Gasser (1986).
5. For evidence on the tendency of routine mutations to become permanent and on several other relevant phenomena, see Neches (1981).
6. The larger phenomenon of transfer (Thorndike and Woodworth 1901) is the subject of a large literature (Singley and Anderson 1989). The concept of transfer can be defined broadly as the degree to which learning that occurs in one situation influences behavior in another situation. Lave (1988) has argued that the empirical evidence does not support a common narrow interpretation, according to which learning transfers from one situation to another because of a mental representation of a formal analogy between the two situations. Alternative interpretations of the concept of transfer exist, such as Pea's (1987) suggestion that material learned in one setting must be interpreted, in the manner of a text, in any other situation to which it might be transferred.
7. Hayes (1975) invented dependencies. So did Stallman and Sussman (1977),

independently and a little later. Doyle (1979) abstracted the functionality of dependencies to produce the first Truth Maintenance System (TMS), a name Doyle and most others now regret. These systems have been used principally to direct backtracking in languages that express domain-specific search strategies. An important early analysis of the theoretical problems that motivated the invention of dependency-directed backtracking appears in Sussman and McDermott (1972). An extensive technology of TMS's has grown up, including de Kleer's *assumption-based* version, the ATMS (de Kleer 1986; de Kleer and Williams 1987). Forbus and de Kleer (1993) codify methods for using TMS's in problem solving systems. For an interesting theoretical treatment of the complexity issues that arise in dependency systems see Provan (1987). On the formal logic of dependencies see McDermott (1991). Dependencies are similar in spirit to Minsky's (1980, 1985) idea of *k-lines* and to Carbonell's (1983) idea of *derivational analogy.* McAllester (1988) has used the idea of accumulating lines of reasoning in logic circuits, in his notion of *semantic modulation* and in the lemma library of his proof checker.

8. For a broader discussion of some related phenomena concerning *reminding* and its relationship to memory organization see Schank (1982) and Kolodner and Cullingford (1986).

7. Rule system

1. David Chapman suggested the outlines of this scheme for the case of IF rules.

8. Planning and improvisation

1. For a later computational treatment of serial order along parallel intellectual lines, see Jordan and Rosenbaum (1989).
2. On the relationship of *Plans and the Structure of Behavior* to Newell and Simon's work, see Simon (1991: 224).
3. In their explicit definition, Miller, Galanter, and Pribram spoke of Plans as "processes," perhaps due to the influence of Lashley's theory, which did not include any notion of a hierarchical symbolic structure. But they do not explain what it would mean to "execute" a process (Hoc 1988: 91). And when they actually discussed the mechanics of Plan use, Plans generally had a thinglike character: they could, for example, be selected from a library or assembled from scratch or by modification of existing Plans. I will persist in referring to Plans as "structures," since it is this understanding of Plans that had the greatest effect on the subsequent AI literature.
4. Miller et al. did offer *some* explication of the process of execution in the notion of a "TOTE (Test Operate Test Exit) Unit" as the basic unit in the

construction of a Plan. The idea, inspired by cybernetic notions of feedback control, is that each Plan step is associated with the condition it is intended to achieve; the organism executes each Plan step repeatedly until its associated condition obtains. They offered the example of hammering a nail repeatedly until it no longer protrudes from the wood. Despite its interactionist character, and perhaps because of it, this notion is not at all integrated into the rest of their theory and has had little influence on subsequent AI work. Moreover, the proposal itself faces serious difficulties, among them explaining the ways in which successive strokes in actual hammering are adapted to specific conditions (how far the nail protrudes, how it is bent, how much clearance is available for swinging the hammer, the consequences of denting the wood, etc.). It is also quite unclear how many kinds of activity can reasonably be described in terms of these nested feedback loops. I therefore disagree with the prevailing consensus (e.g., Edwards 1996: 230–233) that TOTE units constitute the intellectual core of Miller, Galanter, and Pribram's theory.

5. This work is described in several papers, all of which repay close reading: Fikes (1971); Fikes, Hart, and Nilsson (1972a, 1972b); Fikes and Nilsson (1971); Munson (1971). See also Nilsson's later revival and generalization of some of the original ideas from this project in the form of *action nets* (1994); the result is similar to the circuitry developed by RA (see Chapter 9). For the subsequent development of the planning literature see Georgeff (1987); Allen, Hendler, and Tate (1990); Hendler, Tate, and Drummond (1990); McDermott (1992); McDermott and Hendler (1995); and Weld (1994). In a late interview (Agre 1993b: 434–436), Newell asserted that the STRIPS model of planning misconstrued the theory of problem solving in GPS by focusing exclusively on the execution of plans rather than upon planning as simply one internal problem solving method that contributes to an agent's choices of actions.
6. I am simplifying here for expository purposes. In fact the STRIPS search space consisted of partially specified plans that the search process could incrementally elaborate in a variety of ways. Subsequent research has developed and analyzed this approach in much greater detail (Chapman 1987; S. Hanks and Weld 1995).
7. In the AI literature, the program that executes a plan is most commonly called an "executor." The term "executive" is usually associated with the notion of an *executive decision* (see, e.g., Hayes-Roth and Hayes-Roth 1979: 289–290) made in the process of constructing plans. That notion will play no role in my argument.
8. The phrase "reactive planning" which became current among AI people around 1987, reproduces these confusions in an instructive way. The adjec-

tive "reactive" is supposed to mark a distinction from the conventional account of action as purely the construction and execution of plans. The term, however, seems almost always to be opposed to the broad use (reasoning about action) rather than to the narrow use (constructing a plan to execute). As a result, AI vocabulary has made it difficult to conceive that agents might reason about action without executing plans. (This dynamic is of some concern to me, since the phrase "reactive planning" is often incorrectly credited to Agre and Chapman [1987].) For a particularly forceful version of this logic, see Ginsberg (1989). At the same time, I do not wish to condemn those authors who do wish to apply the word "reactive" to their projects, each of which deserves separate analysis (Firby 1987; Fox and Smith 1984; Georgeff and Lansky 1987; Kaelbling 1986; Sanborn and Hendler 1988; Schoppers 1987, 1995).

9. For a typical statement of the view that recipes are at best defective computer programs see Knuth (1983: 6). See also Shore (1975). For more about the actual nature of recipes see Scher (1984).
10. See, however, the relatively sophisticated treatments of goals in Schank and Abelson (1977), though their topic is story understanding and not action as such, and Wilensky (1983).
11. Donald (1989, 1990b), however, describes a technique for plan construction based on "error detection and recovery" (EDR) methods that addresses certain kinds of geometric uncertainty in robot motion.
12. This use of the word "incremental" is distinct from a related use, for example in Hayes-Roth and Hayes-Roth (1979: 304), according to which a plan is constructed entirely in advance of execution by a process of adding successive increments to an initially skeletal plan structure. I will discuss Hayes-Roth and Hayes-Roth's sense of the term momentarily.
13. Knoblock (1989) has extended Anderson and Farley's analysis in an elegant way.
14. On Descartes's mechanism and view of the body, see Jaynes (1970), Mackenzie (1975), and Rodis-Lewis (1978).
15. Chapman (1987: 352–354), for example, points out that certain restricted categories of plan-construction problems are amenable to computationally tractable *intermediate methods*.
16. Recent work (McAllester and Rosenblitt 1991; Weld 1994) has clarified the structure of the search space and found some ways to circumvent Chapman's result. Nonetheless, the classical plan-construction problem still faces a stiff trade-off between the expressive power of its representations and the practicability of its searches.
17. See Sudnow (1978) and the discussion of Lave and Suchman below.
18. In subsequent discussions I will use the phrases "improvised action" and

"situated action" interchangeably. These phrases do not pick out a particular species of action. Instead, they emphasize that *all* action is improvised, in the special sense of the term that I have defined.

19. In related work, Shrager and I have extended this level of Lave's analysis to describe the extremely fine-grained processes of routine evolution that we observed in a detailed study of someone using a photocopier (Agre and Shrager 1990).
20. The term "objective" here is not supposed to imply "infallible" or "certain" but simply "from the theorist's point of view." See Warren (1984) for a general discussion.
21. Bratman (1987) suggests that the necessity of plans in human activity can be demonstrated through computational considerations: a preconstructed plan saves effort that would otherwise have to be expended in moment-to-moment reasoning; and since nobody has the processing capacity to construct fresh plans continually on the fly, plans are indispensable to intelligent action. Bratman is correct in suggesting that plans can conserve computational effort. But it is not so clear that plans are indispensable, given that many other resources can serve a similar purpose – tools, for example (Agre and Horswill 1992; Vygotsky 1978 [1934]). Perhaps the crucial issue for present purposes is whether Bratman's argument implies that plans are like computer programs. Bratman himself does not suggest that it does, but the conclusion may seem plausible anyway. After all, the whole point of computer programs is that they can be interpreted with extremely little computational effort. People, however, are much smarter than the central processing units of computers; they can, and routinely do, use plans in much more varied and complicated ways than computer processors use programs. In arguing that it actually is possible to compose complex plans on the fly, I do not mean to argue against the existence of recipes, schedules, visualized courses of action, memorized scripts, and so forth. My purpose, instead, is simply to place great pressure on the planning–execution distinction in order to see how it breaks and what layers of unarticulated assumptions lie buried underneath.
22. Thus, Fodor (1981b) refers to the research strategy of "methodological solipsism."

9. Running arguments

1. On the changes that words like "decide" undergo as they become cognitivist vocabulary see Coulter (1983: 8–10) and Woolgar (1987).
2. I do to mean to imply that *any* uncertainty or change, etc., makes plan construction impossible. In practice, these negative phenomena open up a huge space of technical trade-offs in which technical tractability can often

be repurchased by imposing additional constraints on the world or by loosening the criteria imposed on the resulting plans. It seems likely that much of this work can find application in constrained environments that possess particular combinations of uncertainty and constraint. For example, see Donald (1990a) on robotic path planning in the presence of uncertainty about spatial position.

3. For research on plan construction in the presence of other agents and autonomous phenomena, see Durfee (1988) and Lansky (1988).
4. On the philosophical analysis of argument see Perelman and Olbrechts-Tyteca (1969 [1958]) and Rescher (1977). On its psychology see Billig (1987).
5. Note that Doyle is not conflating isomorphic streams of thought and action in the manner of the mentalist planning research discussed in Chapter 8; rather, he is effectively redefining thought and action in different terms, so that one category contains the other (cf. McDermott 1978).
6. Doyle refers to his theory as *dialectical argumentation.* Whereas I use the term dialectical in a sense that derives from Hegel, Doyle's version of the term derives from the Greek study of dialog and debate. The word "argument" is ambiguous as well, in both Doyle's text and my own; it refers equally to the individual points raised in a debate and to the whole ensemble of those points that results in a decision.
7. Fodor also suggests that the peripheral systems include linguistic modules of the type that Chomsky (1980: 39; his phrase is "mental organs") has hypothesized (cf. Garfield 1987). Since his primary theoretical focus is on the peripheral systems, he tends to define the central systems in negative terms as not-peripheral – that is, as not-modular. This view aligns strikingly with that of Lashley (1951), who also viewed the brain's central cognitive functioning as a holistic mass of interconnections that interacts with the world through the quasi-grammatical representations of action that I described in Chapter 8.
8. The periphery and part of the world simulation are implemented as another set of Life rules. Also, the modules and their connections are sufficiently abstract that they can be arranged in arbitrary configurations. I will suppress these extra generalities because they play no role in the examples.

10. Experiments with running arguments

1. Note that I am using the word "activity" in two distinct senses. One sense, introduced in Chapter 1, refers in a general way to the things people do all day. The second, more technical sense, used only in this chapter, refers to the number of elements in a circuit whose states are changing. Both uses of the word are conventional in AI.

2. Both Brooks (1991) and Dreyfus and Dreyfus (1988) have filed a related complaint: that the hardest part of an AI problem is already solved when a "world" comes labeled with predicates like ON, LEFT-OF, ABOVE, TOUCHING, and MOVING.
3. Recall that Chapter 9 has compared running arguments with SOAR's universal subgoaling and Chapter 7 has already distinguished RA's rule system from a production system, contrasting the as-long-as semantics of Life rules with the imperative semantics of productions.
4. Another, distinct problem is that most of the rules were written to employ exhaustive forward chaining. An alternative would have been to write the rules in more of a backward-chaining fashion, making more extensive use of the methods described by de Kleer, Doyle, Steele, and Sussman (1977). In that case, the system would effectively have been exploring the space of potential inferences from its perceptions, and most likely it would often fail to discover an inference that would have changed the action it took. Once some situation did happen to provoke that inference, the dependencies would automatically make it again whenever it applies.
5. Only fairly recently, for example, have Gupta and Nau (1992) discovered that the crucial factor in the search for *optimal* (i.e., shortest) blocks-world plans is the beneficial interactions between subgoals. These are hard to predict, and they make it even harder to decompose the plan-construction process for different subgoals.

11. Representation and indexicality

1. "A computer program capable of acting intelligently in the world must have a general representation of the world in terms of which its inputs are interpreted" (McCarthy and Hayes 1969: 463).
2. This is clearly true for *iconic* representations, in which bits of representation correspond one to one with things and properties in the world. But it is also true for representation schemes that employ quantification. In these cases the correspondence is more complex, but it is still a componential mapping.
3. Chatila and Laumond (1985), however, present an instructive study in the complexities of actually trying to maintain an accurate world model, even in a simplified environment.
4. Another influential vocabulary derives from McCarthy and Hayes (1969), who say that a representation scheme is *metaphysically adequate* if it includes formal elements that correspond to the world's basic ontological categories, *epistemologically adequate* if it can express the knowledge that a given agent can actually obtain through its observations of the world, and *heuristically adequate* if it can express the lines of reasoning through which the agent decides what to do.

5. See Frege (1968 [1918]).
6. Since Boden (1970), the argument that computational devices could exhibit genuine intentionality has rested on the strong mimetic understanding of representation that I have outlined. Boden, indeed, cites Craik for the notion of mental model and argues that the behavior of a robot that employed a model of its own body could not be understood except in partially nonphysical terms. To be sure, Boden does not assert that intentionality is necessarily subserved by a mental model; nonetheless, the AI literature regularly implicitly equates the two and does not, as far as I am aware, develop any alternatives. In discussions of intentionality in the cognitivist literature, it is common to restrict attention to examples that clearly call for the imputation of beliefs, desires, or intentions to some agent. (To speak of "intentionality" does not imply the existence of a correlative "intention," in the sense of "intending to do something.") Thus, one is not likely to see "avoiding an obstacle" offered as an example of intentionality, nor indeed any other case in which the intentionality inheres in an action or in a concrete functional relationship between an agent and an object. (An exception is Searle [1980].) Searle (1981) provoked much debate on the intentionality of machines with his "Chinese room" thought experiment, which purports to demonstrate that a computer (or, for that matter, a human being) could not exhibit intentionality simply by following rules that it does not understand. Dietrich (1994) gathers rebuttals seeking to defend various versions of computationalism against this argument.
7. In fact, in *Being and Time* Heidegger uses the terms "intentionality" or "consciousness" only when discussing others' work, and scholars disagree about the precise relevance of *Being and Time* to the issues at hand. I follow the interpretation of Dreyfus (1991: 46–59), but see Olafson (1987).
8. A third mode of relationship, *Mitsein* (or being-with), pertains, roughly speaking, to intentionality toward other people. It also relates to the broader phenomenon that we experience the world of daily life as something shared with others.

12. Deictic representation

1. Smith (1996) argues that the very laws of physics are indexical.
2. Perhaps the most sophisticated discourse on representation in computer science is data modeling (Simsion 1994), a body of techniques for designing databases that exhibits some parallels to AI ideas about semantic networks (Borgida 1991).
3. "Only a being that could have conscious intentional states could have intentional states at all, and every unconscious intentional state is at least potentially conscious" (Searle 1992: 132).

4. See Dennett (1987), Haugeland (1985), Searle (1984), and Woolgar (1994).
5. This proposal is based on a rough analogy with Heidegger's analysis of everyday intentionality in Division I of *Being and Time,* with objective intentionality corresponding to the present-at-hand and deictic intentionality corresponding to the ready-to-hand. Much of my terminology will be taken from this analysis and from Dreyfus's (1991) commentary on it. But the analogy between my proposal and Heidegger's is imperfect. Many of Heidegger's analyses, such as network of reciprocal implication that is characteristic of the phenomenon of equipment, have no equivalents in my scheme, though it would be worth trying to understand what these equivalents would be like.
6. In his logical formalization of indexical propositional knowledge, Lespérance (1991: 43; cf. Burge 1977) usefully contrasts two distinctions: objective vs. indexical and de re vs. de dicto. Objective knowledge is framed in terms of a specific agent and time (e.g., George at noon), whereas indexical knowledge is relative to the agent and current moment (e.g., me now). De re knowledge picks out a particular individual (e.g., the Eiffel Tower) whereas de dicto knowledge picks out an individual through a definite description (e.g., the tallest structure in Paris). My definition of a deictic ontology deliberately collapses these distinctions because the "odd" cases (objective de dicto and indexical de re) are of greater interest for the analysis of propositional knowledge than for the analysis of intentional relationships as such. In my terms, "functional" means roughly de dicto, but without the implication that the agent has access to the definite description in the form of a linguistically analyzable entity.
7. "Our central admonition [for robot designers] is that *one should be careful not to impose excessive knowledge requirements upon agents in formalizing actions; in particular, one should merely require indexical knowledge, as opposed to de re knowledge, when that is sufficient*" (Lespérance 1991: 104; emphasis in the original; cf. Lespérance and Levesque 1995).
8. This concept was originally called "indexical-functional representation" (Agre and Chapman 1987). Since this phrase seemed unwieldy, subsequent publications (Agre 1988; Chapman 1991) have employed the term "deictic representation." Though more compact, it is unfortunately somewhat misleading. The word "deixis," both in classical Greek and in modern philosophical and linguistic vocabulary, pertains to the act of referring to something by directly pointing it out. It is broader than "demonstration" or "ostention" but narrower than the meaning I have in mind here, which is not necessarily linguistic and incorporates the full range of stable causal relationships that one might have to a thing.
9. Hayes (1979b) describes an attempt to formalize the semantics of frames

within an extended first-order logic. Schank (1982: 5) lists the interpretations that work in the Yale School has given to the idea of a script.

10. On active vision see Bajcsy (1988), Ballard (1991), Blake and Yuille (1992), Horswill and Brooks (1988), Larkin and Simon (1987). Early formulations of active vision include Garvey (1976) and Tenenbaum (1973). Several authors have proposed sophisticated models of visual-motor coordination, guided by various combinations of biological and technological inspiration; see Arbib (1989), Grossberg and Kuperstein (1989), and Mel (1990).
11. See Chapman and Agre (1987) and Chapman (1991) for some thoughts on the subject.
12. Rumelhart, Smolensky, McClelland, and Hinton (1986) suggest how a connectionist model might internalize simple ways of using concrete representations.

13. Pengi

1. Whitehead and Ballard (1991) have described a mechanism based on reinforcement learning for synthesizing deictic representations and devising visual routines to register the newly formed deictic aspects. They employ a quantitative reformulation of the notion of distinguishing entities in terms of their functional significance. Johnson, Maes, and Darrell (1995) use genetic programming to simulate the evolution of visual routines. For a study in the historical specificity of visual routines see Baxandall (1972).
2. In terms of the register-transfer model of computing described in Chapter 4, a microinstruction is an instruction that specifies a much smaller unit of processing than an instruction, and a processor with horizontal microinstruction set is capable of executing a large number of microinstructions on each cycle.
3. Chapman (1991) has developed these ideas about visual architecture in greater detail. For more about how visual operators might be organized see J. Mahoney (1987). For an early computational account of visual system architecture see J. Becker (1973).
4. A bus is a bundle of wires that carries a complex value. A three-bit bus – that is, a bundle of three wires – can distinguish eight different values.

References

Abelson, Harold, and Gerald Jay Sussman, *Structure and Interpretation of Computer Programs,* Cambridge, MA: MIT Press, 1984.

Agre, Philip E., "The Structures of Everyday Life," Working Paper 267, MIT Artificial Intelligence Laboratory, 1985a.

Agre, Philip E., "Routines," AI Memo 828, MIT Artificial Intelligence Laboratory, 1985b.

Agre, Philip E., *The Dynamic Structure of Everyday Life,* Ph.D. dissertation, Department of Electrical Engineering and Computer Science, MIT, 1988.

Agre, Philip E., "Formalization as a Social Project," *Quarterly Newsletter of the Laboratory of Comparative Human Cognition* 14(1), 1992, pp. 25–27.

Agre, Philip E., "The Symbolic Worldview: Reply to Vera and Simon," *Cognitive Science* 17(1), 1993a, pp. 61–70.

Agre, Philip E., "Interview with Allen Newell," *Artificial Intelligence* 59(1–2), 1993b, pp. 415–449.

Agre, Philip E., "Computational Research on Interaction and Agency," *Artificial Intelligence* 72(1–2), 1995a, pp. 1–52.

Agre, Philip E., "Institutional Circuitry: Thinking About the Forms and Uses of Information," *Information Technology and Libraries* 14(4), 1995b, pp. 225–230.

Agre, Philip E., "The Soul Gained and Lost: Artificial Intelligence as a Philosophical Project," *Stanford Humanities Review* 4(2), 1995c, pp. 1–19.

Agre, Philip E., and David Chapman, "Pengi: An Implementation of a Theory of Activity," in *Proceedings of the Sixth National Conference on Artificial Intelligence,* Seattle, 1987, pp. 196–201.

Agre, Philip E., and David Chapman, "What are Plans For?" in Pattie Maes, ed., *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back,* Cambridge, MA: MIT Press, 1990, pp. 17–34.

Agre, Philip E., and Ian D. Horswill, "Cultural Support for Improvisation," in *Proceedings of the Tenth National Conference on Artificial Intelligence,* Los Altos, CA: Morgan Kaufmann, 1992, pp. 363–368.

Agre, Philip E., and Jeff Shrager, "Routine Evolution as the Microgenetic Basis of Skill Acquisition," in *Proceedings of the Cognitive Science Conference,* Boston, MA, 1990, pp. 694–701.

Ajjanagadde, Venkat, and Lokendra Shastri, "Efficient Inference with Multi-Place Predicates and Variables in a Connectionist System," in *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society,* Ann Arbor, MI, 1989, pp. 396–403.

Allen, James, James Hendler, and Austin Tate, eds., *Readings in Planning,* Los Altos, CA: Morgan Kaufmann, 1990.

Almasi, George S., and Allan Gottlieb, *Highly Parallel Computing,* 2nd ed., Redwood City, CA: Benjamin/Cummings, 1994.

Alterman, Richard, "Adaptive Planning," *Cognitive Science* 12(3), 1988, pp. 393–421.

Anderson, John R., *The Architecture of Cognition,* Cambridge, MA: Harvard University Press, 1983.

Anderson, John R., *The Adaptive Character of Thought,* Hillsdale, NJ: Erlbaum, 1990.

Anderson, John R., *Rules of the Mind,* Hillsdale, NJ: Erlbaum, 1993.

Anderson, John S., and Arthur M. Farley, "Plan Abstraction Based on Operator Generalization," in *Proceedings of the Seventh National Conference on Artificial Intelligence,* St. Paul, MN, 1988, pp. 100–104.

Arbib, Michael A., "Visuomotor Coordination: Neural Models and Perceptual Robotics," in Jörg-Peter Ewert and Michael A. Arbib, eds., *Visuomotor Coordination: Amphibians, Comparisons, Models, and Robots,* New York: Plenum, 1989, pp. 121–171.

Arbib, Michael A., and Mary B. Hesse, *The Construction of Reality,* Cambridge University Press, 1986.

Arbib, Michael A., and Jim-Shih Liaw, "Sensorimotor Transformations in the Worlds of Frogs and Robots," *Artificial Intelligence* 72(1–2), 1995, pp. 53–79.

Ashby, William Ross, *An Introduction to Cybernetics,* New York: Wiley, 1956.

Athanasiou, Tom, "Artificial Intelligence: Well-Disguised Politics," in Tony Solomonides and Les Davidow, eds., *Compulsive Technology: Computation as Culture,* London: Free Association Books, 1985, pp. 13–35.

Atkinson, J. Maxwell, and John Heritage, eds., *Structures of Social Action: Studies in Conversation Analysis,* Cambridge University Press, 1984.

Bachnik, Jane M., and Charles J. Quinn, Jr., eds., *Situated Meaning: Inside and Outside in Japanese Self, Society, and Language,* Princeton, NJ: Princeton University Press, 1994.

Bajcsy, Ruzena, "Active Perception," *Proceedings of the IEEE* 76(8), 1988, pp. 996–1005.

Ballard, Dana H., "Animate Vision," *Artificial Intelligence* 48(1), 1991, pp. 57–86.

Bartlett, Frederic C., *Remembering: A Study in Experimental and Social Psychology,* New York: Macmillan, 1932.

Barwise, Jon, and John Perry, *Situations and Attitudes,* Cambridge, MA: MIT Press, 1983.

Barwise, Jon, and John Perry, "Shifting Situations and Shaken Attitudes," *Linguistics and Philosophy* 8(1), 1985, pp. 105–161.

Batali, John, "Trails as Archetypes of Intentionality," in *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society,* 1993, pp. 220–225.

Baxandall, Michael, *Painting and Experience in Fifteenth Century Italy: A Primer in the Social History of Pictorial Style,* Oxford: Clarendon Press, 1972.

Becker, Howard S., *Doing Things Together: Selected Papers,* Evanston, IL: Northwestern University Press, 1986.

Becker, Joseph D., "A Model for the Encoding of Experiential Information," in Roger C. Schank and Kenneth Mark Colby, eds., *Computer Models of Thought and Language,* San Francisco: Freeman, 1973, pp. 396–434.

Beer, Randall D., *Intelligence as Adaptive Behavior: An Experiment in Computational Neuroethology,* San Diego, CA: Academic Press, 1990.

Beer, Randall D., "A Dynamical Systems Perspective on Agent–Environment Interaction," *Artificial Intelligence* 72(1–2), 1995, pp. 173–215.

Berggren, Douglas, "The Use and Abuse of Metaphor," *Review of Metaphysics* **16**(2), 1962, pp. 237–258, and **16**(3), 1963, pp. 450–472.

Berkeley, Edmund C., *Giant Brains: Or Machines That Think,* New York: Science Editions, 1961.

Berlin, Isaiah, *Vico and Herder: Two Studies in the History of Ideas,* London: Hogarth, 1976.

Berman, Bruce, "The Computer Metaphor: Bureaucratizing the Mind," *Science as Culture* 7, 1989, pp. 7–42.

Berwick, Robert C., *The Acquisition of Syntactic Knowledge,* Cambridge, MA: MIT Press, 1985.

Billig, Michael, *Arguing and Thinking: A Rhetorical Approach to Social Psychology,* Cambridge University Press, 1987.

Birkhoff, Garrett, *Lattice Theory,* 3rd ed., Providence, RI: American Mathematical Society, 1967.

Black, Max, *Models and Metaphors: Studies in Language and Philosophy,* Ithaca, NY: Cornell University Press, 1962.

Blake, Andrew, and Alan Yuille, eds., *Active Vision,* Cambridge, MA: MIT Press, 1992.

Bloomfield, Brian P., *Modelling the World: The Social Constructions of Systems Analysts,* Oxford: Blackwell, 1986.

Bloomfield, Brian P., ed., *The Question of Artificial Intelligence: Philosophical and Sociological Perspectives,* London: Croom Helm, 1987a.

Bloomfield, Brian P., "The Culture of Artificial Intelligence," in Brian P. Bloomfield, ed., *The Question of Artificial Intelligence: Philosophical and Sociological Perspectives,* London: Croom Helm, 1987b, pp. 59–105.

Bloor, David, *Knowledge and Social Imagery,* London: Routledge & Kegan Paul, 1976.

Bobrow, Daniel G., and Terry Winograd, "Overview of KRL: A Knowledge Representation Language," *Cognitive Science* **1**(1), 1977, pp. 3–46.

Boden, Margaret A., "Intentionality and Physical Systems," *Philosophy of Science* 37, 1970, pp. 200–214. Reprinted in *Minds and Mechanisms: Philosophical Psychology and Computational Models,* Ithaca, NY: Cornell University Press, 1981, pp. 52–70.

Boden, Margaret A., *Artificial Intelligence and Natural Man,* New York: Basic Books, 1977.

Boden, Margaret A., ed., *The Philosophy of Artificial Intelligence,* Oxford: Oxford University Press, 1990.

Bolter, J. David, *Turing's Man: Western Culture in the Computer Age,* London: Duckworth, 1984.

Bono, James J., "Science, Discourse, and Literature: The Role/Rule of Metaphor in Science," in Stuart Peterfreund, ed., *Literature and Science: Theory and Practice,* Boston: Northeastern University Press, 1990, pp. 59–89.

Boole, George, *An Investigation of the Laws of Thought, on Which are Founded the Mathematical Theories of Logic and Probabilities,* London: Macmillan, 1854.

Bordo, Susan, *The Flight to Objectivity: Essays on Cartesianism and Culture,* Albany, NY: SUNY Press, 1987.

Borgida, Alexander, "Knowledge Representation, Semantic Modeling: Similarities and Differences," in Hannu Kangassalo, ed., *Entity-Relationship Approach: The Core of Conceptual Modeling,* Amsterdam: North-Holland, 1991, pp. 1–24.

Boulding, Kenneth E., *The Image: Knowledge in Life and Society,* Ann Arbor: University of Michigan Press, 1956.

Bourdieu, Pierre, *Outline of a Theory of Practice,* translated by Richard Nice, Cambridge University Press, 1977. Originally published in French in 1972.

Bourdieu, Pierre, *The Logic of Practice,* translated by Richard Nice, Cambridge: Polity Press, 1989. Originally published in French in 1980.

Boyd, Richard, "Metaphor and Theory Change: What Is 'Metaphor' a Metaphor For?" in Andrew Ortony, ed., *Metaphor and Thought,* Cambridge University Press, 1979, pp. 356–408.

Brachman, Ronald J., and Hector J. Levesque, "The Tractability of Subsumption in Frame-Based Description Languages," in *Proceedings of the National Conference on Artificial Intelligence,* Austin, TX, 1984, pp. 34–37.

Brachman, Ronald J., and James G. Schmolze, "An Overview of the KL-ONE Knowledge Representation System," *Cognitive Science* 9(1), 1984, pp. 171–216.

Bratman, Michael E., *Intentions, Plans, and Practical Reason,* Cambridge, MA: Harvard University Press, 1987.

Brooks, Rodney A., "A Robust Layered Control System for a Mobile Robot," *IEEE Transactions on Robotics and Automation* 2(1), 1986, pp. 14–23.

Brooks, Rodney A., "A Robot That Walks: Emergent Behaviors from a Carefully Evolved Network," *Neural Computation* 1(2), 1989, pp. 253–262.

Brooks, Rodney A., "Intelligence Without Representation," *Artificial Intelligence* 47(1–3), 1991, pp. 139–159.

Brown, John Seely, Allan Collins, and Paul Duguid, "Situated Cognition and the Culture of Learning," *Educational Researcher* 18(1), 1989, pp. 32–42.

Brown, Peter, *Augustine of Hippo: A Biography,* Berkeley: University of California Press, 1967.

Brown, Roger Langham, *Wilhelm von Humboldt's Conception of Linguistic Relativity,* The Hague: Mouton, 1967.

Bruner, Jerome S., "The Organization of Early Skilled Action," in Martin P. M. Richards, ed., *The Integration of a Child into a Social World,* Cambridge University Press, 1974, pp. 167–184.

Bruner, Jerome S., *Actual Minds, Possible Worlds,* Cambridge, MA: Harvard University Press, 1986.

Buckland, Michael K., "Information as Thing," *Journal of the American Society for Information Science* 42(5), 1991, pp. 351–360.

Burge, Tyler, "Belief De Re," *Journal of Philosophy* 74(3), 1977, pp. 338–362.

Burge, Tyler, "Sinning Against Frege," *Philosophical Review* 88(4), 1979, pp. 398–432.

Burtt, Edwin, *The Metaphysical Foundations of Modern Physical Science,* London: Routledge & Kegan Paul, 1959. Originally published in 1924.

Button, Graham, Jeff Coulter, John R. E. Lee, and Wes Sharrock, *Computers, Minds and Conduct,* Cambridge: Polity Press, 1995.

Cantor, Paul, "Friedrich Nietzsche: The Use and Abuse of Metaphor," in David S. Miall, ed., *Metaphor: Problems and Perspectives,* Brighton: Harvester, 1982, pp. 71–88.

Carbonell, Jaime G., "Derivational Analogy and Its Role in Problem Solving," *Proceedings of the National Conference on Artificial Intelligence,* Austin, TX, 1983, pp. 64–69.

Cartwright, Nancy, *How the Laws of Physics Lie,* Oxford: Oxford University Press, 1983.

Chalmers, David, "On Implementing a Computation," *Minds and Machines* 4(3), 1995, pp. 391–402.

Chang, C. C., and H. Jerome Keisler, *Model Theory,* 2nd ed., Amsterdam: North-Holland, 1973.

Chapman, David, "Planning for Conjunctive Goals," *Artificial Intelligence* 32(3), 1987, pp. 333–377.

Chapman, David, *Vision, Instruction, and Action,* Cambridge, MA: MIT Press, 1991.

Chapman, David, and Philip E. Agre, "Abstract Reasoning as Emergent from Concrete Activity," in Michael P. Georgeff and Amy L. Lansky, eds., *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop at Timberline, OR,* Los Altos, CA: Morgan Kaufmann, 1987, pp. 411–424.

Chatila, Raja, and Jean-Paul Laumond, "Position Referencing and Consistent World Modeling for Mobile Robots," in *Proceedings of the IEEE International Conference on Robotics and Automation,* St. Louis, MO, 1985, pp. 138–145.

Chien, R. T., and S. Weissman, "Planning and Execution in Incompletely Specified Environments," in *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence,* Tbilisi, USSR, 1975, pp. 169–174.

Chomsky, Noam, "Review of B. F. Skinner's *Verbal Behavior,*" *Language* 35(1), 1959, pp. 26–58.

Chomsky, Noam, *Cartesian Linguistics: A Chapter in the History of Rationalist Thought,* New York: Harper & Row, 1966.

Chomsky, Noam, *Language and Responsibility,* translated by John Viertel, New York: Pantheon, 1979. Originally published in French in 1977.

Chomsky, Noam, *Rules and Representations,* New York: Columbia University Press, 1980.

Churchland, Patricia S., and Terrence J. Sejnowski, *The Computational Brain,* Cambridge, MA: MIT Press, 1992.

Churchland, Paul M., *A Neurocomputational Perspective: The Nature of Mind and the Structure of Science,* Cambridge, MA: MIT Press, 1989.

Clark, Andy, "The Presence of a Symbol," *Connection Science* 4(3–4), 1992, pp. 193–205.

Clements, Alan, *The Principles of Computer Hardware,* 2nd ed., Oxford: Oxford University Press, 1991.

Cole, Michael, and Yrjö Engeström, "A Cultural-Historical Approach to Distributed Cognition," in Gavriel Salomon, ed., *Distributed Cognitions: Psychological and Educational Considerations,* Cambridge University Press, 1993.

Collins, Harry M., *Artificial Experts: Social Knowledge and Intelligent Machines,* Cambridge, MA: MIT Press, 1990.

Comaroff, John L., and Simon Roberts, *Rules and Processes: The Cultural Logic of Dispute in an African Context,* Chicago: University of Chicago Press, 1981.

Copeland, Jack, *Artificial Intelligence: A Philosophical Introduction,* Oxford: Blackwell, 1993.

Coulter, Jeff, *Rethinking Cognitive Theory,* London: Macmillan Press, 1983.

Coulter, Jeff, "Logic: Ethnomethodology and the Logic of Language," in Graham Button, ed., *Ethnomethodology and the Human Sciences,* Cambridge University Press, 1991, pp. 20–50.

Coyne, Richard, *Designing Information Technology in the Postmodern Age: From Method to Metaphor,* Cambridge, MA: MIT Press, 1995.

Craik, Kenneth J. W., *The Nature of Explanation,* Cambridge University Press, 1943.

Crevier, Daniel, *AI: The Tumultuous History of the Search for Artificial Intelligence,* New York: Basic Books, 1993.

Culler, Jonathan, *On Deconstruction: Theory and Criticisms After Structuralism,* Ithaca, NY: Cornell University Press, 1982.

Cummins, Robert, *Meaning and Mental Representation,* Cambridge, MA: MIT Press, 1989.

Davis, Philip J., and Reuben Hersh, *Descartes' Dream: The World According to Mathematics,* San Diego, CA: Harcourt Brace Jovanovich, 1986.

de Kleer, Johan, "An Assumption-Based TMS," *Artificial Intelligence* 28(2), 1986, pp. 127–162.

de Kleer, Johan, Jon Doyle, Charles Rich, Guy L. Steele Jr., and Gerald Jay Sussman, "AMORD: A Deductive Procedure System," AI Memo 435, MIT Artificial Intelligence Laboratory, 1978.

de Kleer, Johan, Jon Doyle, Guy L. Steele Jr., and Gerald Jay Sussman, "Explicit Control of Reasoning," in *Proceedings of the ACM Symposium on Artificial Intelligence and Programming Languages,* Rochester, NY, 1977.

de Kleer, Johan, and Brian C. Williams, "Diagnosing Multiple Faults," *Artificial Intelligence* 32(1), 1987, pp. 97–130.

DeMey, Marc, *The Cognitive Paradigm: Cognitive Science, a Newly Explored Approach to the Study of Cognition Applied in an Analysis of Science and Scientific Knowledge,* Dordrecht: Reidel, 1982.

Dennett, Daniel C., "Artificial Intelligence as Philosophy and as Psychology," in *Brainstorms: Philosophical Essays on Mind and Psychology,* Montgomery, VT: Bradford, 1978, pp. 109–126.

Dennett, Daniel C., "Intentional Systems," in John Haugeland, ed., *Mind Design: Philosophy, Psychology, Artificial Intelligence,* Cambridge, MA: MIT Press, 1981, pp. 220–242.

Dennett, Daniel C., *The Intentional Stance,* Cambridge, MA: MIT Press, 1987.

Denning, Peter J., and Walter F. Tichy, "Highly Parallel Computation," *Science* 250, November 30, 1990, pp. 1217–1222.

Derrida, Jacques, *Of Grammatology,* translated by Gayatri Chakravorty Spivak, Baltimore: Johns Hopkins University Press, 1976. Originally published in French in 1967.

Derrida, Jacques, *Edmund Husserl's* Origin of Geometry: *An Introduction,* translated by John P. Leavey Jr., Lincoln: University of Nebraska Press, 1978. Originally published in French in 1962.

Derrida, Jacques, *Positions,* translated by Alan Bass, Chicago: University of Chicago Press, 1981. Originally published in French in 1972.

Derrida, Jacques, "White Mythology: Metaphor in the Text of Philosophy," in *Margins of Philosophy,* translated by Alan Bass, Chicago: University of Chicago Press, 1982a, pp. 207–272. Originally published in French in 1972.

Derrida, Jacques, "Signature Event Context," in *Margins of Philosophy,* translated by Alan Bass, Chicago: University of Chicago Press, 1982b, pp. 307–330. Originally published in French in 1972.

Descartes, René, *The Philosophical Works of René Descartes,* Vol. 1, translated by Elizabeth S. Haldane and G. R. T. Ross, Cambridge University Press, 1972.

Devisch, Renaat, "Symbol and Psychosomatic Symptom in Bodily Space-Time: The Case of the Yaka of Zaire," *International Journal of Sociology* **20**, 1985, pp. 589–616.

Devitt, Michael, and Kim Sterelny, *Language and Reality: An Introduction to the Philosophy of Language,* Oxford: Blackwell, 1987.

Dewey, John, *Experience and Nature,* New York: Norton, 1929.

Dewey, John, and Arthur F. Bentley, *Knowing and the Known,* Boston: Beacon Press, 1949.

Dietrich, Eric, ed., *Thinking Computers and Virtual Persons: Essays on the Intentionality of Machines,* San Diego, CA: Academic Press, 1994.

Dixon, Michael, *Embedded Computation and the Semantics of Programs,* Palo Alto, CA: Xerox Palo Alto Research Center, 1991.

Dodds, E. R., *The Greeks and the Irrational,* Berkeley: University of California Press, 1951.

Donald, Bruce R., *Error Detection and Recovery in Robotics,* Lecture Notes in Computer Science, Vol. 336, New York: Springer-Verlag, 1989.

Donald, Bruce R., "The Complexity of Planar Compliant Motion Planning with Uncertainty," *Algorithmica* 5(3), 1990a, pp. 353–382.

Donald, Bruce R., "Planning Multi-Step Error Detection and Recovery Strategies," *International Journal of Robotics Research* 9(1), 1990b, pp. 3–60.

Donnellan, Keith S., "Reference and Definite Descriptions," *Philosophical Review* 75(3), 1966, pp. 281–304.

Doyle, Jon, "A Truth Maintenance System," *Artificial Intelligence* 12(3), 1979, pp. 231–272.

Doyle, Jon, *A Model for Deliberation, Action, and Introspection,* Ph.D. dissertation, Department of Electrical Engineering and Computer Science, MIT, 1980. Also published as Technical Report 581, MIT Artificial Intelligence Laboratory, 1980.

Drescher, Gary L., *Made-Up Minds: A Constructivist Approach to Artificial Intelligence,* Cambridge, MA: MIT Press, 1991.

Dreyfus, Hubert L., *Alchemy and Artificial Intelligence,* Santa Monica, CA: RAND, 1965.

Dreyfus, Hubert L., *What Computers Can't Do: A Critique of Artificial Reason,* 1st ed., New York: Harper & Row, 1972.

Dreyfus, Hubert L., ed., *Husserl, Intentionality, and Cognitive Science,* Cambridge, MA: MIT Press, 1982.

Dreyfus, Hubert L., *Being-in-the-World: A Commentary on Heidegger's* Being and Time, *Division I,* Cambridge, MA: MIT Press, 1991.

Dreyfus, Hubert L., *What Computers* Still *Can't Do: A Critique of Artificial Reason,* Cambridge, MA: MIT Press, 1992.

Dreyfus, Hubert L., "Heidegger on Gaining a Free Relation to Technology," in Andrew Feenberg and Alastair Hannay, eds., *Technology and the Politics of Knowledge,* Bloomington: Indiana University Press, 1995, pp. 97–107.

Dreyfus, Hubert L., and Stuart Dreyfus, "Making a Mind versus Modeling the Brain: AI Back at a Branchpoint," *Daedalus* 117(1), 1988, pp. 15–43.

Durfee, Edmund H., *Coordination of Distributed Problem Solvers,* Boston: Kluwer, 1988.

Edelman, Shimon, and Tomaso Poggio, "Representations in High-Level Vision: Reassessing the Inverse Optics Paradigm," in *Proceedings of the 1989 DARPA Image Understanding Workshop,* 1989, pp. 944–949.

Edwards, Paul N., *The Closed World: Computers and the Politics of Discourse in Cold War America,* Cambridge, MA: MIT Press, 1996.

Elias, Norbert, *The Civilizing Process,* translated by Edmund Jephcott, New York: Pantheon, 1982. Vol. 1 of *The History of Manners,* originally published in German in 1939.

Engeström, Yrjö, *Learning by Expanding,* Helsinki: Orienta-Konsultit Oy, 1987.

Evans, Gareth, *The Varieties of Reference,* edited by John McDowell, Oxford: Oxford University Press, 1982.

Fahlman, Scott E., "A Planning System for Robot Construction Tasks," *Artificial Intelligence* 5(1), 1974, pp. 1–49.

Feenberg, Andrew, *Alternative Modernity: The Technical Turn in Philosophy and Social Theory,* Berkeley: University of California Press, 1995.

Feigenbaum, Edward A., and Julian Feldman, eds., *Computers and Thought,* New York: McGraw-Hill, 1963.

Feldman, Jerome A., "Dynamic Connections in Neural Networks," *Biological Cybernetics* 46(1), 1982, pp. 27–39.

Feldman, Jerome A., "Four Frames Suffice: A Provisional Model of Vision and Space," *Behavioral and Brain Sciences* 8(2), 1985, pp. 265–313.

Feldman, Jerome A., "Neural Representation of Conceptual Knowledge," Technical Report 189, Department of Computer Science, University of Rochester, 1986.

Fikes, Richard E., "Monitored Execution of Robot Plans Produced by STRIPS," in *Proceedings of the IFIP Congress 71,* Ljubljana, Yugoslavia, Amsterdam: North-Holland, 1971, pp. 189–194.

Fikes, Richard E., "A Commitment-Based Framework for Describing Informal Cooperative Work," *Cognitive Science* 6(4), 1982, pp. 331–348.

Fikes, Richard E., Peter E. Hart, and Nils J. Nilsson, "Learning and Executing Generalized Robot Plans," *Artificial Intelligence* 3(4), 1972a, pp. 251–288.

Fikes, Richard E., Peter E. Hart, and Nils J. Nilsson, "Some New Directions in Robot Problem Solving," in Bernard Meltzer and Donald Michie, eds., *Machine Intelligence* 7, New York: Wiley, 1972b, pp. 405–430.

Fikes, Richard, and Nils Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence* 2(3), 1971, pp. 189–208.

Firby, R. James, "An Investigation into Reactive Planning in Complex Domains," in *Proceedings of the Sixth National Conference on Artificial Intelligence,* Seattle, 1987, pp. 202–206.

Fleck, James, "Development and Establishment in Artificial Intelligence," in Brian P. Bloomfield, ed., *The Question of Artificial Intelligence: Philosophical and Sociological Perspectives,* London: Croom Helm, 1987, pp. 106–164.

Fodor, Jerry A., *Psychological Explanation: An Introduction to the Philosophy of Psychology,* New York: Random House, 1968.

Fodor, Jerry A., "Tom Swift and His Procedural Grandmother," in *Representations: Philosophical Essays on the Foundations of Cognitive Science,* Cambridge, MA: MIT Press, 1981a, pp. 204–224.

Fodor, Jerry A., "Methodological Solipsism Considered as a Research Strategy in Cognitive Psychology," in *Representations: Philosophical Essays on the Foundations of Cognitive Science,* Cambridge, MA: MIT Press, 1981b, pp. 225–253.

Fodor, Jerry A., *The Modularity of Mind,* Cambridge, MA: MIT Press, 1983.

Fodor, Jerry A., *Psychosemantics,* Cambridge, MA: MIT Press, 1987.

Fodor, Jerry A., and Zenon W. Pylyshyn, "Connectionism and Cognitive Architecture: A Critical Analysis," *Cognition* 28(1), 1988, pp. 3–72.

Forbes, David L., and Mark T. Greenberg, eds., *Children's Planning Strategies,* San Francisco: Jossey-Bass, 1982.

Forbus, Kenneth D., and Johan de Kleer, *Building Problem Solvers,* Cambridge, MA: MIT Press, 1993.

Ford, Kenneth M., Clark Glymour, and Patrick J. Hayes, eds., *Android Epistemology,* Cambridge, MA: MIT Press, 1995.

Ford, Kenneth M., and Patrick J. Hayes, eds., *Reasoning Agents in a Dynamic World: The Frame Problem,* Greenwich, CT: JAI, 1991.

Forgy, Charles L., *OPS5 User's Manual,* Technical Report 81–135, Department of Computer Science, Carnegie-Mellon University, 1981.

Forgy, Charles L., "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artificial Intelligence* 19(1), 1982, pp. 17–37.

Forsythe, Diana E., "Engineering Knowledge: The Construction of Knowledge in Artificial Intelligence," *Social Studies of Science* 23(3), 1993a, pp. 445–477.

Forsythe, Diana E., "The Construction of Work in Artificial Intelligence," *Science, Technology, and Human Values* 18(4), 1993b, pp. 460–479.

Foucault, Michel, *The Order of Things: An Archaeology of the Human Sciences,* New York: Vintage, 1970. Originally published in French in 1966.

Foucault, Michel, *The Birth of the Clinic: An Archaeology of Medical Perception,* translated by Alan Sheridan, New York: Pantheon, 1973. Originally published in French in 1963.

Foucault, Michel, *The History of Sexuality,* translated by Robert Hurley, New York: Vintage, 1980. Originally published in French in 1976.

Fox, Mark S., and Stephen Smith, "ISIS: A Knowledge-Based System for Factory Scheduling," *Expert Systems* 1(1), 1984, pp. 25–49.

Franklin, Stan, *Artificial Minds,* Cambridge, MA: MIT Press, 1995.

Frege, Gottlob, "On Sense and Reference," in Peter Geach and Max Black, eds., *Translations from the Philosophical Writings of Gottlob Frege,* 2nd ed., Oxford: Blackwell, 1960, pp. 56–78. Originally published in German in 1892.

Frege, Gottlob, "The Thought," in E. D. Klemke, ed., *Essays on Frege,* Urbana: University of Illinois Press, 1968, pp. 507–536. Originally published in German in 1918.

Friedman, Sarah L., Ellin Kofsky Scholnick, and Rodney R. Cocking, *Blueprints for Thinking: The Role of Planning in Cognitive Development,* Cambridge University Press, 1987.

Gadlin, Howard, and Susan H. Rubin, "Interactionism," in Allan R. Buss, ed., *Psychology in Social Context,* New York: Irvington, 1979, pp. 213–238.

Gallistel, C. R., *The Organization of Action: A New Synthesis,* Hillsdale, NJ: Erlbaum, 1980.

Gardner, Howard, *The Mind's New Science: A History of the Cognitive Revolution,* New York: Basic Books, 1985.

Gardner, Karen M., *A Metaphorical Analysis of the Knowledge Acquisition Process,* Ph.D. dissertation, School of Library and Information Sciences, University of California, Berkeley, 1991.

Garfield, Jay L., ed., *Modularity in Knowledge Representation and Natural-Language Understanding,* Cambridge, MA: MIT Press, 1987.

Garfinkel, Harold, *Studies in Ethnomethodology,* Cambridge: Polity Press, 1984. Originally published in 1967.

Garvey, Thomas D., "Perceptual Strategies for Purposive Vision," Technical Note 117, Menlo Park, CA: Stanford Research Institute AI Center, 1976.

Gasser, Les, "The Integration of Computing and Routine Work," *ACM Transactions on Office Information Systems* 4(3), 1986, pp. 205–225.

Genesereth, Michael R., and Nils J. Nilsson, *Logical Foundations of Artificial Intelligence,* Los Altos, CA: Morgan Kaufmann, 1987.

Geoghegan, William, "Information Processing Systems in Culture," in Paul Kay, ed., *Explorations in Mathematical Anthropology,* Cambridge, MA: MIT Press, 1971, pp. 3–35.

Georgeff, Michael P., "Planning," in Joseph F. Traub, Barbara J. Grosz, Butler W. Lampson, and Nils J. Nilsson, eds., *Annual Review of Computer Science,* Vol. 2, Palo Alto, CA: Annual Reviews Inc., 1987, pp. 359–400.

Georgeff, Michael P., and Amy L. Lansky, "Reactive Reasoning and Planning," in *Proceedings of the Sixth National Conference on Artificial Intelligence,* Seattle, 1987, pp. 677–682.

Ghallab, Malik, "Task Execution Monitoring by Compiled Production Rules in an Advanced Multi-Sensor Robot," in Hideo Hanafusa and Hirochika Inoue, eds., *Robotics Research: The Second International Symposium,* Kyoto, 1985, pp. 393–401.

Gibson, James J., *The Ecological Approach to Visual Perception,* Hillsdale, NJ: Erlbaum, 1986. Originally published in 1979.

Gilbert, G. Nigel, and Christian Heath, eds., *Social Action and Artificial Intelligence,* Aldershot: Gower, 1985.

Gilbreth, Frank B., *Motion Study: A Method for Increasing the Efficiency of the Workman,* New York: Van Nostrand, 1921.

Ginsberg, Matthew, "Universal Plans: An (Almost) Universally Bad Idea," *AI Magazine* 10(4), 1989, pp. 40–44.

Giralt, Georges, Raja Chatila, and Marc Vaisset, "An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots," in Michael Brady and Richard Paul, eds., in *Proceedings of the First Symposium on Robotics Research,* Cambridge, MA: MIT Press, 1984, pp. 191–214.

Glaser, Barney G., and Anselm L. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research,* Chicago: Aldine, 1967.

Goldberg, David E., *Genetic Algorithms in Search, Optimization, and Machine Learning,* Reading, MA: Addison-Wesley, 1989.

Goldstein, Larry Joel, *Abstract Algebra: A First Course,* Englewood Cliffs, NJ: Prentice-Hall, 1973.

Goody, Jack, *The Logic of Writing and the Organization of Society,* Cambridge University Press, 1986.

Greeno, James, "Hobbits and Orcs: Acquisition of a Sequential Concept," *Cognitive Psychology* 6(3), 1974, pp. 270–292.

Grossberg, Stephen, and Michael Kuperstein, *Neural Dynamics of Adaptive Sensory-Motor Control,* 2nd ed., New York: Pergamon, 1989.

Grosz, Barbara J., and Candace L. Sidner, "Plans for Discourse," in Philip R. Cohen, Jerry Morgan, and Martha E. Pollack, eds., *Intentions in Communication,* Cambridge, MA: MIT Press, 1988, pp. 417–444.

Gupta, Naresh, and Dana Nau, "On the Complexity of Blocks-World Planning," *Artificial Intelligence* 56(2–3), 1992, pp. 223–254.

Güzeldere, Güven, and Stefano Franchi, eds., *Constructions of the Mind: Artificial Intelligence and the Humanities,* special issue of *Stanford Humanities Review* 4(2), 1995.

Habermas, Jürgen, *The Theory of Communicative Action,* Vol. 2: *Lifeworld and System: A Critique of Functionalist Reason,* translated by Thomas McCarthy, Boston: Beacon Press, 1987. Originally published in German in 1981.

Hammond, Kristian J. , *Case-Based Planning: Viewing Planning as a Memory Task,* San Diego, CA: Academic Press, 1989.

Hammond, Kristian J., Timothy M. Converse, and Joshua W. Grass, "The Stabilization of Environments," *Artificial Intelligence* 72(1–2), 1995, pp. 305–327.

Hampden-Turner, Charles, *Maps of the Mind,* London: Mitchell Beazley, 1981.

Hanks, Steve, and Daniel S. Weld, "A Domain-Independent Algorithm for Plan Adaptation," *Journal of Artificial Intelligence Research* 2, 1995, pp. 319–360.

Hanks, William F., *Referential Practice: Language and Lived Space Among the Maya,* Chicago: University of Chicago Press, 1990.

Harré, Rom, "Architectonic Man: On the Structuring of Lived Experience," in Richard Harvey Brown and Stanford M. Lyman, eds., *Structure, Consciousness, and History,* Cambridge University Press, 1978.

Haugeland, John, ed., *Mind Design: Philosophy, Psychology, Artificial Intelligence,* Cambridge, MA: MIT Press, 1981.

Haugeland, John, *Artificial Intelligence: The Very Idea,* Cambridge, MA: MIT Press, 1985.

Hayes, Patrick J., "In Defence of Logic," in *Proceedings of the Fifth International Joint Conference on Artificial Intelligence,* Cambridge, MA, 1977, pp. 559–565.

Hayes, Patrick J., "The Logic of Frames," in Dieter Metzing, ed., *Frame Conceptions and Text Understanding,* New York: Walter de Gruyter, 1979a, pp. 46–61.

Hayes, Patrick J., "The Naive Physics Manifesto," in Donald Michie, ed., *Expert Systems in the Microelectronic Age,* Edinburgh: Edinburgh University Press, 1979b, pp. 242–270.

Hayes, Philip J., "A Representation for Robot Plans," in *Advance Papers of the Fourth*

International Joint Conference on Artificial Intelligence, Tbilisi, USSR, 1975, pp. 181–188.

Hayes-Roth, Barbara, and Frederick Hayes-Roth, "A Cognitive Model of Planning," *Cognitive Science* 3(4), 1979, pp. 275–310.

Hayles, N. Katherine, "Designs on the Body: Norbert Wiener, Cybernetics, and the Play of Metaphor," *History of the Human Sciences* 3(2), 1990, pp. 211–228.

Heidegger, Martin, *Being and Time,* translated by John Macquarrie and Edward Robinson, New York: Harper & Row, 1961. Originally published in German in 1927.

Heidegger, Martin, "The Question Concerning Technology," in *The Question Concerning Technology,* translated by William Lovitt, New York: Harper, 1977, pp. 3–35. Originally published in German in 1954.

Heims, Steve J., *The Cybernetics Group,* Cambridge, MA: MIT Press, 1991.

Hendler, James, ed., *Planning in Uncertain, Unpredictable, or Changing Environments, Proceedings of the AAAI Symposium at Stanford,* University of Maryland Systems Research Center Report SRC TR 90–45, 1990.

Hendler, James, Austin Tate, and Mark Drummond, "AI Planning: Systems and Techniques," *AI Magazine* **11**(2), 1990, pp. 61–77.

Hennessy, John L., and David A. Patterson, *Computer Organization and Design: The Hardware/Software Interface,* Los Altos, CA: Morgan Kaufmann, 1994.

Heritage, John, *Garfinkel and Ethnomethodology,* Cambridge: Polity Press, 1984.

Hesse, Mary B., *Models and Analogies in Science,* Notre Dame, IN: Notre Dame University Press, 1966.

Hill, Jane H., and Bruce Mannheim, "Language and World View," in Bernard J. Siegel, Alan R. Beals, and Stephen A. Tyler, eds., *Annual Review of Anthropology* **21**, 1992, pp. 381–406.

Hill, William C., and James D. Hollan, "History-Enriched Digital Objects: Prototypes and Policy Issues," *Information Society* **10**(2), 1994, pp. 139–145.

Hillis, W. Daniel, *The Connection Machine,* Cambridge, MA: MIT Press, 1985.

Hillis, W. Daniel, and Joshua Barnes, "Programming a Highly Parallel Computer," *Nature* **326**, March 5, 1987, pp. 27–30.

Hoc, Jean-Michel, *Cognitive Psychology of Planning,* translated by Constance Greenbaum, London: Academic Press, 1988. Originally published in French in 1987.

Hodges, Andrew, *Alan Turing: The Enigma,* London: Burnett, 1983.

Holland, John H., *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence,* Ann Arbor: University of Michigan Press, 1975.

Holmes, Walter G., *Applied Time and Motion Study,* New York: Ronald Press, 1938.

Holmqvist, Berit, and Peter Bogh Andersen, "Language, Perspectives and Design," in Joan Greenbaum and Morten Kyng, eds., *Design at Work: Cooperative Design of Computer Systems,* Hillsdale, NJ: Erlbaum, 1991, pp. 91–119.

Hoos, Ida R., *Systems Analysis in Public Policy: A Critique,* rev. ed., Berkeley: University of California Press, 1983.

Hopcroft, John E., and Dean B. Kraft, "The Challenge of Robotics for Computer Science," in Jacob T. Schwartz and Chee-Keng Yap, eds., *Algorithmic and Geometric Aspects of Robotics,* Vol. 1, Hillsdale, NJ: Erlbaum, 1987.

Horn, Berthold Klaus Paul, *Robot Vision,* Cambridge, MA: MIT Press, 1986.

Horswill, Ian, "Analysis of Adaptation and Environment," *Artificial Intelligence* 73(1–2), 1995, pp. 1–30.

Horswill, Ian D., and Rodney A. Brooks, "Situated Vision in a Dynamic World: Chasing Objects," in *Proceedings of the Seventh National Conference on Artificial Intelligence,* St. Paul, MN, 1988, pp. 796–800.

Hurlbert, Anya, and Tomaso Poggio, "Making Machines (and Artificial Intelligence) See," *Daedalus* 117(1), 1988, pp. 213–239.

Husserl, Edmund, *Cartesian Meditations: An Introduction to Phenomenology,* translated by Dorion Cairns, The Hague: Nijhoff, 1960. Originally published in German in 1929.

Husserl, Edmund, *The Crisis of European Sciences and Transcendental Phenomenology: An Introduction to Phenomenological Philosophy,* translated by David Carr, Evanston, IL: Northwestern University Press, 1970. Originally published in German in 1954.

Hutchins, Edwin, *Cognition in the Wild,* Cambridge, MA: MIT Press, 1995.

Ilyenkov, E. V., *Dialectical Logic: Essays on its History and Theory,* translated by H. Campbell Creighton, Moscow: Progress, 1977. Originally published in Russian in 1974.

Jaynes, Julian, "The Problem of Animate Motion in the Seventeenth Century," *Journal of the History of Ideas* 31, 1970, pp. 219–234.

Jirotka, Marina, and Joseph A. Goguen, eds., *Requirements Engineering: Social and Technical Issues,* San Diego, CA: Academic Press, 1994.

John-Steiner, Vera, *Notebooks of the Mind: Explorations of Thinking,* Albuquerque: University of New Mexico Press, 1985.

Johnson, Mark, *The Body in the Mind: The Bodily Basis of Meaning, Imagination, and Reason,* Chicago: University of Chicago Press, 1987.

Johnson, Michael P., Pattie Maes, and Trevor Darrell, "Evolving Visual Routines," *Artificial Life* 1(4), 1995, pp. 373–389.

Johnson-Laird, Philip N., "Procedural Semantics," *Cognition* 5(3), 1977, pp. 189–214.

Jordan, Michael I., and David A. Rosenbaum, "Action," in Michael I. Posner, ed., *Foundations of Cognitive Science,* Cambridge, MA: MIT Press, 1989, pp. 727–768.

Jordanova, Ludmilla J., "Introduction," in *Languages of Nature: Critical Essays on Science and Literature,* London: Free Association Books, 1986.

Joseph, Mathai, ed., *Real-Time Systems: Specification, Verification and Analysis,* Englewood Cliffs, NJ: Prentice Hall, 1996.

Kaelbling, Leslie Pack, "An Architecture for Intelligent Reactive Systems," in Michael P. Georgeff and Amy L. Lansky, eds., *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop at Timberline, OR,* Los Altos, CA: Morgan Kaufmann, 1987, pp. 395–410.

Kaplan, David, "Demonstratives," in Joseph Almog, John Perry, and Howard Wettstein, eds., *Themes from Kaplan,* New York: Oxford University Press, 1989, pp. 481–563. Originally circulated in 1977.

Keller, Evelyn Fox, *A Feeling for the Organism: The Life and Work of Barbara McClintock,* New York: Freeman, 1983.

Kempton, Willett, "Two Theories of Home Heat Control," *Cognitive Science* 10(1), 1986, pp. 75–90.

Kirsh, David, "Today the Earwig, Tomorrow Man?" *Artificial Intelligence* 47(1–3), 1991, pp. 161–184.

Kirsh, David, "The Intelligent Use of Space," *Artificial Intelligence* 73(1–2), 1995, pp. 31–68.

Kitano, Hiroaki, and James A. Hendler, *Massively Parallel Artificial Intelligence,* Cambridge, MA: MIT Press, 1994.

Klein, Jacob, *Greek Mathematical Thought and the Origin of Algebra,* translated by Eva Brann, Cambridge, MA: MIT Press, 1968.

Klein, Raymond, "Inhibitory Tagging System Facilitates Visual Search," *Nature* 334, August 4, 1988, pp. 430–431.

Knoblock, Craig A., "A Theory of Abstraction for Hierarchical Planning," in D. P. Benjamin, ed., *Change of Representation and Inductive Bias,* Boston: Kluwer, 1989.

Knoespel, Kenneth J., "The Narrative Matter of Mathematics: John Dee's Preface to the *Elements* of Euclid of Megara," *Philological Quarterly* 66(1), 1987, pp. 27–46.

Knuth, Donald, *The Art of Computer Programming,* Vol. 1: *Fundamental Algorithms,* Reading, MA: Addison-Wesley, 1983.

Koch, Christof, and Shimon Ullman, "Selecting One Among the Many: A Simple Network Implementing Shifts in Selective Visual Attention," *Human Neurobiology* 4(3), 1985, pp. 219–227.

Kolodner, Janet L., and Richard E. Cullingford, "Towards a Memory Architecture that Supports Reminding," in *Proceedings of the Eighth Annual Conference of the Cognitive Science Society,* Amherst, MA, Hillsdale, NJ: Erlbaum, 1986, pp. 467–477.

Kopytoff, Igor, "The Cultural Biography of Things: Commoditization as Process," in Arjun Appadurai, ed., *The Social Life of Things: Commodities in Cultural Perspective,* Cambridge University Press, 1986, pp. 64–91.

Kowalski, Robert A., "Predicate Logic as a Programming Language," *Proceedings of the IFIP Congress 74,* Amsterdam: North-Holland, 1974, pp. 569–574.

Kraft, Philip, *Programmers and Managers: The Routinization of Computer Programming in the United States,* New York: Springer-Verlag, 1977.

Kripke, Saul, "Semantical Considerations on Modal Logic," *Acta Philosophical Fennica* 16(1), 1963, pp. 83–94.

Kripke, Saul, *Naming and Necessity,* Cambridge, MA: Harvard University Press, 1980.

Kronfeld, Amichai, *Reference and Computation: An Essay in Applied Philosophy of Language,* Cambridge University Press, 1990.

Kuhn, Thomas S., *The Structure of Scientific Revolutions,* Chicago: University of Chicago Press, 1962.

Kuhn, Thomas S., "Metaphor in Science," in Andrew Ortony, ed., *Metaphor and Thought,* Cambridge University Press, 1979, pp. 409–419.

Laird, John E., and Allen Newell, "A Universal Weak Method: Summary of Results," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* Karlsruhe, West Germany, 1983, pp. 771–773.

Laird, John E., Allen Newell, and Paul S. Rosenbloom, "SOAR: An Architecture for General Intelligence," *Artificial Intelligence* 33(1), 1987, pp. 1–64.

Laird, John E., Paul S. Rosenbloom, and Allen Newell, "Towards Chunking as a General Learning Mechanism," in *Proceedings of the National Conference on Artificial Intelligence,* Menlo Park, CA: American Association for Artificial Intelligence, 1984, pp. 188–192.

Laird, John E., Paul S. Rosenbloom, and Allen Newell, *Universal Subgoaling and Chunk-*

ing: The Automatic Generation and Learning of Goal Hierarchies, Boston: Kluwer, 1986.

Lakoff, George, and Mark Johnson, *Metaphors We Live By,* Chicago: University of Chicago Press, 1980.

Lange, Trent E., and Michael G. Dyer, "High-Level Inferencing in a Connectionist Network," *Connection Science* 1(2), 1989, pp. 181–217.

Lansky, Amy L., "Localized Event-Based Reasoning for Multiagent Domains," *Computational Intelligence* 4(4), 1988, pp. 319–340.

Lansky, Amy L., and David S. Fogelsong, "Localized Representations and Planning Methods for Parallel Domains," in *Proceedings of the Sixth National Conference on Artificial Intelligence,* Menlo Park, CA: AAAI Press, 1987, pp. 240–245.

Larkin, Jill, and Herbert A. Simon, "Why a Diagram Is (Sometimes) Worth Ten Thousand Words," *Cognitive Science* 11(1), 1987, pp. 65–100.

Lashley, Karl S., "The Problem of Serial Order in Behavior," in Lloyd A. Jeffress, ed., *Cerebral Mechanisms in Behavior: The Hixon Symposium,* New York: Wiley, 1951, pp. 112–146.

Latour, Bruno, "Visualization and Cognition: Thinking with Eyes and Hands," *Knowledge and Society: Studies in the Sociology of Culture Past and Present* 6, 1986, pp. 1–40.

Latour, Bruno, and Steve Woolgar, *Laboratory Life: The Construction of Scientific Facts,* Princeton, NJ: Princeton University Press, 1986. Originally published in 1979.

Lave, Jean, *Cognition in Practice: Mind, Mathematics, and Culture in Everyday Life,* Cambridge University Press, 1988.

Lave, Jean, and Etienne Wenger, *Situated Learning: Legitimate Peripheral Participation,* Cambridge University Press, 1991.

Layton, Edwin T., Jr., *The Revolt of the Engineers: Social Responsibility and the American Engineering Profession,* Cleveland: Case Western University Press, 1971.

Leary, David E., ed., *Metaphor in the History of Psychology,* Cambridge University Press, 1990.

Leder, Drew, *The Absent Body,* Chicago: University of Chicago Press, 1990.

Leith, Philip, "Fundamental Errors in Legal Logic Programming," *Computer Journal* 29(6), 1986, pp. 545–552.

Leith, Philip, "Involvement, Detachment and Programming: The Belief in PROLOG," in Brian P. Bloomfield, ed., *The Question of Artificial Intelligence: Philosophical and Sociological Perspectives,* London: Croom Helm, 1987, pp. 220–257.

Leont'ev, A. N., *Problems of the Development of the Mind,* Moscow: Progress, 1981.

Lespérance, Yves, *A Formal Theory of Indexical Knowledge and Action,* Ph.D. dissertation, Department of Computer Science, University of Toronto, 1991.

Lespérance, Yves, and Hector J. Levesque, "Indexical Knowledge and Robot Action: A Logical Account," *Artificial Intelligence* 73(1–2), 1995, pp. 69–115.

Lighthill, James, "Artificial Intelligence: A General Survey," in *Artificial Intelligence: A Paper Symposium,* London: Science Research Council, 1973, pp. 38–45.

Lilienfeld, Robert, *The Rise of Systems Theory: An Ideological Analysis,* New York: Wiley, 1978.

Lucy, John, *Language Diversity and Thought: A Reformulation of the Linguistic Relativity Hypothesis,* Cambridge University Press, 1992.

Lugowski, Marek W., "Why Artificial Intelligence is Necessarily Ad Hoc: One's

Thinking/Approach/Model/Solution Rides on One's Metaphors," Technical Report 176, Department of Computer Science, University of Indiana, 1985.

Lynch, Michael, *Art and Artifact in Laboratory Science: A Study of Shop Work and Shop Talk in a Research Laboratory,* London: Routledge & Kegan Paul, 1985.

Lynch, Michael, "The Externalized Retina: Selection and Mathematization in the Visual Documentation of Objects in the Life Sciences," *Human Studies* 11(2), 1988, pp. 201–234.

Lyons, Damian M., and Michael A. Arbib, "A Formal Model of Computation for Sensory-Based Robotics," *IEEE Transactions on Robotics and Automation* 5(3), 1989, pp. 280–293.

Lyons, Damian M., and A.J. Hendriks, "Exploiting Patterns of Interaction to Achieve Reactive Behavior," *Artificial Intelligence* 73(1–2), 1995, pp. 117–148.

Mackenzie, Ann Wilbur, "A Word about Descartes' Mechanistic Conception of Life," *Journal of the History of Biology* 8(1), 1975, pp. 1–13.

Maes, Pattie, ed., *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back,* Cambridge, MA: MIT Press, 1990.

Mahoney, James V., *Image Chunking: Defining Spatial Building Blocks for Scene Analysis,* Technical Report 980, MIT Artificial Intelligence Laboratory, 1987.

Mahoney, Michael S., "The History of the Computer in the History of Technology," *Annals of the History of Computing* 10(2), 1977, pp. 113–125.

March, James G., Lee S. Sproull, and Michal Tamuz, "Learning From Samples of One or Fewer," *Organization Science* 2(1), 1991, pp. 1–13.

Markley, Robert, *Fallen Languages: Crises of Representation in Newtonian England, 1660–1740,* Ithaca, NY: Cornell University Press, 1993.

Markus, Gyorgy, "Why Is There No Hermeneutics of Natural Sciences? Some Preliminary Theses," *Science in Context* 1(1), 1987, pp. 5–51.

Marr, David, "A Theory for Cerebral Neocortex," *Proceedings of the Royal Society of London B* 176, 1970, pp. 161–234.

Marr, David, *Vision,* San Francisco: Freeman, 1982.

Martin, Emily, *The Woman in the Body: A Cultural Analysis of Reproduction,* Boston: Beacon Press, 1987.

Martin, Janet, and Rom Harré, "Metaphor in Science," in David S. Miall, *Metaphor: Problems and Perspectives,* Brighton: Harvester, 1982, pp. 89–105.

Maturana, Humberto R., and Francisco J. Varela, *The Tree of Knowledge: The Biological Roots of Human Understanding,* Boston: Shambhala, 1988.

Maunsell, John H. R., and William T. Newsome, "Visual Processing in Monkey Extrastriate Cortex," *Annual Review of Neuroscience* 10, 1987, pp. 363–401.

McAllester, David, *ONTIC: A Knowledge Representation System for Mathematics,* Cambridge, MA: MIT Press, 1988.

McAllester, David, and David Rosenblitt, "Systematic Nonlinear Planning," in *Proceedings of the National Conference on Artificial Intelligence,* Los Altos, CA: Morgan Kaufmann, 1991, pp. 634–639.

McCarthy, John, "Programs with Common Sense," in Marvin Minsky, ed., *Semantic Information Processing,* Cambridge, MA: MIT Press, 1968, pp. 403–418. Originally published in 1958.

McCarthy, John, and Patrick J. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence," in Bernard Meltzer and Donald Michie, eds., *Machine Intelligence* 4, Edinburgh: Edinburgh University Press, 1969, pp. 463–502.

McClamrock, Ron, *Existential Cognition: Computational Minds in the World,* Chicago: University of Chicago Press, 1995.

McCorduck, Pamela, *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence,* San Francisco: Freeman, 1979.

McCulloch, Gregory, *The Mind and Its World,* London: Routledge, 1995.

McCulloch, Warren S., and Walter H. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," in Warren S. McCulloch, *Embodiments of Mind,* Cambridge, MA: MIT Press, 1965, pp. 19–39. Originally published in *Bulletin of Mathematical Biophysics* 5(1), 1943, pp. 115–133.

McDermott, Drew, "Planning and Acting," *Cognitive Science* 2(2), 1978, pp. 71–109.

McDermott, Drew, "Artificial Intelligence Meets Natural Stupidity," in John Haugeland, ed., *Mind Design: Philosophy, Psychology, Artificial Intelligence,* Cambridge, MA: MIT Press, 1981, pp. 143–160.

McDermott, Drew, "A General Framework for Reason Maintenance," *Artificial Intelligence* 50(3), 1991, pp. 289–329.

McDermott, Drew, "Robot Planning," *AI Magazine* 13(2), 1992, pp. 55–79.

McDermott, Drew, "Planning: What It Is, What It Could Be," *Artificial Intelligence* 76(1–2), 1995, pp. 1–16.

McReynolds, Paul, "The Clock Metaphor in the History of Psychology," in Thomas Nickles, ed., *Scientific Discovery: Case Studies,* Dordrecht: Reidel, 1980, pp. 97–112.

Mead, George Herbert, *Mind, Self, and Society from the Standpoint of a Social Behaviorist,* Chicago: University of Chicago Press, 1934.

Mel, Bartlett W., *Connectionist Robot Motion Planning: A Neurally-Inspired Approach to Visually-Guided Reaching,* San Diego, CA: Academic Press, 1990.

Merleau-Ponty, Maurice, *Phenomenology of Perception,* translated by Colin Smith, New York: Humanities Press, 1962. Originally published in French in 1945.

Meyer, Jean-Arcady, and Stewart W. Wilson, eds., *From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior,* Cambridge, MA: MIT Press, 1991.

Miller, George, "Toward a Third Metaphor for Psycholinguistics," in Walter B. Weimer and David S. Palermo, eds., *Cognition and the Symbolic Processes,* Hillsdale, NJ: Erlbaum, 1974, pp. 397–414.

Miller, George A., Eugene Galanter, and Karl H. Pribram, *Plans and the Structure of Behavior,* New York: Holt, 1960.

Miller, Jonathan, *States of Mind,* New York: Pantheon, 1983.

Minsky, Marvin, "Frame-System Theory," in Philip N. Johnson-Laird and Peter C. Wason, eds., *Thinking: Readings in Cognitive Science,* Cambridge University Press, 1977, pp. 355–376.

Minsky, Marvin, "K-Lines: A Theory of Memory," *Cognitive Science* 4(2), 1980, pp. 117–133.

Minsky, Marvin, *The Society of Mind,* New York: Simon & Schuster, 1985.

Mirowski, Philip, *More Heat Than Light: Economics as Social Physics, Physics as Nature's Economics,* Cambridge University Press, 1989.

Moerman, Michael, *Talking Culture: Ethnography and Conversation Analysis,* Philadephia: University of Pennsylvania Press, 1988.

Montgomery, David, *The Fall of the House of Labor: The Workplace, the State and American Labor Activism, 1865–1925,* Cambridge University Press, 1987.

Montgomery, Scott L., "Reflections on Language in Science," *Science as Culture* 6, 1989, pp. 42–77.

Munson, John H., "Robot Planning, Execution, and Monitoring in an Uncertain Environment," in *Proceedings of the Second International Joint Conference on Artificial Intelligence,* London, 1971, pp. 338–349.

Nardi, Bonnie A., ed., *Context and Consciousness: Activity Theory and Human–Computer Interaction,* Cambridge, MA: MIT Press, 1996.

Neches, Robert, *Models of Heuristic Procedure Modification,* Ph.D. dissertation, Department of Psychology, Carnegie-Mellon University, 1981.

Neisser, Ulric, "Computers as Tools and as Metaphors," in Charles R. Dechert, *The Social Impact of Cybernetics,* Notre Dame, IN: University of Notre Dame Press, 1966.

Newell, Allen, ed., *Information Processing Language V Manual,* Englewood Cliffs, NJ: Prentice-Hall, 1961.

Newell, Allen, "HARPY, Production Systems, and Human Cognition," in Ronald A. Cole, ed., *Perception and Production of Fluent Speech,* Hillsdale, NJ: Erlbaum, 1980, pp. 289–380.

Newell, Allen, "Intellectual Issues in the History of Artificial Intelligence," in Fritz Machlup and Una Mansfield, eds., *The Study of Information: Interdisciplinary Messages,* New York: Wiley, 1983, pp. 187–227.

Newell, Allen, *Unified Theories of Cognition,* Cambridge, MA: Harvard University Press, 1990.

Newell, Allen, J. C. Shaw, and Herbert A. Simon, "Report on a General Problem-Solving Program," in *Proceedings of the International Conference on Information Processing,* Paris, 1960, pp. 256–264.

Newell, Allen, and Herbert A. Simon, "GPS, A Program that Simulates Human Thought," in Edward A. Feigenbaum and Julian Feldman, eds., *Computers and Thought,* New York: McGraw-Hill, 1963, pp. 279–293. Originally published in 1961.

Newell, Allen, and Herbert A. Simon, *Human Problem Solving,* Englewood Cliffs, NJ: Prentice-Hall, 1972.

Newman, Denis, Peg Griffin, and Michael Cole, *The Construction Zone: Working for Cognitive Change in School,* Cambridge University Press, 1989.

Nilsson, Nils J., "Teleo-Reactive Programs for Agent Control," *Journal of Artificial Intelligence Research* 1, 1994, pp. 139–158.

Noble, David F., *America by Design: Science, Technology, and the Rise of Corporate Capitalism,* Oxford: Oxford University Press, 1977.

Norman, Donald A., "Reflections on Cognition and Parallel Distributed Processing," in James L. McClelland and David E. Rumelhart, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* Vol. 2, Cambridge, MA: MIT Press, 1986, pp. 531–546.

Norman, Donald A., *The Psychology of Everyday Things,* New York: Basic Books, 1988.

Nye, Andrea, *Words of Power: A Feminist Reading of the History of Logic,* London: Routledge, 1990.

Olafson, Frederick A., *Heidegger and the Philosophy of Mind,* New Haven, CT: Yale University Press, 1987.

Ong, Walter J., *Ramus, Method, and the Decay of Dialogue: From the Art of Discourse to the Art of Reason,* Cambridge, MA: Harvard University Press, 1983. Originally published in 1958.

Onians, Richard Broxton, *The Origins of European Thought About the Body, the Mind, the Soul, the World, Time, and Fate: New Interpretations of Greek, Roman, and Kindred Evidence, Also of Some Basic Jewish and Christian Beliefs,* Cambridge University Press, 1954.

Ortner, Sherry B., "Theory in Anthropology Since the Sixties," *Comparative Studies in Society and History* 26(1), 1984, pp. 126–166.

Pea, Roy D., "Socializing the Knowledge Transfer Problem," *International Journal of Education Research* 11(6), 1987, pp. 639–663.

Pepper, Stephan C., *World Hypotheses: A Study in Evidence,* Berkeley: University of California Press, 1961.

Perelman, Chaim, and Lucie Olbrechts-Tyteca, *The New Rhetoric: A Treatise on Argumentation,* translated by John Wilkinson and Purcell Weaver, Notre Dame, IN: University of Notre Dame Press, 1969. Originally published in French in 1958.

Perry, John, "Frege on Demonstratives," *Philosophical Review* 86(4), 1977, pp. 474–497.

Pervin, Lawrence A., "Theoretical Approaches to the Analysis of Individual–Environment Interaction," in Lawrence A. Pervin and Michael Lewis, eds., *Perspectives in Interactional Psychology,* New York: Plenum Press, 1978, pp. 67–85.

Pervin, Lawrence A., and Michael Lewis, "Overview of the Internal–External Issue," in Lawrence A. Pervin and Michael Lewis, eds., *Perspectives in Interactional Psychology,* New York: Plenum Press, 1978, pp. 67–86.

Pickering, Andrew, *Constructing Quarks: A Sociological History of Particle Physics,* Edinburgh: Edinburgh University Press, 1984.

Pinker, Steven, and Alan Prince, "On Language and Connectionism: Analysis of a Parallel Distributed Processing Model of Language Acquisition," *Cognition* 28(1), 1988, pp. 73–194.

Pollack, Martha E., "The Uses of Plans," *Artificial Intelligence* 57(1), 1992, pp. 43–68.

Popper, Karl R., and John C. Eccles, *The Self and Its Brain,* Berlin: Springer-Verlag, 1977.

Preston, Elizabeth F., *Representational and Non-Representational Intentionality: Husserl, Heidegger, and Artificial Intelligence,* Ph.D. dissertation, Department of Philosophy, Boston University, 1988.

Preston, Elizabeth F., "Heidegger and Artificial Intelligence," *Philosophy and Phenomenological Research* 53(1), 1993, pp. 43–69.

Provan, Gregory, "Efficiency Analysis of Multiple-Context TMSs in Scene Representation," in *Proceedings of the Sixth National Conference on Artificial Intelligence,* Seattle, 1987, pp. 173–177.

Putnam, Hilary, "The Meaning of 'Meaning'," in *Mind, Language, and Reality: Philosophical Papers,* Vol. 2, Cambridge University Press, 1975a, pp. 215–271.

Putnam, Hilary, "Minds and Machines," in *Mind, Language, and Reality: Philosophical*

Papers, Vol. 2, Cambridge: Cambridge University Press, 1975b, pp. 362–385. Originally published in 1960.

Pylyshyn, Zenon W., *Computation and Cognition: Toward a Foundation for Cognitive Science,* Cambridge, MA: MIT Press, 1984.

Pylyshyn, Zenon W., ed., *The Robot's Dilemma: The Frame Problem in Artificial Intelligence,* Norwood, NJ: Ablex, 1987.

Quillian, M. Ross, "Semantic Memory," in Marvin Minsky, ed., *Semantic Information Processing,* Cambridge, MA: MIT Press, 1968, pp. 227–270.

Radin, George, "The 801 Minicomputer," *IBM Journal of Research and Development* 27(3), 1983, pp. 237–246.

Raibert, Marc H., "Running with Symmetry," *International Journal of Robotics Research* 5(4), 1986, pp. 3–19.

Rescher, Nicholas, *Dialectics: A Controversy-Oriented Approach to the Theory of Knowledge,* Albany, NY: SUNY Press, 1977.

Ricoeur, Paul, *The Rule of Metaphor: Multi-Disciplinary Studies of the Creation of Meaning in Language,* translated by Robert Czerny, Buffalo, NY: University of Toronto Press, 1977.

Rodis-Lewis, Geneviève, "Limitations of the Mechanical Model in the Cartesian Conception of the Organism," in Michael Hooker, ed., *Descartes: Critical and Interpretive Essays,* Baltimore: Johns Hopkins University Press, 1978, pp. 152–170.

Rogoff, Barbara, *Apprenticeship in Thinking: Cognitive Development in Social Context,* Oxford: Oxford University Press, 1989.

Rogoff, Barbara, and James V. Wertsch, eds., *Children's Learning in the "Zone of Proximal Development,"* San Francisco: Jossey-Bass, 1984.

Rorty, Richard, *Philosophy and the Mirror of Nature,* Princeton, NJ: Princeton University Press, 1979.

Rosenbloom, Paul S., *The Chunking of Goal Hierarchies: A Model of Practice and Stimulus–Response Compatibility,* Ph.D. dissertation, Department of Computer Science, Carnegie-Mellon University, August 1983.

Rosenschein, Stanley J., and Leslie Pack Kaelbling, "The Synthesis of Digital Machines with Provable Epistemic Properties," in Joseph Halpern, ed., *Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge,* Los Altos, CA: Morgan Kaufmann, 1986, pp. 83–98.

Rosenschein, Stanley J., and Leslie Pack Kaelbling, "A Situated View of Representation and Control," *Artificial Intelligence* 73(1–2), 1995, pp. 149–173.

Roszak, Theodore, *The Cult of Information: The Folklore of Computers and the True Art of Thinking,* New York: Pantheon, 1986.

Rumelhart, David E., and James L. McClelland, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* Vol. 1, Cambridge, MA: MIT Press, 1986.

Rumelhart, David E., Paul Smolensky, James L. McClelland, and Geoffrey E. Hinton, "Schemata and Sequential Thought Processes in PDP Models," in James L. McClelland and David E. Rumelhart, eds., *Parallel Distributed Processing: Exploration in the Microstructure of Cognition,* Vol. 2, Cambridge: MIT Press, 1986, pp. 7–57.

Sahlins, Marshall, *Historical Metaphors and Mythical Realities: Structure in the Early History of the Sandwich Islands Kingdom,* Ann Arbor: University of Michigan Press, 1981.

Sanborn, James C., and James A. Hendler, "A Model of Reaction for Planning in Dynamic Environments," *International Journal of Artificial Intelligence in Engineering* 3(2), 1988, pp. 95–102.

Sandewall, Erik, *Features and Fluents: The Representation of Knowledge About Dynamical Systems,* Oxford: Oxford University Press, 1994.

Sartre, Jean-Paul, *Being and Nothingness,* translated by Hazel Barnes, New York: Philosophical Library, 1956.

Schank, Roger C., *Dynamic Memory: A Theory of Reminding and Learning in Computers and People,* Cambridge University Press, 1982.

Schank, Roger C., and Robert P. Abelson, *Scripts, Plans, Goals, and Understanding,* Hillsdale, NJ: Erlbaum, 1977.

Scher, Bob, *The Fear of Cooking,* Boston: Houghton-Mifflin, 1984.

Schiller, Dan, "From Culture to Information and Back Again: Commoditization as a Route to Knowledge," *Critical Studies in Mass Communication* 11(1), 1994, pp. 93–115.

Schön, Donald A., *Displacement of Concepts,* London: Tavistock, 1963.

Schön, Donald A., "Generative Metaphor: A Perspective on Problem-Setting in Social Policy," in Andrew Ortony, ed., *Metaphor and Thought,* Cambridge University Press, 1979, pp. 137–163.

Schön, Donald A., *The Reflective Practitioner: How Professionals Think in Action,* New York: Basic Books, 1983.

Schopman, Joop, "Frames of Artificial Intelligence," in Brian P. Bloomfield, ed., *The Question of Artificial Intelligence: Philosophical and Sociological Perspectives,* London: Croom Helm, 1987, pp. 165–219.

Schoppers, Marcel J., "Universal Plans for Reactive Robots in Unpredictable Environments," in *Proceedings of the Tenth International Joint Conference on Artificial Intelligence,* Milan, 1987, pp. 1039–1046.

Schoppers, Marcel J., "The Use of Dynamics in an Intelligent Controller for a Space Faring Rescue Robot," *Artificial Intelligence* 73(1–2), 1995, pp. 175–230.

Searle, John R., "The Intentionality of Intention and Action," *Cognitive Science* 4(1), 1980, pp. 47–70.

Searle, John R., "Minds, Brains, and Programs," in John Haugeland, ed., *Mind Design: Philosophy, Psychology, Artificial Intelligence,* Cambridge, MA: MIT Press, 1981, pp. 282–306.

Searle, John R., *Minds, Brains, and Science,* Cambridge, MA: Harvard University Press, 1984.

Searle, John R., "Notes on Conversation," in Donald G. Ellis and William A. Donohue, eds., *Contemporary Issues in Language and Discourse Processes,* 1986, pp. 7–19.

Searle, John R., *The Rediscovery of the Mind,* Cambridge, MA: MIT Press, 1992.

Sengers, Phoebe, "From the Belly of the Devil: Critical Theory in Scientific Practice," *Parallax* 2, 1996, pp. 151–159.

Shore, John, *The Sachertorte Algorithm and Other Antidotes to Computer Anxiety,* New York: Viking, 1975.

Silverstein, Michael, "Shifters, Linguistic Categories, and Cultural Description," in Keith H. Basso and Henry A. Selby Jr., eds., *Meaning in Anthropology,* Albuquerque: University of New Mexico Press, 1976, pp. 11–55.

Silverstein, Michael, "Language Structure and Linguistic Ideology," in Paul R. Clyne, William F. Hanks, and Carol L. Hofbauer, eds., *The Elements: A Parasession on Linguistic Units and Levels,* Chicago: Chicago Linguistic Society, 1979, pp. 193–247.

Simmons, Reid G., "The Roles of Associational and Causal Reasoning in Problem Solving," *Artificial Intelligence* 53(2–3), 1992, pp. 159–208.

Simon, Herbert A., *The Sciences of the Artificial,* Cambridge, MA: MIT Press, 1969.

Simon, Herbert A., *Models of My Life,* New York: Basic Books, 1991.

Simsion, Graeme C., *Data Modeling Essentials: Analysis, Design, and Innovation,* New York: Van Nostrand Reinhold, 1994.

Singer, Dorothy G., and Tracey A. Revenson, *A Piaget Primer: How a Child Thinks,* New York: International Universities Press, 1978.

Singley, Mark K., and John R. Anderson, *The Transfer of Cognitive Skill,* Cambridge, MA: Harvard University Press, 1989.

Smith, Brian C., "Prologue to *Reflection and Semantics in a Procedural Language,*" in Ronald J. Brachman and Hector J. Levesque, eds., *Readings in Knowledge Representation,* Los Altos, CA: Morgan Kaufmann, 1985, pp. 31–40.

Smith, Brian C., "Varieties of Self-Reference," in Joseph Halpern, ed., *Proceedings of the Conference on Theoretical Aspects of Reasoning About Knowledge,* Los Altos, CA: Morgan Kaufmann, 1986, pp. 19–43.

Smith, Brian C., "The Correspondence Continuum," Report CSLI–87–71, Menlo Park, CA: Center for the Study of Language and Information, 1987. Originally in *Proceedings of the Sixth Canadian AI Conference,* Montreal, 1986.

Smith, Brian C., "The Limits of Correctness," in Timothy R. Colburn, James H. Fetzer, and Terry L. Rankin, eds., *Program Verification,* Dordrecht: Kluwer, 1993, pp. 275–293. Originally published as "Limits of Correctness in Computers," Report CSLI–85–36, Stanford, CA: Center for the Study of Language and Information, Stanford University, 1985.

Smith, Brian C., *On the Origin of Objects,* Cambridge, MA: MIT Press, 1996.

Smolensky, Paul, *On Variable Binding and Representation of Symbolic Structure,* Technical Report, University of Colorado at Boulder, 1987.

Spinosa, Charles, "Derrida and Heidegger: Iterability and *Ereignis,*" in Hubert L. Dreyfus and Harrison Hall, eds., *Heidegger: A Critical Reader,* Oxford: Oxford University Press, 1992, pp. 270–297.

Stallman, Richard M., and Gerald Jay Sussman, "Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis," *Artificial Intelligence* 9(2), 1977, pp. 135–196.

Staudenmaier, John M., *Technology's Storytellers: Reweaving the Human Fabric,* Cambridge, MA: MIT Press, 1985.

Steels, Luc, "The Artificial Life Roots of Artificial Intelligence," *Artificial Life* 1(1–2), 1993, pp. 75–110.

Stefik, Mark, "Planning with Constraints (MOLGEN: Part 1)," *Artificial Intelligence* 16(2), 1981, pp. 111–140.

Sternberg, Robert J., *Metaphors of the Mind: Conceptions of the Nature of Intelligence,* Cambridge University Press, 1990.

Subramanian, Devika, and John Woodfill, "Making Situation Calculus Indexical," in

Hector J. Levesque and Ronald J. Brachman, eds., *Proceedings of the First International Conference on Principles of Knowledge Representation,* Los Altos, CA: Morgan Kaufmann, 1989a, pp. 467–474.

Subramanian, Devika, and John Woodfill, "Subjective Ontologies," in *Proceedings of the AAAI Spring Symposium on Limited Rationality,* Stanford University, 1989b.

Suchman, Lucy, "Office Procedures as Practical Action: Models of Work and System Design," *ACM Transactions on Office Information Systems* 1(4), 1983, pp. 320–328.

Suchman, Lucy A., *Plans and Situated Actions: The Problem of Human-Machine Communication,* Cambridge University Press, 1987.

Suchman, Lucy A., and Randall H. Trigg, "Artificial Intelligence as Craftwork," in Seth Chaiklin and Jean Lave, eds., *Understanding Practice: Perspectives on Activity and Context,* Cambridge University Press, 1993, pp. 144–178.

Sudnow, David, *Ways of the Hand: The Organization of Improvised Conduct,* Cambridge, MA: Harvard University Press, 1978.

Sullivan, Harry Stack, *The Interpersonal Theory of Psychiatry,* New York: Norton, 1953.

Sun, Ron, "On Variable Binding in Connectionist Networks," *Connection Science* 4(2), 1992, pp. 93–124.

Sun, Ron, "Logics and Variables in Connectionist Models: A Brief Overview," in Leonard Uhr and Vasant Honavar, eds., *Artificial Intelligence and Neural Networks: Steps Toward Principled Integration,* Boston: Academic Press, 1994, pp. 301–320.

Sussman, Gerald Jay, *A Computer Model of Skill Acquisition,* New York: Elsevier, 1975.

Sussman, Gerald Jay, and Drew McDermott, "Why Conniving Is Better than Planning," MIT AI Lab Memo 255A, 1972. Also in *Proceedings of the Fall Joint Computer Conference* 41, Montvale, NJ: AFIPS Press, 1972, pp. 1171–1179.

Tambe, Milind, and Allen Newell, "Some Chunks are Expensive," *Proceedings of the Fifth International Conference on Machine Learning,* Ann Arbor, MI, 1988, pp. 451–458.

Tambe, Milind, and Paul S. Rosenbloom, "Investigating Production System Representations for Non-Combinatorial Match," *Artificial Intelligence* 68(1), 1994, pp. 155–199.

Tanz, Christine, *Studies in the Acquisition of Deictic Terms,* Cambridge: Cambridge University Press, 1980.

Tarjan, Robert, "Algorithm Design," *Communications of the ACM* 30(3), 1987, pp. 205–212.

Tate, Austin, "Interacting Goals and Their Use," in *Advance Papers of the Fourth International Joint Conference on Artificial Intelligence,* Tbilisi, USSR, 1975, pp. 215–218.

Taylor, Charles, *Human Agency and Language: Philosophical Papers,* Vol. 1, Cambridge University Press, 1985.

Taylor, Frederick, *The Principles of Scientific Management,* New York: Harper, 1911.

Tenenbaum, Jay M., "On Locating Objects by Their Distinguishing Features in Multisensory Images," *Computer Graphics and Image Processing* 2(3–4), 1973, pp. 308–320.

Thorndike, E. L., and R. S. Woodworth, "The Influence of Improvement in One Mental Function upon the Efficiency of Other Functions," *Psychological Review, I,* 8(3), 1901, pp. 247–261; *II.* "The Estimation of Magnitudes," 8(4), 1901, pp. 384–395; *III.* "Functions Involving Attention, Observation, and Discrimination," 8(6), 1901, pp. 553–564.

Toth, Jozsef A., "Review of Kenneth Ford and Patrick Hayes, eds., *Reasoning Agents in a*

Dynamic World: The Frame Problem," *Artificial Intelligence* 73(1–2), 1995, pp. 323–369.

Toulmin, Stephen E., *Cosmopolis: The Hidden Agenda of Modernity,* New York: Free Press, 1990.

Touretzky, David S., and Geoffrey E. Hinton, "A Distributed Connectionist Production System," *Cognitive Science* 12(3), 1988, pp. 423–466.

Truesdell, Clifford A., "The Computer: Ruin of Science and Threat to Mankind," in *An Idiot's Fugitive Essays,* New York: Springer-Verlag, 1984, pp. 594–631.

Turbayne, Colin Murray, *The Myth of Metaphor,* rev. ed., Columbia: University of South Carolina Press, 1970.

Turbayne, Colin Murray, *Metaphors for the Mind: The Creative Mind and Its Origins,* Columbia: University of South Carolina Press, 1991.

Turing, Alan M., "On Computable Numbers, with an Application to the *Entscheidungsproblem,*" *Proceedings of the London Mathematics Society,* 2nd series, 42, London: Hodgson, 1936, pp. 230–265.

Turkle, Sherry, *The Second Self: Computers and the Human Spirit,* New York: Simon & Schuster, 1984.

Ullman, Shimon, "Visual Routines," *Cognition* 18(1), 1984, pp. 97–159.

van Caneghem, Michel, and David H. D. Warren, eds., *Logic Programming and Its Applications,* Norwood, NJ: Ablex, 1986.

van Gelder, Tim, "Defining 'Distributed Representation'," *Connection Science* 4(3–4), 1992, pp. 175–191.

VanLehn, Kurt, *Mind Bugs: The Origins of Procedural Misconceptions,* Cambridge, MA: MIT Press, 1990.

Varela, Francisco J., and Paul Bourgine, eds, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life,* Cambridge, MA: MIT Press, 1992.

Varela, Francisco J., Evan Thompson, and Eleanor Rosch, *The Embodied Mind: Cognitive Science and Human Experience,* Cambridge, MA: MIT Press, 1991.

Vico, Giambattista, *The New Science of Giambattista Vico,* rev. translation of the 3rd ed. (1774) by Thomas Goddard Bergin and Max Harold Fisch, Ithaca, NY: Cornell University Press, 1968.

Vygotsky, L. S., *Mind in Society: The Development of Higher Psychological Processes,* edited by Michael Cole, Vera John-Steiner, Sylvia Scribner, and Ellen Souberman, Cambridge, MA: Harvard University Press, 1978. Originally published in Russian in 1934.

Warren, Scott, *The Emergence of Dialectical Theory: Philosophy and Political Inquiry,* Chicago: University of Chicago Press, 1984.

Webber, Bonnie, Norman Badler, Barbara Di Eugenio, Chris Geib, Libby Levison, and Michael Moore, "Instructions, Intentions and Expectations," *Artificial Intelligence* 73(1–2), 1995, pp. 253–269.

Weizenbaum, Joseph, *Computer Power and Human Reason: From Judgement to Calculation,* San Francisco: Freeman, 1976.

Weld, Daniel S., "An Introduction to Least Commitment Planning," *AI Magazine* 15(4), 1994, pp. 27–61.

Wertsch, James W., ed., *Culture, Communication, and Cognition: Vygotskian Perspectives,* Cambridge University Press, 1985.

West, David M., and Larry E. Travis, "The Computational Metaphor and Artificial Intelligence: A Reflective Examination of a Theoretical Falsework," *AI Magazine* 12(1), 1991a, pp. 64–79.

West, David M., and Larry E. Travis, "From Society to Landscape: Alternative Metaphors for Artificial Intelligence," *AI Magazine* 12(2), 1991b, pp. 69–83.

Wheelwright, Philip Ellis, *Metaphor and Reality,* Bloomington: Indiana University Press, 1962.

Whitehead, Steven D., and Dana H. Ballard, "Learning to Perceive and Act by Trial and Error," *Machine Learning* 7(1), 1991, pp. 7–35.

Whorf, Benjamin Lee, "The Relation of Habitual Thought and Behavior to Language," in *Language, Thought, and Reality,* Cambridge, MA: MIT Press, 1956, pp. 134–159. Originally published in 1941.

Wiener, Norbert, *Cybernetics: or Control and Communication in the Animal and the Machine,* Cambridge, MA: MIT Press, 1948.

Wilensky, Robert, *Planning and Understanding: A Computational Approach to Human Reasoning,* Reading, MA: Addison-Wesley, 1983.

Wilkins, David E., *Practical Planning: Extending the Classical AI Planning Paradigm,* Los Altos, CA: Morgan Kaufmann, 1988.

Winnicott, Donald W., "Mind and Its Relation to the Psyche-Soma," in *Through Paediatrics to Psycho-Analysis,* New York: Basic Books, 1975a, pp. 243–254. Originally published in 1949.

Winnicott, Donald W., "Transitional Objects and Transitional Phenomena," in *Through Paediatrics to Psycho-Analysis,* New York: Basic Books, 1975b, pp. 229–242. Originally published in 1951.

Winograd, Terry, *Understanding Natural Language,* New York: Academic Press, 1972.

Winograd, Terry, "Moving the Semantic Fulcrum," *Linguistics and Philosophy* 8(1), 1985, pp. 91–104.

Winograd, Terry, "Heidegger and the Design of Computer Systems," in Andrew Feenberg and Alastair Hannay, eds., *Technology and the Politics of Knowledge,* Bloomington: Indiana University Press, 1995, pp. 108–127.

Winograd, Terry, and Fernando Flores, *Understanding Computers and Cognition: A New Foundation for Design,* Norwood, NJ: Ablex, 1986.

Winston, Patrick H., *The Psychology of Computer Vision,* New York: McGraw-Hill, 1975.

Winston, Patrick H., *Artificial Intelligence,* 2nd ed., Reading, Mass: Addison-Wesley, 1984.

Winston, Patrick H., Thomas O. Binford, Boris Katz, and Michael Lowry, "Learning Physical Descriptions from Functional Definitions, Examples, and Precedents," in *Proceedings of the National Conference on Artificial Intelligence,* Austin, TX, 1983, pp. 433–439.

Wittgenstein, Ludwig, *Philosophical Investigations,* 3rd ed., translated by G. E. M. Anscombe, New York: Macmillan, 1968. Originally published in 1953.

Woods, William A., "What's in a Link?" in Daniel G. Bobrow and Allan Collins, eds., *Representation and Understanding: Studies in Cognitive Science,* New York: Academic Press, 1975, pp. 35–82.

Woolard, Katherine, and Bambi B. Schieffelin, "Language Ideology," in William H. Durham, E. Valentine Daniel, and Bambi Schieffelin, eds., *Annual Review of Anthropology* 23, 1994, pp. 55–82.

Woolgar, Steve, "Why Not a Sociology of Machines? The Case of Sociology and Artificial Intelligence," *Sociology* 19(3), 1985, pp. 557–572.

Woolgar, Steve, "Reconstructing Man and Machine: A Note on the Sociological Critiques of Cognitivism," in Wiebe E. Bijker, Thomas P. Hughes, and Trevor J. Pinch, eds., *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology,* Cambridge, MA: MIT Press, 1987, pp. 311–328.

Woolgar, Steve, "Re Thinking Agency: New Moves in Science and Technology Studies," *Mexical Journal of Behavior Analysis* **20**, 1994, pp. 213–240.

Young, Robert M., *Darwin's Metaphor: Nature's Place in Victorian Culture,* Cambridge University Press, 1985.

Zimmerman, Michael E., *Heidegger's Confrontation with Modernity: Technology, Politics, Art,* Bloomington: Indiana University Press, 1990.

Author index

Subject index