Technical Report 1204

Vision, Instruction and Action

David Chapman

MIT Artificial Intelligence Laboratory

AI-TR 1204

Vision, Instruction and Action

David Chapman

Massachusetts Institute of Technology
Artificial Intelligence Laboratory

Technical Report 1204 April, 1990

Vision, Instruction, and Action

David Chapman

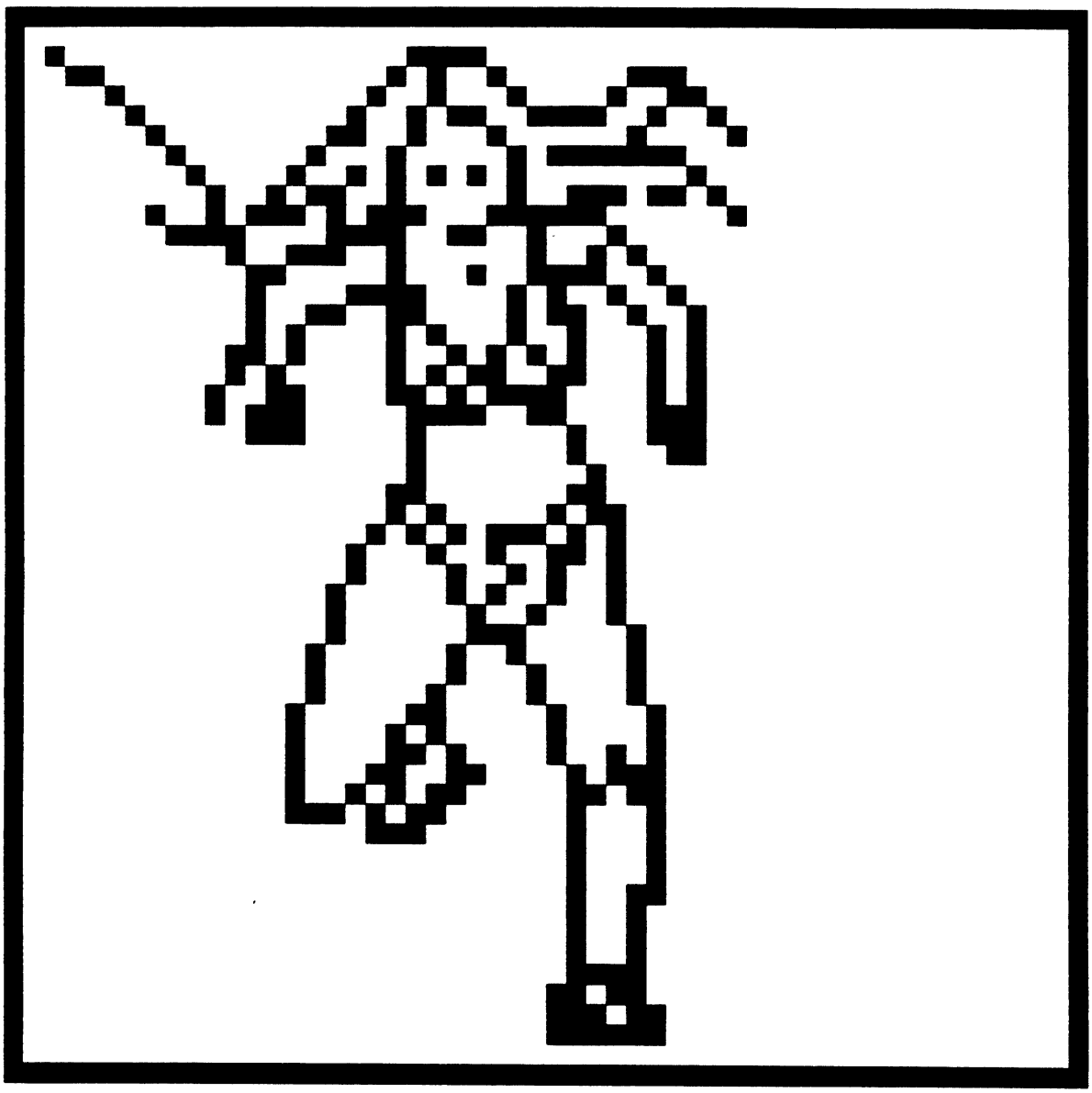
This thesis describes Sonja, a system which uses instructions in the course of visually-guided activity. The thesis explores an integration of research in vision, activity, and natural language pragmatics. Integrated systems are uncommon in AI research, but they provide otherwise unobtainable insights into the relationships between AI problems and techniques.

The design of Sonja is motivated by psychophysical and neuroscientific evidence. The computations the system performs are compatible with the constraints imposed by neurally plausible hardware. The system has access to its domain only via the domain's primitive actions and via a simulated visual system. This visual system addresses the fundamental visual problems of selectively applying visual processing to subsets of the image, finding regions of the image with task-relevant properties, and establishing spatial relationships among parts of the image. It unifies and provides the first psychophysically accurate implementations of a variety of intermediate visual processes, particularly visual search and routines. It models previously unspecified aspects of these phenomena and demonstrates by connecting proposed mechanisms with a natural task domain that they are in fact useful.

Although Sonja can operate autonomously, it can also make use of instructions provided by a human advisor. The system grounds its use of these instructions in perception and action: it interprets them in terms of the situation in which they are given, including the physical configuration of the environment and the system's current and projected activities. The thesis thereby provides a framework for external reference (the ability of representations to designate objects in the world). Sonja uses instructions flexibly; it does work to figure out what they mean, can interleave instruction use with other goal-directed activity, and rejects instructions that make no sense.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract number N00014-85-K-0124. The text of this report is identical with that of the author's PhD thesis, submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Doctor of Science. The author hereby grants to MIT permission to reproduce and to distribute copies of this thesis document in whole or in part.

Copyright © David Chapman, 1990. All rights reserved.



Contents

1 Introduction **10**

1.1 What Sonja does 10
1.2 Contributions 18
1.2.1 Contributions to activity theory 18
1.2.2 Contributions to computational linguistics 18
1.2.3 Contributions to vision research 21
1.3 Guide to the thesis 21

2 The concrete-situated approach **24**

2.1 Routineness 25
2.2 Situatedness 26
2.3 Interactivity 28
2.4 Dynamics 30
2.5 Routines 31
2.6 Improvisation 33
2.7 Deictic representation 34

3 Hardware and architecture **37**

3.1 Essential connectionism 37
3.1.1 The essential connectionist commitments 38
3.1.2 The essential connectionist constraints 38
3.1.3 Advantages of essential connectionist hardware 41
3.1.4 Alternative implementations of essential connectionism 41
3.2 The central system 42
3.2.1 Modularity 42
3.2.2 A three-phase central system architecture 43
3.3 Architectures for vision 44
3.3.1 The purpose of vision 46
3.3.2 Insect vision 47
3.3.3 Some distinctions 48
3.3.4 Visual routines 50
3.3.5 Implications for the central system 53

3.3.6 Visual routines implement deictic representations 56

4 Amazon **57**
4.1 Criteria on a domain 57
4.2 Amazon 60
4.3 Modifications 62
4.4 Realism 63

5 Instruction use **65**
5.1 Video game instruction use 65
5.2 Reference 67
5.2.1 Theories of reference 67
5.2.2 Grounding reference in activity 68
5.2.3 Indexicality 68
5.2.4 Reflexivity 71
5.3 Giving Sonja instructions 72
5.4 How Sonja uses instructions 76
5.4.1 Instructions reorient attention 76
5.4.2 Instructions manage autonomous routines 76
5.4.3 Clearing instruction buffers 78
5.4.4 Some kinds of instructions Sonja can use 78
5.4.5 Generating replies 81
5.4.6 Some issues I'm ignoring 82
5.5 Related work 82
5.5.1 Instruction taking 82
5.5.2 Reference 85

6 The peripheral systems **88**
6.1 Visual attention 89
6.1.1 Psychophysics of attention 89
6.1.2 The addressing pyramid 90
6.1.3 Neuroscientific study of covert visual attention 93
6.1.4 Implementations of covert attention 93
6.2 Visual search 94
6.2.1 Psychophysics of visual search 94
6.2.2 An architecture for visual search 95
6.2.3 Extensions 99
6.2.4 Open questions 100
6.3 Intermediate objects 101
6.3.1 Markers 103
6.3.2 Lines and rays 105
6.3.3 Colors 105
6.4 General implementation considerations 105

6.4.1 Early vision . . . 106
6.4.2 Clocking . . . 106
6.4.3 Precision and accuracy . . . 106
6.4.4 Operator duplication . . . 108
6.5 Criteria on visual operators . . . 108
6.6 The specific operators . . . 109
6.6.1 Distances, directions, and angles . . . 109
6.6.2 Visual search . . . 111
6.6.3 Marker, line, and ray motion; tracking . . . 112
6.6.4 Gaps and edges . . . 113
6.6.5 Coloring . . . 113
6.6.6 Blanking . . . 115
6.7 Other visual systems . . . 115
6.7.1 Segmentation . . . 115
6.7.2 Shape matching . . . 116
6.7.3 Foveation . . . 117
6.7.4 Spatial memory . . . 117
6.8 Sonja's other peripheral systems . . . 118

7 The central system architecture **119**
7.1 MACNET . . . 119
7.2 The arbitration macrology . . . 121
7.2.1 Proposers . . . 121
7.2.2 Conditions and overridings . . . 122
7.2.3 Defaults . . . 123
7.2.4 Hierarchical arbitration . . . 123
7.2.5 Discrimination . . . 126
7.2.6 Arbitration compilation . . . 127
7.2.7 Arbitration in practice . . . 127
7.3 Debugging facilities . . . 129
7.4 Related work . . . 133

8 Sonja **135**
8.1 Scenario . . . 136
8.2 Global implementation issues . . . 138
8.2.1 Real time . . . 138
8.2.2 The allocation of intermediate objects . . . 140
8.2.3 Projection . . . 147
8.2.4 Instruction buffer clearing . . . 147
8.2.5 Defined machine types . . . 147
8.3 Registering the amazon and its aspects . . . 147
8.4 Registration with the general-purpose marker . . . 151
8.5 Registering scrolls . . . 152

8.6 Combat . . . 152
8.7 Navigation . . . 159
8.7.1 Uses of navigation . . . 160
8.7.2 Approach to navigation . . . 160
8.7.3 Head-for-goal . . . 161
8.7.4 Rooms . . . 173
8.7.5 Suggested-go . . . 178
8.7.6 Wandering . . . 178

9 Analysis **179**
9.1 Evaluation . . . 179
9.1.1 Results and approaches . . . 179
9.1.2 Implementations . . . 181
9.1.3 How to evaluate this thesis . . . 182
9.2 The concrete-situated approach . . . 183
9.3 The architecture . . . 185
9.4 Instruction use . . . 186
9.4.1 General discussion . . . 186
9.4.2 Some specific issues in reference and reflexivity . . . 187
9.5 Vision . . . 189
9.5.1 Bypassing early vision . . . 189
9.5.2 Object recognition . . . 190
9.5.3 Three dimensions . . . 190
9.5.4 Criteria on the set of visual operators . . . 191
9.6 The system as a whole . . . 193
9.6.1 Sonja considered as a technology . . . 193
9.6.2 Quality of play . . . 194
9.6.3 Size of the system . . . 195
9.6.4 Deictic representation . . . 196
9.6.5 Navigation . . . 196

A Collaboration **198**

B Skill acquisition **202**
B.1 Some existing skill acquisition methods . . . 203
B.1.1 Domain theory acquisition . . . 203
B.1.2 Compilation . . . 204
B.1.3 Bottom-up skill-acquisition methods . . . 205
B.2 WOMBAT . . . 207
B.2.1 Motivation . . . 207
B.2.2 Implementation . . . 208
B.2.3 Related work . . . 210
B.2.4 Analysis . . . 211

B.3 The phenomenology and dynamics of skill acquisition 212
B.3.1 Resources for skill acquisition 213
B.3.2 The phenomenology of unarticulated skill acquisition 214
B.3.3 Interactions between articulated and articulated skill acquisition 216
B.3.4 Tasks for a skill acquisition system 219
B.3.5 The phenomenology of articulation 221

References **222**

Acknowledgments

Phil Lesh once said in an interview that the Grateful Dead's success could be attributed to the fact that "despite working together for so long we retain the ability to surprise each other." This accurately describes my relationship with my collaborator Phil Agre, to whom my first thanks go.

Rod Brooks supervised this thesis. I couldn't have asked for a better advisor. He knew when to leave me alone and when to twist my arm for text. His advice on difficult problems, technical and political, was unerring.

Barbara Grosz and Tomás Lozano Pérez formed the other two thirds of my committee. Their expertise in fields about which I initially knew nothing was invaluable. Their insistence that I get some of my facts and references straight has improved the thesis immensely.

In addition to these people, John Batali, Brad Blumenthal, Jon Connell, Jim Davis, Gary Drescher, Ian Horswill, Beth Preston, Jeff Shrager, Penni Sibun, Orca Starbuck and Ramin Zabih commented on drafts of the thesis. If it is readable, it is largely due to their efforts.

Christina Allen, Jerry Feldman, Ken Forbus, several members of the Interaction Analysis Laboratory, Jim Mahoney, Eric Saund, Candy Sidner, Susan Stucky, Shimon Ullman, Dan Weld, and many other friends and colleagues have provided useful comments in conversation and by email.

Thanks to all of you.

Chapter 1

Introduction

This thesis describes Sonja, a system which uses instructions in the course of visually-guided activity. The thesis explores an integration of research in vision, activity, and natural language pragmatics. Integrated systems are uncommon in AI research, but they provide otherwise unobtainable insights into the relationships between AI problems and techniques.

Sonja engages in sensible activity in a domain characterized by complexity, uncertainty, immediacy, and continual change. It has access to this domain only via the domain's primitive actions and via a simulated but realistic visual system.

Although Sonja can play autonomously, it can also make use of instructions provided by a human advisor. The system interprets instructions in terms of the situation in which they are given, including the physical configuration of the environment and the system's current and projected activities. It uses instructions flexibly; it does work to figure out what they mean, can interleave instruction use with other goal-directed activity, and rejects instructions that make no sense.

1.1 What Sonja does

In this section I will describe a short scenario which demonstrates some of Sonja's abilities.

Sonja plays a video game called Amazon. An Amazon scene appears in figure 1.1. A player of Amazon (a person or Sonja) controls an icon on the screen representing a woman warrior: the amazon. The amazon can move about in a dungeon. In the dungeon there are walls and various enemies and tools. In this scene, some walls form a room, there are a ghost and a demon that will attack the amazon, and there are three amulets (on chains) that will give the amazon magical powers when they are picked up. The player can cause the amazon to shoot at the monsters and kill them.

Sonja plays a competent beginner's game of Amazon autonomously. It knows how to kill off monsters, how to pick up and use tools, and how to get about in the dungeon maze. Sonja doesn't always know how to choose the right thing to do when there are several plausible courses of action available. In these circumstances, instructions from an advisor can be useful. The scene of figure 1.1 is one in which Sonja would do fine without instructions; I will, however,

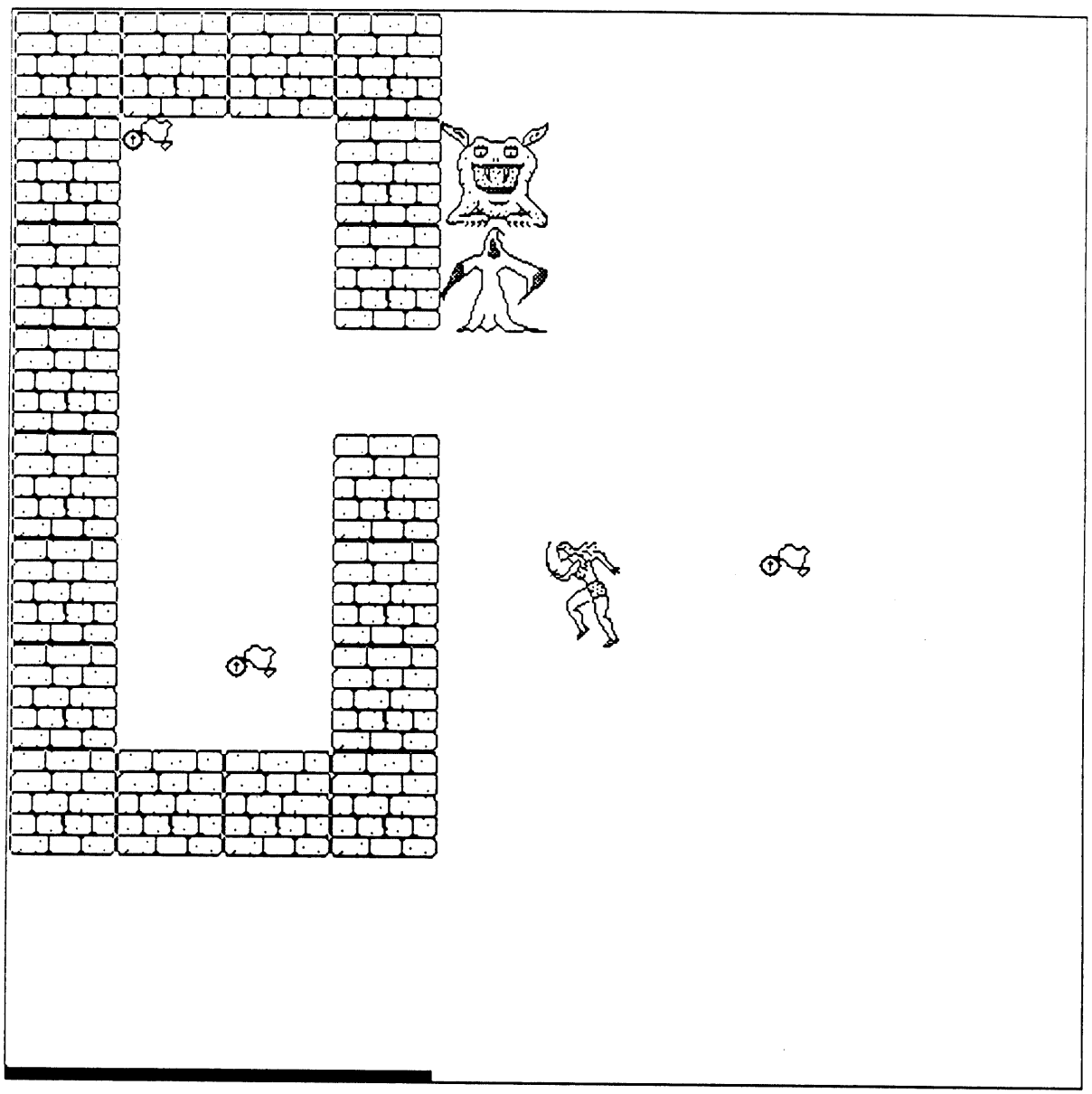


Figure 1.1: An Amazon scene. The amazon is in the center of the picture; one amulet is directly to her right. The frog-like thing with ears is a demon; a ghost is directly below it. In the room there are two more amulets.

micromanage the system to demonstrate how Sonja can use them.

In this scene, an advisor might suggest to Sonja "Go in and get the amulet." To use this instruction, Sonja first has to figure out what "the amulet" refers to. Because Sonja's access to Amazon is mediated by perception, it must perform a visual search of the scene to find the amulet referred to. Sonja's visual system models in detail facts about human visual search. I'll argue that it is important that Sonja and its human advisor have similar visual processes, because communication depends on a shared understanding of the world.

"Go in and get the amulet" doesn't refer to just any amulet; the amulet must be one that is in the room the amazon will go into. The amulet to the amazon's right is not satisfactory, for example; it wouldn't count as carrying out "Go in and get the amulet" to go into the room and then come out and pick up the amulet outside. Sonja's ability to find the right amulet depends on its ability to see whether an amulet is in a room. The ability to recognize rooms and containment relationships is also needed for getting about in the dungeon; to get the amulet, Sonja will have to go through the doorway, which entails finding it.

Sonja identifies the bottom left amulet as the one most likely to have been referred to by the instruction and starts the amazon heading for the doorway. Its passage, however, is blocked by the ghost. Sonja has to kill the ghost to make progress; it does so by shooting shuriken (throwing stars) at it (figure 1.2). Because Sonja lives in a dynamic world in which events are not entirely under its control, it can not treat instructions as though they were programs or subroutine calls. Sonja may have to carry out arbitrary amounts and kinds of work before completing an instruction. Its use of instructions must be intelligent and flexible in order to combine and interleave the use of several instructions.

Having dispatched the ghost, Sonja starts to shoot at the demon (figure 1.3) which also blocks passage through the doorway. At this point the human advisor recommends "Use a knife." Knives in Amazon have several functions; they can be used to jimmy the locks on doors and they can be thrown at monsters and other things to destroy them. While it takes three shuriken and a couple of seconds to kill a monster, a single knife will do the job quickly. On the other hand, knives can only be used once, whereas the supply of shuriken is unlimited.

The instruction "Use the knife" can only be carried out after it has been made sense of and interpreted in terms of the situation at hand. Figuring out what the advisor meant involves looking at the scene for opportunities to use a knife in one of the various ways possible. In this instance, there are no doors to open. On the other hand, throwing a knife at the demon would make sense. Whereas ghosts must touch the amazon to hurt it, demons breathe rapidly-moving fireballs, and so are more dangerous. It is often worth using up a knife to kill one quickly. Sonja accordingly throws a knife at the demon (figure 1.4). This example shows that Sonja can not only carry out several courses of action at once, but can also use several instructions at once; it used the knife *in the course of* going into the room to get the amulet.

Sonja proceeds through the doorway (figure 1.5) and turns to head toward the amulet it has identified as that which the instruction-giver intended (figure 1.6). It turns out, however, that the instruction-giver actually intended Sonja to get the top amulet. That Sonja misunderstood only becomes apparent to the instruction-giver once Sonja is through the door and has turned to head downward. At this point, the instruction-giver says "No, the other one!" to repair the

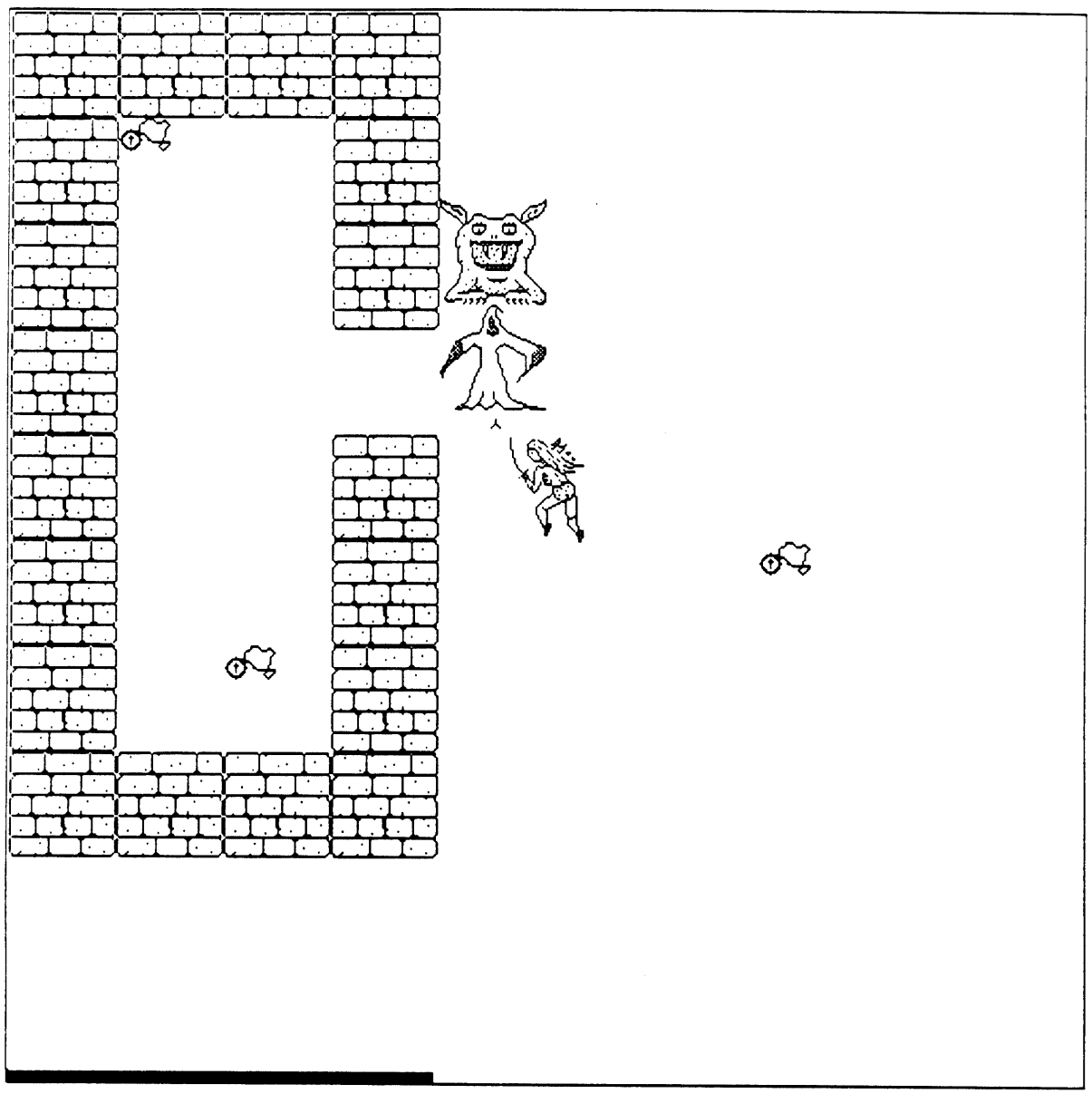


Figure 1.2: Sonja kills the ghost by shooting shuriken at it. A shuriken is visible as a small three-pointed star between the amazon and the ghost.

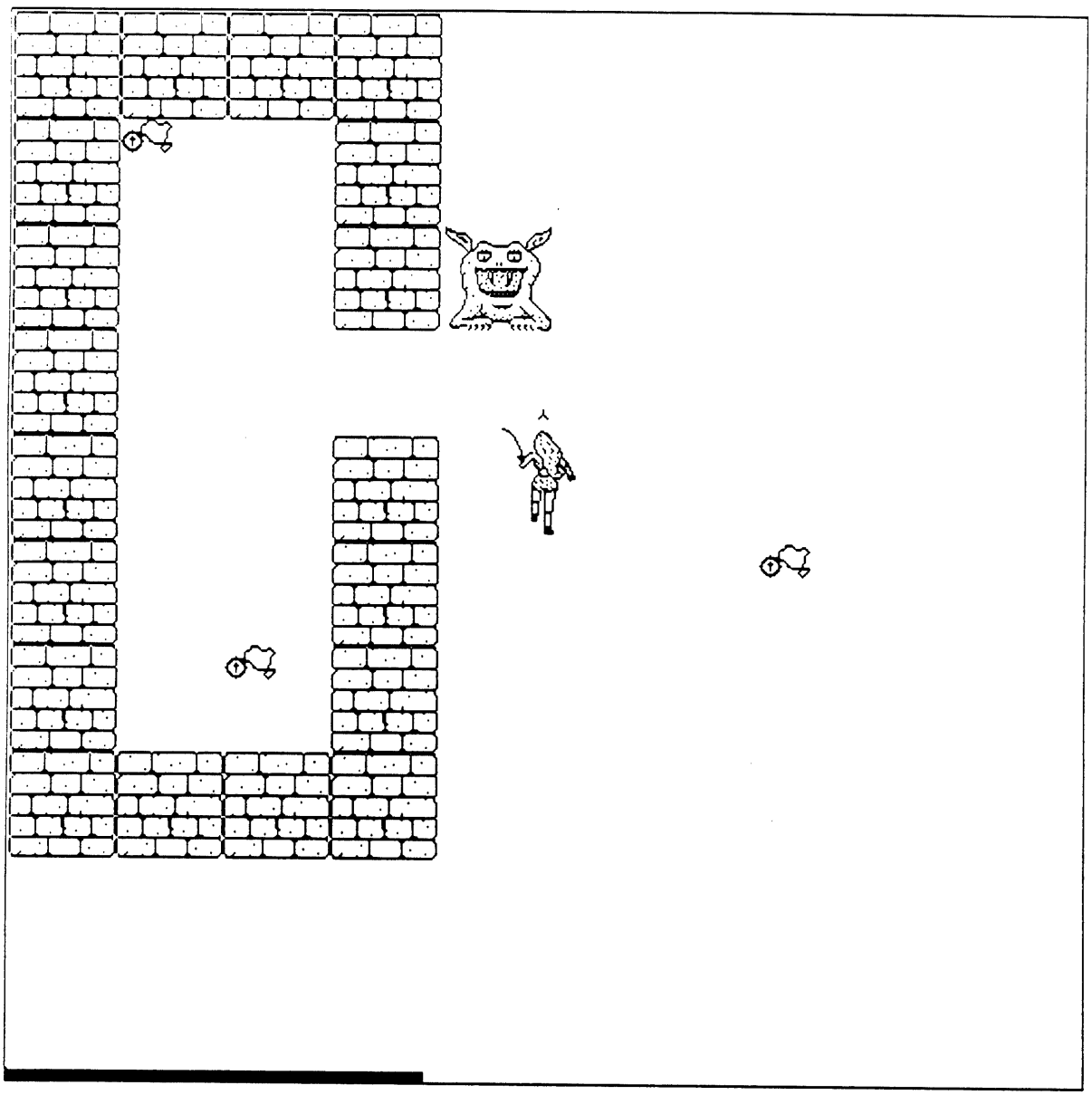


Figure 1.3: Sonja shoots at the demon.

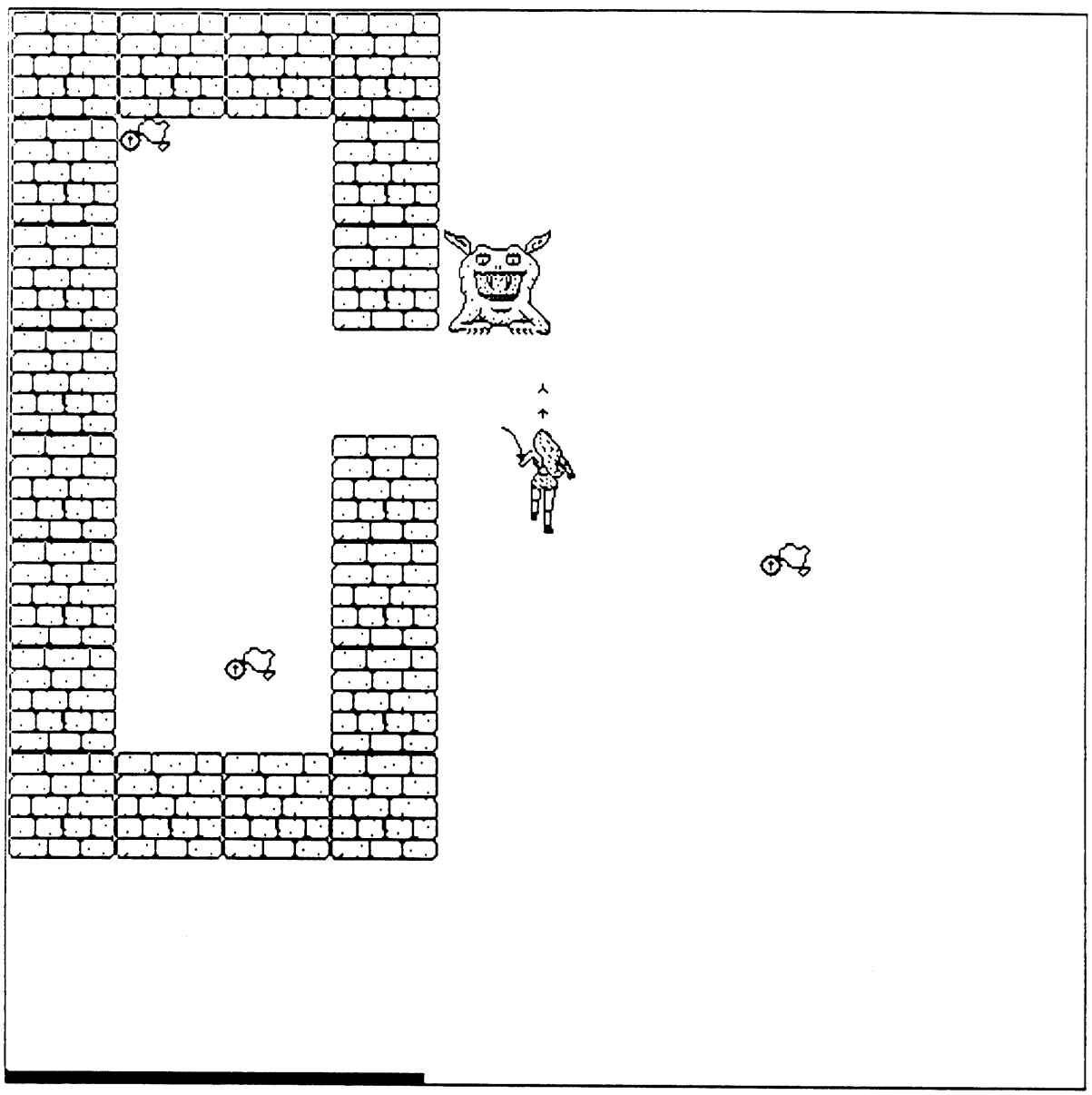


Figure 1.4: The amazon throws a knife at the demon based on the instruction "Use a knife". The knife appears as a small arrow directly above the amazon.

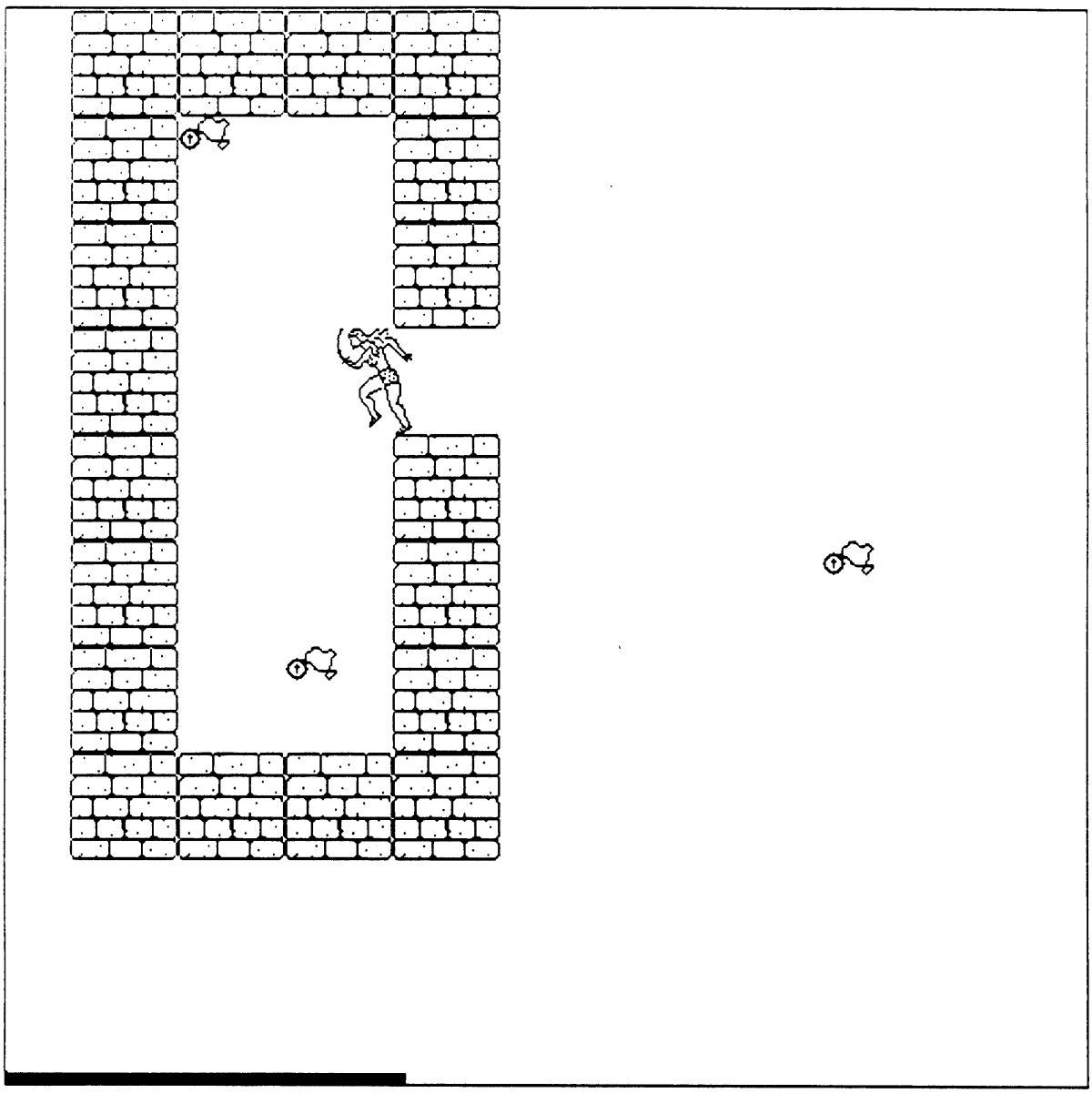


Figure 1.5: Sonja passes through the doorway of the room.

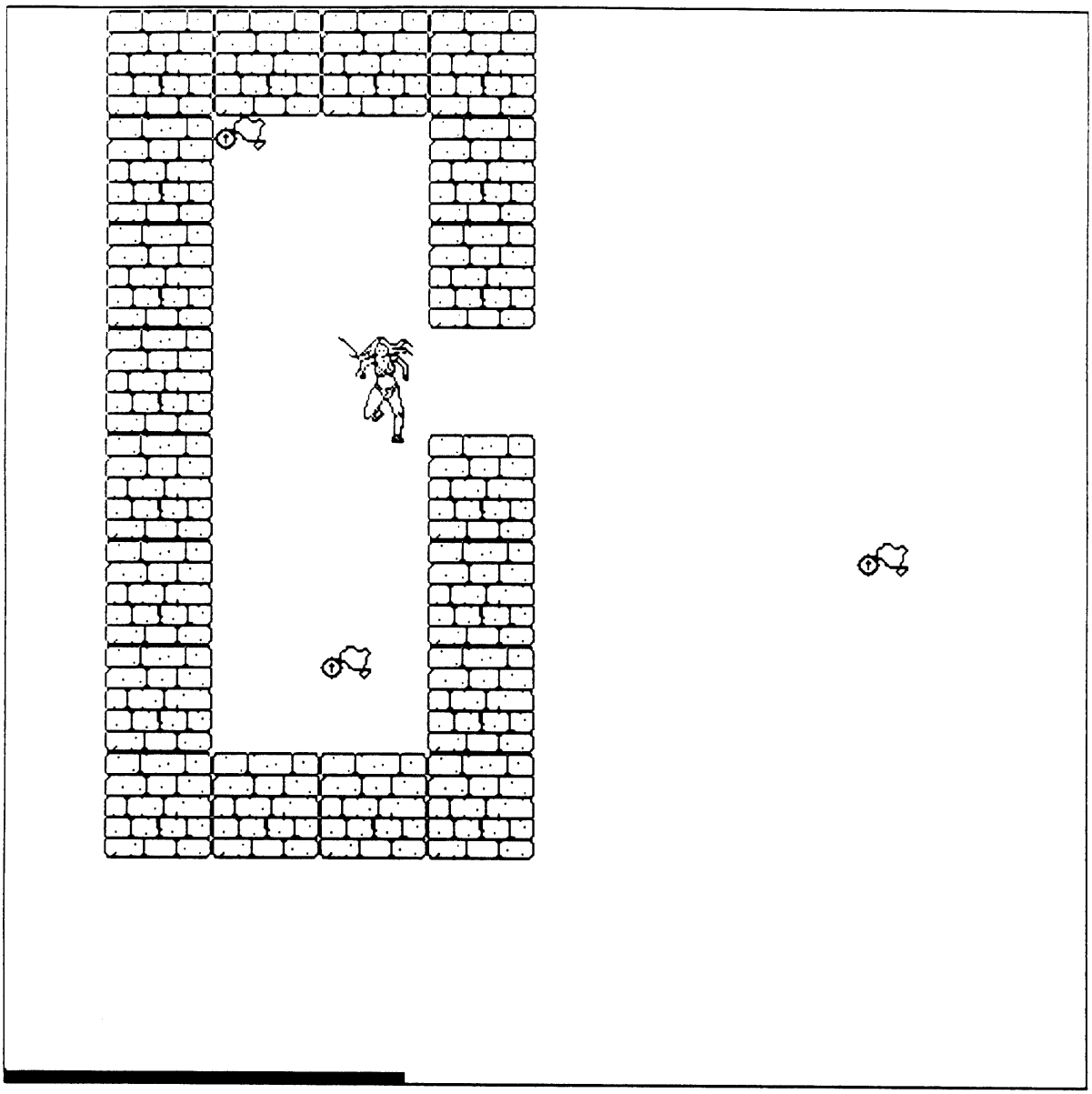


Figure 1.6: Sonja turns downward to get the bottom amulet.

misunderstanding. Sonja finds an alternative referent for "the amulet," namely the top one, heads toward it (figure 1.7), and picks it up (figure 1.8).

1.2 Contributions

This thesis contributes to at least three fields: the theory of activity, computational linguistics, and machine vision.

1.2.1 Contributions to activity theory

A theory of activity is concerned with how agents *do* things. Agre and I have been developing a new theory of activity over the past five years [3, 4, 5, 6, 7, 36, 37, 38].

This theory proposes that activity arises from the interaction of an agent with its environment. The agent can participate in ongoing events by taking continual account of environmental conditions. The theory includes a new, participatory theory of representation. The theory requires and includes a realistic, detailed model of perception.

The theory is motivated by issues of computational and neurophysiological plausibility. We demand that a theory of activity explain how an agent can usually decide what to do before it is too late to do it. We also demand that the theory not contradict established facts about how the brain works.

Sonja is the most recent of a series of implementations of parts of this theory. It extends previous work by addressing linguistic issues. This connection allows and requires that the use of instructions be flexible and sensible, taking account of current goals and circumstances. Sonja also extends our understanding of the architecture underlying activity and incorporates many technical improvements.

1.2.2 Contributions to computational linguistics

The principal linguistic tenet of this thesis is that language use must be grounded in concrete activity. Although I believe this is a universal phenomenon, it is particularly clear in video game instruction use. The ability to understand instructions is inseparable from the ability to use them: that is, to engage in the activity they recommend.

This thesis concentrates, among linguistic phenomena, on reference. Most previous computational work on reference has concentrated on specification: finding a mental representation which refers to the same thing as a given linguistic representation. Sonja is concerned instead with external reference, the ability of representations to designate objects in the world. Sonja implements external reference causally, by grounding reference in language users' interaction with the objects and activities to which their speech refers.

Sonja models several pragmatic issues in addition to reference, in particular linguistic repair processes and the use of temporal and spatial expressions.

Sonja also extends existing work by connecting language use with more detailed and realistic theories of vision and activity.

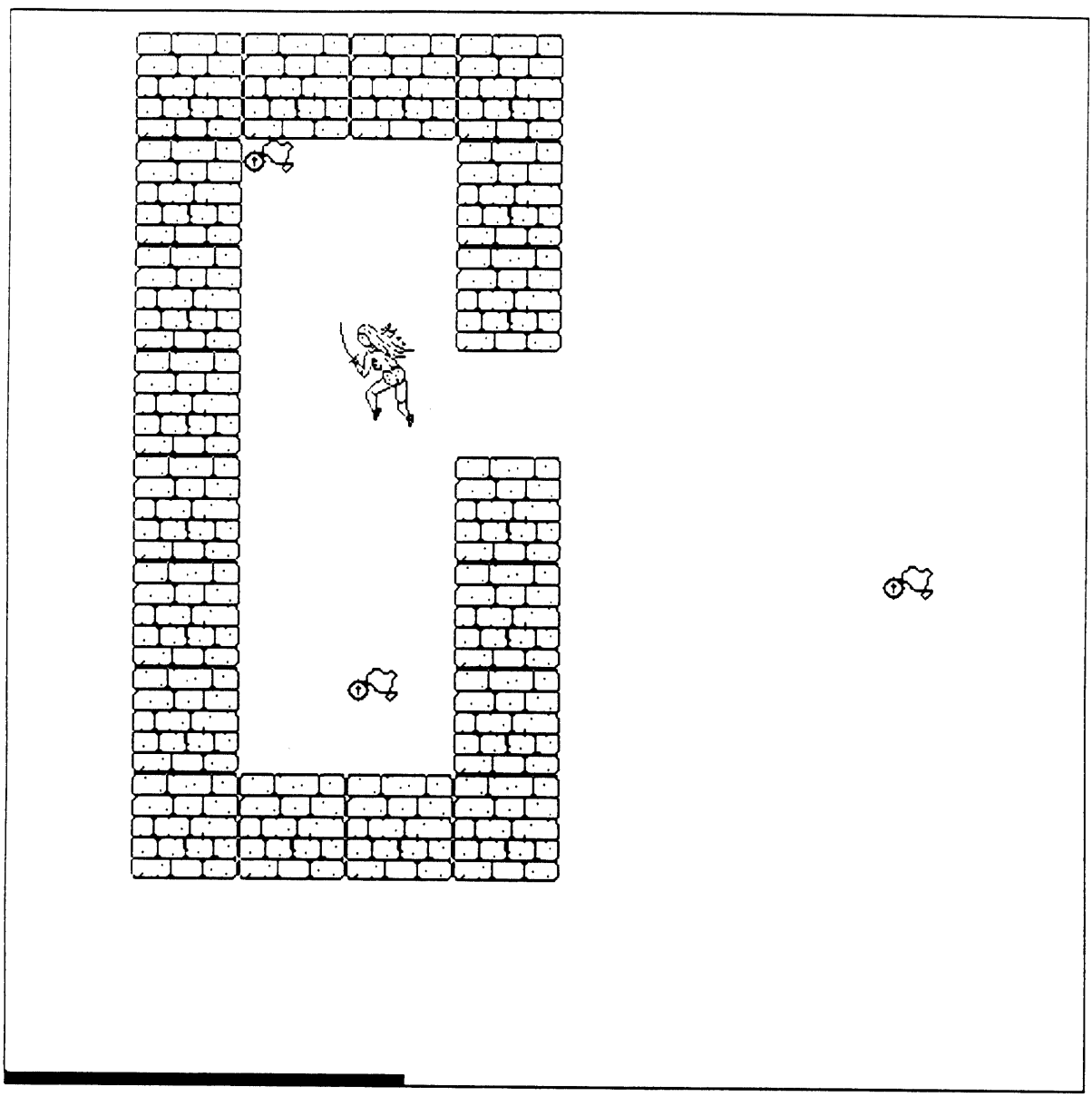


Figure 1.7: Sonja has been told "No, the other one!" and heads for the top amulet instead.

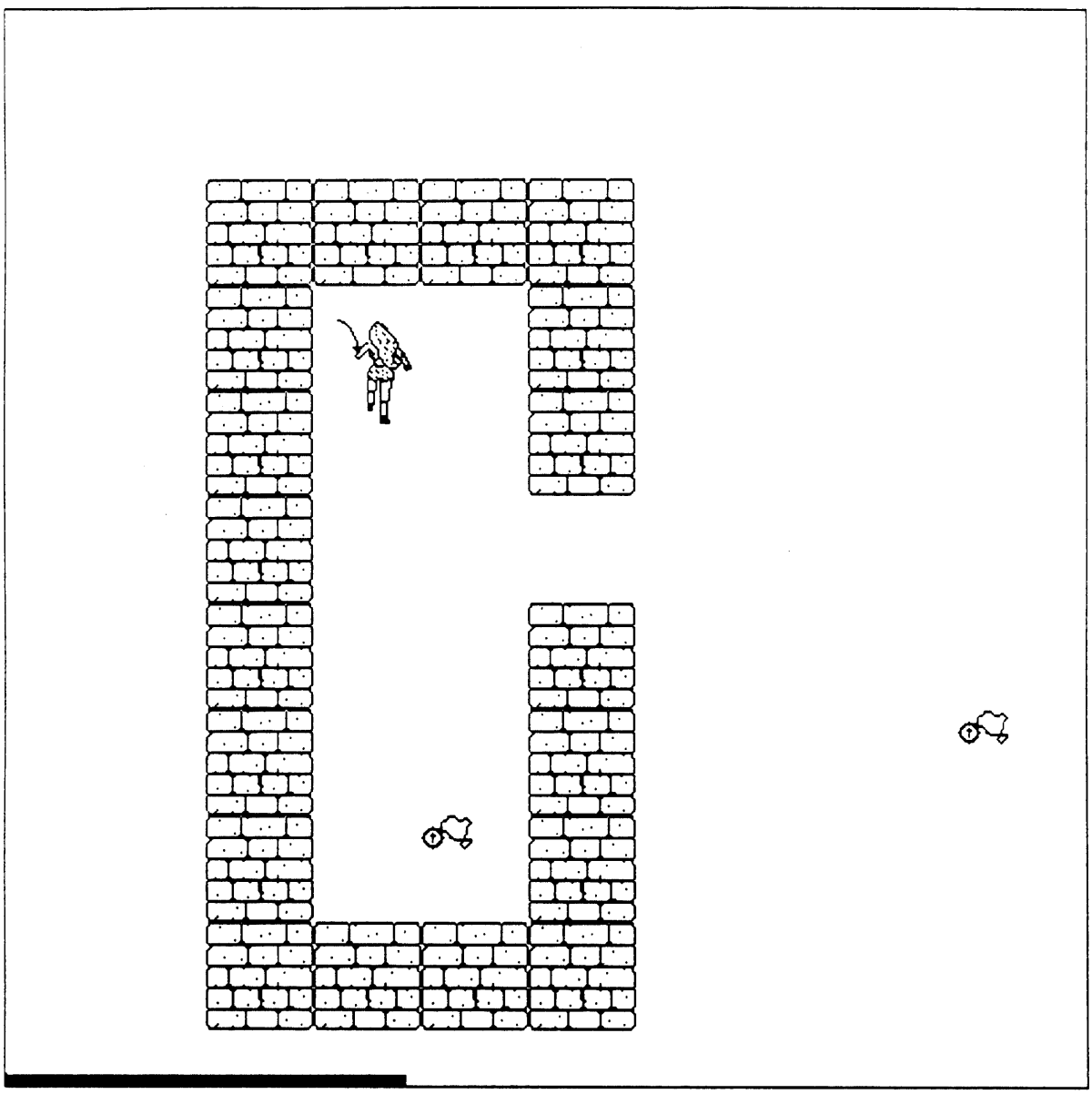


Figure 1.8: Sonja has just picked up the amulet, causing it to disappear from the screen.

There are many linguistic phenomena that Sonja does not model; for example, it does almost no syntactic processing.

1.2.3 Contributions to vision research

Sonja integrates and extends several bodies of work in intermediate vision. The thesis presents an implementation of a unified visual architecture and demonstrates that this architecture can support a serious theory of activity.

Sonja demonstrates for the first time that various visual mechanisms previously proposed on psychophysical, neurophysiological, and speculative computational grounds are in fact useful by connecting them with a natural task domain. The implementation fleshes out many aspects of these theories which have previously been left unspecified. For example, it includes a detailed interface between intermediate vision and central processing.

Sonja does not incorporate a realistic model of early vision. The thesis suggests ways it could.

1.3 Guide to the thesis

This thesis is long and in places dense with technical detail. You may not want to read all or most of it. Here are some suggestions for how to proceed if you only intend to read part of it, or to see if you are interested in more.

- To get more of an idea of what Sonja can do, read the introduction to chapter 8 and then skim sections 8.3 through 8.7, looking particularly at the pictures.
- If you are interested in how Sonja does it, you can start with chapter 3, which describes Sonja's architecture in general terms.
- If you are interested in particular in Sonja's approach to vision, skim chapter 3 and then read chapter 6, which explains Sonja's visual system in detail.
- If you are interested in Sonja's instruction use, read chapter 5.
- If you are interested in Sonja's approach to activity, read chapter 2 for philosophical underpinnings, skim chapter 3 on the architecture, chapter 8 on how Sonja plays Amazon, chapter 9, which analyzes the implementation, and appendix B on skill acquisition.

After skipping around, you may want to read the chapters in order. At any rate, I'll describe them in order in some more detail.

Chapter 2 describes the theory of activity underlying Sonja. You should read this chapter through if you are going to read much of the rest of the thesis, as it introduces terms that are used throughout. Together with chapter 3, it explains why I wrote Sonja the way I did and why that's interesting; Sonja may seem trivial, bizarre, and unmotivated without this explanation.

Chapter 3 describes the theory of cognitive architecture Sonja implements. This theory is principally motivated by arguments from neurophysiology and from vision research. It puts

strong constraints on cognitive theory. For example, it would not be easy, and probably would not be possible, to implement most current cognitive models in this architecture. I present some basic facts about the brain and argue that they imply that the human cognitive machinery makes traditional programming techniques like pointers, variables, datastructures, virtual machines, and dynamic storage allocation expensive, and suggest finding ways to design agents which do not depend on these devices. I argue for a division between peripheral systems (sensory and motor systems) and a central system. I argue for a particular sort of architecture for the visual system based on learnability criteria. This architecture consists of a set of relatively high-level visual building blocks which can be assembled into visual routines by the central system. I propose a particular provisional model of the central system which is intended to allow an agent to act flexibly in realistic domains.

Chapter 4 is about Amazon. You definitely want to read section 4.2, which explains the domain in detail. The rest of the chapter explains why Amazon is and isn't a good domain to explore the issues this thesis is about. It describes the criteria I used in choosing the domain, considers its realism, and discusses some shortcomings of it which became apparent only late in the implementation.

Chapter 5 describes instruction use in Sonja. I describe a framework for grounding reference in concrete activity. I show that naturally occurring instructions must be made sense of in terms of the current situation before they can be used. Making sense of video game instructions involves reorienting attention and requires understanding instructions in terms of activities their user is capable of carrying out autonomously. This chapter describes various sorts of instructions Sonja can use and explains how Sonja uses them.

Chapter 6 describes Sonja's sensory and motor systems, particularly its vision system. The vision system addresses the fundamental visual problems of selectively applying visual processing to subsets of the image, finding regions of the image with task-relevant properties, and establishing spatial relationships among parts of the image. It models a variety of relatively well-understood but previously unimplemented psychophysical phenomena. I discuss visual attention—the ability to direct visual processing to particular areas of an image—and its neurophysiological basis. I also discuss visual search, its psychophysics, its relationship with visual attention, and its implementation in Sonja. Finally, I discuss visual operators—the building blocks of visual processing—and the visual routines assembled out of them. Visual routines perform the spatial reasoning Sonja needs to get about in Amazon.

Chapter 7 describes the implementation technology of Sonja's central system. This system is implemented as a digital circuit. I document the two languages used to design this circuit: a circuit description language and a higher-level arbitration language used to express a three-phase central system architecture. This chapter also documents debugging facilities I implemented to help design Sonja.

Chapter 8 describes the specific digital circuit which is Sonja's central system and explains how it and the peripheral systems together allow Sonja to play Amazon and to take instructions.

Chapter 9 concludes by evaluating the implementation and analyzing its relationship with the theory. I discuss ways in which the implementation technology worked well and ways in which it needs work. I consider whether Sonja demonstrates the points I set out to make.

Two appendices are concerned with future work. Appendix A proposes an extension to Sonja that would allow it to play collaboratively. Appendix B discusses skill acquisition. Much of the theory underlying Sonja makes sense only if it can be made compatible with some theory of skill acquisition. I analyze various existing of skill acquisition in terms of Sonja's architectural constraints, concluding that some popular approaches are impractical or inadequate. The chapter then describes an implementation of an approach to skill acquisition that is compatible with the thesis's approach to activity. Unfortunately this implementation didn't work. I analyze the failure and suggest future directions.

Chapter 2

The concrete-situated approach

This thesis is part of an on-going study of concrete activity: that is, of actively doing something in the world, as opposed to cogitating. I've been conducting this research in collaboration with Agre. Much of our work to date is summarized in our joint papers [5, 6, 7, 38] and in his thesis [4]. Because of the volume and complexity of this work, because most of the central insights are due to Agre, and because they are well documented in his thesis, this chapter is only a summary.

The approach[1] to activity I will present here is quite unlike that currently prevalent in cognitive science. However, in this exposition, I will not explicitly contrast it with the traditional alternatives. The theory is also not without antecedents; in fact, most parts of it have been proposed in literatures other than AI. The principal contribution of our work is in giving these ideas computational grounding. Our paper "What are plans for?" [7] and Agre's thesis discuss the relationship of the concrete-situated approach to previous work in AI; Agre's thesis in addition sketches some of its intellectual history in other fields.

The sections of this chapter explain how the concrete-situated approach

- is principally concerned with *routine* activity (2.1);
- takes an agent's concrete *situation* to be its principal resource in acting (2.2);
- stresses that activity involves *interaction* with an external environment whose influence on activity is equal to the influence of the agent (2.3);
- seeks to describe the *dynamics* of such interaction (2.4);
- in particular seeks to characterize the *routines* an agent falls into (2.5);
- proposes that activity arises from continual *improvisation*, or recomputing what to do (2.6); and
- incorporates a new theory of representation called *deictic* representation (2.7).

[1] I'll talk more about what I mean by an "approach" in section 9.1.1.

These aspects of the theory cohere and reinforce each other. They do not stand alone: the theory of deictic representation, for instance, only makes sense for agents that are interacting with their environments in the course of practical activity. The routineness of activity and dependence on the situation make improvisation feasible. Deictic representation is the principal computational support for improvisation. And so on.

2.1 Routineness

All research begins with simplifying assumptions or idealizations. Most research in AI starts by restricting attention to particular capacities such as vision, language, or planning and studies them largely in isolation. We start instead with a *routineness idealization.* The routineness idealization holds that most activity is *routine.* Routine activity is the regular, practiced, unproblematic activity that makes up most of everyday life. Routine activity does not require that you think new thoughts. (I'll make the notion of a "new thought" more precise in section 3.1.2.) The routineness idealization holds that we can and should study routine activity first, making only occasional reference to the novel elements that are introduced from time to time into in the course of routine activity.

Extensive existing studies of routine activity in several fields demonstrate that it is indeed practical and useful to study routine cases before novel ones. Most of these studies are not computational, but are based in philosophy, psychology, sociology, and anthropology. (For a review, see Agre's thesis [4].) Heidegger [114], for instance, presents a phenomenology of routine activity and argues that this analysis should be the basis of an understanding of what we would call cognitive phenomena, rather than the other way around. (For discussions of the implications of this argument for AI, see Dreyfus [60], Preston [213, 214], and Winograd and Flores [289].) Ethnomethodological sociologists, as another example, study the nature of the routine interpersonal interactions that make up much of everyday life [116]. They are able to reveal much about cognition by studying the ways people say "hello" on the phone or nod their heads in conversation or pay the check in a restaurant.

Computational studies of routine activity begin with Agre's running arguments system [3, 4] and with a system I implemented called Pengi [5]. Pengi illustrated many of the themes I will develop in this chapter; by implementing only the routine case of a particular sort of complex activity, it demonstrated that it was possible to avoid the enormous computational and explanatory overhead required by the then-existing approaches to activity in AI.

There are five reasons to think that it is important to understand routine activity before trying to understand novel activity.

- Routine activity is by far more common. We spend most of our lives engaged in activities like making breakfast, driving to work, reading the paper, typing forms, giving the kids a bath, and grocery shopping; these rarely involve novel elements. Even creative work is mainly routine: it's rewording sentences or painting the background of the scene or playing another gig at the Rat.[2]

[2]These observations are an appeal to your own experience of everyday life. I don't know how to quantify

- Because most activity is routine, the processes which are responsible for novel activity must accommodate and depend on those which are responsible for routine activity. Novel activity is possible only against a background of routine competence. In fact it typically proceeds by assembling routine pieces in novel patterns.

- Routine activity is evolutionarily prior. The activity of animals up to the level of lower vertebrates is wholly routine. For instance, Gallistel [89] argues that learning in animals up through amphibians proceeds only by selective facilitation or inhibition of innate motor patterns. This demonstrates that it is possible to isolate routine activity altogether. It also suggests that if there is distinct neural machinery responsible for novel activity, it must depend on the stable, previously evolved machinery for routine activity.[3]

- As a practical matter, many of the most useful applications of artificial intelligence may be in automating routine activities in order to free people for more interesting pursuits.

These issues are further explored in papers by Agre and myself [4, 38].

Any idealization falsifies the phenomena in certain ways. This is inevitable; we can't do research without one idealization or another. The routineness idealization marginalizes questions about where new thoughts come from and how they connect with the existing machinery for routine activity. Appendix B sketches some starting points for studying these questions.

I believe that the idealization of routineness falsifies the nature of activity less than some other idealizations have. Whether or not this is the case will depend on considerable further research, as computational studies depending on this idealization have only just begun. However, even if not, the idealization may still be useful, because it exposes issues more familiar idealizations hide.

2.2 Situatedness

The concrete-situated approach holds that an agent's most important resource in computing what to do is its concrete situation. In driving, you are responsive to other cars, to road signs, and to the geometry of the road. Pouring milk over your breakfast cereal, your hand constantly adjusts the flow to prevent milk from splashing off the irregular flakes and spilling out of the bowl. There are, of course, other resources available (internal and external memory, for example), but concrete action without perception is virtually impossible.

Use of the immediate presence of the world makes some hard problems easy.

- If you want to find out something about the world that will affect how you should act, you can usually just look and see.[4] Concrete activity is principally concerned with the

routineness, nor do I know of any careful studies of how much of life is routine.

[3] I take no stand on the question of whether machinery specific to novel activity is necessary, desirable, or to be found in the human brain.

[4] This doesn't imply that knowledge is free; as we'll see in section 3.3.4, looking is an active process which requires computation and sometimes external work such as head motion or removal of occluding objects. Looking is, however, usually computationally cheaper than alternatives such as deduction and is to be preferred in such cases.

here-and-now. You mostly don't need to worry about things that have gone before, are much in the future, or are not physically present. You don't need to maintain a world model; the world is its own best representation. (We'll explore this idea further in section 2.7, on deictic representation.)

- If you want to know what action to take, you can often just look and see that as well. In a great many situations, it's obvious what to do next given the configuration of materials at hand. And once you've done that the next thing to do is likely to be obvious too. Complex sequences of actions result, without needing a complex control structure to decide for you what to do. (See section 2.6.)

- If you want to know what the effect of an action will be, or whether a particular approach to a task will work, and if you can't tell just by looking, you can usually safely try it and find out. You rarely need to perform simulations. (See section 2.7.)

- If you want to become expert at a task, you can learn it gradually by practice, rather than solving the general case all at once. The world benignly falls into useful regularities, so you can trust your accumulated experience. (See appendix B.) You aren't playing a game against the world; your breakfast is not out to get you. Doing the wrong thing is rarely fatal.[5]

- The world has people in it as well as things. You aren't on your own; someone else has probably had to do what you want to do before and can help and give advice. (See chapter 5.) After a few rounds of collaboration, you'll become expert yourself. (See section B.3.1.) Many tasks are socially organized as involving division of labor, so that you can participate in activities in which you *never* learn all the roles. Medical activities provide many examples; Hutchins [122] has studied in detail other examples in the domain of learning to navigate naval ships.

The nature of concrete situations makes some traditional approaches to activity unworkable.

- Situations change continually. Algorithms which formally solve your problem are no use if they terminate after the problem has changed or solved itself or turned into a disaster.

- Real situations are immensely messy, always more complicated than any representation of them can be.

- Real situations, in particular, make it hard to know exactly what the outcome of your actions will be.

- Perception is your only access to the situation around you. You can only see and hear things that happen nearby. Some things that matter to you will always be unknown and unknowable. Algorithms which require complete information are no use.

[5]The *natural* world is dangerous. Fortunately, people do not live in a state of nature, but in an embedding culture which eliminates most natural dangers. This fact is depended on extensively by human cognition.

The concrete-situated approach addresses these four difficulties by suggesting that agents

- Intimately interact with the changing situation (section 2.3);
- Represent only relevant aspects of the situation (section 2.7);
- Continually improvise and try out alternative ways of doing things (section 2.6); and
- Ground representation and action in perception (sections 2.7, 3.3.4, and 3.3.6).

2.3 Interactivity

The concrete-situated approach is *interactionist*. It holds that the organization of activity is emergent from interactions between an agent and its environment, rather than being a property of an agent (as in *mentalism*) or of the environment (as in *behaviorism*). Causality rapidly loops in and out of the agent, rather than looping around inside the agent's head and occasionally emerging to affect the world (figure 2.1).

The complexity of activity arises not principally from the complexity of the agent or of the world, but from the complexity of their interaction. Simon's parable of the ant [251] argues that a simple system can engage in complex activity if it is put in a complex environment. But Simon's ant is virtually programmed by its environment; its activity will not be much more complex than the environment is. The complexity of an interaction can be much greater (or much less) than the sum of the complexities of the agent and its environment. A small finite state machine, for instance, can compute any recursive function when placed in the environment of two counters [178]. On the other hand, a photocopier is a complicated machine which does something simple. Like a person and unlike Simon's ant, Sonja is a complex system. However, it is designed so that the complexity of its activity is much greater than the sum of the complexity of its machinery and the complexity of its domain.

An agent that accepts the world as an equal partner in organizing its activity does not continually try to force interactions to conform to a preconceived idea of how things should go. That's futile; other processes and agents in the world would constantly force you off track. On the other hand, taking the world as a *merely* equal partner implies that you can not just react to situations as they arise. Sensible interaction requires forethought.

Locating the determination of activity in interaction implies viewing life as a series of opportunities and contingencies: openings to participate in particular sorts of activities and events that arise as you do so. Life is not a series of problems, each handed to you as a unit and requiring a unitary solution. Life is constantly ongoing. The concrete-situated approach takes dancing and hanging out, rather than solving the eight puzzle or designing a circuit, as prototypical activities. These activities involve other people, who are likely to make it hard to impose your solutions. On the other hand, in these situations the other people will share the work of making interaction simple.

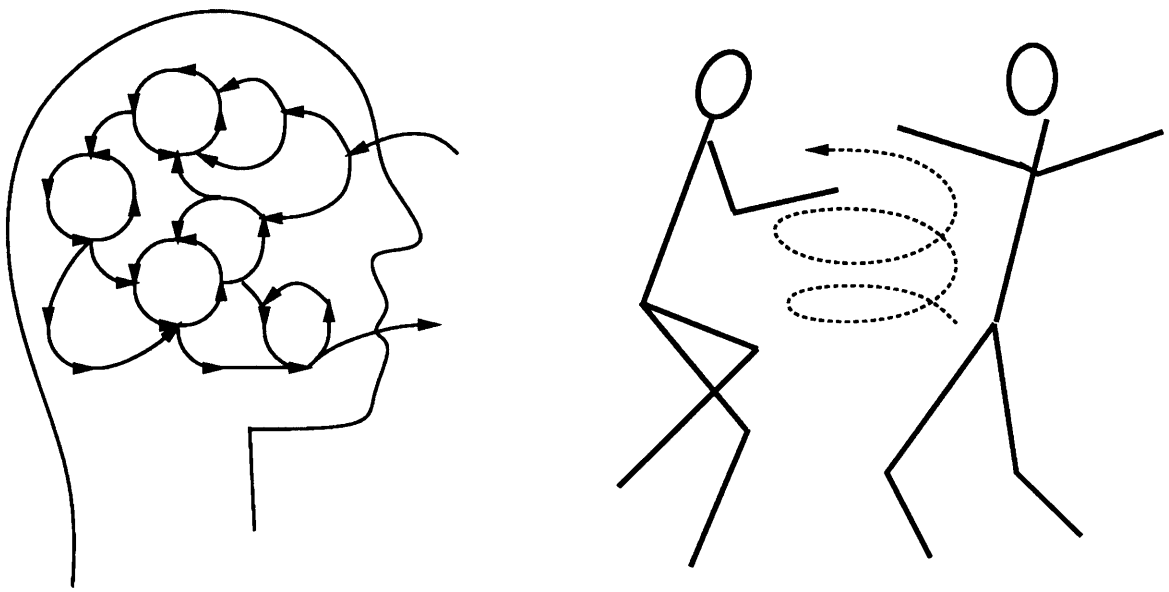


Figure 2.1: Mentalist and interactionist causal loops. In mentalist theories, causality mostly loops around inside an agent's head, occasionally passing from module to module and relatively rarely crossing the borders of the skull. Playing chess is a mentalist prototype of activity. In interactionist theories, causality constantly passes between the agent and the world. Dancing is an interactionist prototype of activity. It emphasizes that people are not disembodied heads, but whole agents interacting with others via sensors and effectors in a social context.

2.4 Dynamics

A *dynamic* is a pattern in the interaction between an agent and its world. What sort of dynamics an agent participates in depends on the machinery that composes it and on its world. What sort of machinery an agent must have depends on what sort of dynamics it needs to participate in and what sort of world it will be situated in. The concrete situated approach prescribes interleaving study of machinery and dynamics.

Here is an example dynamic. If you live in a graduate student household, you probably have in your kitchen cupboard a stack of bowls, not many of which match. When you use a bowl, you put it back at the top of the stack. Over time, the bowls that aren't used often tend to sink to the bottom of the stack. This example illustrates several points about dynamics generally.

- Dynamics are not causal agents. Bowls don't end up at the bottom of the stack *because* of a sinking dynamic; they end up at the bottom because other bowls are taken out from under them.

- A dynamic operates only *ceteris paribus*, when an unbounded set of conditions hold. If you have a big dinner party, you may use all your bowls and they'll end up back in the cupboard scrambled and then will slowly sort themselves out again over the course of the next few weeks.

- Dynamics are not processes in the agent's head. They are patterns of interaction which may be noticed by a theorist. Dynamics are typically not represented by the agent (though some are). Most people with such a stack of bowls probably have never noticed this dynamic. Activity has a coherence and orderliness which is not a product of any representation of it. Other people may have noticed this dynamic, but typically that doesn't make any difference to the operation of the dynamic. Even when dynamics are represented, typically they do not occur in virtue of their representation (though some do).

- A dynamic typically operates in many different domains. For example, if you do not deliberately sort your records and always return them to one end of your shelf, the same dynamic that sorts bowls will ensure that the records you listen to most often will be closest to the end.

- Beneficial dynamics often arise without their having been intended by the agents involved. This is not just an accident, but depends on subtle facts about the structure of activity which are beyond the scope of this discussion; see Agre's thesis [4] for some starting points. The bowl-sorting dynamic is a small example; it often saves you work because the bowl you want is more often at the top of the stack than chance would predict. More important examples come from the dynamics of apprenticeship, in which skills are often transferred by means neither intended nor understood by the expert or apprentice. For example, in any collaborative activity, it is sensible to divide up labor in such a way that the more experienced actor does the hard bits (which she's more likely to get right) and the less

experienced actor does the easier bits. This allocation of tasks is sensible independent of any intention to transfer skills, but it has the effect that the more experienced participant demonstrates competent performance of skills that are just beyond the less experienced participant's ability. This dynamic of unintended demonstration is centrally important in the transfer of skills to preverbal children [143], is a principal mode of skill transfer in many nonindustrial cultures [156], and plays an important role even in situations in which the actors understand themselves to be transferring skills via explicit teaching [30].

The concrete-situated approach shifts explanatory focus from things in the head to dynamics. We postulate new pieces of machinery only as a last resort; this is the principle of *machinery parsimony.* Machinery parsimony is simply good engineering. A well-designed radio is not one that includes every flashy new kind of circuit, it is one that gets the job done with the fewest components. Simple, elegant radio designs usually require a deeper understanding of the physical principles underlying radio operation than do brute force, kitchen-sink designs.

As I will show, powerful dynamics enabling efficient performance of difficult tasks can arise from some simple kinds of machinery. This can be hard to accept at first: in AI, we are trained to find complex machinery impressive, not to find complex explanations of patterns of interaction impressive. It is an easy mistake to dismiss Sonja as trivial because its machinery is uninteresting. What is interesting is the *relationship between* the machinery and its environment.

In studying a particular capability of an agent, AI research has usually begun by asking "What knowledge does the agent need to solve this problem? What machinery does the agent need to manipulate this knowledge?" Frequently, the answer is to propose a new module in the agent's head which gives the capability to the agent. Concrete-situated research asks "what patterns of interaction between the whole agent and its environment will allow it to engage in this sort of activity?" We have found that the deeper your dynamic understanding, the less machinery you need. Bits of machinery, when postulated, do not subserve particular capabilities; the entire agent is applied to every task. Dynamics and bits of machinery do not correspond one-to-one, as they often do in theories based on modularity assumptions. An engineering analogy may again be helpful: while in a crude bridge particular members may be responsible for supporting particular bits of the platform, in a sophisticated truss design the forces are so distributed and combined that every member contributes to supporting every section.

2.5 Routines

A *routine* is a regular, practiced pattern of interaction. Routines are what get work done. You may engage in routines of getting into your car, checking for new email, taking a shower, going for a run, typing your name, saying "hello" to someone you've just been introduced to, scratching your nose, choosing which sandwich to have for lunch, feeding your cat, or making a ski turn.

Routines can have arbitrary temporal extent; you may participate in a yearly academic routine and in a routine of writing your section number on the blackboard. One routine may be

made up of other routines: that is, you may routinely engage in one routine after another. This does not mean, however, that one routine "calls" another. Routines, like other dynamics, are not causal agents. The actions that make up a routine are not the *result* of routine; the agent performs actions for specific reasons which make sense at the time. For instance, you don't put bowls on the top of your pile because that will sort them, you put them there because that's where it's easiest to put them. Routines are not programs, are not bits of machinery, and are typically not represented by their agents.

The routine is the concrete-situated approach's unit of activity. This contrasts with stimulus-response theories, in which the unit of activity is an atomic response to a stimulus or perhaps a fixed action pattern, and also contrasts with mentalism, in which the unit of activity is a plan or other thing-in-the-head that controls activity. Routines are a better way of describing activity than stimulus-response pairs because atomic actions rarely make sense in isolation. Routines extend over time and involve many cycles of interaction with the world; a routine is often useful only because of its eventual outcome. Routines are a better way of describing activity than plans or other things-in-the-head because activity is interactive and does not typically conform in detail to representations the acting agents have of it. Useful explanations of activity must be dynamic, explaining patterns of interaction; explanations of what is in an agent's head are useful only if they relate what's inside to the world outside.

For us, the big question in designing an agent is, what sort of machinery will engage in the sorts of routines we want? The architecture we propose is not the only one that will engage in routines; in fact almost any system interacting with an environment will, because routines typically arise from physical determinism. If you put the same agent into the same situation twice, it will do the same thing. Even though real situations and real agents are never exactly the same twice, people and places change only slowly. Your desires, your daily routine, the arrangement of your office and kitchen, your route to work, and your relationship with your office mate are all relatively constant. An agent's activity can be mainly routine because the world is mainly routine.[6]

If any architecture will engage in routines, then, why make a fuss about them? Because we wish to examine rigorously the relationship between an agent's routines and its machinery. In many theories of activity, an agent engages in a routine in virtue of having a representation of it such as a plan or program or script. This is indeed one way to bring about routines; for reasons discussed at length in papers by Agre and myself [4, 7] we prefer other ones.

Because a routine is an pattern of interactive activity, one cannot literally *implement* a routine; one can only implement a device which will *participate in* a routine when put in a particular kind of environment. However, I will often speak loosely of implementing routines, meaning implementing such a device.

Because life is on-going and mainly routine, because trouble is rarely fatal, and because there are social and cultural supports for skill acquisition, routines can gradually *evolve* over a period of development. An agent rarely *has* to solve a problem that is beyond its routine capacities. It's almost always true that

[6]This means that the approach we advocate is inapplicable to worlds that are not mainly routine. We hypothesize that people would not have evolved as they did in such worlds.

- You've done much the same thing before.
- You don't need the best solution.
- You can do it the obvious brute-force way the first time and worry about optimizations later.
- You can cut corners.
- You can keep trying till you get it right.
- You can work around it.
- You can put it off.
- You can take another tack.
- You can get help.
- You can watch someone else do it.
- You can always just give up.
- You get to *choose* when to try the hard cases.
- So you get to choose when to learn new tricks.

2.6 Improvisation

An embodied agent must always be doing something, even if it is just resting. One way to continually act is to continually compute what is the most sensible thing to do at each instant. We call this approach *improvisation.*[7]

The alternative to improvisation is to spread computing what to do out over time. This approach entails keeping *control state*, which tells you where you are in the process and which computation to perform next. (The paradigm of control state is a program counter.) Control state is frequently also used to tell you what to *do* next, as when executing a plan. The concrete-situated approach advocates minimizing control state. Control state depends by definition on out-of-date information about the world. Out-of-date information about the world may not longer be correct. Steering your car to the left because the value of your program counter is 457 is risky; it's much better to steer your car to the left because there's a car in the lane to your right.

If you have enough computational power to improvise—that is, to avoid maintaining control state—it is better to do so. Our empirical studies of human routine activity convince us that improvisation is usually possible for people. Sonja demonstrates technically that goal-directed activity in at least one complex domain can also be improvised.

Sonja has no control structure. We will see that in spite of this Sonja can abort a routine when conditions change so it no longer makes sense, repeatedly try an action until it succeeds or some other alternative makes more sense, temporarily suspend one routine to engage in another, interleave two or more routines, and combine several routines to achieve multiple goals simultaneously. Such abilities are at the horizon of the state of the art of research on

[7] Not all the connotations of this word are intended: in particular, "improvisation" suggests novelty, and we are principally concerned with routine activity. We haven't found a better term, though. The intended connotation is that of spontaneity and responsiveness.

execution architectures (e. g. [80, 91, 160, 175, 202, 287]), which use elaborate control machinery to achieve them. These architectures need a lot of machinery to selectively break out of a basically sequential architecture, in which what you do at each moment depends principally on what your program counter reads. In Sonja all these abilities emerge automatically from computing what to do at each moment without maintaining control state. Sonja's architecture starts without sequentiality, so it needs no machinery to compute when to overrule a program counter based on sensory information.[8]

Improvisation demands that the computations subserving activity keep close to perception. If you are going to recompute what to do at every instant, you probably don't have enough time to build elaborate abstractions from your sensory input. You don't have enough time to do search or theorem proving or problem solving. You want your representation of your situation to be causally grounded in that situation.

2.7 Deictic representation

The demands of improvisation lead the concrete-situated approach to posit an unfamiliar form of representation. The sorts of representations we are used to are *objective*: they represent the world without reference to the representing agent. *Deictic* representations represent things in terms of their relationship with the agent.[9] The units of deictic representation are *entities*, which are things in a particular relationship to the agent, and relational *aspects* of these entities. For example, *the-cup-I-am-drinking-from* is the name of an entity, and *the-cup-I-am-drinking-from-is-almost-empty* is the name of an aspect of it. *The-cup-I-am-drinking-from* is defined *indexically* [22] in terms of an agent and the time the aspect is used. The same representation refers to different cups depending on whose representation it is and when it is used. It is defined *functionally*, in terms of the agent's purpose: drinking.

Deictic representations are defined in terms of a *causal* connection to the thing they represent. The literal *the-cup-I-am-drinking-from* is just a text string and of no use to concrete activity. The representation this string names is practically useful only if it plays a role in actually drinking from the cup.

The indexical, functional, and causal properties of deictic representations are what make them useful for concrete activity. The part of the world that is relevant to activity is that which is causally connected to the agent (indexicality). Representing things that are irrelevant to the task at hand would be inefficient (functionality). Causal connection with the things represented gives the agent the access needed to manipulate them.

We hyphenate the names of deictic representations because they are typically *noncompositional.* The implementation of *the-cup-I-am-drinking-from* will almost certainly *not* depend on a token representing "I" or on general representations for cups or drinking. These names are

[8] Production systems [190] are also motivated in part by the intuition that dependence on control state is the exception rather than the rule. However, most production system architectures end up with more seriality—and more complex machinery—than Sonja because of conflict resolution and working memory.

[9] Agre originally used the phrase "indexical-functional representation" for this concept, and it appears that way in several of our joint papers. He switched to "deictic representation" in his thesis, and I am following his usage.

a theorist's names, not the agent's names. In particular, they are not linguistic or symbolic entities in the agent's head.

We say that an agent *registers* an entity or aspect with which it has entered into the appropriate causal relationship. An agent will not register most of its entities and aspects most of the time; since I am typing, not driving, I am not registering *the-car-I-am-overtaking.*

Deictic representations are interactive: a deictic representation is not a thing in the agent's head, but a pattern of activity, possibly involving external actions, which allows the agent to get at things in the world which are relevant to its purposes.[10] The causal connection between deictic representations and the things they represent typically goes in both directions. Perception is the most common causal grounding in one direction: you can register an entity if you can see or feel it.[11] This makes questions about the nature of perception central to questions about the nature of representation. Consequently, much of the rest of this thesis will be about perception, particularly vision. Manipulation of the thing represented is the most common grounding in the other direction. Together these imply that rather than *modeling* the world, an agent can be *part* of it, letting signals flow through it. The agent uses the world directly as a source of information about the world.

Any theory of representation must provide an account of *abstraction*: how a representation which originally applied in one situation can apply in another. Familiar forms of representation provide abstraction by means of variables which can be bound to constants. You may have a representation that says that for all times t, cars c, and agents a, if at time t car c is coming directly at a, it is rational for a to head orthogonal to the vector from c to a at time $t+1$. You may also know that at time 34558, (which happens to be *now*), car C0786 is coming directly at agent A7787 (which happens to be you). Then by binding c to C0786, a to A7787, and t to 34558, you can deduce that it is rational for A7787 to move in a direction orthogonal to the vector from C0786 to A7787 at time 34559. Then some (necessarily) non-representational machinery can actually cause A7787 to make the appropriate movement.

Deictic representations provide a very different means for abstraction: they don't represent identity, only relationships. A sensible agent is wired up so that when it registers that *there's-a-car-coming-directly-at-me*, it will register *the-direction-to-the-car-coming-directly-at-me* and use that directly to get out of the way. In a situation like this, you don't care what time it is, you already know who you are, and you don't care which car *the-car-coming-directly-at-me* is, you just want to get out of the way. This *passive* account of abstraction requires no computation beyond that already needed for registration. Passive abstraction is a good example of a deeper understanding of dynamics leading to simplified machinery. Having understood that the interactions involved in concrete activity rarely require you to represent the identity of objects, but only your relationship to them, you can eliminate the machinery for unification and instantiation which any system using objective representations requires.[12] Rather than

[10] It is, in fact, unclear whether or not deictic representations should be called "representations" at all. They have some but not all the properties traditionally attributed to representations. Since there is no generally accepted technical definition of "representation" the question admits no definite answer.

[11] Another common way is via mechanical connection (as in *the-keys-in-my-pocket*).

[12] In rare circumstances, it is important to distinguish perceptually indistinguishable objects. In these cases, more complex schemes are required; some of these are compatible with the current theory, which only suggests

offering new machinery for controlling universal instantiation (which is usually combinatorially explosive), deictic representation offers a dynamic understanding which allows us to eliminate instantiation altogether: the machinery that registers *the-car-coming-at-me* never knows which car it is.

Because a deictic representation must be causally connected with its referent, part of its responsibility is to *constitute* an object. The real world is not neatly divided into discrete objects with identity labels on them. What *counts as* an object depends on the task. Is a window frame part of the window? It is when you are the carpenter nailing the window into the wall; it isn't when you are the cleaner squeegeeing it off. *The-window-I'm-nailing-into-the-wall* and *the-window-I'm-squeegeeing* have to do the work of separating their referents out from the surrounding world.

Some of the aspects of a situation that a sensible agent must register live in the future. In order to choose what to do, you must know what is likely to happen next, or what would happen if you took a particular course of action. We call determining probable futures *projection.*

One way to project futures is by *simulation*: you can build a model of the current situation and apply rules that transform it according to your expectations. Many theories of action privilege simulation as the sole method of projection and make machinery for simulation a central part of the architecture. However, simulation is generally expensive (and, as we'll see in section 3.1, incompatible with neurally plausible hardware). Luckily, there are several alternatives. It seems unwise, then, to make simulation the only means of projection.

Sonja projects by *visualization*, using visual machinery to literally *look to see* what will happen next. The detailed mechanisms of this are discussed in chapter 8. Still other means of projection are possible; a more sophisticated agent ought to incorporate several.

We will see in section 3.3.6 that deictic representations can be efficiently implemented in neurally plausible hardware. We'll see in section 3.1.2 that there are reasons to doubt that objective representation can be so implemented.

Deictic representation is not the only sort of representation people use, nor is it intended as a *general* theory of representation. The instructions given to Sonja, for example, are not deictic representations. There are many propositions that can not be stated in a deictic representation scheme. There are purposes for which a photograph, a visual image, a shopping list, or an equation is the appropriate sorts of representation.

---

that it is better to avoid representing identity when possible.

Chapter 3

Hardware and architecture

This chapter discusses the theories of *hardware* and *architecture* that motivate Sonja's design. I use these terms as they are used by computer designers: hardware is the stuff out of which machinery is made; architecture is the way it is organized. I argue that some simple and well-established facts about the brain put strong constraints on cognitive architecture. These constraints are violated by much current work in AI. If one's interest in AI is the engineering of intelligent systems by whatever means, this may not be cause for concern. Sonja, however, is intended to illustrate a *biologically plausible* theory of activity, and its design is based on neuroscientific and psychological evidence.

Section 3.1 describes "essential connectionism," a set of facts about neural hardware that are well established, constrain architecture, and are obeyed by Sonja but not by some other cognitive models. These facts suggest avoiding the use of pointers, variables, inspectable data structures, interpreters, and dynamic storage allocation. Section 3.2 argues for a particular top-level modularity for the brain, dividing peripheral systems from a central system, and argues that the central system itself is not modular. It proposes a particular provisional non-modular central system architecture. Section 3.3 concerns the architecture of the visual system. It argues for a particular model of visual processing, the visual routines theory. This theory too puts strong constraints on cognitive theories; it would be difficult and perhaps impossible to interface most current cognitive theories of central systems processing with this model of vision, because it delivers different sorts of representations than they require as inputs. I propose a particular interface between the visual and central systems.

3.1 Essential connectionism

The kind of hardware an agent is built out of puts engineering constraints on its architecture. If computation is cheap and communication between computational units is expensive, for instance, a Von Neumann architecture makes sense. If computation is expensive and communication is cheap, other architectures make more sense.

Much of the architecture of the human brain is still mysterious, but some basic facts about human brain hardware are certain. These facts are playing an increasingly large role in AI with

the increasing popularity of connectionism. I believe that these facts and their implications, which together I will term *essential connectionism*, ought to be of central importance in cognitive science, because they put strong, surprising constraints on cognitive architecture. Sonja respects these constraints.

3.1.1 The essential connectionist commitments

The essential connectionist facts[1] are that the brain

— 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.[2]

This organization is strikingly different from that of a Von Neumann computer and also from cognitive models such as inference engines-with-databases. It implies that computations that operate in a second or less must involve no more than about a hundred sequential steps [76]: each step must involve a neuron firing, which takes a few milliseconds. Almost all perceptual computations operate in under a second. So do many action choices: video game players routinely respond in less than a second to changing circumstances; conversational interactions involve coordination processes which also operate in under a second. These are complex computations which must involve much more than a hundred neuron-like computations, implying that massive parallelism must be used [76].

These facts make most of the software technologies AI systems have mainly been built out of prohibitively expense and suggest that alternative techniques must be found. In particular, we must find ways to avoid using pointers, variables, inspectable data structures, interpreters, and dynamic storage allocation. I'll explain why in the next section.

3.1.2 The essential connectionist constraints

Pointers are expensive

Any computation involves computational units (data items, for example) that must talk to each other. Any computational architecture must provide a communication scheme, or router, that allows them to.

One way to allow computational units to talk to each other is via connections. If two units need to talk to each other, you literally wire them together. It is possible to implement many sorts of computation this way; many of the purposes to which semantic networks have been put can be implemented with connections, for instance [74, 75, 244].

[1] These facts are reviewed in any introductory neuroscience text, for example that of Kandel and Schwartz [138].

[2] There are also global interactions based on chemical diffusion. The role of these interactions in neural computation is uncertain; I ignore them in this thesis.

Most AI programs use a more flexible means of communication: pointers. Pointers are more general than connections; they allow *selective* communication between arbitrary computational units. To implement pointers, a communication scheme must allow a piece of hardware to talk to different other pieces of hardware at different times. Pointers are, by definition, side-effectable.

Pointers can be implemented easily on a Von Neumann machine. The computational units are arranged in a star configuration, with the ALU at the hub and memory locations on the rim. Most units don't directly talk to each other, they talk to the ALU, which combines two at a time. This has a high cost, however. Only a tiny fraction of the hardware in the machine does anything at any given time: the ALU and whichever memory locations are being accessed. (Pipelining and memory hierarchy improve this situation only by a constant factor.) Almost all of the hardware is devoted to memory, which just sits there, and to addressing logic and memory management, which just connect memory locations to the ALU, rather to than computing anything useful [118].

If your hardware computes slowly, as does the brain's, this approach won't work. You've *got* to get millions of computational units doing useful work at any given time because individually they are useless. But this makes the communication problem much harder. If you have n computational units that can point to each other, you have to implement n^2 communication paths. Some sort of crossbar or router is needed. It's not clear how many computational units are needed for human intelligence, but a million seems like a lower bound [118]. We don't *know* that the brain doesn't include an n^2 crossbar of this size, but the neuroanatomical evidence is generally against it [48]. As an engineering matter, building a trillion-connection crossbar seems hard; if it is a prerequisite for intelligence, it will be a long time before we'll see progress in AI. We would do well to look for alternatives, both for practical reasons and because it is likely that if there is a way to avoid this quadratic overhead, evolution would incorporate it into the human cognitive architecture.

Finding an efficient implementation for pointers has been termed the *variable binding problem* in the connectionist literature [192]. All the approaches I know of require at least n^2 hardware.[3] This is an area of active current research, and a better solution may be found, but it seems likely to me that we need instead to find architectures that use pointers as little as possible. The architecture proposed in this chapter and implemented in Sonja is one. Such an architecture is evidence that those who dismiss connectionism on the grounds that it does not support pointers and so can not implement cognition [86] may be mistaken.

Having given up the full generality of pointers, which allow anything to talk to anything else, we might go to the opposite extreme and adopt an architecture in which each processing element can only talk to a small fixed set of neighbors. Although I think this is worth exploring (as it has been by connectionists and others, for instance Brooks [27]), I think a middle ground is possible. We may want to build small special-purpose routers when the number of things that need to communicate form a small subset of whole system. Sonja, for instance, uses addressing logic to access its retina; see section 6.1.2.

[3] Touretzky and Hinton [264, 265, 266] have described an elegant approach that (as I've argued elsewhere [34]) seems to require n^3 hardware. Drescher [59] provides a straightforward solution that is n^2. There are many more in the literature.

An architecture in which pointers are expensive requires deeply rethinking AI engineering issues. Without pointers, you can't bind variables to symbolic values, so you can't implement unification, so you can't use objective representation schemes which require universal instantiation. Without pointers, you can't have inspectable, compositional data structures that can be passed around freely and taken apart and put together in different places. Without pointers, you can't build or traverse graphs. Most well-understood AI technologies depend on one or more of these abilities. We need to find alternatives; Sonja provides some.

Dynamic storage allocation is expensive

Dynamic storage allocation, or "consing," is the ability to find unused computational unit and connect it to a given one. Dynamic storage allocation is extensively used in AI systems to create new representations in the course of activity.

Standard dynamic storage allocation techniques require pointers. There are alternatives that are better suited to essential connectionist architectures. In an essential connectionist architecture, finding an unused computational unit is easy; since all computational units can be computing all the time, one thing they can do is to keep track of whether they are in use. The hard part is creating new connections. Feldman [71, 72] describes *recruitment* schemes for doing so. These schemes would not be a good way to implement pointers, because they are slow and allow only a relatively small number of units to ever communicate with a given one. They are unlikely, therefore, to support creating large numbers of new representations in the course of routine activity. They are plausible candidates for the basis of long-term changes, as might be required by learning, for example.

I can now make precise the hypothesis of section 2.1 that routine activity is that which does not require "new thoughts." The hypothesis, technically, is that all routine activity can be engaged in by an architecture with a fixed, limited connection pattern. By fixed I mean that no new connections are formed and that limited I mean that the architecture does not make extensive use of pointers. Sonja is an existence proof that at least *some* complex routine activity can be implemented in such an architecture. Appendix B talks about thinking new thoughts and making new connections.

Virtual machines are expensive

Virtual machines, or interpreters, are a staple AI implementation technique. They allow an architecture of one sort to act like an architecture of another sort. This simulation ability has been used as the basis of arguments that facts about the brain can be ignored by cognitive scientists because however the brain works at a low level, it can simulate the architecture of your choice at a high level. I think such arguments are wrong. The fastest RISC machines, for instance, have a gate depth of ten; executing an instruction takes ten times as long as the basic gate delay. These machines are optimized for gate depth; most virtual machines have a simulation cost much greater than ten. If neurons compute at less than 1 kHz, any virtual machine would compute at well under 100 Hz. While such a machine might be used to implement slow processes such as conscious thought, it couldn't run fast enough to serve as the basis of

concrete activities such as video game playing. Such activities must be implemented directly; computation must stay "close to the hardware" rather than involving layers of abstraction.

3.1.3 Advantages of essential connectionist hardware

Thus far I have described essential connectionist hardware as a liability. Yet it has many advantages. Later in this chapter, I'll show that this sort of hardware naturally and efficiently implements the kinds of dynamics—highly interactive routines and deictic representations—which I believe for other reasons must underlie concrete activity. The massive parallelism it provides is required to sustain the massive signal bandwidth which must pass through a situated agent. Von Neumann machines simply can't keep up with the demands of real-time vision. Similarly, they don't have time to take into account the countless factors that are relevant to computing, in each instant, what to do. The sections of this chapter starting with 3.2 are devoted to showing that essential connectionist hardware can.

3.1.4 Alternative implementations of essential connectionism

Most connectionists, in additional to the essential connectionist commitments, hold some variant of a particular model of the computation performed by the individual neurons. This model supposes that

- neurons are connected randomly or uniformly
- all neurons perform the same computation[4]
- 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.

Unfortunately, these propositions are not neurophysiologically plausible in detail (see Crick [47] and Crick and Asanuma [48] for a discussion of the relevant neurobiology and an evaluation of the faithfulness of connectionist models).

I have adopted an alternative essential connectionist implementation medium, *depth-bounded digital circuits*. Like neurons and connectionist units, a digital circuit is made up of many components (gates), which perform simple computations based on information derived from the components they are connected to. A depth-limited digital circuit is one in which paths between the inputs and the outputs are limited in length. Depth-limited circuits must use broad parallelism to compute anything interesting, because individual paths through them are too short to do much work. Such circuits can do useful work even with large gate delays (corresponding to slow neural computation).

Depth-limited circuits are not a plausible model of neural functioning in detail, but they do reflect the essential connectionist constraints, so that a designer working with such circuits has

[4] These first two assumptions are rejected by increasingly many connectionists; Feldman [75] discusses of some of the relevant issues.

to work around most of the same problems a designer working with other connectionist hardware has to work around. In fact, they are equivalent within a constant factor to any other essential connectionist hardware technology in which each component does limited computation. For example, a standard connectionist unit can be simulated by a small digital circuit (assuming limited-precision arithmetic) and standard connectionist units can similarly simulate gates.

The advantages of digital circuits are that they are well understood, have existing design methodology to draw on, and can be simulated more cheaply than real-valued connectionist units. Adopting them is, however, an engineering move only, and not a theoretical commitment. I have, in fact, used real-valued components in some parts of Sonja where they seemed useful. My circuit simulator does not distinguish analog and digital components, so these can be thought of as having been implemented either way.

3.2 The central system

3.2.1 Modularity

It is common in cognitive science to make a first division of the mind and brain into *peripheral systems* and *central systems*. The peripheral systems are concerned with sensory and motor processing. The central systems are concerned with cognitive processes such as learning, reasoning, and planning. This division is almost universally accepted (an interesting dissent being that of Brooks [25, 26, 27]). I will adopt a version here also. This version is similar to that proposed by Fodor [85], and I will make use of his arguments for it here.

The reason to believe in the central/peripheral split is that much evidence (some of it reviewed later in this chapter) shows that perceptual and motor processes are in large part innate, immutable, localized to specific brain areas, and task- and domain-independent. There are, moreover, good engineering reasons for these facts. On the other hand, people and other intelligent agents learn, so there must be some other part of the brain that is the locus of learning, development, and storage of task- and domain-dependent skills.

The modularity of the peripheral systems (that is, how many and which ones there are) is becoming clearer as a result of neurophysiological and psychophysical studies. The modularity of the central systems is less clear. Fodor argues that the central systems are *not* modular; that, in fact, there is a single homogeneous central system. His main argument (much simplified) is that anything you know can potentially be brought to bear on any cognitive task, so there's nowhere to draw modularity boundaries.

I've come to the same conclusion in different ways.[5] In a previous attempt to build an integrated agent, combining planning, acting, learning, and reasoning [31, 32], I found that research from these AI subareas each saw their job as transforming representations in one format into representations in another format, but that the interfaces didn't match up. Natural language would produce a "semantic" representation of an English string, but no other module wanted that representation as input. Planning wanted "goals" as inputs, but no one else

[5] Brooks's work is motivated by similar observations [27].

supplied them.[6] Planning thought that perception was supposed to deliver a different sort of representation than perception was willing to deliver. Every theory of learning had different ideas about what learning's inputs and outputs were supposed to be. In sum, connecting together the modules most frequently proposed in cognitive science research is hard. It may be, however, that a different modularity is possible, or that further research on connecting together modules will succeed.

Another reason to avoid modularity is that much of the work required by the modules seems to result from the modularity boundaries themselves. Each module, since it doesn't have access to the workings of other modules, has to solve problems in the general case, which is usually much harder than solving specific concrete problems. Each modularity boundary implies representations that are passed across it; these representations typically require pointers, which are expensive. A good example of these difficulties is provided by the modularity boundary between planning and execution. This boundary requires that a planner solve problems without access to ongoing circumstances. It implies a representational interface, plans. Agre and I have argued that eliminating this modularity boundary makes sensible action much easier [7]. An agent without this modularity doesn't have to solve the provably intractable problems a planner does. Plans, which require pointers to represent, can be eliminated.

These considerations argue for a single, homogeneous, nonmodular central system, which is what Sonja has.

3.2.2 A three-phase central system architecture

I don't have a good theory of how the central system works. I've adopted a scheme, described in this section, which is adequate for Sonja and will probably generalize some. I make no theoretical claims for it.

In concrete activity, the job of the central system is to compute what to do. Routine activity, according to the hypothesis of section 3.1.2, involves no allocation of new processing elements or connections. This suggests statically allocating hardware for each task. Since these bits of machinery are permanently allocated to specific tasks, there's no issue of freeing up hardware to allocate to the current task, so it makes sense for all these bits of machinery to keep thinking all the time; otherwise they will simply be idle.

Sonja's central system, then, is composed in part of *proposers*, bits of circuitry which each continually figure out whether it would be sensible to do some particular thing right now. (There are many similar schemes in the literature, perhaps most famously Minsky's Society of Mind [180].) Proposers make *proposals* for action. Since proposals are domain- and task-specific, there are as many proposers as things you might do. Proposers thus effectively exploit essential connectionist hardware by keeping most of the hardware working most of the time. This parallelism, I argued earlier, is a requirement on any cognitive architecture that respects the constraints of essential connectionism. Because proposers are allocated statically, the architecture scales well: each new ability the agent acquires requires new proposers only in proportion

[6] Some natural language understanding systems (e. g. Allen's [9]) transform requests or commands into goals, but this is not enough to keep an autonomous agent going; most goals must be generated internally.

to the complexity of that ability.

A proposer often proposes actions which are incompatible with those suggested by other proposers. *Arbiters*, another kind of central system component, choose among suggested actions (figure 3.1). Collectively the arbiters form an *arbitration network.*[7]

Arbiters and proposers typically both take into account the situation at hand, information about which comes from the sensory systems. However, the information coming from the sensory systems is in sensory terms, not in terms of the current task. *Registrars*, the final sort of central system component, interpret sensory inputs in terms of relevant aspects of the situation. Registrars and arbiters, like proposers, are small circuits that all operate continuously and in parallel.

Is this three-phase architecture right? What is the internal structure of the components? How are they organized? Not enough is known about either activity or brain structure to answer these questions. Lacking such answers, I've fallen back on engineering intuition. I believe that ultimately criteria of learnability will provide most of the constraint on the nature of the central system. Agre's thesis uses this constraint to argue for a particular sort of additive construction of the central circuitry. Neither Pengi nor Sonja is constructed in this way, because it is not a very strong constraint, and because Agre leaves open the question of what adds the circuitry and how. I believe that the central system must incorporate a statistical induction engine, and the nature of this engine probably will provide strong constraints when we understand it. (This point is taken up again in appendix B.)

Even apart from these questions, it is clear that this model of the central system leaves many things out. There is no model of memory, for example. I've resisted the temptation to add one because I don't think enough is known about memory to choose between incompatible competing models and because the best models are so complex that implementing them would have been a thesis in itself.

3.3 Architectures for vision

This section argues for a particular theory of vision, the visual routines theory. Section 3.3.1 argues that vision must be task-specific in part. Section 3.3.2 argues that task-specificity is not by itself a sufficient constraint on vision for creatures much more complex than insects. Section 3.3.3 introduces some terminological distinctions which allow section 3.3.4 to describe and argue for visual routines. Section 3.3.5 describes the implications of the visual routines theory for the central system and explains the nature of the interface between the visual and central systems. Finally, section 3.3.6 explains how visual routines can implement deictic representations.

[7]The function of this arbitration network is superficially similar to that of the conflict resolution phase of a production system [172], but is actually quite different. Production systems are intended to model slow, serial cognitive processes [190]. The job of the conflict resolution phase is to enforce seriality by choosing just one of the applicable productions to fire on each cycle. Sonja's central system architecture is intended to model fast parallel subconscious processing. Typically many proposals for action are adopted simultaneously; by default there is no conflict. Arbitration is required only when two proposers suggest incompatible courses of action.

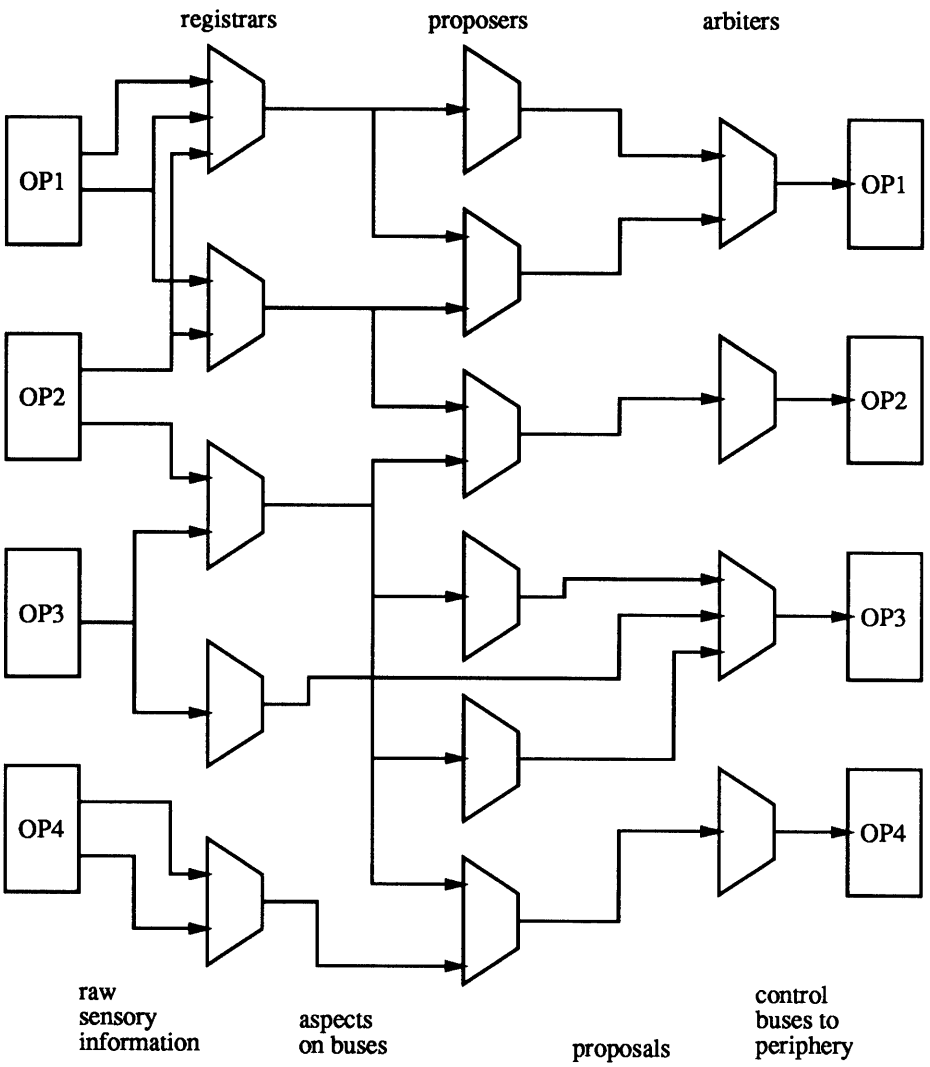


Figure 3.1: The three-phase architecture. In this diagram, trapezoids represent small circuits and lines represent buses connecting them. Registrars compute aspects of the situation based on inputs from sensory operators. Proposers suggest actions based on these aspects. Arbiters choose between conflicting actions, often based on information provided by registrars (as in the case of the third arbiter from the top). The outputs of arbiters are vectors of control signals that go to peripheral operators. Registrars, proposers, and arbiters are distinguished only by their role in computing what to do; there are no restrictions on what form the circuits that implement them may take.

3.3.1 The purpose of vision

Much of the vision literature supposes that the purpose of vision is to deliver a complete model of a scene. This model, similar to those used in CAD systems, would identify all of the objects visible in the scene and specify their shapes, locations, and orientations. Most AI literature concerned with cognition also supposes that vision delivers complete scene models, though of a somewhat different sort, one that describes properties of objects and relationships between them in an expressional representation such as predicate calculus ground terms. (This project is sometimes termed "pixels to predicates" [203].)

I think that scene models such as these are neither necessary nor generally very useful, and that they are much harder to generate than the information that vision actually needs to extract. I will propose a different model, after arguing against these alternatives. Most of these arguments have been made earlier and at greater length by Ballard [20], Edelman and Poggio [63], and Horswill [125].

- The models these theories propose are constructed independently of the specific purposes to which the seeing agent will put them. In the concrete-situated approach to activity, you are almost always engaged in some specific activity, and the purpose of vision is to support that activity. Different sorts of objects are important in different sorts of activity; most of the things you see at any given time are irrelevant to whatever you are doing, and recognizing and locating them would be wasteful.

- As we saw in section 2.7, what counts as an object, and the boundaries of objects, depend on your purposes. As another example, the top of a paint can should be considered part of the can when you are carrying paint around, but a separate object when you are dipping a brush in it. This too suggests that objects cannot be recognized in a domain-independent way, and that the viewer's goals must be taken into account by object recognition.

- Extracting precise metrical information and surface descriptions from images is a principal concern in machine vision. It seems to be a very hard problem, in principle as well as in practice; it is numerically unstable, computationally expensive, and depends on finding "constraints" which seem to be hard to come by [125]. It seems likely that going to such trouble is not necessary. It's rare for activity to depend on precise metrical information; work in robotics [168] shows how compliance (sliding) can achieve precise positioning without precise measurements.

- Specific concrete activities demand that you know particular aspects of a given situation. For example, in playing Amazon, you'd like to know which monster is nearest to you, roughly how far away it is, and whether you've got a clear line of fire at it. These aspects are *relational* in character, and they would not be made explicit by the sort of CAD model that most vision research proposes. Such a model *cannot* make relational properties explicit, because there are too many (vastly more than there are properties of individual objects), and a model would have to make *all* of them explicit because it is constructed without reference to the seer's purposes. Although relational properties *are*

made explicit in the sorts of representations cognitive AI researchers hope to get from vision, the same combinatoric arguments apply. In practice, since cognitive AI almost never connects with vision, the researcher selects the relational properties needed for the task.

- Computing the relevant relational aspects would require much additional work beyond that of constructing the model. It is, furthermore, not clear that the model would be much help in computing these aspects; it is not clear that some other representation wouldn't be much more useful.

- The constant terms involved in an expressional representation (such as `block-7691`) refer to specific, objectively given individuals. A visual system which generates such representations must determine the identities, as well as the types, of objects. This is usually difficult and often impossible. The visual system would have to provide a different constant symbol for every corn flake in your bowl and keep track of which is which. The world is full of indistinguishable things, and we clearly don't distinguish as many things as we could.

3.3.2 Insect vision

Why do we need vision, if not to produce a model of the world? We need vision because we want to act, and action must be responsive to environmental conditions. The considerations of section 3.3.1 suggest that vision must be *task-specific*, at least in part. One approach to visual system design is to posit that the doctrine of visual task specificity, when added to engineering intuition and the constraints of essential connectionism, is sufficiently constraining. This approach does not mandate a division between visual and non-visual processing, or the central/peripheral split. The way to go about building creatures, on this view, is to just "wire them up to do the right thing" (figure 3.2) This is roughly the position taken by Brooks [25], though he adds an additional architectural principle of layered design.

Perhaps the most interesting example of this approach is Connell's mobile robot Herbert [44, 45]. Herbert navigates about an office environment collecting Coke cans. This task requires several different sorts of visual perception and corresponding sorts of spatial reasoning, for navigation, can recognition, arm motion, and grasping. These sorts of perception and reasoning are implemented in Herbert by independent task-specific modules. The architecture has no central/peripheral split; individual modules or small clusters of modules are each responsible for both certain forms of perception and related forms of action. (I'll describe Herbert further in section 7.4.)

This is an approach for which I have a great deal of sympathy. I believe, in fact, that it is entirely appropriate for insects. Insects have a fixed wired-in repertory of activities they can engage in. I don't think it will work for modeling people and other more sophisticated animals who can learn new skills, and I have not adopted it in Sonja. I'll explain why in section 3.3.4.

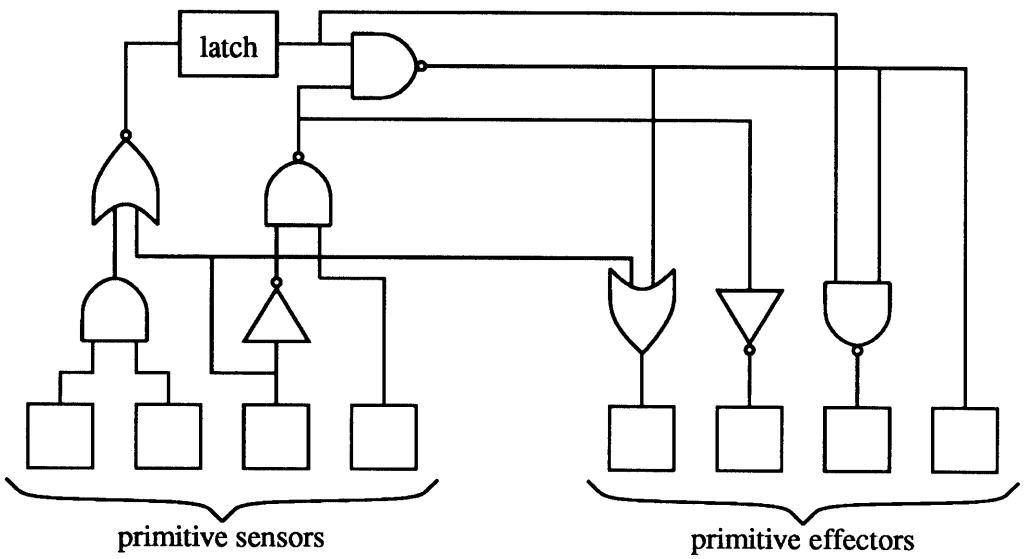


Figure 3.2: An agent wired up to do the right thing. In this approach, the agent consists of some unstructured circuitry interposed between sensors and effectors.

3.3.3 Some distinctions

Before we can go further, I need to introduce some terminology.

Bottom-up and top-down

Bottom-up visual processing is that which depends only on the retinal image. A process is bottom-up if and only if the same computation occurs whenever the same image is presented. Bottom-up processing, thus, can not depend on any non-visual contextual factors, on memories or other state, or on what it is the creature is up to. It is generally agreed that both bottom-up and non-bottom-up visual processing are necessary. Efficiency is the main reason that vision cannot be entirely bottom-up. There are too many different sorts of things that *might* need to be computed, depending on what matters to you. This argument has been made rigorous by Tsotsos [273, 272, 274], who uses counting arguments to show that the human brain does not have nearly enough hardware to perform shape matching for object recognition using bottom-up computation only. Thus, what visual processing you do has to depend on what you are looking for. On the other hand, as I'll explain in section 3.3.3, there is strong neurophysiological and psychophysical evidence that much of visual processing is bottom-up, so a combination is evidently involved.

Early, intermediate, and late

In this thesis, I will distinguish *early, intermediate*, and *late* visual processing. Although these terms are current in the literature, and "early" vision is well-defined, different researchers use the terms "intermediate" and "late" vision differently.[8] I will give these terms fairly precise meanings which roughly align with current practice and which express distinctions important in the design of Sonja.

The retina is a two-dimensional array of light sensors. Extensive neuroscientific evidence shows that the optic nerve (which connects the eye with the brain) and the parts of the brain to which it is immediately connected preserve this retinal topology [169]. This *retinotopic* neural processing comprises *early vision.* We also know that the body's effectors are controlled by small non-retinotopic nerve bundles. So there must be some point at which retinotopic processing stops and some other representation begins. The evidence suggests that this transition occurs fairly early on in processing, well before the level of motor control. It seems plausible that the later representation consists, at least in part, of *narrow buses*, consisting of small groups of neurons which together represent a particular global property of the image. This is corroborated by neurophysiological evidence; for example Perrett et al. describe experiments that suggest that individual monkey neurons respond selectively to faces [205]. (The interpretation of these and related experiments is still controversial; Maunsell and Newsome [169] provide a review.) Feldman [75] presents engineering arguments and neurophysiological evidence that encoding is probably predominantly in terms of small groups of neurons, rather than individual ones. Let us therefore term visual processing which uses these narrow encodings *late* vision, and whatever the process is which reduces retinotopic to narrow encodings *intermediate* vision (figure 3.3).

Early vision is coming to be quite well understood [120, 167, 169]. In addition to being retinotopic, early vision is bottom-up and applies uniformly and in parallel over the image. It produces "unarticulated" output representations, typically a single continuously variable or boolean value at each point in the image. Early visual processing is performed by a set of fixed, innate, retinotopically organized machines called *early maps.* The identity and function of some of these maps are known; new maps are still being discovered and the functional properties of some remain to be determined. There appear to be around fifteen maps; among them are ones that compute color, edge orientation, stereoscopic depth, and various properties of motion such as speed, direction, and size change.[9] These claims about the nature of early vision are by now backed up by much neuroscientific and psychophysical evidence [169].

Intermediate and late vision are less well understood. There is little applicable neuroscientific evidence and the computational constraints are less clear. The approach taken to them in this thesis is, accordingly, exploratory and speculative in some respects.

[8] The term "early" vision originates with Marr [166]. The terms "low-level" and "high-level" are also commonly used to express similar concepts.

[9] Early work by Zeki [291, 292] suggested a one-to-one correspondence between retinotopic maps and types of visual information. It is now known that the correspondence is many-to-many [169]. I will ignore this complexity.

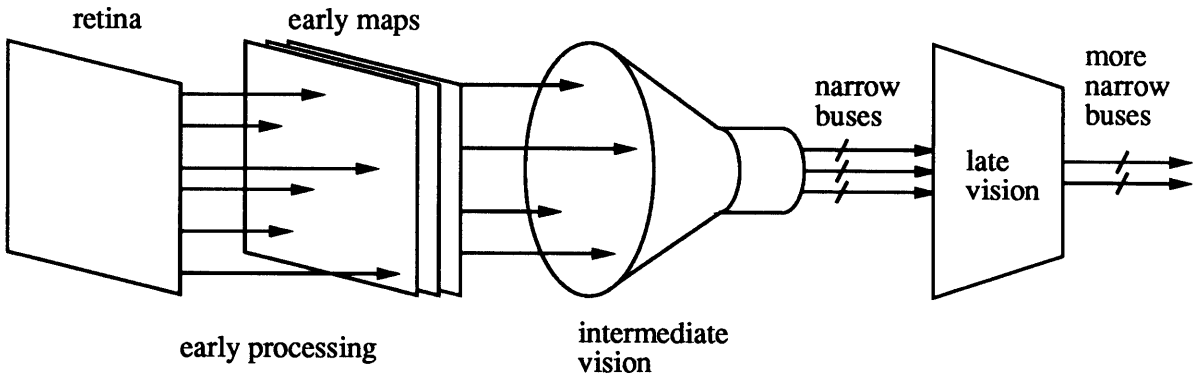


Figure 3.3: Early, intermediate, and late vision. Early processing computes, point by point, retinotopic maps from the retinal image; intermediate vision crunches these maps down into narrow buses; late vision computes with narrow encodings on such buses.

3.3.4 Visual routines

People regularly learn new skills. Since vision is task-specific, this means that learning to pilot a glider or play a new video game or make pancakes involves building new visual machinery. Unless some strong constraints are put on the architecture of the visual system, I believe this is a very hard problem. If vision were made up of completely independent task-specific modules, as in the insect vision model, learning a new task would require building a whole new pathway between the primitive sensors and effectors. As well as contradicting neuroanatomical evidence, this seems like an impossibly difficult learning task. More generally, the learning task of building new visual hardware seems difficult, and I believe that ameliorating this problem should be considered one of the most important constraints on visual architecture.

I believe that this constraint must be satisfied by including in the visual system a clean set of relatively high-level primitives which can act as a tinkertoy-like kit from which new visual machinery can be built. Making the primitives high-level means that the learning combinatorics are much reduced, and making the set "clean" makes it easier to discover useful combinations of primitives. (The lesson here is similar to that of Eurisko [157]: learning is easy if you have a set of primitives which generate a space in which useful combinations are relatively dense [158].)

A model of this sort, the *visual routines model*, has been proposed by Ullman [276], although learnability was not among his concerns. This model posits a small, innate set of *visual operators* which perform particular sorts of visual work. These operators collectively constitute a *visual routines processor (VRP)*. The VRP is not the only visual system (I'll discuss other ones in section 6.7), but it is the one most involved in acquiring new visual skills.

The visual routines theory posits *visual routines*, patterns of activation of different visual operators over time. As an example, Ullman describes a visual routine for computing whether

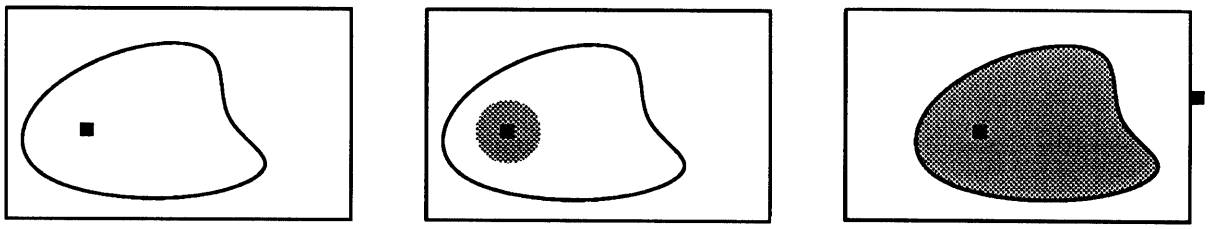


Figure 3.4: A visual routine for computing containment. Starting from a point marked with a solid square (first picture), a wave of "coloring" is spread (second picture). The wave stops when it hits a boundary. Then, a "point at infinity" is tested to see whether it is colored (third picture). In this case, the point at infinity is marked with another square; it is not colored, and so the first square must have been inside a boundary.

or not a particular point is contained within a closed curve in the image. This routine has two steps. First, a wave of "activation" is propagated in the image, starting from the point of interest, and expanding in parallel in all directions, but stopping when a boundary is reached. (See figure 3.4.) This computation can be performed efficiently on a parallel two-dimensional grid machine. Second, a "point at infinity"—any point that is for some reason guaranteed to lie outside any curve—is tested to see whether it has been activated. If it has, we know that the activation has "leaked out" of any surrounding curve, and that the original point is not in fact contained in a closed loop. If it hasn't, we know that there is a containing curve.

This example is rather abstract. You might also have a visual routine for checking your speedometer, for glancing at your keyboard to align your fingers in home position, for checking a pancake to see if it is ready to flip, or for finding safe footing when walking in the mountains.

The containment routine involves two specific visual operators. I won't talk more about what other operators there are until section 6.6. However, I can describe some of their general properties.

The visual routines theory is a theory of intermediate vision: it describes an interface between early vision and late vision, so the job of the visual operators is to crunch retinotopic maps down into narrow buses. A visual operator is a piece of hardware dedicated to one particular sort of crunching. In the most general case, illustrated in figure 3.5, a visual operator takes its input in the form of one or more of the retinotopic maps and in addition one or more control signals from the central system; maintains state, which typically is shared with other operators; and produces as output to the central system one or more narrow buses. Operators can be degenerate in all the corresponding possible ways: an operator may not depend on retinotopic maps, or may not take control inputs from the central system (in which case it operates *bottom-up*), or may not depend on state, or may not side-effect the state it does use, or may not produce any values for the central system, but rather operate by side-effect only.

Early vision is uniform, retinotopic, and local. Intermediate vision, since it is producing *narrow* values from retinotopic images, must compute *non-local* properties of images: that is,

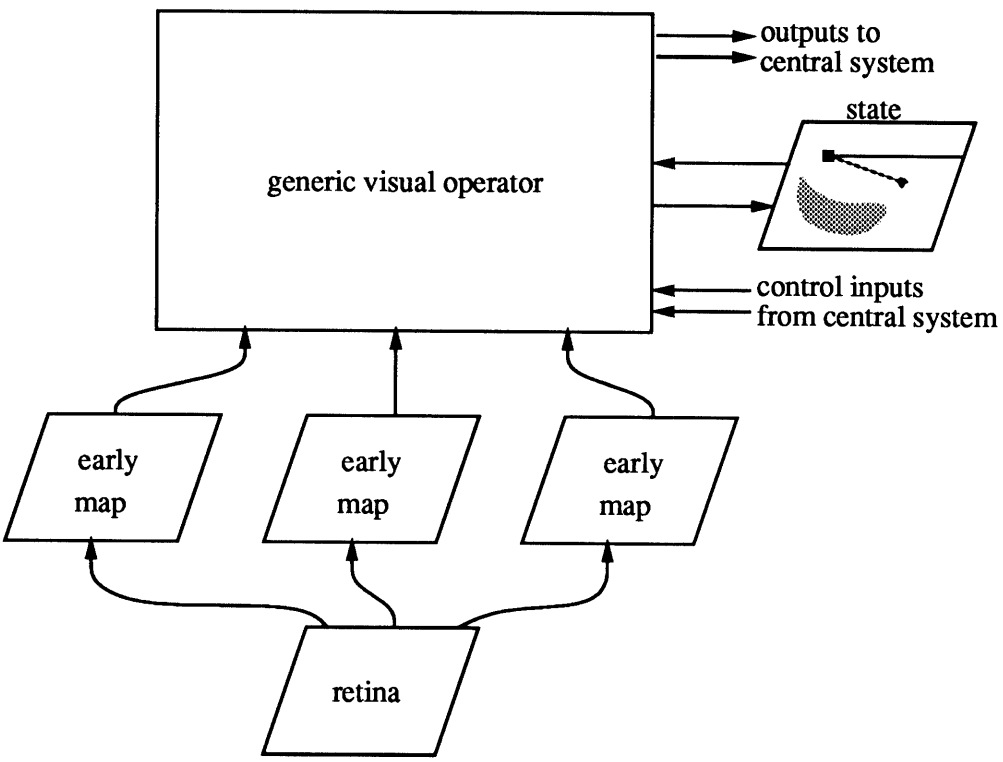


Figure 3.5: A generic visual operator. The visual operator, on the basis of control inputs from the central system, produces narrow outputs from the retinotopic maps. It may maintain some state, which can be shared with other visual operators.

it must perform computations which depend on large parts of the image. Furthermore, while it is logically possible that intermediate vision would also be uniform (so that all parts of the image would contribute equally to all intermediate computations), it seems clear that much of vision requires analyzing specific subparts of the image which correspond to distinct objects in the world. Thus it's also the job of intermediate vision to extract *spatially-specific* information from chosen parts of the image. This means that there must be operators which are capable of *finding* interesting subsets of the image, and others which are capable of *manipulating* them.

3.3.5 Implications for the central system

Ullman says little about what is responsible for causing the right visual operations to happen at the right times. He writes about visual routines as programs which are executed in the usual way. He talks briefly about a process of compilation which synthesizes new routines from specifications, and suggests that these routines are stored for later use. He seems to suggest that this compiler is in the visual system, and so presumably specific to vision. This compilation process is in fact akin to classical planning: it produces programs (plans) from specifications (goals) and stores them for reuse. In section B.1.2 I'll argue that this is a bad model for skill acquisition in general; the same considerations apply to visual activity.

Agre has suggested reinterpreting the notion of visual "routines" in terms of his use of the word "routine" rather than the use as "program fragment." In this model, visual routines are not programs to be executed, but rather patterns of activation of visual operators, just as routines more generally are patterns of activity. Agre has further suggested that rather than having special machinery for executing and synthesizing visual routines, that visual operators could be activated directly by the central system.

These suggestions amount to some strong claims:

- The interface between the central and visual systems consists of a fixed set of buses, with visual information running from the VRP to the central system and visual commands running from the central system to the VRP. It seems plausible to generalize this model to other peripheral systems.
- Late vision is not distinct from cognition, and is supported by the same central machinery. Accordingly, I draw the line between peripheral and central systems rather differently than most researchers; I put more of the responsibility for perception into the central system than is usual.
- The central system must be able to provide new control outputs for the sensory systems within tens of milliseconds after getting new inputs from the sensory systems, because visual routines may involve several cycles of interaction and operate in well under a second. Sonja's central system architecture satisfies this constraint; others do not.

For purposes of vision, the central system's job is to get the right arguments to the right visual operators at the right times. The architecture of figure 3.6 is in many respects similar to that of a horizontally microcoded computer. The central system plays the role of the control logic and the peripheral systems play the role of the datapaths. The central system has narrow,

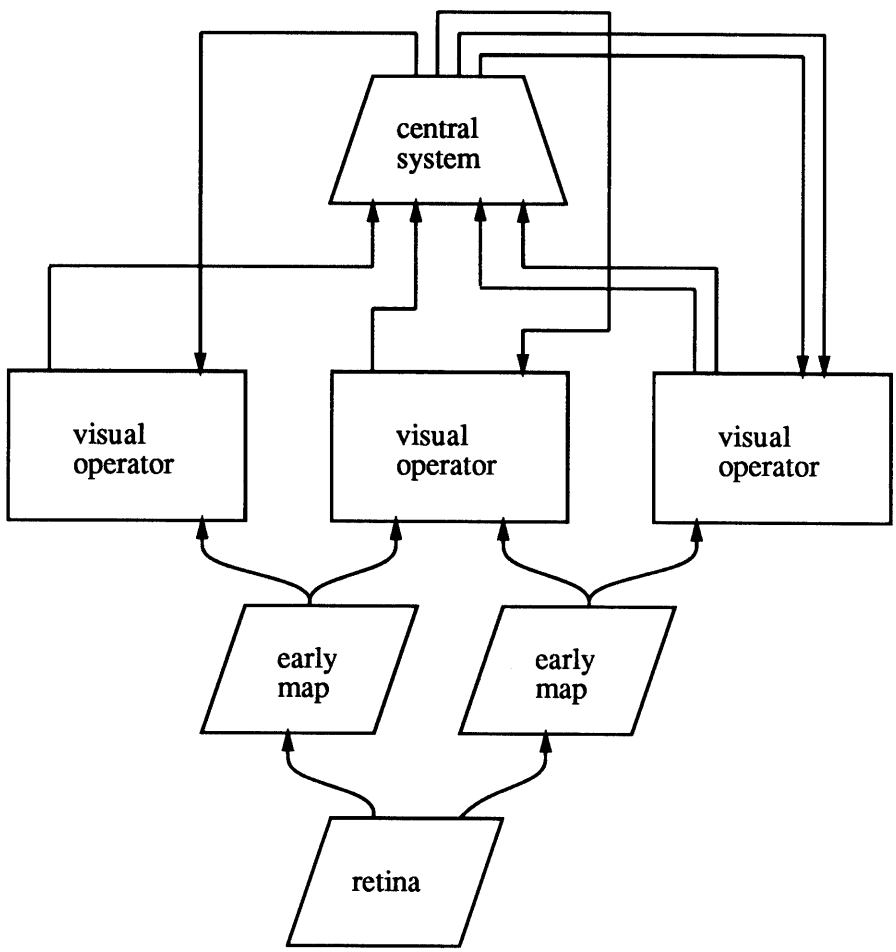


Figure 3.6: Overall modularity. The central system takes inputs from the visual operators and produces outputs for them.

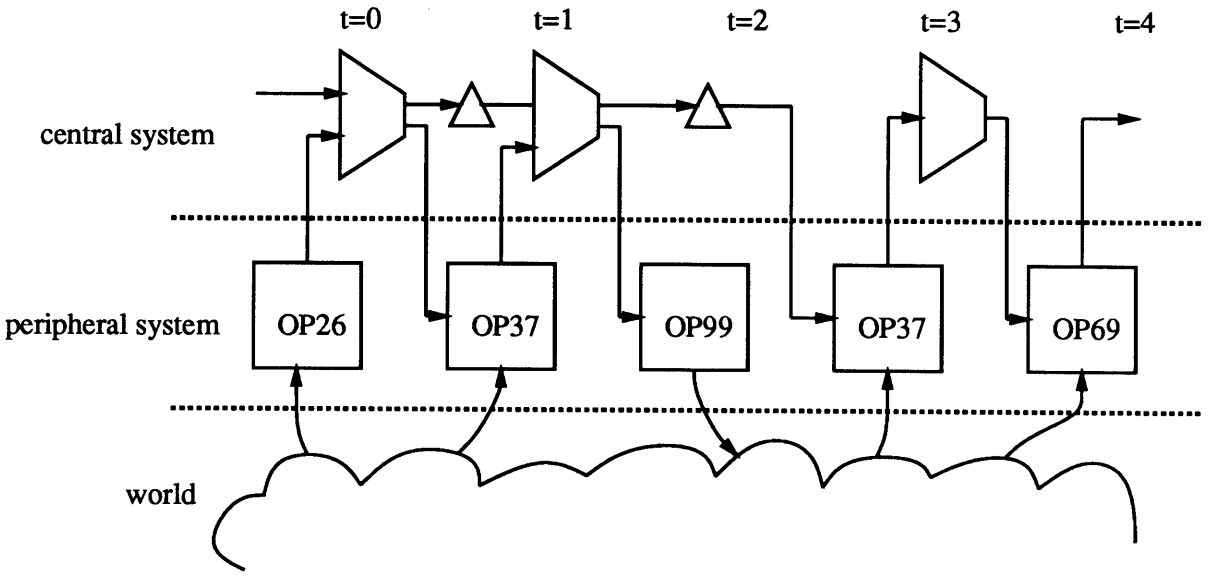


Figure 3.7: A generic visual routine. This diagram multiplexes time and space along the x axis; it shows how the central system, on successive ticks, produces control inputs for visual operators based on their outputs on the previous tick. The trapezoids represent the central system as a whole; of course it has more inputs and outputs than are illustrated. The triangles represent delay elements that store information within the central system from tick to tick. The squares represent peripheral operators, which also take time to compute.

mostly one-bit buses; the peripheral systems, like datapaths, operates on wide data values. Like some current horizontally microcoded architectures, termed *very long instruction word* machines [82], there are many datapaths all of which operate in parallel on every cycle (though some of them may not do anything useful, if there is no computation of their sort to be performed on that cycle). The job of the central system is to compute on each tick the values of the parameters controlling the peripheral operators; the vector of the values is analogous to the vector of control bits provided by a horizontal microinstruction.

Now we can put the pieces together. Figure 3.7 illustrates a generic visual routine. This figure is somewhat confusing in that it multiplexes both time and space along the x axis. In this diagram, causality weaves in and out between the central system, the peripheral system, and at times the world. That is the major difference between this architecture and a horizontally microcoded architecture: visual routines are interactional patterns, not programs. Along the top is pictured the central system, as a circuit, at several moments in time. Each trapezoid represents an instantaneous snapshot of the whole central system. The middle row represents various visual operators at particular moments in time; it suppresses from view all but the relevant operators.

The routine starts with the world presenting inputs that cause the bottom-up operator OP26 to produce a new input to the central system. This corresponds to *noticing* something unexpected. Based on this input, and possibly also on state information from within the central system, the central system directs OP37 to report on some aspect of the visual input. Based on this result, the central system directs OP99 to take some action, an eye movement perhaps, and also gives directions to OP37 (perhaps new directions but possibly the same ones; the central system has to give every operator inputs on every tick). Based on this result, the central system gives directions to OP69, which delivers another value, and so the process continues.

This visual routine actually incorporates some motor work. More generally, the notion of a "visual" routine is an abstraction. Late visual processing is usually in service of some specific perceptual-motor task, and as there is no separate hardware for late vision and as the task will typically involve tight feedback between vision and action, late vision is inextricably tied up with everything else.

3.3.6 Visual routines implement deictic representations

In section 2.7, we saw that deictic representations are indexical, functional, and grounded causally, particularly in perception. Deictic representations in Sonja are implemented with visual routines which discover particular aspects of the visual scene. A visual routine is task-specific and performed only when it is useful; this gives deictic representations their functional character.[10] Visual routines discover properties of the specific circumstances that surround the agent that engages in them; this makes deictic representations inherently egocentric and thereby indexical. Your visual routine that implements *the-cup-I'm-drinking-from* is relative to *you* because you are always looking in front of your face and that's where the cup will be. Additional indexicality is introduced by visual routines making explicit *relational* properties of the scene. An Amazon player may have a visual routine which finds *the-monster-closest-to-the-amazon*, an entity that is defined indexically, in terms of *the-amazon*.

[10]This is analogous to Ballard's hypothesis that "the internal representation of the image is minimal with respect to the goals" [19].

Chapter 4

Amazon

This chapter describes the video game Amazon, Sonja's domain. Section 4.1 explains how I chose this domain, drawing on earlier experience with writing a similar system for a similar domain. Section 4.2 explains Amazon in detail: what sorts of things are in the domain and how they interact. Section 4.3 describes some modifications I made to Amazon to make it easier to implement Sonja. I argue that these modifications do not vitiate the points the thesis makes. Section 4.4 considers whether or not Amazon is a realistic domain; I conclude that it is, though it is in some ways an atypical one.

4.1 Criteria on a domain

Sonja descends from an earlier system called Pengi [5] which I wrote to illustrate the concrete-situated ideas discussed in chapter 2. Pengi's domain was a video game called Pengo. My reasons for choosing Pengo as a domain for Pengi were different from my criteria for choosing Amazon as a domain for Sonja, but I'll explain them because they affected the choice of Amazon.

Pengi was intended to illustrate the point that the then-standard "planning" view of activity was inadequate for domains that are complex, uncertain, or real-time. Thus, I wanted a complex, uncertain, real-time domain. Pengi was also designed to demonstrate concrete-situated ideas about activity. These ideas emphasize perceptual connection with the world, so I wanted a domain which required substantial perceptual interaction, but which would not require that I spend years solving known difficult problems in vision research.

Video games have just these properties. A snapshot of a Pengo game appears in figure 4.1. In this game the player controls a penguin, which appears near the upper left corner, which is fighting against some bees in a maze made of ice cubes. The penguin and the bees can both kick ice cubes, so that they go skidding across the underlying ice surface until they run into something. If the something is a penguin or bee, it dies. Bees hatch from larvae inside ice cubes.

This scene has about two hundred and fifty objects in it. Simply representing the coordinates of all these objects would take at least as many propositions. To encode some basic information about their spatial relationships (what's next to what, where there are regions of freespace,

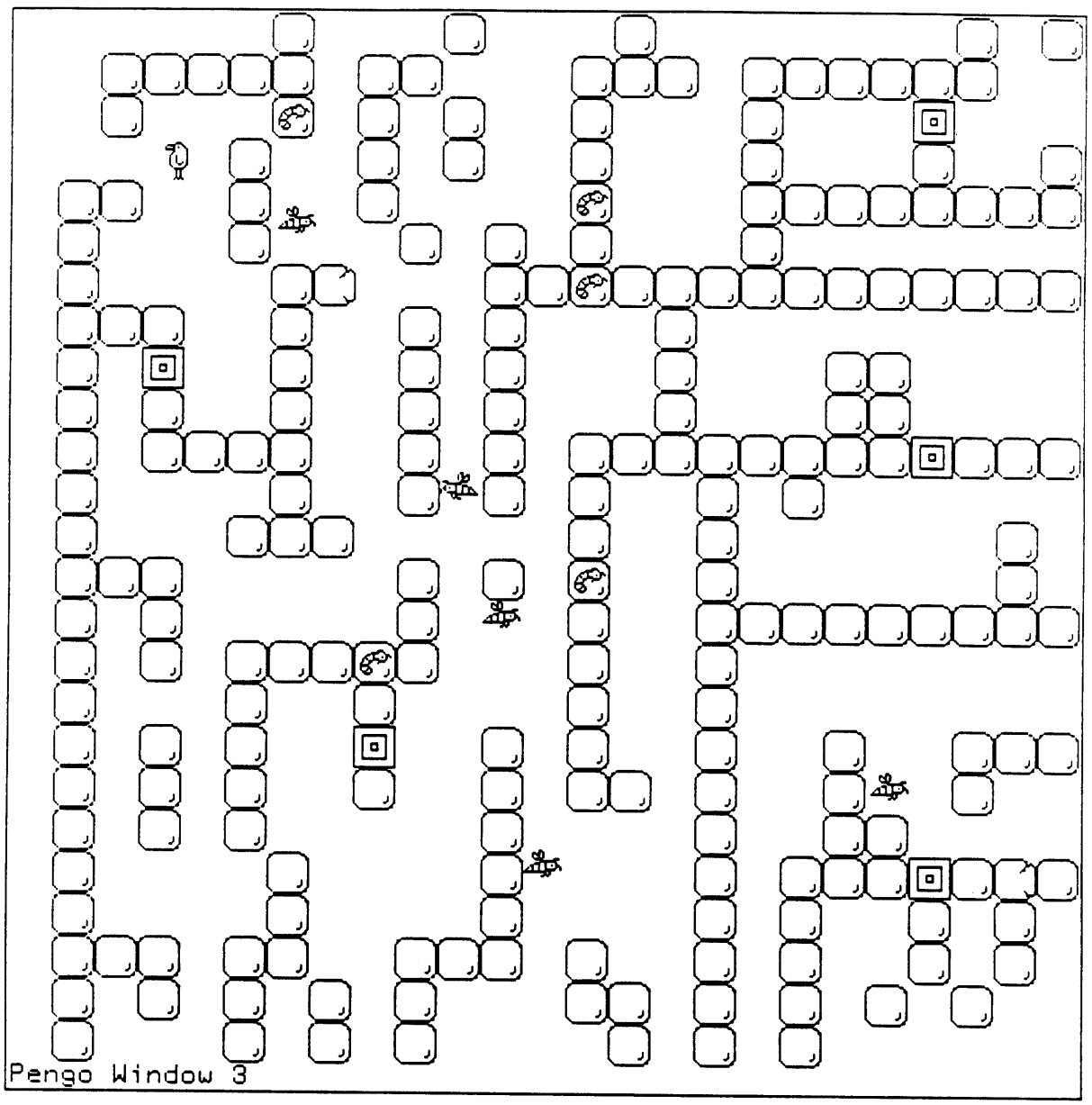


Figure 4.1: A Pengo game in progress. This scene has several hundred objects in it (it is complex); it changes several times per second (it is real-time); and in it bees act partly at random (it is uncertain).

where there are connected blobs of cubes and what their shapes are) would require a few thousand propositions. The game, therefore, is more *complex* in terms of sheer size than typical AI domains. Pengo is *uncertain* because the bees' behavior has a random component. It is *real-time*, with the penguin and the bees moving around at about twenty Hertz. Although a player might not have to track the game at this rate, surprising things happen quickly enough that a player's response time has to be well under a second; maintaining a complete and consistent model of the screen would require updating hundreds of propositions per second.

These properties together ensure that planning and problem solving techniques, which depend on exponential search and theorem proving, would be swamped by Pengo. This was one major point of our paper on the subject.

On the other hand, Pengo was in many ways an easier domain than some others. For one thing, it didn't require building an actual robot. I believe that studies involving actual robots are vitally important [26], but I didn't want to take one on. Pengo also didn't require building a complex simulation, just implementing a video game, which took about a month.

Video game playing is a strongly perceptual activity for humans; I incorporated into Pengi the best available theories of human vision. Because video games are two-dimensional and because the images involved are generated by a computer, I could bypass hard problems in early vision such as noise tolerance, stereo, and shape-from. This made it possible to implement Pengi in a few months.

I based Sonja's design on that of Pengi in many ways, and I took it as given that Sonja's domain would be another video game. However, I wanted to make different points in this thesis, and I needed a different sort of video game.

- The game had to be amenable to being played by a system similar to Pengi.
- For advice to be useful, it had to be the case that the player should often have several plausible mutually exclusive alternative courses of action available. Each of these things should be relatively easy to do; deciding which thing to do should be relatively hard. This is not generally the case in playing Pengo, in which there are few strategic choices to make.
- For there to be enough to talk about, the game had to be be ontologically complex. In Pengo, ice cubes, bees, and larvae are all there is; and few of each of these have any interesting properties at any given time.
- In Pengo, almost all the relevant aspects of a situation change from second to second. I wanted the game to be one in which some things change more slowly, so that longer-term intentions would be useful. This would allow instructions to talk about events in the future, and would require that a player project probable futures.
- Ideally, I wanted the game to be non-violent. I was bothered by the violence of most video games on political grounds, and also felt that it actively distorted the design of Pengi.

Gauntlet, a popular Dungeons and Dragons-like arcade game, immediately came to mind. It met all the criteria except the last one. I spent a few days trying to design a non-violent

game that met the other criteria and failed. So I gave up and implemented a game modeled fairly closely after Gauntlet: Amazon.[1]

4.2 Amazon

Sonja plays Amazon from the same perspective a human does: looking at the screen, which is to say as if looking down on the dungeon from above. It does *not* play it from the point of view of the amazon icon. I emphasize this because many people have been confused about Pengi in this way.

Figure 4.2 displays a scene in which all of the sorts of things in the Amazon domain appear. This diagram is a snapshot of the Amazon window as it appears to the player. This window actually provides a view into a small part of a larger underlying dungeon. The window tracks the *amazon*, which, as described in section 1.1, is the player icon. When the amazon gets close to the edge of the screen, the window smoothly moves over the underlying scene to keep the amazon within bounds, revealing new parts of the dungeon as it goes.

The amazon can move in the eight king's-move directions only; motion is continuous (actually a pixel at a time and fast enough that there is no noticeable flicker). The dungeon is broken up by *walls*. The motion of monsters and the amazon is *compliant* against walls: that is, if one runs into a wall on a diagonal, it will slide along it. Walls frequently form *rooms*, such as that directly above the amazon.

Ghosts and *demon bunnies* also inhabit the dungeon. (I tried to draw demon toads, but they just wouldn't cooperate.) The monsters can also move only on king's-move axes, but always head as nearly as possible toward the amazon within that constraint. (There are eight different icons for each monster type, corresponding to their headings.) The player has a continuously variable *health*, which is represented by the black bar of varying width at the bottom of the screen. When a monster runs into the amazon, it continuously saps your health. When your health runs out, you die. The demon bunnies also breathe *fireballs* at the amazon; five are visible in the figure as a line of small lozenges. Fireballs take a certain amount of health when they hit the amazon. You can fight back by making the amazon hurl *shuriken* at the monsters; these are the small three-pointed icons to the right of the amazon. It takes three shuriken hits to kill a monster. After the amazon fires a shuriken, she is exhausted by the effort, and can not move or shoot again for a fraction of a second. Shuriken and fireballs mutually annihilate when they collide. The amazon has an inexhausible supply of shuriken.

Monsters are intermittently produced by *monster generators*. *Piles of bones* produce ghosts and *bunny hutches* produce demons. You can destroy a pile of bones with ten shuriken hits, but hutches are invulnerable. Just when and where next to a generator a new monster will appear is random. Piles of bones also unpredictably have a *dark shadow* fall upon them, which renders them invulnerable until it passes. (The dark shadow is visible on the upper right and not on

[1] It appears that Gauntlet was modeled after a game called Dandy, which was implemented as a thesis by Palevich, then a student at MIT [197]. Though Dandy is a violent game, Palevich designed it to foster cooperation as opposed to conflict among video game players. Gauntlet and Amazon retain these cooperative characteristics; I'll discuss them further in chapter 5 and in appendix A.

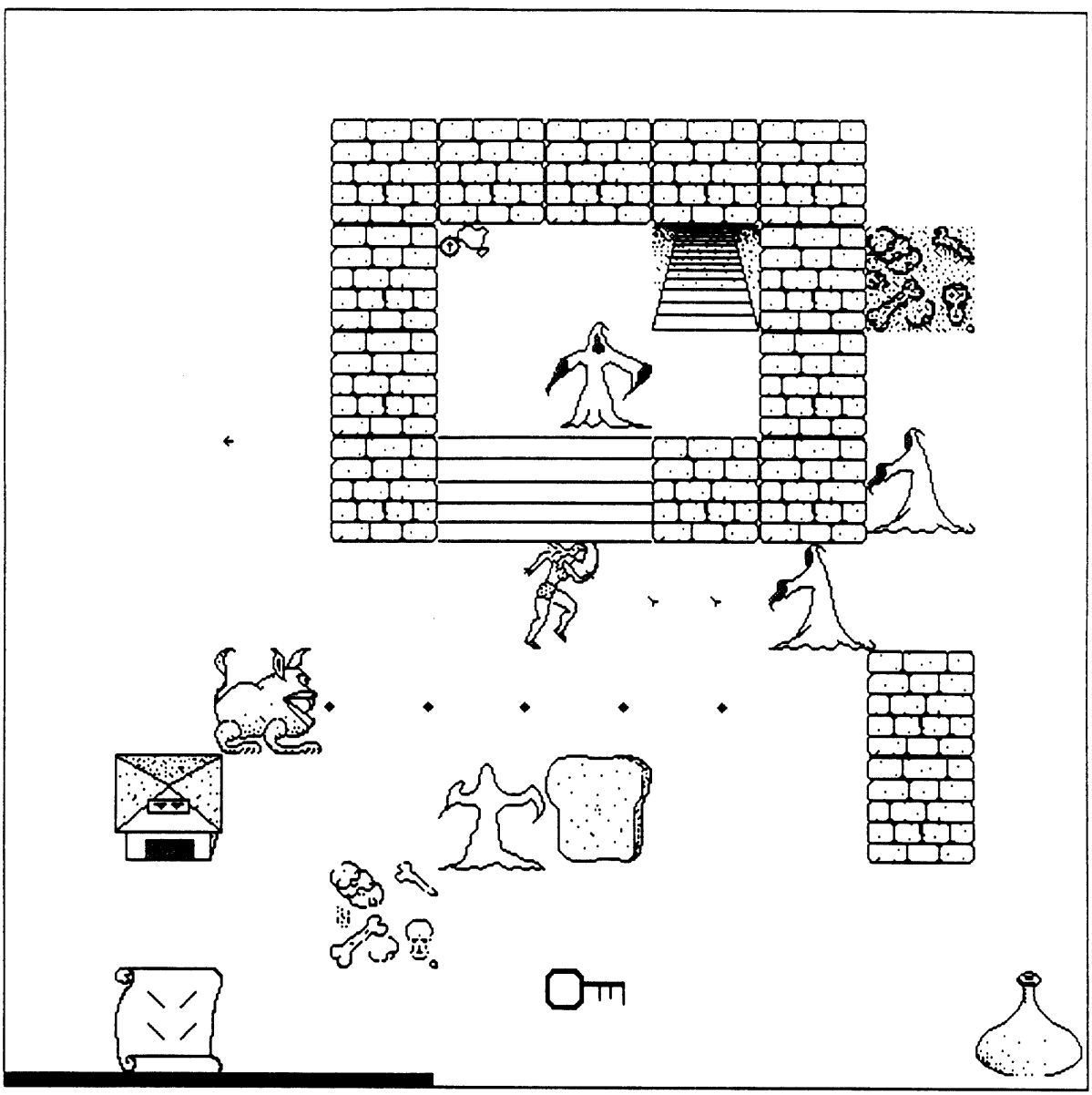


Figure 4.2: An illustrative Amazon scene. All of the sorts of things that can appear in an Amazon game appear in this scene.

the lower left pile of bones in the figure.)

There are several types of *goodies* you can pick up by running into them: amulets, scrolls, keys, knives, potions, and pieces of bread.

If you pick up a magic *amulet*, it grants you one of five magic powers at random: your shuriken may bounce off of walls (rather than being absorbed by them), you may repel monsters (instead of attracting them), you may find that you are able to recover from throwing shuriken much faster (so that you can effectively fire a denser stream of them), your speed may increase by a factor of five (so you go zooming about, easily outmaneuvering monsters), or you may be given a magic shield (so that monsters sap your strength more slowly). (An amulet, with its chain, is visible in the room in the figure.) These effects wear off after a minute or two. Magic *scrolls* work similarly, except that the effects are permanent. (A scroll appears near the bottom left corner of the figure.) Scrolls have four *characters* on them, which are diagonal strokes. A scroll is *ready* when all four characters point inward. You can only pick up a scroll when it is ready; otherwise it acts just like a wall. The characters on a scroll slowly mystically change themselves around.

Keys are needed for opening *doors*. (The horizontal striped area in the figure represents the door to the room.) *Knives*, once picked up, can be thrown against monsters, killing them instantly, or they can be used to jimmy the lock on a door. (Knives appear as small arrow-shaped icons; one is to the left of the room in the sample scene.) You must be facing a door and right up against it to jimmy it, and knives are used up once they are used for either purpose. Knives, unlike shuriken, are conserved; you can only use one once.

A *potion* (appearing as a bottle in the figure) can be picked up and saved for future use. When activated, a potion kills every monster on the screen.

Pieces of *bread*, when picked up, increase your health, counteracting the damage done by the monsters.

The *stairwell* leads down to the next level of the dungeon: a whole new space to explore.

4.3 Modifications

I wrote Amazon before I had more than vague ideas about what Sonja was going to do. This, together with its similarity to Gauntlet, made it seem "real" to some degree. I was for this reason not very happy to have to modify it for Sonja's sake.

I made two modifications. The first was that the game as I originally wrote it was not ontologically complex enough. Some of the complexity I described in the last section I introduced later in order for there to be interesting things to give advice about. Since this was an *addition* to the game, it did not seem serious.

More seriously, I found I had to redesign the layout of the dungeon. It turned out that getting about in the dungeon was a hard task for Sonja, and I had to restrict the allowable configurations of walls to make it easier. The reason for this is recounted in section 8.7. This was disappointing, but as getting about the dungeon was not a focus of the work, it did not seem like a real cheat.

Late in the implementation of Sonja, I realized that Amazon was a bad domain in ways

I hadn't anticipated, because I hadn't articulated some domain properties that Amazon lacks but that turn out to be important. Lack of these properties result in limited opportunities for certain types of visual work and certain sorts of natural language reference which I would have liked to explore further. I'll discuss these properties in sections 9.4.2 and 9.6.4. I discovered these problems too late to correct them by changing domain.

4.4 Realism

People often say of Pengo as a domain that it is as much a simulation as any other AI domain, and that this undercuts my arguments about the importance of using real world domains. Although I think there is a lot of truth in this, it is not literally correct: Pengo and Amazon are real, not simulated domains. Playing video games is a real human activity which people pay money to engage in.

There are two ways in which this criticism is valid, however. The first is that although the *domains* of Pengi and Sonja are real, their *visual systems* are in fact simulated. I don't use a TV camera; Sonja and Amazon coexist in the same computer and communicate not through light rays and joystick motion but via Lisp datastructures. This, I think, would be quite easy to rectify, and probably not much would be learned in the process, because Sonja's visual system, though simulated, is realistic; for discussion, see section 9.5.1.

The second, more serious issue is that playing Amazon, though a real human activity, is in some ways atypical.

The vision problem in Amazon is both easier and harder than in most other domains. It is easier in that the visual scene is two-dimensional, without occlusion, with uniform "illumination," with clearly delineated, brightly colored objects moving about in it. It is harder because in you often have to keep track of many monsters trying to attack you at once; this stretches one particular ability of the human visual system (tracking) to its utmost. (This, I imagine, is one reason video games are fun.)

The extreme time pressure of video games, though shared with some other human activities, is atypical. You don't generally have to make split-second decisions while making breakfast. (On the other hand, you do have to do some things on time: you can't go catatonic proving theorems while you've got eggs on the stove.) Extreme time pressure makes it difficult to engage in cognition. More typical activities do permit somewhat more explicit, articulated thought about what you are doing (though probably a lot less than has usually been posited).

In most domains, not everything relevant to your activity is perceptually accessible. This is true of Amazon also: the screen displays only a small part of the underlying universe. Further, we'll see that Sonja's visual system allows it to access only a small portion of the screen at any time. However, most of what is off the screen is irrelevant to most play. Amazon is atypical in that almost everything you need in order to decide how to act is perceptually accessible with minimal work. This makes it possible to make do with almost no state; activity in many other domains will require greater use of memory.

The notion of goals serves in AI both to individuate activities and as an account of motivation. Goals and activities are more clearly defined in video games than in most other domains.

Video game activity is strictly instrumental and has an objective measure of success (a numerical score). Most activity is less easily described in terms of determinate goals. The rich structure of dinner conversation, for instance, is hard to account for in these terms. Understanding activity in such domains will require better theories of the structure of activity and of motivation.

Most human activities do not involve hostile agents (such as the monsters in Amazon). Cooperation is far more characteristic, but has been studied much less. I'll have more to say about this in appendix A and section B.3.1.

Chapter 5

Instruction use

This chapter explains how Sonja uses instructions. Although instruction use is an interesting phenomenon in its own right, the issues discussed in this chapter are interesting principally for their implications for language use more generally. Sonja provides a framework for grounding instruction use in concrete activity. Such a framework can be an implementation medium for theories of pragmatic issues such as reference.

Section 5.1 begins by describing human video game instruction use. Video game instruction use is a form of language use with many special characteristics which are exploited by Sonja. I'll argue, though, that some characteristics of video game instruction use show up in language use more generally. In particular, video game instruction use highlights the problem of reference, discussed in section 5.2. In that section, I present alternative theories of reference and adopt a causal theory. A main contribution of Sonja is showing in detail how reference can be causally grounded in activity, particularly perception. I discuss the issues of indexicality and reflexivity, which are simultaneously problems in and resources for resolving references. Section 5.3 explains how you give Sonja instructions and how Sonja keeps track of the instructions you've given it. An instruction buffer mechanism provides for compositionality of processing. The section provides a table of all the instructions Sonja can use. Section 5.4 explains how instructions affect Sonja's activity. Instructions act by redirecting Sonja's attention and by managing its routines. Section 5.4.4 describes various specific linguistic constructs Sonja uses, and section 5.4.6 describes various linguistic issues that Sonja does not address at all. Finally, section 5.5 describes previous computational research on instruction use and on reference.

5.1 Video game instruction use

I was inspired to implement Sonja by observation of human video game instruction use. I spent time in video arcades watching video game players giving and using instructions. I video taped and transcribed some of these interactions. Video game instruction use differs from other forms of language use, and from other forms of instruction use, in many ways. These are largely determined by the social and physical context of video game playing, and are reflected in Sonja's instruction use.

Video game playing is a *social* activity for many practitioners. People go to arcades with their friends and play together or watch each other play. Video games are most quickly learned by collaboration, imitation, and from explicit instruction. (I'll talk more about these sorts of learning in section B.3.1.)

Video game instruction use is *situated*: the instruction giver and taker are right next to each other and have the same perceptual access to the situation. Many video games are in fact collaborative (two players cooperate in fighting off simulated enemies), in which case the instruction giver may be an involved participant and not merely a kibbitzer. This contrasts with written instructions and those given over the phone. Video game instruction givers and takers depend heavily on this shared access (see section 5.2.4), as does Sonja.

Some instructions are *immediate*: intended to apply only to the current situation. "Pick up that item" is an example. "Immediate" does not mean *instantaneous*, though; carrying out "Pick up that item" may take a second or two. Other instructions are intended to convey *general* rules which will apply in many situations. An example in my video corpus is "Turn your perspective before you go around the corner." This is *always* good advice in playing Xybot, and the instruction taker probably understood that he should permanently change his going-around-a-corner routine accordingly. Sonja models only immediate instruction use: you say something like "Use the knife!," it figures out, based on what's currently going on, what you must mean by that, uses the knife in the appropriate way as soon as feasible, and then forgets all about the suggestion. Sonja does not learn from advice it is given. (Appendix B suggests ways it might be extended to do so.)

Video game players understand that learning to play a video game involves learning a series of game-specific skills in a characteristic order that varies little from player to player. Thus players understand that individuals can be ranked in a nearly total order with respect to their understanding of a particular game. There is relatively little room for difference of opinion about how to play. This ranking makes instruction-giving a natural, useful, and in fact common function. Players understand that instructions given them by a more expert player are probably correct, and are therefore are willing to take them partly on trust. Sonja, likewise, tries to make sense of instructions when it can, and rejects instructions that obviously make no sense, but must take others on trust.

Video game instructions are typically accepted without any response. This contrasts with instruction use in many other domains, in which it is common for instruction users to acknowledge instructions, to report success or failure, and to negotiate their meanings with the instruction giver by asking for elaborations of them and engaging in conversation about their application to the situation at hand [40]. These phenomena are rare in video game instruction use, probably because there isn't time for them; by the time you could have negotiated the meaning of an instruction, the situation to which it is relevant is often long gone. Sonja, accordingly, is not equipped to engage in extended conversation about individual situations or objects.

Probably again because of the time pressure of video game playing, immediate video game instructions are brief. The mean length of utterance in my transcripts is around five words. There are many one and two and three word utterances ("Danger!" "Other way," "We're losing

energy.") The longest utterances are about three simple sentences. Video game instructions are also syntactically simple. For example, there are *no* subordinate clauses in my transcripts, which represent about an hour of interaction and about three hundred utterances. As we'll see in section 5.3, Sonja does not make much use of the syntactic structure of the instructions given it.

5.2 Reference

"Get the ghost" can be used in a particular Amazon-playing situation to *refer* to a particular ghost. How? In this section I'll review theories of reference and discuss the problem of indexicality. I won't propose a general theory of reference, but rather will adopt a version of the *causal* theory, which grounds reference in physical causality. The contribution of this thesis to the theory of reference is to show in greater detail than has previously been given how such a grounding is possible.

5.2.1 Theories of reference

Most theories of reference explicitly or implicitly divide the problem in two by positing that there are, in addition to natural language representations, "mentalese" representations. The first part of the problem, which following Sidner [248] I'll call *specification*, is figuring out how natural language expressions correspond to mentalese representations; the second part, *external reference*, is figuring out how mentalese representations correspond to external objects. For example, the name "Jerry Fodor" is taken to specify a particular mentalese representation, which in turn refers externally to a particular person. The problem of specification is considered a problem for linguistics, and one on which progress might be made relatively straightforwardly, because it is only a matter of transforming one sort of representation into another. External reference is usually taken to be a harder problem, and one for philosophy rather than linguistics.

There are, loosely speaking, two approaches to the problem of reference: *correspondence* theories and *causal* theories. (Some theories fit in neither category; Millikan's [177], for example.) Correspondence theories (such as those of Davidson [52], Montague [55, 200], and Tarski [261]) seek to explain compositionality of reference. For example, the meaning of "snow is white" is said to be that snow is white, and we know this because we know that the meaning of "snow" is snow and that the meaning of "white" is white. Such theories only address the specification problem, because they do not explain how atomic representations (such as "snow") correspond to their referents. Correspondence theories of reference in linguistics are concerned mainly with finding the canonical mentalese representation specified by a linguistic one. For example, it might be useful to discover that "The US President in 1990" specifies $GEORGE-BUSH.

Causal theories, such as those of Evans [69] and Kripke [149] ground reference in physical causality. For instance, it seems that a tachometer needle represents engine speed in virtue of its connection to the engine via cables and wires; that a computer vision system represents a scene in virtue of its connection to them via light rays and a TV camera; and that a shopping list represents the food items it lists in virtue of your ability to recognize them on a shelf when you

see them. On the other hand, causal theories are unable to explain reference to abstractions, because abstractions do not participate in physical causality. What causal paths connect "3" or "Rudolph the Red-nosed Reindeer" to their referents?

5.2.2 Grounding reference in activity

This thesis proposes no general solution to the problem of reference. Causal grounding seems to explain nicely reference in the case of video game instruction use, and so I have adopted it. On the other hand, this thesis is not concerned with specification, because Sonja has no mentalese. (Sonja forms an existence proof that an agent can make sensible, complex use of external representations without addressing specification.)

A causal theory of reference must specify *which* causal connections constitute references. This is relatively easier for immediate, situated instruction use than for some other cases, because reference can be *grounded in activity.* The causal connections that constitute the reference of the instruction "Pick up that item" are among those that let the instruction taker carry the instruction out: the perceptual processes that let the instruction taker find the referent of "that item." More generally, resolving reference may require physical action as well: for example to enable perception by moving your head or by pulling aside an occluding obstacle.

Elucidating the grounding of reference in activity has implications for linguistics more generally. Fillmore [78, 79], Herskovits [117], Jackendoff [127], Lakoff [155], Miller and Johnson-Laird [176], Stucky [255], and Talmy [260], among others, argue for grounding pragmatics, semantics and some syntactic phenomena in perception, bodily space, and activity. They propose, for instance, that the meanings of many words and perhaps some syntactic constructions ground out in spatial reasoning and representation. Some but not all of this work explicitly supposes that this spatial reasoning is related to vision: either that it actually uses the innate visual hardware, or that it has the form it has because it is the result of the internalization of visual activity. Although these linguistic studies are inspiring starting points, they are limited by lack of a connection to serious theories of activity and perception. Their theories of vision are schematic and much of the time are theories of Cartesian space rather than of vision per se. That is, meanings are taken as depending on Cartesian spatial models, without reference to visual processing. As I've argued earlier, the sorts of representations that vision actually delivers may be quite unlike Cartesian models.

This thesis suggests only a *framework* for grounding linguistic theories in activity. The specific "linguistic theory" implemented in Sonja, composed of routines that interpret specific instructions, is crude. Implementing a serious linguistic theory in this framework would be an enormous project, but Sonja suggests it is possible.

5.2.3 Indexicality

Indexicality is the dependence of the reference of a representation on its situation of use. Modeling indexical reference was one of the main motivations for implementing Sonja.

Indexicality is a pervasive phenomenon which manifests in many ways. (Fillmore [78] provides a review of categories of indexicality.) The classical cases are words such as "now," "here,"

and "I," whose referents depend solely on the situation of use.[1] Sonja doesn't use this sort of indexicality (though it easily could, as will become evident).

Another class of cases is concerned with the dependence of reference on the *linguistic context*: that is, on what has already been said. Here is an example of this phenomenon (adapted from Grosz [100]):

A: I'm going camping next weekend. Do you have a tent I could borrow?
B: Sure. I have a two-person backpacking tent.
A: The last trip I was on there was a huge storm. **The tent** collapsed and I got soaked.
B: What kind of tent was it?
A: A tube tent.
B: Tube tents don't stand up well in a real storm.
A: Uh huh.
B: OK, I'll bring in **the tent** tomorrow.

Here, the two instances of **the tent** refer to *different* tents due to their distinct linguistic contexts. Considerable attention has been given to this phenomenon within pragmatics and computational linguistics, particularly under the rubric of *anaphora* [100, 99, 105, 109, 128, 248, 284]. Since Sonja maintains little linguistic history, it rarely uses the linguistic context to ground indexical references. Sonja can get away with this because video game instruction use also makes little use of the linguistic context. This is probably because in video game discourse it is usually easier to refer to objects in terms of their their appearance on the screen. Sonja does make use of the linguistic context in resolving indexicality in one case, the directive "No, the other one!", which can refer to another monster or amulet or pile of bones depending on what instructions have previously been given. It would be interesting to extend Sonja to keep track of more of the linguistic context and to make more use of it in grounding indexical references. I think that existing techniques for doing this could be incorporated straightforwardly; it would be more interesting to try to find mechanisms that accurately model human memory for linguistic context.

The phenomenon of indexicality is broader than either of these forms. It is not restricted to a particular subset of words; the meanings of almost all words depend on context. For example, Enç [65] argues from examples involving tense scoping that all nouns and verbs must be treated as indexicals whose denotations can only be determined pragmatically. For example, in "All virgins will go to heaven," "virgins" denotes all those people who are virgins when they die, not, as it would in other contexts, all those who are virgin at some particular moment. Other arguments to the same end are proposed by Fillmore [79], Nunberg [195], and others.

What counts as "context" can be arbitrarily broad. Besides the linguistic context, it includes the physical, social, and cultural circumstances of the instance of use, and it includes the purposes, memories, and histories of the interlocutors. A nice example is provided by Winograd and Flores' analysis of "Is there any water in the refrigerator?" [289].

[1] *Situation semantics* [22] is a compositional semantics devised particularly to address this class of indexicals.

A: Is there any water in the refrigerator?

B: Yes.

A: Where? I don't see it.

B: In the cells of the eggplant.

If A intends to drink the water, this is an unhelpful answer. On the other hand, it is "literally correct," and if A intends to clean the inside of the refrigerator with a chemical that reacts violently with trace amounts of water, it might also be a useful warning. It's easy to construct examples like this in which, depending on almost arbitrary factors, words can be made to mean almost arbitrary things.

In the end, as Katz and Fodor [139] and Nunberg [193, 194] have argued, we shouldn't expect a theory of reference that is both general and detailed, because reference is inescapably domain- and task-specific. Reference is, moreover, *locally organized*, in the ethnomethodological sense: it's an ongoing collaborative achievement of the participants, who can and will make use of arbitrary aspects of the situation [40, 110, 116].

Sonja is not able to use all the sorts of context I've described. However, it is able to use its understanding of the ongoing Amazon situation and its own purposes in interpreting utterances. Sonja can find the referents of instructions such as "Get the demon," which in context often picks out a unique monster that it currently *makes sense* to get, though several may be visible. Schematically, making sense of such references is simple: referents are found by engaging in visual activity. Although it is widely acknowledged that perception is important in determining reference, few specific theories have previously been offered. On the other hand, there is a large linguistic and computational literature on the semantics and pragmatics of noun phrases [22, 105, 121, 150, 250] describing many phenomena that are not reflected in Sonja. For example, Sonja interprets all noun phrases *referentially*, rather than *attributively* [150], and *specifically*, rather than *generically* [250]. Sonja does not process determiners, quantifiers, or demonstratives, and so does not distinguish between "the stairs," "a stairwell," "any stairwell," and "those stairs."

The utterances in my video corpus are highly indexical. Here are some examples. They are quite typical and not chosen for extreme indexicality.

- "Hey, that's mine." This is said with no conversational context. It refers to an object ("that") on the screen. Both players know which object must be meant; only one makes sense.
- "Forty percent energy." In context, this is understood as "You have forty percent of your energy left." In a different context, it would mean "I have forty percent of my energy left." Which is meant turns on whose energy level is relevant in jointly deciding what to do next. For instance, I may be pointing out that I have more than you do, so I should be the one to deal with the next batch of monsters, or I may be warning you that you are losing energy quickly and had better retreat.
- "Are we gonna go through the time warp again?" "We" is traditionally an indexical; it's trivially understandable in this context, though. Definite noun phrases have often been

analyzed as non-indexical, but in fact here determining the referent of "the time warp" is much more complicated than that of "we". Both players know which time warp is meant, though it is not visible and though it is a *different* time warp than the one they went through last time ("again"). It's the time warp that it might *make sense* to go through this time.

5.2.4 Reflexivity

Instruction use, like other language use, is organized *reflexively*,[2] meaning that the speaker can depend on the hearer to understand the situation in the same way the speaker does, and to understand that the two share an understanding, and so on recursively. For example, an instruction giver can only expect "No!" to communicate if the instruction taker understands herself to be engaged in the particular activity "No!" recommends against; the taker can only make sense of the instruction if she imagines that the advisor considers her to be engaged in that activity; the giver must further be able to count on the taker imagining this; and so on. Likewise, in video game instruction use, both participants must reflexively understand that "Get that guy" picks out a mutually-agreed-upon enemy.

Video game players depend on instruction givers and users seeing the evolving game the same way, despite its many shifting issues. They *must* make this assumption. If they didn't the instruction giver could never finish specifying everything that would be necessary to relate an instruction to the evolving game situation. I'm sure the players couldn't list their shared understandings if they had to. In my video tapes of collaborative play, most coordination *goes without saying* [224]. For the most part, talk is required only when there are minute differences in understanding, and it depends extensively on the shared background of understanding. Communication doesn't pick up a "meaning" from my head and set it down in yours. Instead, communication is part of the work of maintaining a common reality. The instruction giver and taker share a common reality because they are competent players and because they use language to keep the shared reality in good repair.[3]

The phenomenon of reflexivity has been accounted for by several authors [24, 39, 105, 107, 148, 204, 209] under the rubric of *mutual knowledge*. In this framework, the interlocutors have beliefs about each other's beliefs, and about each other's beliefs about each other's beliefs, and so on. Not all the authors I have cited have specified how these beliefs are to be implemented; the most obvious way, which has been adopted by many of them, is that beliefs are *mental representations*. I don't think this account can be right in general; it requires more representations than there is time to reason with.

In Sonja, I take an alternative, *passive* approach to reflexivity.[4] Sonja and a human interlocutor can make sense of each other because they have a similar enough understanding of the ongoing Amazon game. When you tell Sonja "Get the ghost" it will probably know which

[2] I am using this term as it is used in ethnomethodology [116]. This is unrelated to the use of "reflexive" to refer to situations in which the subject and object of a verb have identical referents.

[3] This point is central in the ethnomethodological conversation analysis literature [116]. It has also been recognized by some computational researchers, for instance Grosz [101].

[4] A similar approach is taken by the linguist Rommetveit [224].

monster you mean because it is competent at finding the most important ghost to clobber at any given time. Sonja doesn't, and doesn't need to, reason about your beliefs, because they are almost all shared.

Passive reflexivity depends not only on the interlocutors sharing an understanding of the specific domain of discourse, but also on their having homologous computational machinery. We'll see that advice such as "Get the amulet" is useful to Sonja not only because it supports one activity (getting the amulet) over others, but because it tells Sonja how to allocate a scarce computational resource, namely visual attention. Advisors would not give such advice if they did not reflexively understand that players have limited visual attention. Exactly what sorts of structural homology between conversational participants are required probably depends on specifics of the conversation and its context. Section 9.4.1 proposes an experiment to see whether Sonja has adequate homology for its task.

Passive reflexivity is more efficient than representational reflexivity, so we should see how far we can get with it. It is not hard to find situations in which people *do* reason about each other's beliefs and the effect differences in beliefs will have on communication. The same principle suggests that we should seek minimally representational (and so maximally efficient) accounts for such situations. I have not done this in Sonja; it is an interesting problem for future research.

As in human communication, differences in understanding can lead to miscommunication; as in human communication, Sonja can participate in *repair* of misunderstandings. This is an instance of the situated, interactive, improvisational organization of activity doing work in the concrete-situated approach that requires representation and reasoning in other approaches. Common situatedness makes it highly probable that interlocutors share an understanding; interactivity allows them to repair miscommunications; improvisational organization means that repair processes are not disruptive. I'll discuss repair further in section 5.4.4.

5.3 Giving Sonja instructions

Figure 5.1 shows the screen as it appears to you as an advisor of Sonja. You can watch Sonja play Amazon in one window. When it seems that an instruction would be useful, you hit the **suspend** key, which stops the action and gives you a prompt in another window devoted to linguistic interaction. You type your instruction in this window, hit **end**, and Amazon starts running again.

Few instructions can be carried out instantaneously. Accordingly, Sonja must store instructions in some way, so that they effectively hang around until the activities they suggest are completed. Since I have little idea how people do this, I adopted the simplest mechanism I could think of for Sonja. *Instruction buffers* to keep track of which instructions have been given. (A complete table of Sonja's instruction buffers appears in figure 5.2.) Whenever an instruction is given, one or more instruction buffers are set.

Each instruction buffer has a *valid bit*, which tells whether a corresponding instruction has been given. A buffer remains valid until it is explicitly cleared by the central system. Each buffer has a *clear line* input; when it is set high the valid bit is cleared. The central system

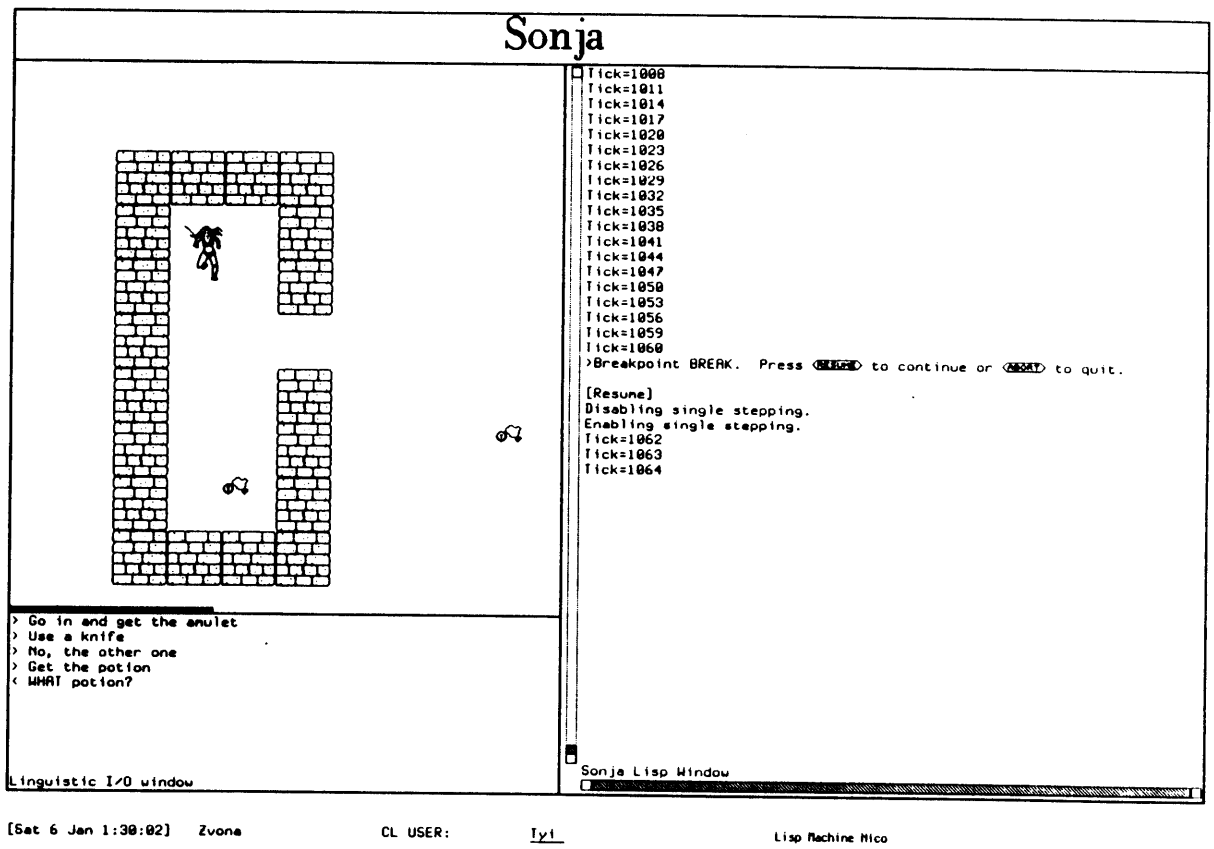


Figure 5.1: The lisp machine screen as it appears to an advice giver. The Amazon window is on the left. Below it is a linguistic interaction window. The advisor's input appears in this window after a greater-than sign prompt. Sonja's linguistic output appears with a less-than sign prepended. The window on the right is a lisp interaction window which is used in debugging the system. The top window is just a title.

does this when it has succeeded or failed in carrying out the instruction, or when it rejects it.

In addition to its valid bit, an instruction buffer may have named *fields*, which are registers holding data values extracted from instructions. In Sonja, no buffer has more than one field, and few have any.

Compositionality of processing is provided in three ways. Several instructions may set the same buffer; for example, both "Get the potion" and "Get the amulet" set the buffer `pick-up-the-goody`. Buffer fields also contribute: the instructions "Head right" and "Head left," for example, set the same instruction buffer, `suggested-go`, but set its `direction` field differently. Further compositionality is provided by the organization of the central system, in that the same piece of machinery may take input from several instruction buffers. For example, the instruction buffers `register-the-amulet` and `register-the-potion` are used by the same visual search routine. These mechanisms for compositionality are effectively equivalent; it would, for instance, involve changing about five lines of code to make "Head down those stairs" and "Don't go down yet" share a single instruction buffer with a boolean `negated?` field, rather than having their own buffers `go-down-the-stairwell` and `dont-go-down-the-stairwell`. Since instruction buffers are not intended as a real theory of memory for instructions, I've made arbitrary choices between these mechanisms in the implementation.

Instruction buffers can be *chained* to store instructions involving a temporal sequence (for instance "Get the potion and set it off"). A chained instruction buffer has a *next pointer* to another buffer that represents the next activity in the sequence. When the buffer is cleared, indicating that the first activity has been completed, the valid bit on the next buffer is set high. Next pointers, like instruction buffers themselves, are not intended to be plausible as cognitive models, which is why it doesn't bother me that, as pointers, they violate essential connectionism. A real account of how instructions are buffered would be part of a real account of memory more generally, and such an account should respect the constraints of essential connectionism.

Figure 5.2 shows how instruction buffers are set based on an advisor's input string. From it we can see that, roughly speaking, Sonja does little syntactic processing. (I say "roughly" because it's not really clear how in Sonja to draw the line between syntax and semantics.) The system can accept only few dozen instructions; the natural language input window could be replaced with a mouse menu. The system does not incorporate a parser; a single Lisp procedure implements the table of figure 5.2. Depending on how "semantics" is defined, Sonja also does no semantic processing; certainly it constructs no semantic representation of its input.

On the other hand, the compositionality of Sonja's instruction use makes me believe that it would be straightforward to add to the system as it stands a parser that would set instruction buffers. Such a parser would currently be limited by the relatively small number of sorts of activities Sonja can engage in and consequently the relatively small number of sorts of advice Sonja could make use of. A deeper linguistic project is to determine how instructions are in fact buffered and what sort of processing connects linguistic input with specific proposals for action.

Instruction	Instruction buffer(s) set	Field
Get the monster/ghost/demon	`kill-the-monster`	`monster-type`
Don't bother with that guy	`dont-kill-the-monster`	
Head down those stairs	`go-down-the-stairwell`	
Don't go down yet	`dont-go-down-the-stairwell`	
Get the bones	`kill-the-bones`	
Ignore the bones for now	`dont-kill-the-bones`	
Get the *goody*	`pick-up-the-goody`	
	`register-the-`*goody*	
Don't pick up the *goody*	`dont-pick-up-the-goody`	
	`register-the-`*goody*	
Head *direction*	`suggested-go`	`direction`
Don't go *direction*	`suggested-not-go`	`direction`
Go around to the left/right	`go-around`	`direction`
Go around the top/bottom	`go-around`	`direction`
Go on in	`go-in`	
OK, head out now	`go-out`	
Go on in and down the stairs	`in-the-room`	
	`go-down-the-stairwell`	
Go on in and get the bones	`in-the-room`	
	`kill-the-bones`	
Go in and get the *goody*	`register-the-`*goody*	
	`in-the-room`	
	`pick-up-the-goody`	
Get the potion and set it off	`register-the-potion`	
	`pick-up-the-goody`	
	`use-a-potion` (chained)	
Scroll's ready	`scroll-is-ready`	
On your left! *and similar*	`look-out-relative`	`rotation`
On the left! *and similar*	`look-out-absolute`	`direction`
Use a knife	`use-a-knife`	
Hit it with a knife when it goes light	`hit-it-with-a-knife-when-it-goes-light`	
Use a potion	`use-a-potion`	
No, the other one	`no-the-other-one`	

Figure 5.2: Sonja's instruction buffers and the instructions that set them. Some instructions have variant forms. `pick-up-the-goody` and `use-a-potion` are chained together when the instruction "Get the potion and set it off" is given.

5.4 How Sonja uses instructions

Intelligent use of advice requires that you *make sense* of it. (Video game instructions are in this way unlike computer program instructions, which can be simply executed.) Making sense can in general require unbounded types and amounts of work. Sonja's sense-making is largely perceptual; it understands instructions by relating them to the current Amazon-playing situation. When Sonja is given an instruction, it registers the entities the instruction refers to and uses the instruction to choose between courses of action that themselves make sense in the current situation. An instruction can fail to make sense if it refers to entities that are not present in the situation in which it is given, or if the activity it recommends is implausible in its own right.

5.4.1 Instructions reorient attention

Interpreting a noun phrase typically requires and permits new registrations. For example, if you are told "Use a knife" and you don't have one, you can look in likely places and do whatever it takes to recognize something as a knife. You might also remember where relevant knives are, or ask, or engage in an open-ended collection of other ways of finding one. Once you have found *the-knife*, you can register aspects like *the-knife-is-easily-accessible,* and this immediately suggests the proposal *pick-up-the-knife.* Similarly, the processing of verb phrases permits and requires new projections. If I say to you (as someone does in my video corpus) "Don't throw the barrel that way—throw it this way," you will project what would happen if you throw it "this way," which you might not have otherwise done. Then you'll see (if I'm right) that that makes better sense than throwing it "that way."

To summarize, *as part of taking an instruction you register aspects of the situation you weren't previously aware of, and project new possibilities. Based on these new registrations you engage in new activities.* Advice, to put it another way, serves to reorient attention; it tells you not only *what to do* but *what to look at.* This point has been made by Cohen [41]. An analogous one is made by Grosz and Sidner [101, 106], who stress the importance of attentional state in discourse. Grosz and Sidner's point is broader; visual attention is only one part of attention more generally, and the only part modeled by Sonja.

5.4.2 Instructions manage autonomous routines

Instructions recommend courses of action; Sonja's arbitration network takes account of such recommendations. Proposers (described in section 3.2.2) are each responsible for deciding, at every moment, whether the sort of activity they propose makes sense. The arbitration network is responsible for choosing among competing plausible proposals; instructions are taken into account. Because instructions can only influence arbitration when they make sense in terms of activities Sonja could engage in autonomously, their role in Sonja is one of *management* only.

In my video corpus, there are many instances of someone being told "Turn left!" when doing so would run him right into a wall. In this case, the player will turn left when he gets to the upcoming junction. But the instruction is not merely a syntactic ellipsis for "Turn left at the

junction coming up"; I have examples of someone ignoring the instruction for a while longer, because when he gets to the junction an enemy starts shooting at him, and turning left would be suicidal. "Turn left" here 'means' "Turn left when it *makes sense* to do so." This isn't a matter of syntactic ellipsis; the same qualification applies to every instruction. It's not a matter of semantic amplification, either; what *makes sense* means depends on all the particulars of the specific situation. Relevance cannot be determined *a priori*; the use of "Turn left" is a matter for indefinite amounts and types of situation-specific interpretation.

This non-literality of instructions, combined with their management role, supports the notion using the outputs of instruction buffers as inputs exclusively to the arbitration network. An instruction can't force you to do something or tell you what's so; it helps you decide among competing alternatives.

A suggestion like "Get the amulet" does not tell Sonja what to do in the sense that a program would. Getting the amulet can take a lot of work; Sonja may have to navigate about the dungeon and fight off monsters along the way. Sonja is autonomously competent at getting amulets and many other routine Amazon activities; and you can no more tell it to perform the available primitive actions than you can tell me to send a particular sequence of neural impulses to my arm muscles. Instructions do not act like calls on high-level subroutines, either; getting the amulet may have to be interleaved with almost every other activity Sonja is capable of. "Get the amulet" is useful because it tells Sonja that getting the amulet, rather than (for example) going down the nearby stairwell, is the right thing to be doing at the moment.

The contents of the instruction buffers are inputs to the arbitration network (along with suggested activities from proposers and registrations of aspects from registrars). How, specifically, are instructions taken into account in choosing what to do? An instruction buffer's valid bit will typically *support* a particular proposal. This support, described technically in section 7.2.5, allows the system to resolve several types of arbitration decisions that are otherwise difficult:

- Before an instruction is given, there may not have been either any support for the proposal nor any objection to it. In this case the proposal is not taken. The instruction constitutes support for the proposal, so it can be taken.
- There may be several mutually exclusive proposals on the table, among which Sonja has no way to choose. The arbitration network must choose arbitrarily in such a case. If an instruction supports one rather than the others, Sonja can choose it.
- There may be several mutually exclusive proposals on the table, none of which is unambiguously preferable, but one of which is weakly preferred by default. If an instruction supports one of the others, minimal trust (of the sort video game instruction takers routinely extend to instruction givers) leads Sonja to take the supported proposal.
- An instruction may suggest an activity at a higher level of abstraction than any currently active. For example, at one point in my video corpus someone says "Try to get to the third level," when the hearer had just been wandering around aimlessly. Such a suggestion, if taken, can organize other activity.

Sonja implements instances of all these ways an instruction can affect arbitration except the last, which I haven't gotten to. Section 7.2.5 explains in more detail the arbitration machinery

that allows instructions to manage activity.

5.4.3 Clearing instruction buffers

If I say to you "Stop shooting," you may be counted as following the direction even if you start shooting again later. For example, I may have said it because you were in danger of shooting something valuable; as soon as it is out of your line of fire, the instruction is void. Suggestions have temporal extent, which is rarely explicitly demarcated. You can ignore my suggestion as soon as it no longer makes sense. The instruction buffer clear lines (section 5.3) implement this: Sonja must explicitly clear its instruction buffers once a suggestion is no longer relevant. How can it tell when that is?

We shouldn't expect a general answer to this question, any more than to any other concerning real human activity. There are many specific cases in which the answer is clear. (All the instructions illustrating these cases are from my video corpus.)

- Some instructions suggest activities that have an inherent endpoint: "Turn left!" Having turned left once (even were that a durative activity) you would not continue by turning left again.
- Some instructions make sense for a good reason which stops being true: the "Stop shooting" example, for instance.
- Some suggestions become obviously nonsensical at some point. "You want to head, ahm, straight down" will eventually run you into a wall, at which point you'd better turn.
- Some suggestions may be explicitly countermanded. A complex example is " 'Kay, now go ba—keep going—take—left! left! Now go left again, now go down that, that alleyway... Turn—straight!—the way you are going now. Now take a right—take the first right."

I've implemented instances of all of these except the last.

5.4.4 Some kinds of instructions Sonja can use

This section describes some sorts of advice Sonja can use and reviews literature on these sorts of advice. This taxonomy of advice types is intended for expositional purposes only, not as any sort of linguistic theory. The classes are *indexical reference to objects, indexical reference to activities, negations, temporal expressions, spatial expressions, indirect speech acts*, and *repairs*.

I have implemented only one to a few examples of each of these classes. As Sonja is a pilot study of a new architecture, the intention is to demonstrate that constructs like these are possible and to suggest some mechanisms, rather than to achieve full coverage of particular phenomena. These mechanisms are explained in greater detail in chapter 8.

Indexical reference to objects

Typically, instructions support both proposals to take action in the world and proposals to do visual work. For instance, "Head down those stairs" supports both going to a stairwell and performing a visual search to find a stairwell if Sonja is not already registering one. This

implements *indexical reference to objects.* Section 6.2 describes the mechanisms of visual search in more detail.

Indexical reference to activities

Not only the objects used in an activity but the activity itself suggested by an instruction can depend on context. For instance, in Amazon, "Use a knife" can suggest throwing one at an enemy or using one to jimmy a door, and Sonja can figure out which is meant. (Section 8.6 explains how.) I know of only one previous study of this phenomenon, due to Robinson [221], who extended Grosz's focusing algorithm [102], intended originally for nominal anaphora understanding, to disambiguate indexical references to activities.

Negation

Most of the instructions Sonja can use have corresponding *negated* forms: "Don't go in there," "Don't pick up the amulet." The interpretation of these instructions shares machinery with the positive forms. Specifically, the instruction buffer for a negated instruction objects to the same outward activities the buffer for the positive form supports. On the other hand, both instruction buffers support registering the relevant entities if they are not already registered. Thus a negated instruction draws attention to the same entities as the corresponding positive instruction.

Temporal expressions

Temporal expressions in my video corpus include "Turn it *now*—there you go," "When you come back, make sure you take the first right," "Keep goin', straight, then take the right at the end of the hall," and "That dude should be comin' right about now." Sonja implements a few instructions of this sort; "Get the potion and set it off" and "Hit it with a knife when it goes light" are examples.

There are large linguistic and computational literatures on temporal expressions; see, for example, [10, 43, 55, 78, 182, 201, 285, 290]. These literatures are concerned with mapping linguistic temporal expressions to internal temporal representations, typically in an interval algebra.

Sonja implements something quite different: its use of temporal expressions is *in* time as well as *about* time. Sonja's understanding of temporal expressions is constituted by its ability to manifest activity with the same temporal structure the expressions describe. For example, Sonja's use of "Get the potion and set it off" involves two activities, getting the potion and setting it off, which have the temporal relationship (succession) described by the instruction. I know of only one related project, Stucky's Pidgin (personal communication), an artificial language designed for situated communication between people and machines, which is similarly intended to be interpreted *in* time.

Spatial expressions

Earlier in this chapter I have reviewed literature on the spatial grounding of language. Although this phenomenon is ubiquitous in Sonja's instruction use, it is particularly clear in the case of expressions involving prepositions; Sonja can use several instructions such as "Go in and get the amulet" and "Go around the top". Sonja's ability to use these instructions is grounded in its visual routines for parsing the world in terms of spatially extended objects such as rooms and obstacles and for figuring out spatial relationships between objects.

In my video corpus there are several instances of one player warning another that a particularly obnoxious monster has just appeared on the screen with an expression such as "Look out behind you!" These expressions fall into two classes: ones such as "Look out behind you!" that refer to directions relative to the heading of the player icon (the amazon in the case of the game Amazon), and others such as "Look out below!" which refer relative to the coordinate system of the screen. These classes correspond to the *intrinsic* and *extrinsic* uses of prepositions [217].[5] Sonja uses such warnings to switch attention to the indicated monster.

Indirect speech acts

The instruction "Head left" in context usually does not mean simply to go left, but to go left in order to achieve a particular goal which requires going left, and to continue by actually achieving the goal. "Head left" is, thus, an indirect speech act [159, 204, 239], referring to some activity other than that it directly describes. Sonja uses the context to figure out what proposal "Head left" actually supports, and remembers this information so it can carry it out.

Similarly, I may say something declaratively from which I expect you to draw the inference that I am recommending a particular activity. Some examples from my video corpus are "Dynamite's on the ground!" "Here they come!" "There's two guys in front of you," and "We've got both barrels over here." These "instructions" result in visual activities that lead you to reregister the situation and consequently to act differently. The inference involved in understanding these indirect speech acts seems to be a matter of perception and action, not theorem-proving or symbol-manipulating. Having seen that "we've got both barrels over here," which you might not have noticed before it was pointed out to you, it may be obvious that the right thing to do is to pick them up and throw them at the enemies. The sole example in Sonja of a declarative instruction is "Scroll's ready."

Repairs

Human language use is never perfectly communicative; people regularly misunderstand each other. Misunderstandings often result from differences in the set of entities the conversational participants are attending to, for instance. Typically people rapidly recover from miscommunication using any of several linguistic repair mechanisms [40, 130, 219, 236]. Repair is common

[5] Retz [217] actually contrasts three uses, deictic, intrinsic, and extrinsic. Deictic uses are relative to the point of view of the speaker. Since all participants in video game discourse always face the screen, deictic and extrinsic usages are indistinguishable.

in some discourse tasks, particularly those in which the interlocutors do not share perceptual access to the topic of discourse, for example when one gives instructions to another over a telephone line. Repair is rare in video game instruction, however. This is probably partly because there isn't time for it and partly because shared perceptual access to the situation means that misunderstandings are uncommon in the first place. (Two of the three examples of repair in my corpus occur in a short segment when an expert is instructing a complete novice who hasn't yet figured out what the icons on the screen mean.) It is also common, both in video game and other discourse, for minor misunderstandings to go undetected without significant consequence [130].

Sonja implements one repair, "No, the other one," which can be given as an instruction when Sonja has misidentified an ambiguous reference. In section 5.2.3, I explained that the interpretation of "No, the other one" depends on the linguistic context; which entity Sonja reregisters depends on what instruction has been given. For example, if Sonja is carrying out "Head down those stairs," "No the other ones" causes it to look for a new stairwell, whereas if Sonja is carrying out "Get the demon," "No the other one" causes it to look for a new demon. It is not clear what Sonja should do if it is carrying out more than one instruction when told "No, the other one." I think that a good theory of this would require serious analysis of turn taking and the temporal structure of discourse and other complex phenomena. Accordingly I have made an arbitrary implementation decision: Sonja reregisters *all* the entities that have been referred to by pending instructions when it is told "No, the other one."

A related approach to repair is taken in Goodman's FWIM system [95]. FWIM, like Sonja, is motivated by the observation that a hearer has limited resources for determining reference and that repair processes are consequently often required. Like Sonja, FWIM actively searches for referents; it differs in that this search is over a space of internal representations, rather than over external objects, and in its mechanism, which is relaxation. FWIM, like Sonja, uses spatial information in resolving references, but unlike Sonja its model of spatial computation is ad hoc. Also unlike Sonja, FWIM does not take purposes into account in choosing referents; it has a fixed task-independent preference ordering on possibilities.

5.4.5 Generating replies

Sonja has a simple linguistic output mechanism for generating replies to instructions. All of these replies are complaints, generated when Sonja can't make sense of an instruction. Examples are "I haven't *got* a key," "*What* demon?", "Go in *where*?", and "Use a knife for *what*?"; a complete list appears in figure 8.4 on page 140. There are many more situations in which Sonja might usefully generate complains or other replies. For instance, Sonja doesn't complain when given a completely ambiguous reference, but chooses a referent arbitrarily.

Sonja only complains when an instruction can't make any sense: specifically, when it determines that one of the entities it refers to does not exist. Accordingly, Sonja clears the instruction buffer it complains, thereby rejecting the instruction. Since Sonja complains only when instructions could not possibly make sense, it does not allow the instruction-giver to reply to complaints. You can't tell Sonja which knife when it complains "*What* knife." This is acceptable because no reply (other, perhaps, than "Oops, sorry") could make sense. Sonja's existing

instruction-taking mechanisms are sufficient to make sense of clarifying replies if Sonja were to complain of ambiguities. For Sonja to engage in more extended negotiation of reference [40] would in many cases require mechanisms for storing and using more of the linguistic context.

5.4.6 Some issues I'm ignoring

This thesis concentrates on understanding certain aspects of instruction use, and necessarily ignores many more. There are innumerable phenomena even in my video tapes that Sonja does not model. This section will discuss some of them; the list is not intended to be exhaustive.

Sonja makes less use of the timing of instructions as a resource for determining reference than it might. In my video corpus, the temporal course of an instruction or stream of instructions often tracks the activities it recommends. For example: " 'Kay, now go ba—keep going—take—left! left! Now go left again, now go down that, that alleyway... Turn—straight!—the way you are going now. Now take a right—take the first right." I don't see any way to model these phenomena without real time speech input; in Sonja, the game stops while instructions are typed. Davis's Back Seat Driver [53] (described further in section 5.5.2) carefully times the instructions it gives (such as "Bear left here") so they will end at the moment they are needed.

Sonja does not model many phenomena that arise from the interactive nature of situated instruction use. Several large literatures, such as discourse analysis [105, 131, 207], interaction structure theory [61], and conversation analysis [116, 159] study phenomena such as anaphora, rhetorical structure, gaze direction, gesture [173], turn taking, adjacency pairs, and prosodics [119, 206]. Sonja addresses none of these issues.

Sonja does not model many phenomena that arise from the embedding of discourse in a social as well as a physical situation. Issues of politeness, deference, formality, the maintenance of shared social as well as physical reality, and management of the participants' emotions are ignored. (Fillmore [78] provides a catalog of such phenomena under the rubric of *social deixis*; see also Grosz [101].)

5.5 Related work

5.5.1 Instruction taking

I'll review AI research first on instruction taking in general and then on natural language instruction taking in particular. I'll also describe some work on instruction giving, but not provide a full review, since Sonja does not address that issue.

AI's major model for instruction use is in the planning literature. (For reviews and analysis of the planning literature see for instance Agre and Chapman [7], Chapman [33], Georgeff [90], and McDermott [171].) A planner, given a goal and an initial situation, constructs a plan which is used by an executive. Executives are programming language interpreters, possibly with minor extensions; they execute a sequence of instructions blindly and mechanically. Several models of instruction use in the literature take instructions to be plans or plan fragments to be used by an executive; others take instructions to be goals given to a planner. (I'll discuss examples of each in this section.) I find neither model satisfactory, as I don't believe the planning model

of activity; this thesis is motivated in part by a desire to replace this notion of plan use with a more flexible and hence more useful and realistic one. For further discussion, see Agre and Chapman [7] and Chapman [36].

More sophisticated ways of thinking about instruction use begin with McCarthy's proposed Advice Taker [170]. McCarthy described his motivation thus: "A machine is instructed mainly in the form of a sequence of imperative sentences [like a plan or program], while a human is instructed mainly in declarative sentences describing the situation in which action is required together with a few imperatives that say what is wanted." The Advice Taker was to be instructed as a human is. The Advice Taker was never implemented, but a program inspired by it, Mostow's "operationalizer" FOO [184], gets at some of the relevant intuitions. In these models of advice taking, as in Sonja, advice is not executable, but must be made useful. They differ in many other ways, however. The Advice Taker and FOO are given "high-level" descriptions of a domain, where the advice given to Sonja is quite concrete and specific to the immediate situation. Sonja takes advice at "run time," where FOO and the Advice Taker are given it at "compile time." FOO and the Advice Taker transform advice into an "operational" procedure that can be mechanically executed later, where Sonja uses advice only to manage an already-ongoing activity. FOO and the Advice Taker use theorem proving or related exponentially inefficient reasoning techniques to effect this transformation, where the machinery of Sonja is simple and seems more likely to scale.

Laird et al.'s Robo-Soar [154] takes advice in a real, incompletely-modeled robot arm domain. Robo-Soar, like Sonja, is not based on the planning model of activity, and accordingly its operation has some of the same flavor. As in Sonja, the advice given to Robo-Soar applies directly to the current situation, and is not construed either as plans or goals. Unlike Sonja, Robo-Soar learns from advice, reusing it in situations similar to those in which it was originally given. The advice given to Robo-Soar is of two sorts. First, when the system is unable to decide between alternative actions, the advisor can specify one. This is analogous to Sonja's use of advice to manage autonomous competences, except that that this advice must be couched in terms of Robo-Soar's internal processes, rather than in terms of the domain [94]. The advisor can therefore give direct advice about what may be rather abstract internal control decisions; this makes Robo-Soar a less natural fit to human advice-taking. Putting advice in terms of the system's structure rather than in terms of the domain also bypasses some of the problems of learning. Much of the difficulty of learning from the advice given to Sonja would be in figuring out which aspects of the situation make the advice valid and so which situations the advice should be applied to in the future; Robo-Soar does not face this problem. Robo-Soar can also be given a sort of advice that is used in recovering from an incorrect domain theory. This advice explicitly identifies to Robo-Soar the features of the domain relevant to the success or failure of particular strategies. This advice must be exhaustive, unlike the sorts of advice people typically give each other.

There's little work in the AI literature on the use of natural language instruction use. Winograd's SHRDLU [288] translated English instructions into procedures (plans) and executed them. The system did nothing except when instructed. The instructions were immediate, like Sonja's, not high-level like the Advice Taker's. My principal disagreement with this work is

with the plan-based theory of activity that underlies it.

Vere and Bickmore's "basic agent" Homer [278] translates natural language commands into goals and passes them to a planner and executive. Homer operates in a simulated marine world and can take a broad variety of types of commands. Homer and Sonja are outwardly similar in their use of instructions, though Homer's sophistication and breadth is the more impressive. The two systems, furthermore, share a deeper motivation than exploring instruction use: the designers of both wished to go beyond AI research into particular areas such as planning, learning, or parsing, to construct a unified intelligence. Our approaches, however, are quite different, and this is reflected in the underlying architectures. Vere and Bickmore write that "The underlying thesis of this work is that AI component research and computer hardware have in fact progressed to the point where it is now possible, by a resolute effort, to construct a complete integrated agent." I disagree, for two reasons. First, I believe that existing AI theories of particular cognitive modules (components) are in fact inadequate. Planning is a clear example [7]; Vere and Bickmore do not address the problem of search control for planners, even though they report as a difficulty with Homer that, in its tiny simulated world, plan generation takes up to several minutes. Second, I think that existing AI theories reflect deeper misunderstandings. For example, I think that the particular modularity that most AI work assumes is incorrect, and that as a consequence theories of activity and of natural language that are not connected with perception from the outset probably never can be. Vere and Bickmore dismiss this concern by describing their agent as operating "at a cognitive level." Homer's simulated perceptual component is unrealistic, and I doubt that an autonomous cognitive level can exist.

Grosz has initiated a line of research on *task-oriented dialogues*: that is, conversations that naturally occur as a part of concrete activity. Her first empirical studies [100] were of experts directing apprentices in a mechanical assembly task. Her analysis emphasizes the importance of discourse context, including the configuration of the task equipment and the state of completion of the task [103], and accordingly the importance of shared visual access to the task equipment [100] and the resulting shared understanding of the activity [101]. These concerns are reflected in this thesis. Grosz further argues that the attentional state of the conversational participants is central to the structure of the discourse [100, 102]; she has developed this point in a recent paper with Sidner [106]. In Grosz's model attention focuses on parts of a semantic network representation. She argues for a hierarchical structure for discourse paralleling a hierarchical structure for activity; this hierarchical structure is supported by an attentional focus stack that is pushed and popped as subtasks are initiated and completed [99, 100, 102, 103, 106]. Sonja too makes attention a central structuring device for discourse. The attention mechanism I've implemented is narrower in scope than that of Grosz (as mine encompasses only visual attention). Sonja does not privilege hierarchical organizations of activity; I view hierarchical organization as an emergent phenomenon rather than as a correlate of particular machinery [4, 7]. Accordingly Sonja has no focus stack. However, it also does not address a hard problem the focus stack helps solve, namely resolving anaphora by consulting the linguistic context. Grosz's task-oriented dialog studies have been followed up by many other researchers; Grosz et al. [105] provide a survey.

Suchman's ethnomethodological study of an instruction-giving system based on the instruc-

tions-as-plans model [256] was one of the principal sources of the concrete-situated view of activity presented in chapter 2. It was also an inspiration for the design of Sonja. Her analysis emphasizes the importance of the indexicality of instructions. She points out that much of the work in using instructions is in resolving references (both to objects and to activities), so that instructions are not simply executed, but must be made relevant to the concrete situation. She found that in practice the instruction-giving system's plan-execution model of activity led to communicative breakdown. Users of the system would run into trouble in the use of instructions and then understood themselves as initiating repairs of various sorts, which the system had no understanding of. Consequently the system and its users would understand instructions differently, leading to further confusions. Sonja is intended to model aspects of the "situated inquiry" theory of instruction use Suchman proposes as an alternative. In particular, Sonja models the process of making-relevant; the process of linguistic repair, which is central in Suchman's analysis, Sonja models only sketchily. Another aspect of instructions Suchman discusses is their retrospective use to explain what has been done. Sonja does not model this at all.

5.5.2 Reference

There has been much work in both linguistics and AI on specification [13, 14, 15, 16, 104, 105, 150, 199, 248, 249]. I won't review that work here because Sonja isn't concerned with the subject. Some of these studies have explicitly mentioned the problem of external reference and have proposed grounding it in vision, but none have given an account of how to do so. Cohen's empirical study of the pragmatics of referring [41, 42] comes closest, as he is explicitly concerned only with perceptually grounded reference, but he grounds out in a generic PERCEPTUALLY-ACCESSIBLE predicate.

All systems I know of that provide external reference do so causally. Many AI systems have given causal grounding to their representations; the vast majority are vision systems (which represent what they see) and robotics systems (which represent what they manipulate or navigate about in). However, a few previous AI systems have given causal grounding to natural language representations.

Winograd's SHRDLU [288] pointed in the right direction. It allowed an interlocutor to refer to imaginary blocks in a simulated world. Reference was a product of the system's ability to perform simulated actions on the simulated blocks. This sort of reference is arguably causal and arguably even external, though the intimacy of the connection between SHRDLU and its blocks world makes the causal grounding a bit too easy. (The same perhaps could be said about Sonja, but interposing a realistic model of vision between language and Amazon is intended to deflect this criticism.) Similarly, Davey's tic-tac-toe player [50, 51] generated accounts of games it had actually played. His system deliberately exploited indexicality of reference to generate expressions like "the middle of the edge".

The TDUS system of Grosz and her colleagues [100, 221, 280] engaged in a dialog with a human user who was assembling an air compressor. TDUS could refer to objects with expressions such as "the aftercooler elbow"; this expression was causally connected with the actual aftercooler elbow by virtue of the human's ability to recognize it. This provides unarguably

external reference, but offloads the work of achieving it onto the interlocutor.

Davis's Back Seat Driver [53] gives route instructions to a human driver. Like TDUS, it achieves external reference causally in virtue of its human user's perceptual abilities. It uses an annotated map in its model of the driver's visual abilities. For example, based on map locations, it generates instructions like "Look for Merit Gas on the left side." The system has no detailed model of the driver's visual processes, and its judgments of whether or not the driver will be able to see things is limited by the imprecision of the map.

Most systems that attempt causal grounding do so through machine vision. So far as I have been able to determine, none of these has quite made a full causal connection; in each some part of the process relating camera data to linguistic input or output is simulated.

The earliest such system I've found is Badler's [17], which anticipates the form of all the others. His intention was to build a system which would produce natural language descriptions from sequences of images of moving objects. His principal contribution was a general-purpose representation for motion and spatial relationships. This representation combined features of semantic networks and metrical representations from graphics algorithms. He presented unimplemented algorithms that would translate from image sequences to these representations and from the representations to natural language descriptions. The vision algorithms, we can say confidently with fifteen years of hindsight, wouldn't work. More seriously, there is reason to believe that the sort of metrical representations he wanted to extract from images cannot be had (see section 3.3.1), and that the sort of semantic network representations he proposed to give to a text generator are incompatible with essential connectionism. Nonetheless, his account of the aggregation of primitive motions and relationships into more abstract, meaningful ones is valuable.

The ALVEN system of Tsotsos et al. [271] built on Badler's work and incorporated more realistic vision algorithms with provision for noise and occlusion. Its final output was to be a frame representation suitable for text generation; the authors did not actually connect it to a text generator. Whether the implementation ever actually ran is unclear from the paper.

Four groups, working independently in the late Seventies, used representations similar to Badler's, though not all seem to have known of his work. Waltz and Boggess [281, 282, 283] considered the problem of constructing from a natural language description a spatial model to use in spatial reasoning. The system ran in some simple cases. Okada [196] and Abe et al. [1] describe implemented systems which constructed representations from stories with accompanying cartoon images. The visual and natural language processing in these systems seem in retrospect ad-hoc and unlikely to generalize. A similar, unimplemented proposal was made by Marburger et al. [165].

This last work has been followed up by a series of projects in Germany which have continued through this decade [191, 217, 218, 237], particularly in the VITRA (visual translator) project. These systems produce natural language descriptions from sequences of images of moving objects. They are particularly concerned with the use of prepositions and motion verbs, and draw on a linguistic literature on the grounding of prepositions in spatial primitives; see, for example, Adorni et al. [2] and Herskovits [117]. The systems are all fairly similar for our purposes; I'll describe CITYTOUR [217, 237], which is best documented. CITYTOUR's visual system

segments moving objects from the images, yielding a database of (object, location) pairs. A geometrical analysis system produces qualitative descriptions of the motions of the objects in the same ontology used by the natural language system. This ontology includes notions such as passing, being close to, following a path, and stopping. Finally a generation system produces text from this description.

CITYTOUR is the best implemented connection between language and vision to date. However, it has several weaknesses; most important is that the model of vision is crude and ad hoc. It requires a prior geometric model of the stationary background of the motion sequences and identifies only the bounding boxes of moving objects. In order to actually connect it to the language system, the types and orientations of the objects have to be entered by hand. It is not general purpose or motivated by a broader conception of vision. These deficiencies can probably be rectified. Were this done, the resulting system would differ from Sonja principally in that it would be a passive observer describing events, rather than an involved participant; and consequently it would provide more limited opportunities for linguistic interaction.

Chapter 6

The peripheral systems

This chapter explains Sonja's peripheral systems, exclusive of the linguistic input system, which I've already described in chapter 5. Almost all of the chapter is concerned with vision, as the other peripheral systems are trivial. In particular it is concerned with the visual routines processor, as I have not implemented other visual systems.

As I mentioned in section 3.3.4, Sonja's vision model is based on that of Ullman's visual routines theory. Ullman's proposal, however, is sketchy in many respects. Although other researchers have worked in the visual routines framework (for instance Mahoney [164] and Romanycia [223]), no one has produced an implementation that uses visual routines in a natural task domain. Thus I have had to extend the proposal and fill in many details. I've specified in greater detail the interfaces to the operators Ullman suggests, I've extended the set of operators, I've specified much more concretely the interface between the VRP and the central system, and I've worked out ways to compose operators into routines, which he addressed only briefly.

Section 6.1 describes visual attention, the ability to access subsets of the early retinotopic representations. Converging forms of evidence suggest that visual attention is implemented with an addressing pyramid that routes information from the selected location to the central system; this device is a key locus of the "crunching" of retinotopic representations down into narrow buses. Sonja incorporates such a device.

Section 6.2 describes visual search, the ability to find locations in an image that have specified properties. Visual search has been shown psychophysically to depend on visual attention; in many cases it proceeds by serially enumerating and testing locations. Sonja models the psychophysical properties of human visual search.

Section 6.3 describes intermediate objects, which are the state variables maintained by visual operators and used in communication between the central system and the visual routines processor. There are four sorts of intermediate objects, called markers, lines, rays, and colors.

Section 6.4 describe various technical issues in the implementation of the visual operators. Particularly it explains how I've bypassed early visual processing.

Section 6.5 describe criteria for choosing operators. I describe local criteria on individual operators and global criteria on a set of operators. (Section 9.5.4 evaluates the set of visual operators I chose according to these criteria.)

Section 6.6 describes the specific visual operators used in Sonja. Figure 6.9 in that section gives a complete table of them with pointers to the subsections of 6.6 in which they are described.

Section 6.7 describes various human visual systems, other than the VRP, that are not modeled in Sonja. Modeling them might eliminate some weaknesses in Sonja. The systems I describe are bottom-up segmentation, shape matching, foveation, and spatial memory.

Section 6.8 describes Sonja's non-visual peripheral systems, particularly the motor systems.

For a minimal reading, you will want to read the first parts of sections 6.1 and 6.2 on visual attention and search, skipping the end bits where they get technical, and only the introduction of section 6.3 on intermediate objects. You'll want to look at figure 6.9 and flip through section 6.6 on the specific visual operators, because they are made extensive use of in chapter 8. The rest of this chapter is dispensable.

6.1 Visual attention

Visual attention is the ability to differentially apply visual processing to a subset of a scene. It is taken as consisting of two components: *overt* visual attention, or gaze direction, which can be studied with an eye tracker; and *covert* visual attention, which is neurally mediated and can only be studied indirectly. Following standard usage, I will use "visual attention" to mean covert visual attention when this will not result in confusion.

Visual attention is a central intermediate visual process, and one that is important in Sonja's operation. There are large psychophysical and neuroscientific literatures on the subject; I will review some of this literature in this section. While the data are uncertain and sometimes contradictory, there is broad agreement about some general facts.

6.1.1 Psychophysics of attention

The primary evidence for visual attention comes from psychophysical studies, for instance those of Posner *et al.* [210]. In a typical experiment, subjects are required to react to the onset of an event (such as a light coming on) somewhere in the visual field. The results of such experiments are that

- Reaction times are lower when the subjects are told where in the field the event will occur, suggesting that visual resources can more effectively be brought to the detection task when the location of the event is known.
- Covert visual attention is independent of (overt) gaze direction: it operates even when the subjects do not foveate the indicated location [210, 189], and brain lesions that eliminate voluntary eye movements do not affect covert attention [211].[1]
- Visual attention is at least partly cognitively penetrable and under voluntary control; the event location can be indicated by non-natural cues [210].

[1] There may be weak interactions between covert and overt attention. Kröse, for instance, presents evidence that the detectability of a "T" in a background of "L"s is a function of retinal eccentricity [152]. Other researchers (such as Nakayama and Mackeben [189, p. 1639]) have failed to find such effects.

- The bulk of the evidence suggests that attention can be directed only to a single contiguous subset of the image [68, 210].[2]
- The diameter of the attended subset can be varied voluntarily [68, 132, 267, 269]. The possible shapes it can assume and the distinctness of its margin are uncertain [267].[3]

These observations are summarized as the *spotlight* model of attention, in which attention "illuminates" a chosen subset of the image. The nature of these subsets is unclear, so I will refer to them neutrally as "locations"; I'll have more to say about them in section 6.7.1.

6.1.2 The addressing pyramid

Koch and Ullman [147] have proposed an *addressing pyramid* as the hardware supporting the attentional spotlight. (This proposal was inspired by neuroanatomical speculations of Crick [46]; similar proposals have been made by Anderson and Van Essen [12], Treisman and Gormican [270], Tsotsos [274], and others.) The addressing pyramid is similar in function to the addressing hardware of a Von Neumann machine: it routes information from a selected part of a peripheral array to a central location. In the case of a Von Neumann machine, this information is the contents of a memory location; in the case of the attentional hardware, it is the contents of the early representations in a connected region of the retinotopic array. The pyramid gets its name from its two-dimensional hierarchical tree organization. It consists of a series of exponentially smaller stacked layers that route information upwards to a central node (figure 6.1). Each level is composed of an array of nodes, each of which selects one of the four nodes beneath it to route to its superior.[4] Thus the system as a whole acts as a recursive *winner-take-all network* [76], eventually routing the contents of just one leaf node up to the root. These leaf nodes actually each contain the values of the early representations at one retinotopic location. I'll describe how pyramid nodes choose among their subnodes in section 6.2.

A variable diameter spotlight can be implemented by having some nodes send up a combination of the values of its inferiors, rather than choosing a single one (figure 6.2). This has been suggested by Treisman and Gormican [270], who propose that interior nodes in the pyramid can, selectively, pass up the average of the early values of their inferiors, rather than passing up the exact value of a single chosen inferior.

This addressing pyramid, then, is a key locus of the "collapse" of retinotopic representations into narrow buses that is criterial of intermediate vision. In effect it implements pointers into the image. In section 3.1.2 I argued against the use of pointers; the resolution involves three factors. Pointers into the image are not *general* pointers, from anywhere to anywhere, but range only over early representations. Moreover, they can only be accessed through a single root node, so the hardware cost is only linear, not quadratic, in the size of the retinotopic representations. Finally, addressing the visual image is probably so evolutionarily important

[2] Earlier studies by Shaw and Shaw [241, 242] suggesting that attention can be split over arbitrary subsets of the image have not been confirmed by more recent studies.

[3] Eriksen and St. James [68] present evidence for an indistinct margin, with processing efficiency decreasing gradually from the center.

[4] This organization is termed a *quad tree* in computational geometry [232].

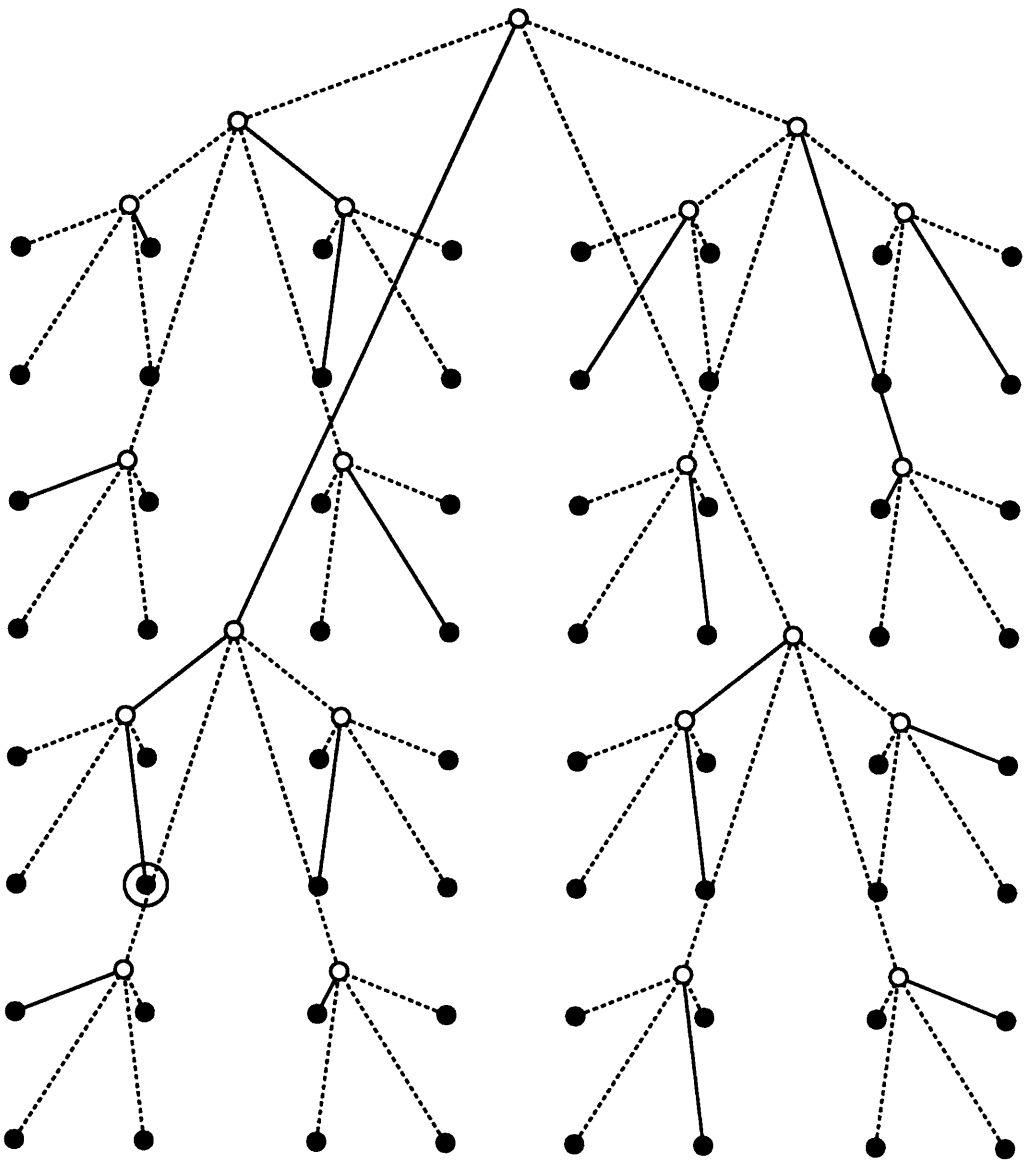


Figure 6.1: The addressing pyramid. Leaf nodes (solid circles) contain buses of early properties. Interior nodes (open circles) choose information from one of their four inferiors to route to their superior. Selected communication paths are drawn as solid lines; the selected leaf node is circled.

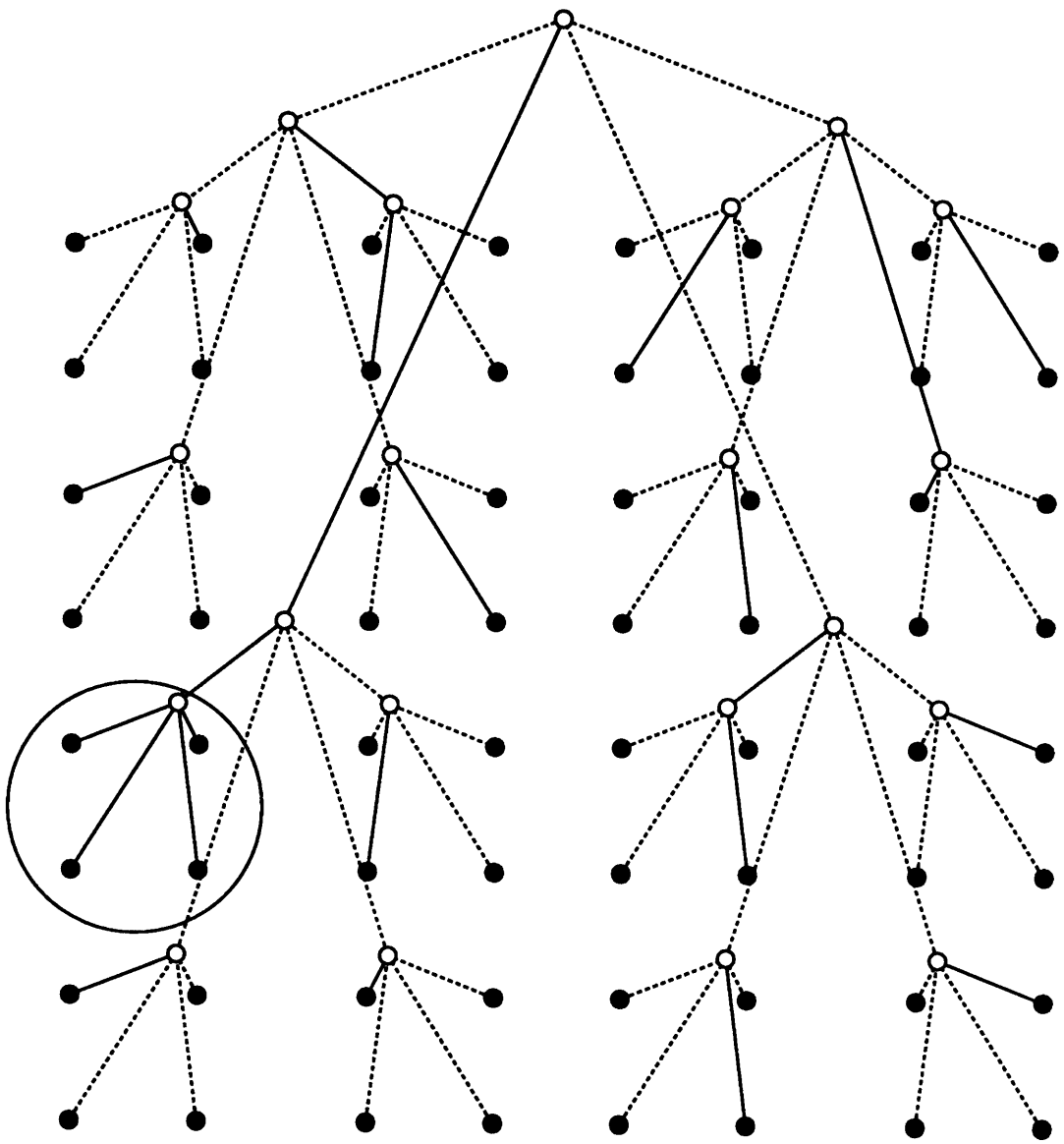


Figure 6.2: Variable diameter addressing. Here the whole encircled region (containing four leaf nodes) is addressed. The interior node immediately above this region passes up the average of the four leaf values, rather than selecting one. As drawn, this architecture allows for addressing only of regions that align with exponential quads. More general addressing could be provided by using many overlapping, tangled quad trees.

that it would be worth devoting a substantial chunk of brain mass to the task. (An intriguing possibility is that this addressing machinery is reused for other purposes for which pointers would otherwise be necessary. Visual imagery might work by allowing the central system to "paint" on retinotopic representations using the addressing pyramid. Much of human thinking (of the sort often modeled with pointer computations in AI) might proceed by painting and accessing images through the addressing pyramid.)

I implemented an addressing pyramid in Sonja using a circuit similar to that proposed by Koch and Ullman. I'll describe it further in section 6.2.

6.1.3 Neuroscientific study of covert visual attention

There is convincing neurophysiological evidence for the existence of visual attention, but the details of its operation are still murky. Moran and Desimone [183] found that the receptive fields (RFs) of neurons in areas V4 and IT, but not V1, are shifted by changes in visual attention. (V4 is an intermediate step in a neural pathway from V1 to IT.) Specifically, V4 and IT neurons whose RFs included the attended location responded strongly to stimuli at this location and weakly elsewhere in the RF. However, V4 neurons whose RFs did not include the attended location responded strongly to stimuli anywhere in the RF. Most neurons in IT, on the other hand, have RFs that subtend much of the visual field, and almost none of them will respond to unattended stimuli. This suggests that IT is at or above the root of the pyramid, whereas V4 is somewhere in the middle and V1 is at or below the leaf plane. This would mean that the addressing pyramid is actually tangled up with early processing, rather than coming after it as in my implementation.

Crick [46] has proposed an alternative theory, that the thalamic reticular complex is the locus of operation of visual attention. In particular, he suggests that it acts as a winner-take-all network which selectively gates portions of the retinal signal arriving at the lateral geniculate nucleus (LGN) into the cortex.

Mountcastle *et al.* [187] found that attention acted differently on neurons in V4 and in area PG of the parietal lobe, and proposed a functional explanation for the difference. This suggests that attention is not be a unitary phenomenon, as in the spotlight/pyramid theory, but has several components.

6.1.4 Implementations of covert attention

Feldman and Ballard [76] provided the first suggestion I have found for a computational implementation of covert attention. They intended both to model Treisman and Gelade's [268] psychophysical studies and to solve the connectionist crosstalk problem. They suggested using a winner-take-all network; their discussion is abstract and apparently the suggestion was not implemented. Koch and Ullman [147] similarly did not implement their proposal.

The earliest implementation I've found is due to Fukushima [87]. He used a hierarchical winner-take-all network of connectionist units. His implementation seems to have been motivated principally by engineering concerns; it does not try to model psychophysical data accurately. For example, the attended subset of the image does not need to be contiguous.

Strong and Whitehead [254] present an implemented model of *overt* visual attention inspired by Feldman and Ballard's work and similarly intended to solve the crosstalk problem.

Mozer's [186] implementation of covert attention models the psychophysical data better than Fukushima's; it can attend only a single contiguous region. He uses a winner-take-all network implementation similar to that of Koch and Ullman. Like me (and unlike Fukushima) he implemented both content and pointer addressing modes; I'll explain these in section 6.2.2. Mozer's winner-take-all network is not implemented hierarchically (as a pyramid) but rather retinotopically. Koch and Ullman argue that their hierarchical organization results in faster convergence of the winner-take-all computation than would a locally connected network such as Mozer's. Mozer's network uses real-valued units and relaxation, rather than boolean switching elements. He implemented continuously variable diameter; I didn't. Mozer's network operates by selective enhancement of signals from the attended region, rather than by selective inhibition of signals elsewhere (as does Fukushima's implementation and mine). Moran and Desimone's results suggest that the human attentional system operates by selective inhibition; Tsotsos [274] argues that selective inhibition should make the winner-take-all network converge more quickly. These implementation differences are minor. Mozer did not use attention in serial visual search; I did.

6.2 Visual search

6.2.1 Psychophysics of visual search

Many of Sonja's visual operators are devoted to finding parts of the image which have specified properties. These operators are inspired by psychophysical research on *visual search.* (For surveys, see Julesz [132] and Treisman and Gelade [268].)

The central result of this literature, due to Treisman and her colleagues [268, 270] and replicated by many other researchers, concerns the distinction between *parallel* and *serial self-terminating* search. The experimental paradigm motivating this distinction examines the time required to determine whether or not an object with specified properties exists somewhere in an artificial scene. The results depend on the nature of the property and also on what objects are found in the scene (figure 6.3). Tasks varying on these dimensions segregate strongly into two classes. In the first class, the time required is independent of what is in the scene. In the second class, time is a linear function of the number of "distractor" objects in the scene, and on average is twice as long in cases in which the object to be found is not present than when it is present. Treisman interprets the first class as indicating that certain properties are computed in parallel and in constant time over the entire visual field. In cases in which the desired property is one of those computed this way, determining whether or not any object with the property is present can be computed in constant time as a global OR over the resulting retinotopic map. The object, if present, is said to "pop out" of the display, and such properties are called *pop out* properties. Treisman interprets the second class as indicating that, in cases where properties are not computed in parallel, visual attention must be applied sequentially to each item in the field to determine whether or not it has the desired property. In these cases

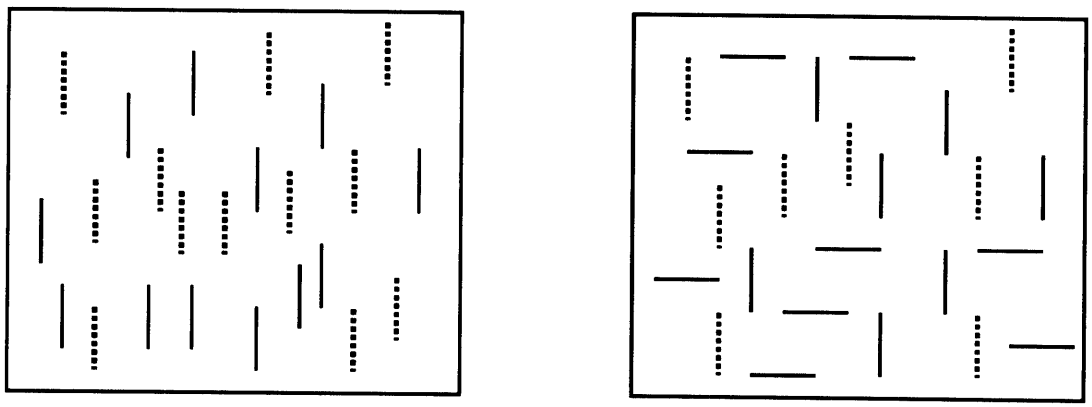


Figure 6.3: Psychophysical displays requiring parallel or serial search. Determining whether or not there is a horizontal line or a dashed line in a display such as the first one takes time independent of the number of objects. Determining whether there is a horizontal dashed line among vertical dashed and horizontal solid lines (as in the second display) requires serial search and takes time linear in the number of objects.

if a single object of the desired type *is* present, on average half the objects in the field will be examined before it is found; if one is *not* present, every object in the field must be examined. This "serial self-terminating search" accounts neatly for the reaction time data.

Given this paradigm, we can ask what features pop out. Treisman and Gormican [270] report that colors, grey level, line curvature, line orientation, line length, proximity and numerosity of clusters of lines, line termination, direction of motion, and stereoscopic depth are pop out properties. These results are particularly interesting because there is convergent neurophysiological evidence for early retinotopic representations of many of these properties [167, 169]. On the other hand, intersection, line juncture, angle, connectedness, containment, and aspect ratio are not pop out properties. Neither are any conjunctions of pop out properties.[5]

6.2.2 An architecture for visual search

These results suggest an architecture like that of figure 6.4. Early modalities compute retinotopic maps bottom up.[6] Each retinotopic map is retinotopically connected to an *activation map*. Let us say that an early property consists of a *dimension* (which is a particular early modality) and a *value* on that dimension. Thus color is a dimension and red is a value. An activation map acts as a value filter; it has binary elements which are "on" at points where

[5] Nakayama and Silverman [188] have found an exception: conjunctions of stereo depth information with one other early property pop out. Thus, for example, blue things eighteen inches away can be found with parallel search. There is probably special-purpose hardware to support this adaptively important case.

[6] This is an abstraction from the neuroscientific data, which show that retinotopic maps are actually computed in a cascade of stages which are probably interwoven with the attentional pyramid [169].

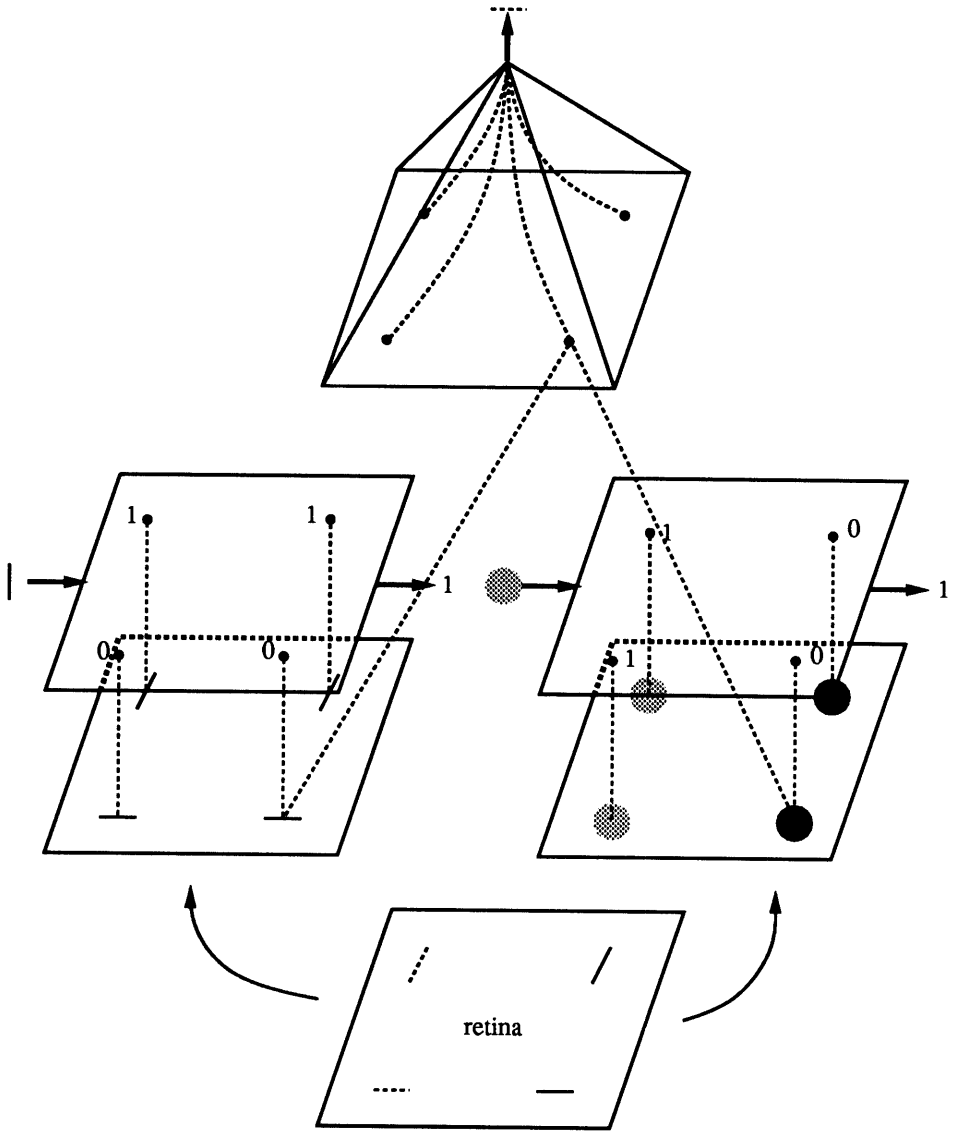


Figure 6.4: An architecture for visual search. In this example the retina is presented with lines varying in orientation (horizontal, vertical) and color (symbolized by solid and dashed). Two early maps compute these properties. Activation maps compute whether a desired value (vertical or dashed, input from the left) is present in the corresponding early maps at each point. A global OR (output on the right) supports parallel search. The addressing pyramid supports serial search, routing to the root, and thereby combining, all the early properties corresponding to a particular addressed location (the lower left in this case). I have omitted most lines connecting to pyramid leaf nodes to reduce visual clutter.

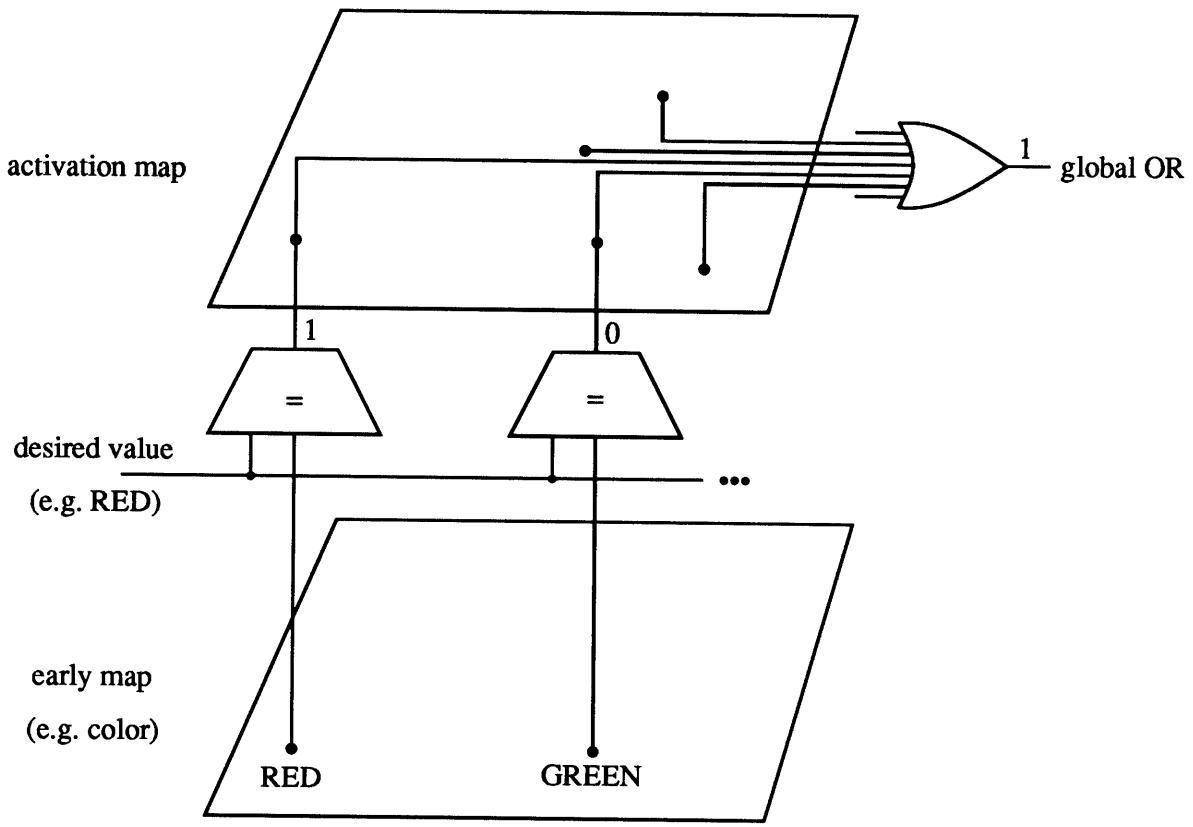


Figure 6.5: Structure of the activation maps. At each point the early value is compared with the desired value to give a boolean activation value. A global OR of activation values is computed over the entire map.

corresponding elements in the early map have the desired value. A network extending globally over the activation map distributes the desired value to all the activation elements. (An alternative implementation would use a separate activation map for each early value. This would correspond to *value unit encoding*, which seems to be the rule for cortical neurons [18].) Another global network computes the global OR (figure 6.5).[7]

This much machinery is sufficient for tasks that require one to decide whether or not there is a point in a scene which has a particular early property, and therefore accounts for parallel visual search. What about combinations of early properties? A straightforward solution would be to provide activation maps for all possible combinations. There's a good engineering reason not to do this: there are too many combinations. Assuming that there are a dozen early maps,

[7] Alternatively, as Treisman and Gormican [270] have suggested, the global OR could be computed using the addressing pyramid by adjusting the diameter of the attended area to span the entire visual field.

there would be $2^{12} = 4096$ combination maps. (Value unit encoding of the individual activation maps would increase the exponent substantially.) Since retinotopic maps are known to each take up a significant chunk of the cortex [169], this is infeasible. An alternative is serial application of visual attention, as proposed by Treisman.

Serial visual search requires enumerating candidate and testing to see if they have the desired property. Various enumeration schemes are possible; I'll propose a simple one which matches the psychophysical data. To enumerate candidates, you pick one of the early dimensions involved in the compound desired property and enumerate all the items that have the desired value on that dimension. For instance, if you are looking for a vertical blue line, you can enumerate all the vertical things and check if they are blue or enumerate all the blue things and check if they are vertical. Enumeration involves repeated application of two primitives, *content addressing* and *return inhibition*, which affect the way addressing pyramid nodes select among their subnodes. I'll describe content addressing, candidate testing, and return inhibition in the remainder of this section, and show how they can be combined into an algorithm for visual search.

In content addressing, late vision specifies an early dimension and value, and the addressing