Book Review

David Chapman,
*Vision, Instruction, and Action**

Damian M. Lyons

Philips Laboratories, North American Philips Corporation,
345 Scarborough Rd, Briarcliff Manor, NY 10510, USA

Recieved June 1992
Revised September 1992

1 Introduction

Chapman's aptly named book *Vision, Instruction, and Action* is a first-rate exposition of the *concrete-situated* approach that he and Phil Agre have been advocating for some time. The book describes the theory and implementation of Sonja: an *integrated* system that uses instruction in the course of visually guided activity to produce sensible behavior in a domain characterized by complexity, uncertainty and immediacy. The domain is a video game called Amazon — a game with most of the complexities of the Pengo video game (the domain for Sonja's forerunner, Pengi[2]) plus some. While Sonja is an integrated system, it's not by any means a complete system: Chapman emphasizes one kind of visual system, the use of advice in decision making, and the selection of action, in that order. Nonetheless it provides a good concrete illustration of the tenets of the concrete-situated approach. Researchers familiar with this approach as well as those new to the area will find this book useful.

This review is structured as follows. Section 2 contains a description of the approach and constraints that Chapman brings to the problem of building an integrated system. Section 3 describes the architecture and some of the mechanisms used to build Sonja. Section 4 is a description of the system in action. Section 5 is an overview of the appendix material:

*(MIT Press Cambridge, MA, 1991); 295 pages, $35.00

proposed extensions to handle collaboration and skill-acquisition.

2 Approach

2.1 Concrete-Situated Approach

In previous work [8], Chapman has argued that the 'classical' approach to planning (e.g., STRIPS, NOAH) has problems with computational complexity, and has suggested that it might be appropriate to take an entirely different approach to the problem of producing sensible behavior in complex environments. That new approach, called the *concrete-situated approach,* has since been described in a series of research papers with Phil Agre [1, 2]. The book does a lucid job of explaining this approach in Chapter Two, and I found this section to be both compelling and rewarding to read.

There are seven key aspects of the approach espoused by Chapman:

1. Routineness. The approach assumes that activity is mostly routine — regular, non-problematic, practised. In routine activity, no 'new thoughts' (a term which Chapman defines more precisely) are required. The approach doesn't deny the need for non-routine activity. However, it states that the mechanisms that support activity should *primarily* support routine activity.

2. Situatedness. The key decision-making resource available to an agent is its concrete situation — the configuration of the environment in which the agent is situated, the here-and-now. Appropriate use of this resource, through perception, avoids the problems of approximate and out of date representations.

3. Interactivity. One of the most interesting aspects of this approach is that it proposes that the organization of activity is not a sole property of the agent nor its environment. Rather the organization of activity is *shared* between them. Chapman describes this by saying "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." This aspect of the approach shifts the focus from building agents that attempt to completely control the environment to agents that exploit patterns of interaction with the environment.

4. Dynamics. A dynamic is a pattern in this interaction of agent and environment. Such patterns may be noted by an external observer but are not explicitly represented in their entirety in either the agent or the environment. Dynamics are acausal; although an observer can

attribute effects to a dynamic, the dynamic does not necessarily exist just for the purpose of producing those effects. This is not to say that dynamics cannot be exploited to accomplish work.

5. Routines. A routine is "a regular, practiced pattern of activity" — a dynamic that has been exploited so that the agent gets work done. The routine is the unit of activity in the concrete-situated approach. Designing an agent requires considering what mechanisms will allow the agent to engage in appropriate routines with the environment.

6. Improvisation. The concrete-situated approach claims that the best strategy for an agent to adopt in determining what to do next is for the agent to continually inspect its concrete-situation and use it to redecide what to do. Chapman calls this improvising. It ensures that an agent's actions are always chosen based on the latest data about the environment; uncertainty about the state of the environment is simply not allowed to build up. Improvisation strongly constrains what mechanisms can be used for computing what to do next — there is not sufficient time to engage in theorem proving or problem solving.

7. Deictic Representation. The concrete-situated approach espouses an unusual kind of representation referred to as 'deictic'. The environment is segmented and represented in units which are *relative* to the agent and its ongoing activities. There is no objective view of the environment. (Subramanian and Woodfill [18] describe some of the advantages and trade-offs implicit in this agent-centered view of representation).

2.2 Choice of Domain

Pengi [2] was constructed by Agre and Chapman as a first illustration of the concrete-situated approach. Pengi played a video game called Pengo. This video game modeled many of the characteristics of real-world domains that were problematic to classical planning systems: there was an enormous number of objects that had to be tracked, objects in the game exhibited unpredictable behavior, and the game required a response time of well under a second. Chapman has also designed a video game, Amazon, as the domain for Sonja. The game is similar to Pengo in demanding a timely response to a complex, uncertain and changing environment, but was also designed to test advice taking, handling longer-term interactions, and handling a more ontologically complex domain.

Amazon is a dungeon-and-dragons style video game. The objective of the game is to make the amazon wander the dungeon, which is partitioned into rooms, collecting goodies (amulets, scrolls, keys, etc.) and not get it killed by the hordes of monsters that inhabit the place (ghosts, demon bunnies, piles of bones, etc.). The amazon can fight by throwing shuriken (nasty three-pointed objects) or by using a knife. As you might guess, amulets and scrolls have special powers which if understood can be used to fight more effectively. Tools such as knives can be used for multiple purposes. Amazon health is a variable quantity which is decreased when monsters attack and increased when food is eaten. Amazon lacks the treasure 'reward' normally part of such video games. Chapman doesn't allude to this point; however, its clear from his implementation that Sonja explores without recourse to reward motivation. It's important to note that Sonja plays Amazon from the same perspective a human would, an overhead view of a portion of the dungeon.

Chapman emphasizes that Sonja's domain is *not* a simulation. The point is that Sonja's domain is *not* exploring dungeons; rather it is playing the video game called Amazon. Sonja is not the amazon object in the game; it moves the amazon around the same way a human player does and with the same kind of visual input. Playing video games is a realistic and demanding human activity. Admittedly, the interface to the video game is quite simplified: rather than using a camera and mechanical hand hooked to a joystick, Chapman hooks Sonja's perceptual and motor structures up to the video game data structures. Human video game players seem to 'identify' themselves with the agent in the game. Even reading about video game playing seemingly invokes this effect, and it can be difficult at times to appreciate the separation between Sonja's perceptions and those of the amazon.

2.3 Essential Connectionism

We find out in Chapter Three that Chapman has two objectives for his work; not just to build an integrated, intelligent system but to do so with *biologically plausible* hardware. To address this second objective, he sets forth the following set of well-established facts about the brain by which, he claims, an implementation must abide to be biologically plausible.

- The brain is composed of a great many components,
- each of which is connected to many others, and
- each of which carries out some relatively simple computation,
- slowly, and
- using information mainly from its connections.

Chapman calls these the *essential connectionist* constraints and argues that they prohibit the use of pointers, variables, inspectable data structures, interpreters and dynamic storage. Pointers are disallowed because they cannot be easily implemented in a connectionist framework (they would require a crossbar switch); and, while the jury is still out, the evidence is against the brain implementing them. If pointers are disallowed then variables and inspectable data structures cannot be implemented in a general fashion.

Disallowing pointers also makes the usual computing concept of dynamic storage more difficult to implement. However, in a connectionist framework the concept of dynamic storage maps to the concept of creating *new connections* between components. Chapman additionally claims that his routineness assumption (no 'new thoughts') rules out this analog of dynamic storage. Routine activity is therefore defined as activity that can be engaged in by an architecture with a fixed connection pattern.

Interpreters (virtual machines) are ruled out for speed reasons. Given the slowness assumption of essential connectionism, if connectionist hardware were to 'emulate' other hardware, then this emulated machine would run too slowly to handle activities such as video game playing. Connectionist computation "must stay 'close to the hardware'."

Chapman chooses depth-bounded digital circuits — a digital circuit with a bound on the length of the paths between inputs and outputs — as his implementation medium. While these are not a plausible model of detailed neural functioning, they do obey the list of essential connectionist constraints.

2.4 Criteria of Success

Chapter nine presents an interesting discussion of how Chapman would like to see his work evaluated. I think this discussion is worth reading early on since it motivates the style and content of the book.

Chapman argues that AI needs an evaluation criterion — a definition of what constitutes good work — of its own. He claims that an implicit criterion seems to have arisen, and cites as evidence the fact that several highly regarded and seminal AI papers (Minsky's Frames [13], Hayes' Naive Physics[10], and McCarthy's Advice Taker [12]) would fail evaluation criteria from math, science, philosophy or engineering. Nonetheless they are widely regarded as examples of good AI work. These papers do not focus on results, on demonstrations of superior technology or on producing new theorems. Rather what these papers offer are *approaches*: "a way of doing research: a way of looking at phenomena, of choosing issues to explore, of stating and attacking problems." From this viewpoint, the role of implementation is not to test a theory or demonstrate superior technology, but to provide concrete illustrations of an approach. An implementation is successful when it persuades other researchers to adopt the approach. Chapman cites NOAH as a particularly successful implementation since it has engendered much imitation.

This argument is in part a response to the school of thought that considers AI to be an unscientific pursuit simply because it has as yet featured little rigorous experimentation and few widely known theoretical results. Chapman admits this lack. However, he maintains that this state of affairs is fine because, "AI is *largely about* approaches," as evidenced by the Minsky, Hayes and McCarthy papers. Thus he argues for an explicit, privileged role for work describing approaches. I think he's taking the point a little too far here. Approach papers do indeed play a strong and vital role in AI. I would not consider this due to the intrinsic nature of the field, however, just to its relative immaturity. A field matures by finding which approaches are good for which problems — by the evaluation of works (such as this book) which propound an approach and attempt to establish its usefulness. On the other hand, the point that implementations are exemplary of approachs is well made. The evaluation of approach papers can be difficult precisely because they lack concrete results. Describing the implementation of a suggested approach is one way to make the work more concrete and hence more open to evaluation and (potentially) imitation. It is important to value implementations for this reason *as well* as for reasons of performance testing.

It is worth bearing in mind when reading this book that Chapman does not set out to produce formal results or to exhaustively compare Sonja with other work. The main objective of the book is to describe his approach in a concrete fashion. The book is rewarding when read from this perspective: I found the implementation clarified issues raised in the descriptive early chapters and I found a number of ideas I plan to make use of myself. However, I would have liked also to see the results of some test runs, some performance figures and some more detailed comparisons with other technologies. I wanted to see these not necessarily because I believe scientific or engineering evaluation criteria should be applied, but because these kinds of results are also illustrative devices, aids to understanding the Chapman approach.

3 Architecture

Sonja's architecture follows precedent in being divided into a central system (concerned with choosing what to do) and a peripheral system (concerned with sensory and motor processing). The exact division between central and peripheral systems, however, lies along an unusual line. The central system is homogeneous and non-modular, and Chapman does not argue for its generality. The design of the peripheral system is modular and follows neurophysiological evidence. The only peripheral system considered is vision. Sonja also takes and uses advice; however, advice input is not seriously modeled as a peripheral system. Sonja's motor system is a direct connection to the video game controls.

3.1 The Central System

Sonja's central system has three kinds of components: registrars, which interpret sensory data in a deictic fashion; proposers, which signal the appropriateness for specific courses of action based on registrar input; and arbiters, which arbitrate between the proposed courses of action. Registrar, proposer and arbiter components are each small parallel and interconnected circuits that remain continuously active. Chapman cites Minsky's Society of Mind [14] for comparison; another example would be Selfridge's Pandemonium [17].

Note that the segmentation of sensory data along deictic lines is done in the central system. This gives an important advantage. In more conventional renditions, such as Albus' RCS[3], sensory data is segmented in a separate sensory hierarchy. The large amounts of raw sensory data that occupy the bottom levels of the hierarchy are compressed into more compact, "object" related representations at higer levels. This hierarchy can usually communicate laterally with a parallel modelling or task hierarchy (the equivalent of Chapman's central system). However, such communication is restricted to occur only between nodes at the same level. Thus, not only is a task restricted to the sensory segementation provided in the hierarchy, it is also restricted to just the sensory primitives available at its own level. Thinking of perception in this way, as a separate, general-purpose module, is both inaccurate from a neurophysiological perspective and is a barrier to generality.

It is reasonable to assume that perception and action developed in animals hand-in-hand [21]. Perception was used to solve quite specific and time-constrained problems, and the solutions so derived were not necessarily generally applicable. Arbib [4] cites the existence of multiple, distinct visual systems in the frog — one for prey acquisition, another for preditor avoidance, another for barrier navigation, etc. This argues for considering perception-action units as the basic unit, rather than considering a perception action split. Sonja's central system captures this distinction nicely for (at least one kind of) vision.

Separating off perception also limits the generality of an implementation.

In robotics the 'standard' approach is to build a sensory subsystem capable of recognizing the occurrence of any one of a set of geometric object models in a scene. This sensory subsystem then updates a central database [11]. The remaining components of the robot control system query this data base when they need perceptual information. Although this approach sounds eminently reasonable, it is next to impossible to implement in a general fashion in a nontrivial environment. For a start, the notion of what constitutes 'objects' in a task — obvious to humans — often does not coincide with what geometric matching pulls out of the image. The system ends up doing either too much work, or too little. It ends up doing too much work if precise geometric information is not important for the task; It ends up doing too little if the correct recognition criteria were not geometric but were, say, functional ('cup-ness' for example). Even when the environment is controlled sufficiently to support geometric matching, a good set of models for one task is often not so convenient for another.

The problem is not with geometric modeling or returning precise descriptions. The problem is with the 'one size fits all' approach to breaking the environment down into the right perceptual units for the task at hand. Different tasks require different definitions of 'objectness'. Their requirements include functional and relational as well as geometric models and they require differing degrees of data precision. Sonja's architecture supports this flexibility by including registrars as part of the central system.

Chapman uses MACNET, a LISP-based digital circuit specification language to write the central system. MACNET translates a circuit specification program into a (somewhat optimized) circuit description that can then be executed on a circuit emulator. Use of MACNET enforces the digital circuit hardware constraint. The book presents a special macro language designed to support the easy specification of proposers and arbiters (registrars are written directly in MACNET). The macro language provides for defining proposers and associating activation conditions with them. It provides for defining default, override and hierarchical abstraction relationships between proposers. A 'competition and cooperation' style of computation [4] is provided for by allowing proposers to support or object-to other proposers and proposals.

The book contains some useful comments on building and debugging this kind of circuitry. One interesting problem arises from the 'fixed-connection' assumption. Various proposers may want to use the same chunks of, or connections to, the peripheral system at the same time. This gives rise to a problem because rather than there being a subroutine that calculates, say, the distance between two points, there is a piece of circuitry that does this. If two proposers need to evaluate the distance between different sets of points, then this circuitry has to be shared out sequentially between them, and the results must then be routed to the right proposer. This complicated allocation approach was used in Pengi. In Sonja, the approach is to duplicate the circuitry as necessary. This makes sense for operations in which speed is essential. However, both allocation and duplication seem inadequate solutions in general and I suspect that a limited kind of instantiation framework is necessary.

3.2 Vision

Sonja's vision system is described in detail. It is inspired by, and meant to model, theories of human vision. Chapman argues strongly that vision needs to be task-specific in the sense that it must support the activities in which the system engages. Amazon requires relational data of low precision. Relational data is tough to make explicit in a 'standard' (geometric-model/common-database) vision system because of the sheer quantity of relations and the problems of updating them when the scene changes. One solution is have the vision system produce only the requisite data needed by actions; i.e., to wire actions up to get the very specific pieces of information they need from sensors. This approach, which Chapman calls 'insect vision' and attributes to Brooks [7], does not require distinguishing vision processing from task or action processing.

The insect approach is somewhat too task-specific: it requires that an entire new pathway between sensors and effectors be constructed for each new task or subtask. This is the one place where Chapman uses learning as a constraint in addition to the tenets of the concrete-situated approach and essential connectionism. He argues that learning implies the usefulness of a set of relatively high-level visual primitives to "act as a tinkertoy-like kit from which new visual machinery can be built." Ullman's *visual routines* model [20] provides this framework for Sonja.

The visual routines model says that visual processing is accomplished by applying the members of a small set of *visual operators* in various combinations and sequences, *visual routines,* to an image. Visual operators model what is called intermediate vision and visual routines model late vision. Early vision refers to the retinotopic phase of vision. Intermediate vision refers to the phase where retinotopic mappings are reduced to more compact encodings. Late vision refers to processes that use these compact encodings. In Sonja, the registrars of the central system implement late vision. Sonja does not engage in any pixel-level early processing, instead the visual system is connected to the game data structures. This sidesteps one of the key problems in early vision — noise. Chapman agrees that whether this structure will handle noisy input well is an open question. He notes that video game screens are designed to be easy to process visually anyway.

Early properties are made available to the central system via *visual attention.* The mechanisms of visual attention allow the differential application of visual processing to parts of the image. Chapman implements a 'spotlight' model of attention using an addressing pyramid. The pyramid is a tree structure in which each level is exponentially smaller than the one below it. Leaf nodes in the pyramid contain information on early properties of regions of the image. Interior nodes can either average the information at all their inferior nodes, or select information from one of their inferior nodes, and pass that upwards. Sonja has a 16X16 resolution pyramid. The pyramid turns out to be prohibitively slow on a serial machine, so Sonja can run with or without the pyramid (in which case the game data structures are accessed directly). Chapman points out that the concept of an addressing pyramid rescues pointers, at least in a very limited form, from the prohibition of the essential connectionism constraints.

Sonja employs two kinds of visual search described in the psychophysics literature: *parallel* and *serial-self-terminating* [19]. Parallel search involves computation of early properties at all points in an image. Not all properties can be computed in this fashion; some properties have to be computed on one candidate region of the image at a time. Chapman uses retinotopic maps to implement parallel search and uses the attention pyramid (routing all the early properties of an addressed candidate location to the root, candidate by candidate) to implement serial search. Chapman reports that Sonja's speed at serial search roughly matches human performance (about 17 locations per second). Although, it's not clear if this is with or without the full pyramid mechanism (and hence if it's a real or estimated performance measure).

Visual routines need to maintain state between calls to operators. For example, to find the nearest instance of an object, it's necessary to 'mark' the current position of the agent and then check the distance between this and each object instance. Chapman uses a set of globally accessible marker objects, *intermediate objects*, to store intermediate state. There are four types:

- markers (which flag point locations),
- lines (which flag the line segment between two points),
- rays (which flag a direction), and
- activation planes (which flag regions).

There are a fixed number of each kind of object and they are uniquely named.

Each visual operator used in Sonja was chosen to be generally useful, abide by essential connectionism and have some biological evidence to support it. Chapman admits this last constraint was not always met. The set itself was chosen to span the domain and facilitate writing/learning new visual routines. Without a formal definition of the domain, it's very hard to show that the set of operators spans it. This gets back to the 'standard' robotics concept of a separate and self-sufficient perceptual system that can be used for any task. I'm still not convinced this sort of question is even well-defined; but given that it could be made so, then I would expect it to be easier to determine if a set of visual operators spans a domain than to determine if a 'standard' object recognition system spans the domain. Sonja's operators include tracking, determination of distances, directions and angles, and various region operations. Again, because of the fixed hardware assumption, it's necessary to have multiple copies of those operators which could potentially be needed simultaneously.

3.3 Advice

One of the major ways that Sonja differs from Pengi is in its use of advice. Agre and Chapman have maintained that the concept of plans-as-programs — plans composed of detailed instructions which are directly executed by the agent — is flawed. They argue instead for plans as resources that can be consulted by an agent in deciding what to do "on an equal basis with other resources such as the configuration of your equipment, external memory devices like a string tied around your finger or a scratch pad, or your feelings"[1]. Sonja's use of instructions illustrates this approach.

Sonja's vocabulary of instructions consists of a small set of imperative sentences and their negatives. Chapman chooses to ignore many linguistic issues in favor of concentrating on the issues of grounding the instructions in concrete activity and on the use of instructions as decision-making resources. So, for example, Sonja does not look at the syntax of the instructions, nor will it engage in a running dialog with an advice-giver.

To understand what is involved in video game advice-giving, Chapman videotaped a number of human players and their advice-giving friends during play. Human players rely on the fact that the advice-giver shares the same sense of the game that they do — sees things in the same way and has understood objectives. In this kind of framework, communication becomes part of maintaining a shared view of reality. Chapman advocates a passive approach to this issue — Sonja and its advisor share a common world view not so much because they actively establish it, but because Sonja's mechanisms embody this common view and because much of Sonja's design is modeled on human psychophysical studies. So, for example, 'visual attention' is a phenomenon common to Sonja and its advisor; this means instructions such as "on your left" (a warning to watch out for something on the left) can be understood and are actually of some use to the system. At a more general level, this suggests that the reasons for using biological models in AI are not limited to inspiration or Cognitive Science modeling, but also include the need to ensure that an adequate level of communication is possible between humans and the AI system.

The mechanism for issuing instructions to Sonja is a bit stiff: at a point in the game when advice is to be offered, the advisor hits suspend, types in the advice, and then resumes the system. Sonja stores instructions using a fixed set of instruction buffers. Instruction strings (e.g., "on your left") activate one or more of the instruction buffers (and buffers can be activated by one or more instructions). A buffer stays active until it has been consulted and explicitly cleared by the central system. Instruction buffers can be chained to achieve sequential ordering of instructions. For example, the instruction string "get the potion and set it off" activates the chain of instruction buffers `register-the-potion,` then `pick-up-the-goody,` and then `use-a-potion`. This sequence would motivate Sonja to determine if there is a magic potion nearby that it would make sense to get, and then possibly to acquire and use that potion.

Sonja understands instructions by relating them to the current Amazon situation. Instructions (and instruction buffers) are *not* high-level programs. So `use-a-potion` is not a program/subroutine that results in motor signals being sent to Sonja's effectors. Rather the instruction simply adds in an extra 'vote' for the proposed action. This vote competes and cooperates with all the other proposals active in Sonja's arbitration network. The instruction is consulted to arrive at a decision, not executed to produce motor commands.

An instruction may support or object to courses of action already under consideration, or it may permit or even require new registration. For example, "use a knife" or "no, the other one" (setting buffers `use-a-knife` and `no-the-other-one` respectively) may cause Sonja to look around for a sensible candidate object to use and discover new possibilities for using it. The instruction buffer `no-the-other-one` effects what Chapman calls a *repair*, a correction of a misunderstanding. It causes Sonja to reregister "*all* the entities referred to by pending instructions."

4 Sonja at Play

About one sixth of the book is devoted to describing Sonja in action. I think Chapman missed a chance here to drive home the concepts of dynamics and routines — agent and environment sharing the responsibility for organizing complex behavior — that he describes so well in the early chapters. The description of Sonja in action is filled with detailed descriptions of mechanisms and how they give rise to specific kinds of behavior. It reads a bit too much like the sort of maintenance documentation that programmers are urged to provide for their latest creations. Having come from the 'high' of reading the early chapters, I expected this section to give a description of Sonja's behavior from an observer's viewpoint: what routines are engaged in, what part Sonja plays and what part the environment plays in each routine, how the resultant interaction exploits the dynamic to achieve work. Given that perspective, then it would be appropriate to describe in detail how Sonja's mechanisms cause it to play it's 'part' in the routine (and thus illustrate why they are appropriate mechanisms). This section of the book does contain a description of the routines Sonja engages in, but these seem to take a back seat to the details of the mechanism.

I separated out four interesting routines in which Sonja engages and I list them in the next few subsections in the 'observational' (as opposed to mechanistic) style that I had expected. The first two, wandering and bypassing obstacles, rely on relatively static contributions from the environment. The second two, both combat related routines, incorporate dynamic contributions from the actions of entities such as ghosts and demons. In almost all cases I have simplified out the special cases from the routines, and I have omitted a detailed discussion of Sonja's mechanisms.

4.1 Wandering

In this relatively simple routine Sonja interacts with the environment to effect a random wandering behavior. The relevant parts of the environment are obstacles (which will cause the amazon to stop moving if it moves against them) and the amazon's direction of motion (which can be sensed). The relevant parts of Sonja's mechanism are the two local rules: 1) if the amazon is not moving then it is kicked into motion in a random direction; 2) if the amazon is moving then it is advanced in its current direction of motion.

Sonja and the environment interact as follows: The amazon is stopped if the game has just started, if Sonja has been engaged in some activity, or if the amazon has just hit a wall; in all these cases rule 1 produces a new random heading. As long as the amazon is moving, Sonja uses that current direction of motion to continue motion (rule 2). The wandering routine may, of course, be interrupted or interleaved with other routines such as passing obstacles or fighting ghosts.

4.2 Passing Obstacles

Sonja engages in this routine to get by obstacles. The relevant parts of the environment are obstacles (which can be sensed visually). The relevant parts of Sonja's mechanism are the two rules: 1) if there is an obstacle between the amazon object and the goal location, then rotate the direction of motion until the obstacle is no longer in the way; 2) if passing an obstacle, every so often rotate the heading back towards the goal until the heading intersects either the goal or the obstacle.

Sonja and the environment interact as follows. When an obstacle is sensed, rule 1 results in Sonja choosing and following a new unobstructed direction of motion close to one side of the obstacle. Rule 2 results in Sonja continually choosing a direction directly to the goal (if it is reachable) or encountering the obstacle again. Encountering the obstacle again triggers rule 1. Note that in this routine, two parts of Sonja's mechanism communicate via the obstacle.

4.3 Fighting Ghosts

The two previous routines relied on some rather static contributions by the environment. Fighting routines must incorporate dynamic contributions from the actions of entities such as ghosts and demons. The 'fighting ghosts' routine is used by Sonja to kill off ghosts which otherwise might attack it. The relevant parts of the environment are any ghost instances. Ghosts travel at a constant speed, slower than the amazon object, and always move towards the amazon. They will sap the amazon's strength if they ram up against it, but can't fire at it from a distance. The relevant parts of Sonja operate as follows. Sonja selects the most dangerous ghost: the nearest ghost which has a net component of motion towards the amazon object. If such a ghost exists, then Sonja fires at it until it vanishes or ceases to be the most dangerous ghost.

Sonja and the environment interact as follows. Ghosts are always drawn towards the amazon, but since the amazon can travel faster, only those ghosts it is moving towards have a chance of catching it. Sonja always selects the nearest ghost moving towards it and targets that ghost. This routine (on its own) thus results in Sonja killing all ghosts which could kill it, *unless* the ghost appears extremely close to the amazon icon[†] or a ghost 'wall' of sufficient size and distance advances on the amazon object (a detailed analysis of the interaction should produce numeric values for size and distance). Interleaving this routine with other routines may result in

[†] I don't know if this is allowed in Amazon.

weakening its ghost busting abilities.

4.4 Fighting Demons

Sonja also has a routine for fighting demons. The relevant parts of the environment are instances of demons and the fireballs they generate. Demons, unlike ghosts, are monsters that can fire (fireballs) from a distance; thus demons are always dangerous. Apart from this, they behave the same as ghosts, being drawn towards the amazon at a constant speed. Fireballs are destroyed when hit. The relevant parts of Sonja operate as follows. If a demon is the nearest monster, then the mechanisms described under ghost fighting will be triggered for the demon. Detecting an instance of a fireball causes Sonja to fire back along the trajectory of the fireball.

Sonja and the environment interact as follows. If a demon registers as the nearest monster then a routine similar to the ghost fighting routine will result. If a fireball is detected, then Sonja fires back along the fireball. This will result in the destruction of the fireball and, since demons always move towards the amazon, also in the eventual destruction of the demon. Notice that shots fired at a single demon might be the result of interleavings of the fireball-based and nearest-monster based routines. And of course, with multiple ghosts and demons, Sonja's behavior will be a complex interleaving of instances of these routines. The very fact that these routines are *not* perfectly guaranteed to kill off all threats means that advice is a useful commodity to Sonja.

5 Collaboration and Learning

The book has two appendices. The first describes proposed extensions to Sonja to allow it to play in a collaborative fashion with another player. The second describes an experiment in skill acquisition and some conclusions and discussions.

The key extension needed to enable Sonja to play collaboratively is for it to be able to register the intentions of its collaborator. This activity is often called *plan recognition*. The general case of plan recognition is hard. However, Chapman claims plan recognition is relatively easy in a concrete domain such as Amazon. The concrete situation not only constrains what the agent can/should do, it also constrains what any other agent can/should do in the same situation. Thus, determining what a collaborator's intentions are boils down to determining what your intentions would be in the same situation.

The skill-acquisition experiment consisted of building a learning system called WOMBAT. This work predated Sonja, and its application domain was a video game called Robots. Chapman's objective was to combine top-down and bottom-up learning methods in a synergistic fashion. WOMBAT used a temporal-difference method combined with backpropagation as the bottom-up component, and brute-force, breadth-first search with dependency analysis generalization as the top-down component. Chapman reports that system wasn't very successful, for a host of reasons. Backpropagation temporal-difference methods were too slow and unreliable and required domain-specific tuning of various parameters. Dependency analysis offered little advantage in generalization for the Robots domain. The remainder of the section, taking off from the failure of WOMBAT, is a very interesting discussion of filling out the concrete-situated approach to address skill acquisition.

6 Conclusions

On the whole, I liked this book. Chapman has demonstrated that the concrete-situated approach can be used to build an agent capable of intelligent action in a complex and time constrained domain, a domain which has real-world flavor and in which a classical planning approach would be a liability. The book gives us two very interesting concrete illustrations of the approach: the model of advice and the model of vision; and Appendix B contains a provoking discussion of the application of the concrete-situated approach to learning.

However, I would argue with the use of the phrase "biologically plausable" to describe hardware that obeys connectionist constraints. The neural network literature distinguishes work on artificial neural neworks (connectionism) from work on biological neural networks (computational neuroscience). Connectionists have abstracted computational characteristics from neural models and used them to construct artificial systems. Chapman clearly falls in this class. Computational Neuroscientists attempt to build computational models of biological systems and such models have to match the behavior of the actual neural systems, e.g., Arbib's *Rana Computatrix,* a model of the neural circuitry underlying visuomotor coordination in frog and toad [5]. Implementations guided by the essential connectionist constraints can be called, at most, biologically inspired. Chapman does claim that components of Sonja's visual system are sufficiently precise "that they could be taken as theories about the human brain and subjected to psychophysical experiment." However, he expresses no interest in following up on this avenue of investigation.

Chapman comments that navigation took an inordinate amount of effort: "Half the central system code and well over half the time I spent implementing Sonja went into navigation." Motion, like vision, is one of those things that animals do without much conscious effort; and hence it appears simple. There is much evidence that motion control has highly specific neural control circuitry (e.g, basal ganglia, cerebellum, superior colliculus) [16, 9, 4]. The experience in robotics has been that motion planning is well addressed with motion-domain specific techniques and algorithms, e.g., [6]. Sonja attempts to solve general motion problems completely within its central system. It is more likely that only 'late' motion (by analogy with late vision) should be addressed in Sonja's central system.[‡] Other highly specialized and parallel systems should handle 'intermediate' (e.g., obstacle avoidance) and 'early' (e.g., actuator control) motion.

Looking back at his effort in building Sonja, Chapman writes that he initially thought the arbitration network would be the most complex component to build. Instead he found that the visual systems and the registrars were the problem. This (and other evidence) leads him to claim that for concrete domains action selection is in general easy and the harder problem is perception. Hence, future research should concentrate in that area. I think it's important to be careful here. Studying perception on its own is as bad as studying action selection on its own. Perception and action evolved together in animals [21] and the concrete-situated approach argues well from a problem-directed viewpoint that they need to be tightly coupled. Sonja's visual system is constructed so that its only connection with action selection is through the late vision registrars of the central system. It effectively studies just *one* perception-action system (and one which apparently won't handle navigation well). In a system with many perception-action modules, what we need to study is not just perception or action, since these components may differ remarkably between modules, but rather ways to *integrate* perception and action.

I have already mentioned that I would have preferred to have seen something more in the way of performance studies and comparisons with other technologies. I think Chapman ignores the illustrative powers of such studies as a matter of principle. He wants to be clear that, for example, the sort of tests Sacerdoti runs on NOAH [15] would not stand up as an engineering evaluation of the technology, so he doesn't provide any such tests for Sonja. To a certain extent I agree, but I'd still argue that these tests have illustrative value and would be of use in understanding and duplicating parts of

[‡]Sonja's central system seems to have a functionality corresponding to what Ewert [9] calls *command neurons* — cells which are tightly involved in the decision-making to carry out a specific behavior.

Sonja.

In summary, this book presents us with a very concrete example of the approach that Chapman and Agre have been advocating for some time. Chapman does a first-rate job of presenting that approach, especially in the early chapters. My only real quibble is that he misses a dynamite chance to drive home the concrete-situated notions of dynamics and routines in the section describing Sonja in operation.

References

[1] P. Agre and D. Chapman. From reaction to participation. In *DARPA Planning workshop*, pages 123–154, Santa Cruz, CA, Oct. 1987.

[2] P. Agre and D. Chapman. Pengi: an implementation of a theory of action. In *Proc. AAAI-87*, pages 268–272, Seattle, WA, Oct. 1987.

[3] J. Albus. RCS: a reference model architecture for intelligent control. *IEEE Computer*, 25(5):56–59, May 1992.

[4] M.A. Arbib. Perceptual structures and distributed motor control. In V.B. Brooks, editor, *Handbook of Physiology – The Nervous System II. Motor Control*, pages 1449–1480. American Physiological Society, Bethesda, MD, 1981.

[5] M.A. Arbib. Levels of modelling of neural interactions underlying visuomotor coordination. Technical Report 85-36, University of Massachusetts, Computer and Information Science Department, Amherst MA 01003, 1985.

[6] M. Brady *et al.* *Robot Motion.* MIT Press: Cambridge, MA, and London, England, 1984.

[7] R. Brooks. A robust layered control system for a mobile robot. *IEEE J. Rob. & Aut.*, RA-2(1):14–22, Mar. 1986.

[8] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–377, 1987.

[9] J-P. Ewert. *Neuroethology.* Springer-Verlag Berlin Heidelberg New York, 1980.

[10] P. Hayes. The naive physics manifesto. In D. Mitche, editor, *Expert Systems in the Micro-Electronic Age*. Edinburg Univ. Press, 1978.

[11] E. Kent, M. Shneier, and T. Hong. Building representations from fusions of multiple views. In *IEEE Int. Conf. Robotics & Automation*, pages 1634–1639, 1986.

[12] J. McCarthy. The advice taker. In M. Minsky, editor, *Semantic Information Processing*. MIT Press, 1968.

[13] M. Minsky. A framework for representing knowledge. In P.H. Winston, editor, *The Psychology of Computer Vision*. McGraw-Hill, 1975.

[14] M. Minsky. *The Society of Mind*. Simon and Schuster, 1986.

[15] E.D. Sacerdoti. *A Structure for Plans and Behavior*. Elsevier, new York, 1977.

[16] R.F. Schmidt (Ed). *Fundamentals of Neurophysiology*. Springer-Verlag New York Heidelberg Berlin, 2nd edition edition, 1978.

[17] O.G. Selfridge. Pandemonium: A paradigm for learning. In *Mechanisation of Thought Processes*, pages 511–531. Her Majesty's Stationary Office, 1959.

[18] D. Subramanian and J. Woodfill. Subjective ontologies. In J. Hendler, editor, *AAAI Spring Workshop on Planning in Uncertain, Unpredictable or Changing Environments*, Stanford CA, Mar. 27–29 1990. Systems Research Center, Univ. of Maryland.

[19] A. Treishman and S. Gormican. Feature analysis in early vision: Evidence from seach asymmetries. *Psych. Rev.*, 95(1):15–48, 1988.

[20] S. Ullman. Visual routines. *Cognition*, (18):97–159, 1984.

[21] C. von Hofsten. Catching. In *Perspectives on Perception and Action*, pages 33–46. Lawrence Erlbaum Associates, 1987.