Made-up Minds:
A Constructivist Approach to Artificial Intelligence

by

Gary L. Drescher

B.S. Massachusetts Institute of Technology (1977)
S.M. Massachusetts Institute of Technology (1985)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 1989

Signature of Author .....
Department of Electrical Engineering and Computer Science
September 1, 1989

Certified by .....
Seymour A. Papert
Professor of Learning
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Made-up Minds:

A Constructivist Approach to Artificial Intelligence

by

Gary L. Drescher

Submitted to the Department of Electrical Engineering and Computer Science
on September 1, 1989, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

The schema mechanism is a general learning and concept-building mechanism intended to simulate aspects of Piagetian cognitive development during infancy. A computer program that implements the schema mechanism, MARCSYST I, has replicated several early milestones in the Piagetian infant's invention of the concept of permanent object. In Piaget's *constructivist* theory, an infant first represents the world only in terms of simple sensory and motor elements; initially, there is no concept of persistent, external objects—objects that exist even when not perceived. The infant must construct this concept, working backward from the perceptions that manifest external objects. This conceptual leap is first of a long series of such constructions, extending through adult-level intelligence.

The schema mechanism connects to a simulated body in a microworld. The mechanism learns from its experiences by processes of *induction*, *abstraction*, and *invention*. A novel induction technique builds schemas, each of which asserts that a given action, in certain contexts, has particular results; contexts and results are expressed in terms of binary state elements called *items*. Crucially, the schema mechanism not only thus discovers relations among existing representational elements (actions and items), but also constructs new such elements. For any achievable result, the mechanism can define a new, abstract action, the action of achieving that result. Most important, the mechanism can synthesize new state elements to designate aspects of the world that the existing repertoire of representations fails to express, thus inventing new concepts.

The schema mechanism builds schemas, expressing the context-dependent results of actions, by an induction technique called *marginal attribution*. Discovering the results of actions is complicated by the fact that a particular action typically has different effects on different occasions. Until the corresponding context conditions have been identified, a result is therefore difficult to identify as such, and vice versa.

The marginal attribution facility solves this chicken-and-egg problem by identifying a *relevant* state transition—one which, even if it follows a given action only rarely, is even more rare in the absence of the action. Then, the mechanism searches for conditions under which the relevant result follows more reliably.

The schema mechanism can define a new state element to represent the validity conditions of an unreliable schema. For example, suppose a given schema asserts that moving the hand to a certain body-relative position results in a tactile sensation. The schema mechanism defines a new state element to represent whatever unknown condition must hold for the schema to be valid; in this case, the condition is that there be a palpable object at that position. The concept of palpable objects is not built in; rather, this synthetic item itself implements the mechanism's first approximation to that concept, thus *reifying* the schema's validity conditions, treating the schema's validity as a thing-in-itself. Having defined a new state element, the mechanism begins an open-ended process of finding the element's verification conditions, which tell about its state.

Thesis Supervisor: Seymour A. Papert
Title: Professor of Learning

Acknowledgements

I have had the pleasure and privelege to discuss this research with many friends, colleagues, and teachers: Hal Abelson, Phil Agre, John Batali, David Chapman, Jim Davis, Ken Haase, Ed Hardebeck, Danny Hillis, Dan Huttenlocher, Henry Minsky, Margaret Minsky, Marvin Minsky, Seymour Papert, Ron Rivest, Cynthia Solomon, Gerry Sussman, Tom Trobaugh, Lucia Vaina, and Patrick Winston. Their insights and criticisms have been valuable. In addition, Seymour Papert (my advisor) and Hal Abelson, Ron Rivest, and Gerry Sussman (my readers) have provided support and encouragement as well as technical guidance. I especially thank David Chapman for his thorough, deep and detailed comments on a draft of this thesis. I alone, of course, deserve credit for all remaining errors.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and overview

This dissertation presents the *schema mechanism*, a general learning and concept-building mechanism inspired by Piaget's theory of human cognitive development. The schema mechanism is intended to simulate aspects of cognitive development during infancy, with possible relevance to later development as well. A computer program that implements the schema mechanism, MARCSYST I, has replicated several early milestones in the Piagetian infant's acquisition of the concept of permanent object.

I begin with the background of this research program—the issues of cognitive science and artificial intelligence that motivate and inform this work. I then present a detailed description of the schema mechanism itself and of the results from experiments with its implementation. I discuss the schema mechanism in relation to cognitive science, other research programs in artificial intelligence, and philosophical issues of meaning and epistemology.

1.1 The nature of learning

The scope of human skill and knowledge is startling. For some domains of expertise, such as visual processing, it is clear that the human species is genetically endowed with hardware that embodies knowledge about the domain. For other domains, such as

language, the question of built-in knowledge is controversial. But for many domains—physics, architecture, economics, chess, juggling, cinematography, computer programming, composing symphonies—there can be no corresponding innate mental modules; the subject matter of these domains did not even exist (or become accessible) until so recently that evolution could not have had a chance to provide customized, built-in support. In these domains and myriad others, dramatic advances in knowledge occur within individual lifetimes—sometimes in mere years, days, even seconds. Uniquely, successive generations of human beings inherit a progressive accumulation of competence arising not by the rearrangement of genes but by the creativity of minds, and propagating not through biological reproduction but through the tutelage of culture.

We must, of course, be endowed with innate machinery that is responsible for our ability to construct and acquire such diverse knowledge. We may term such machinery a general learning mechanism—general in the sense that it spans a diverse, open-ended set of domains that were not specifically "anticipated" by evolution. It is natural to wonder what portion of human intelligence grows out of a general learning mechanism, and what part is due to innately specialized, domain-specific processing modules.

1.1.1 Learning in human beings

Jean Piaget and Noam Chomsky stand at two extremes of the nativism vs. learning spectrum. Piaget's work is at the foundation of the modern empirical study of the genesis of intelligence in individual humans. Piaget proposes a radically *constructivist* account in which even the basic notion of an object—the notion that visual and tactile sensations are related to each other, as manifestations of some external thing; that this thing exists even when not perceived, etc—is not innate, but is abstracted from the infant's interactions with its world. Similarly, notions of logic, classification, and number, conceptions of people and of self, and of the rest of the world, are all gradually constructed. Moreover, intelligence itself—seen as a gamut of strategies for pursuing

goals or solving problems or exploring terrains, literal or figurative—is constructed, bit by bit, with ever-increasing sophistication.

At the other extreme, Noam Chomsky champions a radically nativist theory of cognitive development. Chomsky doubts the very intelligibility of the notion of learning (Chomsky 1988, Piattelli-Palmarini 1980), particularly with regard to *cognitive universals*, that is, concepts normally acquired by all persons; virtually everyone, for example, comes to understand the rudimentary properties of physical objects, and gains facility with fundamental principles of language. Such knowledge may not be functional at birth—it may not be available for use by the infant, and it may not yet even exist in a format that permits it to be used. But what is innately present, by any account, is a mechanism that, interacting with any normal environment, eventually develops a functional version of such knowledge. Chomsky then poses the rhetorical question: how is that nearly inevitable development different from, say, the nearly inevitable development of limbs by a zygote? Limbs are not present at conception, nor is any miniature model of limbs, nor is there even (necessarily) any local region of the genome specifically dedicated to the control of limb development. Nonetheless, the growth of limbs is an innately specified maturational process; it would be absurd to say that each individual *learns* to have limbs. Why should the learnedness of cognitive universals be judged by different standards than morphological universals?

Escaping the snare of this clever, provocative question prepares us for more substantive investigations of the human mind. Here is an easy, sensible way out. Consider, for example, knowing the names and layout of the streets in one's neighborhood. This knowledge is untendentiously learned, in that *information is gained* when this knowledge is acquired. No examination of a zygote could yield a street map of its neighborhood; the information is not present. But examining the brain of a person with that knowledge could—in principle—reveal that information.

In contrast, the process of growing limbs yields no new information that limbs will exist. Looking at a zygote, one could—in principle—deduce that the mature

organism will develop limbs, if nurtured in a normal environment. Thus, that information is already present in the zygote; the mature organism bears no additional such information.

Let us say that a mechanism is a learning mechanism if its function is to gain information. Chomsky's point, recast in these terms, is that the acquisition of cognitive universals entails no information gain; as much can be determined (in principle) about the subject matter of such knowledge by examining a zygote as by examining an adult. The same can be said for analytic—necessarily true—knowledge, such as $2 + 2 = 4$. In the case of nonanalytic universals, either examination is fruitful; in the case of analytic knowledge, both examinations are superfluous.

Nonetheless, in opposition to Chomsky's conclusion, such knowledge might develop as the product of a learning mechanism. That is, a mechanism whose function is to gain information might *also* develop usable forms of universal or analytic knowledge, acquired by the same principles of operation that produce information gain. A study of the knowledge acquired by a learning mechanism would have no cause to exclude those of its acquisitions that turn out to be inevitable, hence universal to mechanisms of that class; all of a learning mechanism's acquisitions are sensibly called learned.

Thus, universal and analytic knowledge can be learned, if produced by a learning mechanism—a mechanism for gaining information. But even if cognitive universals are in fact learned, it remains sensible to say that they are innately specified and develop maturationally. These claims are reconcilable if the innately specified developmental process is a learning process, in the sense just given.

The substantive question, then, is which such knowledge, if any, is in fact learned by human beings, and which, if any, is either present at birth, or arises by a nonlearning maturational process.

1.1.2 Learning in artificial systems

A parallel question arises when designing an artificial intelligence (A.I.). To what extent is it reasonable to seek powerful general mechanisms of learning; to what extent should research focus on more domain-specific mechanisms? The question about humans is distinct from the question about A.I. Even if general learning mechanisms are feasible, human beings might not be designed that way; conversely, even if much of human intelligence does flow from a general learning mechanism, engineering a replica of that mechanism may be an intractible problem.

Indeed, early A.I. work, pursuing *self-organizing systems*, tried and failed to find just such a mechanism. As this approach became discredited, there followed a generation of *knowledge-based* A.I., characterized by the principle that intelligence, especially learning, derives its power from knowledge: about domains, about reasoning, about space and time and so on. There must be a wealth of structure to support the acquisition of new structure. From this point of view, bootstrapping from meager initial knowledge seems unlikely.

On the other hand, the failure to develop *tabula rasa* systems may have been due to problems not intrinsic to the very attempt. In particular, the study of self-organizing systems (like some philosophical and psychological inquiries about the innateness vs. acquisition of human knowledge) has been handicapped by lack of attention to empirical evidence. The relevance of empirical data to the question about humans is clear: by observing the intellectual development of humans from infancy, one can hope to obtain evidence as to the early presence or absence of certain abilities or knowledge. With regard to the A.I. approach, it is good to be reminded of a maxim of Seymour Papert: in order to think seriously about thinking, one must think about thinking about *something*. But what is a self-organizing system to "think" about? Not the things that human adults think about: adult tasks are predicated upon much acquired knowledge, not available to a *tabula rasa* machine. On the other hand, whatever it is that human *infants* think about is a plausible candidate for the subject matter

of a learning-based A.I. mechanism. The infant's learning achievements offer target abilities for the mechanism, providing a basis for the mechanism's design. Without data about infants (and without a plausible constructivist theory to characterize that data) there is no good source of inspiration as to what, specifically, a constructivist mechanism ought to *do*.

The methodological flaw—not having a clearly specified target domain—is compounded by a second problem in research about self-organizing systems: it is traditional to set up a chaotic gaggle of interacting elements and then wait for order to emerge from the chaos. But extracting order from chaos is hard work. A mechanism to do this work must be designed not just to amass atomic facts, but to organize data into functional units, to abstract essential attributes and discard useless ones, to verify suspected regularities and pursue variations on them, to develop new kinds of representation as old ones prove inadequate—the sort of activity that must be involved in any serious effort to make sense of the world.[1]

1.1.3 Humanlike learning in artificial systems

Piaget's work offers an antidote to both the lack of a clear target domain and the naive-order-from-chaos problem. Piaget gives an elaborate description of the course of cognitive development from infancy through childhood and adolescence, taking account of the evolution of primitive problem-solving and domain-specific knowledge. Piaget characterizes ways in which, at a given point, a person uses existing knowledge and skills to achieve specific goals, and to create new knowledge and skills. Piaget's intricate roadmap of the course of development—especially during infancy—specifies a target domain for artificial systems, a sequence of cognitive acquisitions for a mechanism to achieve.

[1]The resurrection of self-organizing systems in the guise of connectionism avoids the primary pitfalls of earlier such research. Present-day connectionism tends to focus more modestly on providing alternative computational gools for solving particular problems. Section 6.1.2 compares connectionist work with the approach advocated here.

Piaget documents some striking uniformities throughout cognitive development, and refers to these as the *functional invariants* of intelligence. These invariants amount to a loose characterization of an underlying developmental mechanism. Departing from the older empiricist tradition, Piaget's characterization of developmental invariants emphasizes the importance of well-designed activities of organizing, structuring, and abstracting from experience, and of the purposive application of knowledge and exploration in the pursuit of goals, in contrast with merely accumulating data from and being conditioned by the environment. Piaget's loose description of functional invariants falls far short of a formal specification of a developmental mechanism; still, it furnishes an important alternative to naive-order-from-chaos as a starting point for a precise specification.

My research program, then, is to design and implement a mechanism that corresponds to Piaget's sketch of the functional invariants of cognitive development. This endeavor has two broad goals: to help understand the human mind, and to help design an artificial mind. As mentioned above, questions about the nature of intelligence might have different answers for cognitive science than for A.I. Nonetheless, the program advocated here (not as the sole promising approach, but as one of them) is to try to build an intelligent mechanism by taking human intelligence as the inspiration—that is, by trying to reverse engineer the mechanism of the human mind.

I presume, as a working hypothesis, the approximate correctness of Piaget's theory (or rather, of a significant revision of the theory in light of modern evidence, as discussed in section 2.8.3). That is, I assume that a general learning mechanism resembling Piaget's is indeed present in the human mind, and is of central importance to the development of intelligence from infancy to adulthood. I present results to show that a mechanism designed along the lines of Piaget's theory can account for some early aspects of the Piagetian developmental progression. The motivation for stressing early (rather than later) development is threefold: early development is simplest; the most detailed observations of human development concern early development; and the

developmental mechanism is most clearly discerned in its earliest operation, before its own constructs obscure it by complicating its observable activity.

I present here what I call the *schema mechanism*, a proposed approximation to the mechanism of Piagetian development. Two fundamental influences contribute to the design of the schema mechanism: the Piagetian characterization, and engineering constraints. The mechanism is intended both to help explain the themes of Piagetian development, and to be well-motivated from a engineering standpoint, given the computational demands of the learning tasks involved. I avoid machinery that accords with only one of these two principles: machinery that is rigged to replicate this or that developmental event, but without any good reason for a learning system to incorporate such machinery; or apparatus that builds in sophisticated abilities which, however, are not initially present in Piagetian development. These exclusions stem from the goal of reverse engineering the Piagetian mechanism. Mere replication of developmental events is only of interest here to the extent that those events reflect the operation of a reasonably designed learning mechanism. And reasonably designed mechanisms that do not correspond to the human apparatus are certainly worthy of investigation, but belong to a different program of research.

Piaget's theory is symbiotic with the schema mechanism:

- As just noted, Piaget outlines the main themes of cognitive development, and details much of the content of its early learning. This gives a first approximation to the mechanism, and a set of target achievements.

- The schema mechanism adds precision to Piaget's characterization of constructivism. A more concrete statement of Piagetian theory makes possible more specific tests and evaluations of the theory.

- Implementing a mechanism for Piagetian development is itself a partial test of his theory. Successful replication of Piagetian milestones by a plausibly engineered learning mechanism is circumstantial evidence that such a mechanism

is involved in human development. Unsuccessful attempts at such replication may point to places where the theory is wrong, or needs to be supplemented.

In sum, the project of designing and implementing the schema mechanism explores Piaget's theory by the methodology of artificial intelligence: testing a theory of the mind by building a mechanism that works according to that theory, and seeing if the mechanism does what it was intended to do.

1.2 The schema mechanism: an overview

The schema mechanism is a general learning mechanism. It engages in the discovery of regularities in the world, and the construction of new concepts, which then form the vocabularity for expressing further regularities. The schema mechanism uses the knowledge it acquires to guide its actions, both for the sake of specific goals and in order to gain further knowledge.

The schema mechanism connects to what I call an *animaton*, a simulated robot body that appears on a computer screen, in a two-dimensional simulated world (a *microworld*); see figure 1-1. The animaton includes a crude visual system, and a single, mobile hand with tactile sensors and the ability to grasp and move objects. Like a neonate, the animaton at first lacks the ability to move itself from place to place.

The schema mechanism has been implemented by a computer program, MARCSYST I (an acronym for Marginal Attribution and Representation Construction System). I intend *schema mechanism* to be a generic term (like *internal combustion engine*); it designates any learning mechanism that operates more or less as described here, no matter whether the mechanism is instantiated biologically, electronically, or is just an unimplemented abstraction. MARCSYST I is the particular implementation of the schema mechanism that provides the results presented here. Except where otherwise stated, all aspects of the schema mechanism discussed here have

been implemented in MARCSYST I; and all described learning achievements of the mechanism, unless otherwise noted, have been demonstrated by the implementation.

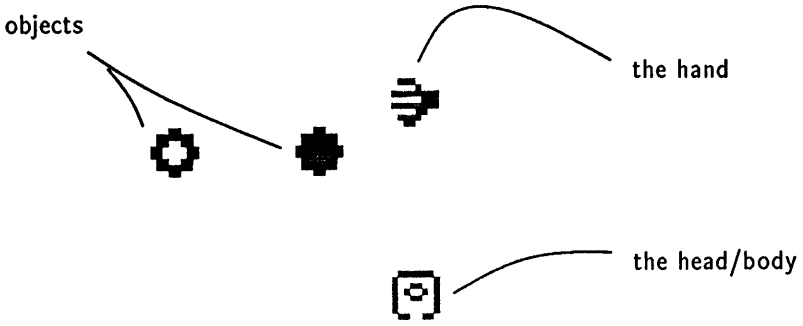


Figure 1-1: The schema mechanism controls an animaton (simulated robot) in a two-dimensional microworld.

The schema mechanism focuses on Piaget's description of cognitive *schemas.* As modeled here, a schema (figure 1-2) is an assertion that a certain *action* (here, action a) has a specified *result*, provided that certain *context* conditions are met; the assertion is subject to a reliability factor that the schema maintains on the basis of actual trials. A schema makes no assertion about what happens if the action is taken when the context conditions are not all satisfied. To *activate* a schema is to initiate its action when the schema's context conditions are satisfied; the schema's activation finishes when its action terminates.

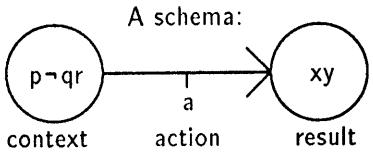


Figure 1-2: The schema: a basic unit of representation.

A schema is a unit of knowledge, both declarative and procedural. Declaratively, a schema makes a factual assertion, an assertion about what would happen under certain circumstances. Procedurally, a schema can pursue a goal; the goal may be in the schema's own result,

or some other result that that schema helps pursue. A schema is also a unit of experimentation, comparing what happens when an action is taken to what happens

without it, and comparing what happens with vs. without various context conditions. As explained below, new schemas arise from such experiments.

The schema mechanism supports three kinds of goals: primitive goals, which are built in; instrumental goals, which are momentarily of value as prerequisite steps toward achieving some other goal (eg, the intermediate states in a chain of schemas); and delegated goals, which receive lasting value from their general tendency to facilitate other goals.

Schemas' contexts and results are represented in terms of binary (On-Off) state elements called *items*. Each context designates zero or more items; some may be negated. In figure 1-2, the context consists of nonnegated items p, r, and negated item q; the result comprises nonnegated items x and y. A context is satisfied when and only when all of its non-negated items are On, and all of its negated items are Off. A result similarly contains zero or more (possibly negated) items, which are expected, subject to the schema's reliability factor, to turn On (or Off, if negated) when the schema completes its activation.

Each item corresponds to some partial state of the world. If the item is On, it asserts that that state currently obtains; if Off, it asserts that that state does not obtain. An item can also be in an Unknown state.

The schema mechanism's primitively supplied items all correspond to perceptual information, such as *there's something touching the hand* or *there's some object at the upper left of the visual field.* Each primitively supplied action corresponds to some simple motor activity, like moving the hand incrementally forward or glancing incrementally to the left. Calling the initial actions and items *primitive* is just to say that they comprise the initial representational vocabulary, in contrast with later elements, which the mechanism itself constructs. What the primitive items designate, and how they are computed, need not be simple; the visual items, for example, may correspond to information that (in humans) is the result of a complicated analysis of a visual scene to extract information about three-dimensional structure.

However sophisticated the processing may be that supplies primitive information to the schema mechanism, the schema mechanism itself is, at first, wholly ignorant of what the primitive actions and items correspond to, or how they might relate to one another. It does not know, for example, which items are visual and which tactile, or even what it would mean to be visual or tactile. It does not know that two items designating similar kinds of information—for example, two tactile items corrseponding to contact with adjacent regions of the hand—have any closer relationship to one another than to arbitrary other items. And the mechanism does not even *have*—let alone understand—any primitive items that designate persistent objects that exist independently of how (or even whether) they currently appear. It is the schema mechanism's task to learn about the relations among its units of representation, both primitive and constructed.

A constructivist mechanism is like a programming language in that its character is defined not so much by its primitives as by its ways of combining structures to form larger ones, and by its means of abstraction—its means of forming new units of representation that allow the details of their implementation to be ignored.[2] The schema mechanism, like a good programming language, is *extensible*: instances of its basic units of representation—schemas, items, and actions—can all be constructed by the mechanism's means of combination and abstraction. More than this, the schema mechanism is *self*-extensible—it is the mechanism itself (rather than a programmer or other external agent) that manufactures these extensions.

The schema mechanism interacts with the world, and based on its experiences, learns about the world by processes of *induction*, *abstraction*, and *invention*.

- Induction. The schema mechanism builds schemas that connect items and actions to express discoveries about regularities in the microworld—particularly causal relationships.

[2]This analysis of programming languages is borrowed from (Abelson and Sussman 1985).

- Abstraction. For any achievable result, the mechanism can define a *composite action*, the action of achieving that result, regardless of the details of just how the result is achieved.

- Conceptual invention. The mechanism synthesizes new state elements, synthetic items, to designate aspects of the world that were not previously represented.

Discovering the results of actions is complicated by the fact that a particular action typically has different effects on different occasions. Even a result that follows very reliably under the appropriate circumstances might follow only rarely in general; moreover, the result, even when it does occur, may accompany a large number of unrelated events. For example, the action of moving my hand backward reliably results in touching my chin, but only if the hand had been just in front of my chin. Had the hand begun in another position, some other result would have obtained instead. And when chin-touching does occur, any other of other events might also happen by coincidence.

Until the corresponding context conditions have been identified, a result is therefore difficult to designate as such; but until a result is so designated, it is difficult to look for conditions under which it occurs reliably. The schema mechanism incorporates a novel induction technique, *marginal attribution*, to solve this chicken-and-egg problem. Marginal attribution tries to identify a *relevant* state transition—one which, even if it follows a given action only rarely, is even more rare in the absence of the action. Then, the mechanism searches for conditions under which the relevant result follows more reliably.

But it isn't enough to discover connections among existing representations. A constructivist system's greatest challenge is to transcend its initially supplied terms of representation, to extend its own ontological vocabulary, to designate kinds of things that are radically different from any that it had previously been able to represent.

The schema mechanism can define a new state element, a synthetic item, to represent the validity conditions of an unreliable schema. For example, suppose a given

schema asserts that moving the hand to a certain body-relative position results in a tactile sensation. The schema mechanism defines a synthetic item to represent whatever unknown condition must hold for the schema to be valid; in this case, the condition is that there be a palpable object at that position. The concept of palpable objects is not built in; rather, this synthetic item itself implements the mechanism's first approximation to that concept, thus *reifying* the schema's validity conditions, that is, treating the schema's validity as a thing-in-itself.

The construction of a synthetic item works backward from some previously conceived manifestation of a thing (for example, the tactile manifestation of an object) to postulate the previously unconceived-of thing that is manifested. The new kind of thing may be radically different from any concept that preceded it—as, for example, the sort of thing that an object is is *nothing like* the sort of thing that a sensory impression is, despite the fact that direct perception is among the kinds of evidence by which we know an object.

Having defined a new synthetic item, the mechanism begins a process of finding the item's verification conditions, which tell about its state. For example, the presence or absence of a palpable object at some location may be verified by feeling there, by looking there, and by many other means. Verification conditions set the state of a synthetic item, turning it On or Off according to whether the associated schema is believed valid at the moment. Crucially, a synthetic item's verification conditions do not *define* the synthetic item. The item is defined to represent the validity conditions of the associated schema, whatever those turn out to be; the currently specified verification conditions are just the mechanism's best approximation so far to the schema's validity conditions, and are always subject to extension and revision.

The continual variability of a synthetic item's verification conditions compounds the radical novelty of the concept-invention accomplished by building synthetic items. Not only does such an item designate a kind of thing that is very different from its previously-conceived-of manifestations—it is also inexpressible, in practice, as any

fixed function of those manifestations.[3]

Synthetic items correspond to Piagetian *conservation* phenomena, wherein an individual postulates some new kind of thing that remains invariant even when all manifestations of it change or cease. This is arguably the most powerful and dramatic capability of human intellect.

1.3 Guide to the dissertation

This dissertation includes a synopsis of the schema mechanism's replication of aspects of the Piagetian infant's reconstruction of the world in progressively more objective, less egocentric terms. Marginal attribution promotes learning on a given level of reconstruction, while synthetic items and composite actions facilitate ascension to the next level.

The synopsis documents how the schema mechanism makes its way through aspects of the stages that Piaget outlines within the first, so-called *sensorimotor* period of intelligence; development culminating in Piaget's fifth stage is discussed, with emphasis on the construction of the concept of a physical object. At first, the infant represents the world only in terms of sensory impressions and personal actions; visual and tactile sensations, for example, are not even known to be related to one another. Sensorimotor development includes learning about the visual and tactile effects of eye and hand motions—eg, learning how to look directly at an object, or to move a hand into view; and the organization of that knowledge to designate the tactile properties of "visual objects", and vice versa—e. g. knowing how to touch an object that is seen. This paves the way to a sensory-mode-invariant representation of objects and space. A sense of the *permanence* of objects also arises—the idea that an object not currently perceived still exists, and its sensory manifestation can be recovered. The synopsis reinterprets these developments—spanning from the first to the fifth

[3]In principle, such a function could express the meaning of the item, by addressing all possible verification conditions. See section 3.4.3 for elaboration.

sensorimotor stage—in terms of the schema mechanism.

Five chapters follow the present one:

- Chapter Two gives a synopsis of the initial, *sensorimotor* period of Piagetian development (and touches briefly on subsequent periods); this developmental sequence is the target scenario for the schema mechanism. This chapter assesses the status of Piaget's theory in light of contemporary data.

- Chapter Three describes the schema mechanism: its data structures, and its machinery for building new instances of those structures.

- Chapter Four presents a synopsis of the schema mechanism's implementation's achievement of some of the developmental milestones described in the second chapter, and proposes a hypothetical scenario of further achievements.

- Chapter Five raises the possibility that the basic learning mechanism, acting in concert with its own constructs, can implement more sophisticated virtual structures and mechanisms.

- Chapter Six appraises the schema mechanism with respect to its accord with cognitive science, and its relation to other A.I. research programs.

Chapter 2

Synopsis of Piagetian development

This chapter discusses Piaget's theory of the development of sensorimotor intelligence, as described in (Piaget 1952a) and (Piaget 1954). I present a summary of the original theory, and suggest a reinterpretation in light of modern evidence which reveals infant knowledge that is inexplicably precocious by Piagetian theory. To anticipate, the reinterpretation suggests that, although some cognitive modules have, for example, extensive built-in knowledge about properties of physical objects, there is also a distinct learning mechanism, in charge of governing actions, which starts without such knowledge, and develops it for itself according to the Piagetian script. Thus, while Piaget's story turns out not to be true of the infant as a whole, the story may well be true of one important cognitive module.

The point of departure of Piaget's theory is the schema: a unit of behavior and knowledge which, by Piaget's biological metaphor, interacts and evolves with its physical environment, and with other schemas. The initial schemas are merely those of reflex responses. For quite some time, the infant's schemas are closely associated with her own actions. Later sophistications, involving the combination of schemas, abstraction above specific acts and perspectives, and the *interiorization* of schemas' activity, will allow the schema to transcend a literal dependence on physical action, while retaining its procedural flavor. Schemas of looking, grasping what's seen, swing-

ing, dropping, hiding one object under another, pushing one object with another, are examples of post-reflex schemas.

Piaget identifies as the functional invariants of intelligence *assimilation* and *accommodation*: respectively

- a schema's use of things in the world (including other schemas) as part of its own functioning; and
- the modification of schemas in adjustment to novelties in the world.

Of course, Piaget doesn't try to present complete, explicit rules governing the activity and modification of schemas. But his theory does try to characterize such rules and to give an intricate chronicle of the low-level results of their functioning.

The sensorimotor period (from birth until about age two) is the first of four broad periods of development in Piagetian theory. Sensorimotor intelligence is expressed solely in *actions* that affect the world. In the later phases—the *preoperational* phase, then the phases of *concrete operations* and *formal operations*—the *truth* of assertions about the world becomes the focus of intelligence, first for assertions about the *real* world, and later in the realms of the hypothetical and the abstract. (See Piaget and Inhelder 1969.)

Piaget distinguishes six stages within the sensorimotor period. Each successive stage is characterized by schemas that embody a new elaboration of problem-solving[1] or goal-pursuing activity (which never implies the eradication of less sophisticated schemas, or even that such schemas stop being created). The elaborations characteristic of a given stage do not appear simultaneously; the "stage" is just the period during which such appearances first peak. A stage's uniformity is thus a descriptive invention, and doesn't imply rigid chronological partitioning.

The infant's representation of reality—space, objects, causation, time—exhibits corresponding stages of development. In fact, Piaget argues that progressively more

[1]The infant's earliest behavior is only a zeroth-order example of "problem-solving"; later stages do greater justice to the term.

sophisticated techniques of intelligence, and progressively more sophisticated representations of reality, are two indissociable aspects of the same development. At the outset, problem-solving is just the dynamic expression of the infant's representation of reality—a natural enough idea, since the infant's schemas are procedural: a thing is understood in terms of what can be done to it or with it. So, more advanced problem solving results from the application of the same mechanism to more sophisticated representations of reality, and vice versa. Eventually, of course, the child acquires explicit knowledge *about* thinking that can be used to improve methods of thought; but there is substantial maturing of intelligence long before such meta-knowledge is evident in the child.

One critical feature of the infant's intelligence, not well captured by this summary, is the *incremental* quality of its development. At least at the outset, each new capability observed in the infant is only slightly different than what was previously exhibited; the infant shows only minor adjustments of activity, in apparent response to experience in prior activity. It should be kept in mind that the actual steps are of much finer grain than are presented here. As intelligence progresses and there come to be more powerful schemas for interpreting the world, the steps grow bolder, and, in ways that I'll discuss, less dependent upon specific experience. So, the change from trivial to powerful steps is a smooth one; the increments by which intelligence improves are, in effect, of size proportional to the power of existing schemas, so the development has an exponential character.

2.1 First stage: reflex activity, solipsist images

The infant's initial schemas are those of reflex activity: for example, closing the hand in response to a touch on the palm, or sucking something that touches the lips. These schemas are exercised either in response to the appropriate stimuli, or else spontaneously, as though for play or practice.

From the outset, schemas admit of modifications in response to experienced results of their activity. For example, after many instances of disorderly reflexive groping for a nipple touching the mouth, an infant's sucking schema appears to notice that when the nipple touches (say) the left cheek, turning to the left will be propitious. Groping in adjustment to the nipple thus assumes a gradually more coherent appearance, as clues such as cheek-contact are exploited.

The early development of schemas also shows generalization and differentiation. For example, the sucking schema adjusts itself not only to the nipple, but also to other objects frequently presented to it: e. g. a finger or a toy. Often, the infant will suck such an object as contentedly as if it were a nipple. But when hungry, the infant responds with enthusiasm to the nipple while crying instead if given a finger to suck. The appearance of this discrimination suggests that, despite the production-like character of schemas' early, stimulus-triggered activity, the desired *result* of a schema's activity also affects its course.

The first few months of life also see the first so-called *primary circular reactions.* These are patterns of action, derived by gradual differentiation of reflex schemas, that tend towards repetition. For example, the grasp-reflex schema gives rise to a alternately-hold-then-release-object schema, and to a scratch-object schema, and so on. As with pure reflex schemas, these sometimes repeat "emptily", that is, without any stimulus or object to interact with.

Visual schemas developing at this time include those of tracking a slowly moving object, of visually exploring a stationary object, and of alternate glances between one object and another.

A striking feature of these early schemas is that they haven't yet "intertwined". For example, tactile stimuli elicit no visual response; things seen inspire no effort at prehension. Moreover, when for example a watched object passes beyond the infant's field of view, the infant either loses all evident interest in it, as though it no longer existed; or else, with apparent expectation of seeing it again, either continues to look

off in the same direction, or gazes back to where the object was first seen. Similarly, an object that is touched but not seen may be repeatedly grasped then released; but if, say, it falls to a new position, the infant will neither search for it visually, nor move her hand to search for the object in a different position than where just grasped.

These observations imply that the infant's model of the world—in the sense of what aspects of the world the infant can react to or exploit—is (metaphorically) solipsist in nature: the infant's universe contains not objects of substance and permanence viewable from different perspectives, but rather "images", some visual, some tactile, etc., that change state in response to personal actions (themselves "known" only by the transformations they produce). The infant's early schemas organize the world into various solipsist spaces, each giving a group (in the mathematical sense) of operations: the operations are primitive motor actions (or, sometimes, passive expectation), and the things operated on are sensory states.

2.2 Second stage: the coordination of primary schemas

As reflex schemas elaborate into primary circular reactions, they also begin to intercoordinate and thus to bridge the gap between sensory modes. The primary circular reactions, and the intercoordinations, both appear to have the same character of development: a schema acquires differentiated responses to, and anticipations of, sensory signals with which it was previously unacquainted. If the new signals of one schema are already familiar to another, then a functional intercoordination results, as when schemas of hand movements combine with sucking to form in integrated thumb-sucking schema.

Initially, an infant will suck her finger (or other object) only if it comes in fortuitous contact with the infant's mouth (or, slightly later, cheeks etc.). (Even then, the infant doesn't know how to keep her hand in place, and the hand is quickly pulled away.)

But random hand movements may accidently brush the hand against the vicinity of the mouth. Not only will this trigger attempts to suck, but also, future hand trajectories will converge to the mouth more and more directly. Eventually, the infant can smoothly and spontaneously move her hand to her mouth, and insert and suck on a finger. Later, a more profound development is seen: the infant is capable of carrying a grasped object to her mouth and sucking on it; thus, prehension is coordinated with sucking.

More striking still is the coordination that develops between vision and prehension. Piaget discerns a number of milestones in this development:

- The infant watches the movements of her hand, and gradually learns to bring her hand into her visual field, and keep it there while watching it.
- The infant watches while grasping and releasing objects.
- The infant subsequently will turn to look at an object when the object touches her hand, or will move the object into her visual field to look at it.
- At some point, the infant will reach for an object, but only if the object and the infant's hand are seen together.
- Eventually, the sight of the object alone will suffice to trigger a successful attempt to grasp it.

Of course, each of these bits and pieces of eye/hand coordination develops not as a sudden leap, but by gradually improved groping.

The acquisition of visual/tactile coordination has an important consequence: hereafter, the infant's learning and attention become oriented around "objects", not just particular sensory impressions. The appearance of this more objective behavior marks the onset of the next sensorimotor stage.

2.3 Third stage: secondary circular reactions, objects of subjective permanence

The third sensorimotor stage usually begins four or five months after birth, and continues until eight or nine months of age.

Secondary circular reactions are characteristic of third stage behavior; these consist of the repetition of actions in order to reproduce fortuitously-discovered effects on objects. For example:

- The infant's hand hits a hanging toy. The infant sees it bob about, then repeats the gesture several times, later applying it to other objects as well, developing a "striking" schema.

- A strange sound is made by accidentally striking the crib wicker with a toy. The infant reproduces the motion involved, and after more occasional fortuitous contacts, will rub the toy deliberately against the wicker. However, spatial contact between the objects is not understood as such. If the infant's position is changed such that the customary gesture fails to achieve contact with the crib, she repeats the gesture anyway, doing nothing that adapts to the altered situation.

- The infant pulls a string hanging from the bassinet hood, and notices that a toy, also connected to the hood, shakes in response. The infant again grasps and pulls the string, already watching the toy rather than the string. Again, the spatial and causal nature of the connection between the objects is not understood; the infant will generalize the gesture to inappropriate situations.

In these reactions, the infant responds quickly to a novel result by using a familiar schema to reproduce the result, even though the schema had never previously been used for that purpose. However, the effect is discovered by accident, and only the particular schema involved in the accident is used to reproduce the effect.

Nonetheless, thanks to the intersensorial schemas of the previous stage, the current schemas transcend particular primitive motor actions and sensory images. This, together with the more complex chain of actions involved in, say, seeing, grasping, moving, or rubbing an object, give secondary circular reactions the appearance of being goal-directed (where the goal is to reproduce the surprise effect), in contrast with the stimulus-bound appearance of the primary circular reactions.

The sense in which the third stage initiates the representation of objects rather than images is perhaps best described as follows: if one were to write a program that did the sorts of things that a third stage infant does, the program would most naturally be written on a level of abstraction that designated objects; a program to mimic earlier stages would most naturally lack such a level, and would instead be oriented around sensory images.

To the extent that they deal with objects rather than images, the secondary circular reactions can begin to designate interactions, and hence practical relations, between objects—but with the limitation that the relationship is given only by a schema with a particular motor action, implying both unnecessary restrictions, and inappropriate generalizations, of the relation (as in the wicker-striking and hood-pulling examples above).

Similar progress, and limitations, appear in the third stage representation of objects' permanence and position:

- *Deferred circular reactions* appear. An infant, playing with a toy (via a secondary circular reaction schema) is momentarily distracted but soon turns back to where the toy was left and resumes playing with it. This is similar to, but more complicated than, the earlier feat of looking again at one image after shifting gaze to another; here, a coordination of body and hand movements, guided by vision, is required to recapture the object.

- When the infant is watching an object that falls, moving too quickly to track so that she loses sight of it, she will look downwards for it. At first this happens primarily when it was the infant who held and dropped the object, and is also catalyzed by the sound of the fallen object, or by tracking it momentarily when it starts to fall. Eventually, the reaction becomes reliable even in the absence of such clues.

- Similarly, if the infant holds (without looking at it) an object that falls, or is taken, from her hand, she learns at this stage to extend her hand and reclaim the object.

Thus, the third stage infant apparently conceives of objects as occupying particular positions at which they can be reclaimed if they vanish from view. Moreover, in contrast with the previous stage, the object can be sought in a *new* position, rather than the first or last place that it was recently perceived. However, closer observation shows that this reclamation is only understood with respect to a particular schema of action. The infant confronted with an object's sudden disappearance tries to recapture it either by extending the activity of a schema already invoked to keep sight of the thing—e. g. for the falling object—or by reusing a schema just used to secure the thing in the first place—e. g. reaching to regrasp an unseen object removed from the hand. In this latter case, if that particular gesture fails to rediscover the object, the infant will *not* (until the next, fourth, stage) employ perpendicular motions in a systematic search for the thing, but may instead revert to looking for it in its original position.[2]

That the position of vanished objects is first conceived only in terms of particular action schemas is further attested to by the reaction of an infant to the intervention of an obstacle. If an infant of this stage is presented with a toy which, as she watches, is covered with a cloth, the infant will not attempt to raise the cloth to recapture the

[2]This reversion to cruder techniques when more advanced ones fail tends to occur through all stages of sensorimotor intelligence, and later intelligence as well.

object—despite the fact that the infant is quite capable of picking up a cloth when that itself is of interest. When the toy disappears, the infant either loses interest, stares at where it was, or looks back at where it was first seen (if that was a different place), but does not reach for it—or, if already reaching for it when sight of it is blocked, will immediately give up. In fact, even if the infant's attempt to grasp a toy is thwarted by a barrier that doesn't block sight of the toy, the infant appears to be oblivious to the barrier, making no attempt to displace it or move around it. The infant does, however, learn during this stage to grasp and extricate the hidden toy if *part* of it is visible.

The need to *rotate* an object presents intellectual difficulties similar to those posed by the need to move an obstacle. Suppose a third stage infant is presented with a bottle, but the bottle is held with the nipple facing away from the child, so that the nipple cannot be seen. Thus the important part of the bottle is obscured, not by a foreign object but by the rest of the bottle itself. The infant exhibits problems similar to those produced by a separate obstacle, giving up on the nipple when it is no longer perceived. The difficulty is not a lack of the motor skill required to rotate an object, since while the nipple is visible, the infant will turn the bottle to make the nipple accessible; this is done quite unsystematically, but persistently until fortuitous success is achieved. So the difficulty is again a representational one, characteristic of this stage: the "potential nipple" (as opposed to the nipple when actually perceived) is understood only in connection with certain schemas known to actualize it. There is not yet a schema of rotation; the successes in orienting a visible nipple appear to be due to a series of separate movements, each guided crudely by the current perception of the nipple, and not organized into a coherent activity of reorientation. When, in the next stage, these attempts are arranged in a coordinated structure, there will indeed be a schema of rotation, with respect to which the potential nipple can be represented.

Finally, it should be noted that during the third stage, a "potential-X-with-respect-to-prehension" is not strongly coordinated with a "potential-X-with-respect-to-vision". For example, an infant of this stage who has looked at, but not touched, an object that falls below her gaze may look downward for it, but will not make any tactile search for it.

2.4 Fourth stage: coordination of secondary schemas

The fourth stage brings a coordination of secondary schemas analogous to the second stage's intertwining of primary schemas. Just as the second stage allowed the infant's representation of the world to transcend specific primitive motor sequences and sensory impressions, and abstract these to *acts* upon *objects* (the subject of third stage learning), so the fourth stage coordinations will allow the infant's understanding to become independent of particular acts, preparing for fifth stage elaboration of the activity of objects themselves, and their interrelationships.

The fourth stage infant is capable of using a familiar schema for a new purpose in a new situation. This contrasts with the previous stage, whose secondary circular reactions did allow familiar schemas to be used for new effects, but only if these effects had previously been empirically (and fortuitously) produced.

A classic example of this is the removal of an object blocking the prehension of a desired toy. This may be catalyzed by the accidental displacement of the intervening object when the infant initially ignores it. But at some point, the infant's attention is focused specifically on moving the obstacle (at first clumsily, but successive efforts develop a well coordinated schema of -displacement- by picking up and moving, or by striking). The infant's behavior makes clear that she is not interested in the obstacle itself, since it is discarded and the desired toy is then grasped. The obstacle displacement was thus subordinated to that goal. (Interestingly, it isn't until shortly after this displacement coordination that Piaget observes the advent of the infant's

ability to release one toy being held in order to pick up another.)

An important variation of the above displacement coordination is the removal of an object that blocks the view of a desired toy. In transition between the third and fourth stages, an infant might continue to reach for and grasp a toy whose view was blocked, provided that the infant had already started to reach when the object disappeared from sight. This, along with the extrication of partially hidden objects (from the previous stage), and the displacement of non-hiding obstacles, leads to the ability to react to the complete covering of an object by removing the cover and claiming the rediscovered object. This is quickly generalized into a game of repeatedly hiding and recovering an object.

Recall the third stage inability to, say, respond with prehension to a "potential visual object". During the fourth stage, "potential" (in contrast with actually perceived) objects with respect to different schemas are united in a way reminiscent of the second stage's marriage of visual and tactile perceptions. The ability to uncover a hidden object extends this unity: not only is there a prehensile remedy to a visual disappearance, but the remedy is complicated, involving a pair of secondary schemas that deal with two distinct objects. Thus, both the permanence and spatial localization of vanished objects are now understood, not just with respect to a given secondary schema, but with respect to coordinated pairs of such schemas. This begins to put objects in spatial relationship to one another. Similarly, the infant of this stage becomes capable of:

- Systematic search. Eg, when the infant drops an object, her hand will not only be moved down to find it, but will also be moved perpendicularly in exploration of the immediate vicinity.

- Systematic rotation. The infant can recover the obscured reverse side of an object.

- Exploitation of perspective. The infant can shift her head to look around an

obstacle.

- Imitation of familiar but invisible movements. During the third stage, only visible actions, producible by existing schemas, are imitated; e. g. grasping a toy. (Interestingly, there is no imitation of a sequence, such as opening and closing a hand, that is exercised as a part of various familiar schemas, but not yet differentiated in its own right.) In the fourth stage, the infant will imitate an action (such as sticking out the tongue) that she has taken many times, but without having seen its effects. (Prior visual/tactile exploration of faces, in conjunction with sounds sometimes accompanying the gesture, provide clues that assist that identification.)

- Systematic exploration of novelty. When presented with a new object, the infant applies in succession many familiar schemas to the object: shaking, striking, rotating, etc. During the third stage, a new object would tend to excite some schema or other, but the current emphasis is different: the schemas now seem focused on the object, while previously, understanding of the object seemed focused on a particular schema. (An unexpected effect of some exploratory action—say, the production of an unusual sound—may give rise to a secondary circular reaction repeating that effect. Piaget calls such a reaction *derived* to denote that it arose in the context of more structured activity, namely the exploration.)

Despite these advances, the fourth stage representations of reality still exhibit many limitations of subjectivity. The most striking of these is the fourth-stage *place error*, shown by the following experiment. The infant plays with a toy that is then taken away and hidden under a pillow at the left. The infant raises the pillow and reclaims the object. Once again, the toy is taken and hidden, this time under a blanket at the right. The infant promptly raises, not the blanket, but the pillow again, and appears surprised and puzzled not to find the toy.

This sort of confusion is observed repeatedly during the fourth stage. It is a remarkable analog to the earlier reaction to disappearance by searching in the first or last place that the thing was recently perceived, or in a new position by extending a reclaiming schema. Then, hidden position was represented only with respect to the comparatively simple schemas that existed. Now, hidden position is understood in terms of combinations of such schemas, which relate pairs of objects. Although more complex, the representation is still procedural, and the procedures involved have only developed to the point of saying something like: "when this toy disappears, displacement of the pillow will rediscover it."

So the relationships among objects are yet understood only in terms of pairwise transitions, as in the cycle of hiding and uncovering a toy. The intervention of a third object is not properly taken into account. Moreover, the infant still comprehends the displacement of an object relative to herself rather that to another object. For instance, an infant who can easily turn a block around does not yet learn to orient it relative to a box so as to fit inside. Similarly, there is no comprehension of the need to put a stick in contact with a semi-distant toy in order to move the toy. These feats will be possible in the following stage.

2.5 Fifth stage: experiments on objects

During the fifth sensorimotor stage (usually beginning about a year after birth) the so-called *tertiary circular reactions* appear. These are little "experiments" that the infant conducts to see what an object will do. For example, an infant may repeatedly drop a toy, paying evident attention not to the act of dropping, but to the behavior of the *object* as it falls. Similarly, the infant experiments with varying ways of placing an object on an inclined surface to watch it roll, or perching it at the edge of a table so that it tumbles to the ground, etc.

These experiments extend the focus on an object's behavior, rather than per-

sonal action, noted during the last stage. But where fourth stage explorations merely use the object in existing schemas, the present experiments vary the exploratory schemas—not just in *response* to surprise results (as with the derived secondary reactions noted in the previous section) but in *provocation* of unexpected behavior. (Indeed, the specific autonomous activity of an object is yet unexpected by the infant, as evidenced by systematic inability to account for it when necessary. For example, an infant trying to dispose of an obtrusive cushion repeatedly pushes it back against a wall, but in such a position that it must fall back in the way again.)

Tertiary (like secondary) circular reactions can be coordinated with other schemas in a means-end relationship. For instance, an infant reaches through the bars of a playpen to grasp a long toy. The infant doesn't anticipate the solidity of the bars, which block the toy from being drawn closer. (The fourth stage infant learned about the solidity of an obstacle to prehension, but that was only with respect to movement of the hand itself! Here, the infant must learn that one object also blocks the motion of another object.) Although the infant already knows how to rotate an object (say to find its reverse side), there is not yet a schema for rotating one object relative to another, as is called for here so the toy can be oriented to allow passage through the bars. But, lacking such a schema, the infant nonetheless appears to identify the collision as the source of difficulty, and for a long while gropes for different ways of placing the object against the bars. Eventually, a successful orientation is found. On subsequent attempts, the infant's gropings converge more and more quickly to the solution, and a reliable schema of object-relative rotation evolves.

The gropings of this example are tertiary circular reactions, as they involve deliberate *variations* of a repeated action, and with interest in the effect on the object (i. e. whether it is making progress through the bars), rather than in the action itself. Now there is an additional feature: the experiment is directed toward the *goal* of bringing the toy closer. Thus, many schemas influence the activity:

- the grasping schema, which specifies the goal.

- the schema of turning an object, relative to one's self, which gives a point of departure for the new means needed to fulfill the goal.

- importantly, the many schemas that by now exist to describe objects and space; these are needed to interpret meaningfully the results of the experimental variations, to direct refinements of the evolving rotation schema.

- the intermediate approximations to the eventual object-relative rotation schema.

From the observer's point of view, the coordination of these schemas results in an important amplification of the infant's intellectual capabilities: for the first time, the infant responds to an unexpected obstacle by "inventing" a way to overcome it, rather than just relying on an already-existing schema. Piaget concludes that this capability essentially falls out of

- quantitatively, the myriad schemas that can be brought to bear on a situation; and

- qualitatively, the higher level of abstraction on which the schemas now represent things, focusing on objects as such;

thus allowing the same principles of interaction of schemas to yield more sophisticated results.

Similar examples of the invention of new means are found when the infant learns to use a stick, an underlying support, or an attached string, to move a given object. You may recall that some secondary circular reactions involved influencing one object by pulling another connected to the first by a string. But that effect was discovered entirely by accident, and with no appreciation of the physical connection. During the present stage, the infant wishing to influence a remote object learns to search for an attached string, visually tracing the path of connection. As with the object-relative rotation schema, a great deal of intermediate groping is required to develop schemas for using a string, support, or stick. One interesting intermediate situation

that Piaget observes regarding the use of a stick is that an infant who is trying to grasp an object just out of reach, and who has previously succeeded in using a stick to draw the object closer, will not think of doing that unless she is already holding the stick, or unless the stick is presented to her. This is somewhat like the state of a second stage infant who is learning to grasp what is seen, but only when the hand is *seen* next to the object.

These developments add to the infant's conceptions of objects and space. Through the tertiary circular reactions, objects are endowed with autonomous behavior; and the direction of such reactions towards goals involving a second object teaches the infant about the solidity of objects, and relationships among objects themselves. This progress is also reflected in the fourth-stage place error, described above. During that stage, some improvement is made in selecting the right place to look for a vanished object, but the accomplishment has an empirical character and the selection is often wrong, as though the infant had learned that "looking under the *blanket* sometimes works instead" but without really getting the point. On the other hand, the fifth stage infant learns reliably to search the place at which the object was seen to disappear.

2.6 Sixth stage: simulation of events

The fifth stage infant shows no sign of mentally "simulating" the activity of objects and learning from the simulation instead of from actual experimentation. But the sixth stage furnishes evidence of this ability. An infant who reaches the sixth stage without happening to have learned about (say) using a stick may invent that behavior (in response to a problem that requires it) quite suddenly, with dramatically less groping than for similar inventions of the previous stage. Piaget argues that some "interiorization" of physical activity is responsible for this capability.

In addition, the infant now becomes capable of interpreting situations whose understanding requires representation of events not actually observed. For instance,

consider yet another form of hidden object confusion, which the fifth stage infant exhibits: A toy is placed in a small box, without a lid, so that the infant still sees it. Before the infant has a chance to recover the toy from the box, the box is moved beneath a blanket where, hidden from the infant's view, toy is dumped out. The box is brought to view again, empty. The infant is surprised that the toy is no longer in the box, and does not attempt to search under the blanket. Analogously to fourth stage progress with the place error, the fifth stage infant does learn, empirically and unreliably, to search under the blanket. But when *two* screening objects are used in succession, a remarkably parallel confusion results: the infant does not understand the need to look specifically under that cover from which the box emerged. But now, during the sixth stage, the infant deals successfully with these situations, apparently able to represent the unobserved displacement of the toy under the screen.

The above developments are a small sample of the explosion of intellect and knowledge of the sixth stage. The ability to represent one's own body in objective spatial terms, to understand personal orientation (for example, being able to point back to a house that's no longer in sight), and the beginning of language all arise during this stage. The sixth stage thus forms a bridge between sensorimotor intelligence and the later periods.

2.7 Subsequent periods: preoperational, concrete operations, and formal operations

Throughout the sensorimotor period, the infant's intelligence is concerned with the effects of actions on present reality. Even the first manifestations of language, towards the end of the sensorimotor period, are concerned with the expression of desires and commands, rather than the communication of ideas. But in the period to follow—the Preoperational period—the child begins to manipulate the truth of propositions, via inference and classification, just as earlier she had manipulated the state of objects

via physical actions. The child begins to think and speak of past or distant events, of causation and number and time, of other peoples' perspectives.

During the period of Concrete Operations, the child becomes able to reason more systematically about the subject matter of the previous period; as during the various sensorimotor stages, previously uncoordinated fragments of representation become properly connected. A preoperational child, for example, confuses the relative duration of two time intervals with the ordering of their beginnings or ends; a child at that period tends to believe that the older of two people was born later. A preoperational child has not grasped conservation of number (or at least, conservation of 1-1 correspondence); consider the following fascinating (and typical) protocol, taken from a conversation with a child of five years (Piaget-1952b, p. 26):

> What are these?—*Little green* [A2]*and red* [A1]*beads.*—Is there the same amount in the two glasses?—*Yes.*—If we made a necklace with the red ones and another with the green ones, would they be the same length?—*Yes.*—Why?—*Because there's the same height of green and red.*—If we put the beads in there [L], what would happen?—*They would be higher.*—Would there be the same amount?—*No.* —Where would there be more?—*There* —[L].—Why?—*Because it is narrow.*—[A1 was poured into L] Do you really think there are more beads there [L] than here [A2]?—*Yes.*—Why? —*Because it is narrow and they go higher.*—If I poured them all out [making as though to pour the red beads on one side and the green on the other] would they be the same or not?—*More red ones.*—Why?—— *Because that one [L] is narrow.*—And if I make a necklace with the red beads and one with the green beads, will they be the same, or not?—*The red one will be longer.*—Why?—*Because there'll be more in there* [L].—[The red beads were put back into A1.] And now?—*They're the same height again.* —Why?—*Because you've poured them into that one*[A1].—Are there more red ones or green ones? —*The same.*

These and other illuminating confusions are corrected during the period of concrete operations.

The final period of intelligence—the period of formal operations—begins approximately at the onset of adolescence. Just as the ascension from sensorimotor intelligence brought with it the ability to represent abstract truth instead of just current state, the passage to formal operations brings the capacity to represent abstract validity instead of just actual truth. Previously the individual could use one proposition to imply others in a variety of ways; but now implicability itself—i. e. , validity—becomes an "object" about which the individual can reason. Reasoning about validity as such makes formal reasoning possible—reasoning separated from the content of the propositions reasoned about. In a similar vein:

- True hypothetico-deductive reasoning appears: a person gains the ability to devise appropriate experiments to test hypotheses, systematically varying one factor, then another, while holding the others constant. Previously the individual maneuvered in a space of propositions linked by (more or less) logical entailment; now, an entire such space is a single point in a new space, where going from point to point corresponds to changing a hypothesis.

- The ability to generate systematic permutations appears. The concrete operations individual could reason about sets of things; to generate all possible permutations among a collection of objects, a person must reason about a set of sets, each of the sets being one permutation of the objects.

In all these examples, *relations* among concrete-operations objects in turn become the objects of formal reasoning.

Piaget describes the progression to concrete and then formal operations as a development of more powerfully expressive logics. In reply, (Fodor 1975) argues that such a progression, if indeed it occurs, cannot occur by learning. The essence of Fodor's argument is that less powerful logics, by definition, simply cannot express, and therefore cannot build, systems that embody more powerful logics. This objection, and a

way around it, can be understood by an analogy between logics and classes of computational entities. A finite-state automaton is strictly less powerful than a Turing machine: a Turing machine can simulate a finite-state automaton, but not vice versa; hence, a finite-state automaton cannot possibly learn to be a Turing machine.

Nonetheless, any physically realized digital computer, though conventionally regarded as Turing-equivalent, is really just a finite-state automaton. It is considered Turing-equivalent via the reasonable and customary idealization that its memory is infinite. There are no precise rules governing the suitability of this idealization; roughly, the idealization is appropriate when a finite-state automaton has a large number of state elements that it uses more or less uniformly. Conceivably, a finite-state automaton might have an initial state that does not lend itself to an infinite-memory idealization, but might later enter a state for which that idealization becomes suitable. A Fodor-like argument is still correct, but only as a technicality: formally, there has been no increase in expressive power. Nonetheless, for reasonable practical purposes, by plausible customary idealizations, the system has indeed evolved from being a finite-state automaton to being (virtually) a Turing machine. A similar possibility with regard to the development of logics of varying power circumvents Fodor's impossibility argument concerning the learning of concrete and formal operations.

2.8 Themes of Piagetian development

Several recurrent themes of Piagetian development are illustrated in the foregoing sections (in some detail for the sensorimotor stages, and hastily for the subsequent periods). These also serve as central themes for the design of the schema mechanism.

- Intelligence develops by building state-spaces to represent the world:
  - by discovering how states and transformations are related; and
  - by constructing new elements of the space, and new transformations, whose relations must in turn be discovered.

From motions of physical objects to inferences among propositions, this theme is repeated throughout Piagetian development.

- New schemas are formed as incremental differentiations or generalizations of existing ones.
- Schemas are intercoordinated to form composite structures that abstract above the details of the component elements.
- Another important kind of abstraction involves *conservation*—the discovery of a new kind of thing in the world, found by noticing the possibility of returning to some manifestation of it.

2.8.1 Fragmented representation

Perhaps the most powerful theme, composed of the above strands, is that the bootstrapping of intelligence involves the assembly of concepts from special-case fragments. That is, many apparently atomic or fundamental concepts are in fact composites of a large body of constituent schemas, from which the "atomic" thing arises. For example:

- Knowing that the ball is on the table entails the expectation that it can be detected there by sight, or by touch (or by weighing the table and noticing the extra weight...); and entails that it won't be found elsewhere at the moment (such as on the floor); and that it must have gotten there somehow, that it used to be in a different position but moved; etc.
- Knowing that four things are present entails that adding another will make five; that if none are added or removed, there will still be four; that if they are counted, in any order, with each counted exactly once, the result will be "four"; etc.

For each of these concepts (and many others), Piaget demonstrates that certain "entailed consequences" of the concept can be seen coming into use for the first time (thus, by implication, first existing) at different stages of development. Gradually, they are organized into a coherent whole. Only in the eventual mature result are the constituent parts of the concept so well coordinated, their mutual entailment so "automatic", as to give rise to a functional unity.

2.8.2 Stages

The role of *stages* in Piagetian theory is often over-emphasized. As mentioned above in section 1.1, the apparent simultaneity of the innovations of a given stage is an expository device; the actual uniformity is only approximate. Moreover, even for some particular strand of development, the invariance of the ordering along the sequence is both less absolute, and less important, than is often thought. There are several reasons that development A might be observed to preceed B, or on the other hand to be contemporaneous with B, in a typical individual's development. For example:

- A and B might each derive quickly and independently from some common ancestor C, and thus tend to appear at the same time.
- A and B might develop (mostly) independently, with A just being "simpler" than B, so that A would appear first.
- A and B might be comparable points along two similar but independent sequences of constructions, whose analogous developments are roughly contemporaneous.
- Some of A's structures might be included as components of B's; A's structures are then a prerequisite for B, so A must appear first.

In the first three cases, it is plausible that the typical order of A and B might be altered by circumstances that cause the individual to focus an unusual amount of

attention on one or the other. Thus, it is not surprising that (White and Held 1966), for example, have shown that by varying the prominence of a hanging, brightly-colored object in infants' early environments, experimenters can induce variations in the order of acquisition of hand-regard and "swiping" behavior. Even in the fourth case above, where the ordering constraint is the strongest, it is possible that alternative paths of development can bypass certain prerequisites, especially when unusual conditions (say, physical handicaps) block the "most natural" path.

Indeed, there is no *a priori* reason to expect a constructivist mechanism to exhibit stagelike regularities at all; the space of plausible developmental paths might be large enough for each individual to pursue her own idiosyncratic construction, in some or all domains. Alternatively, there may be domains where a particular next step is always so "obvious" that there's little room for variation. But in fact, some domains do show strong developmental regularities among different individuals, and it is natural for the study of constructivist mechanisms to begin there. For by observing similar developments among different individuals, the experimenter can partially compensate for being unable to repeat, with controlled variations, the same development for a given individual. Hence, a reason for the preponderance of stagelike developments in the discussion of constructivism.

2.8.3 Constructivism vs. nativism

A constructivist account of the development of intelligence holds that the difference between the mind of an adult, and that of an infant, lies in mental structures built by the individual. Even when a given concept is attained *universally* (e. g. the idea of a physical object), it is because the concept is prominent in reality, in a way that is accessible to the mechanism of learning. A nativist account, on the other hand, holds that universal knowledge is innate, and is either already operative in the neonate, or unfolds according to a predetermined, nonlearning process.

The debate between constructivist and nativist accounts of human intelligence

extends back to antiquity. In a famous dialog, Socrates leads a student to a difficult conclusion by a series of leading questions; Socrates concludes that the student must have known the conclusion all along, since the teacher communicated no facts, just questions (eg Russell 1945, p. 92).

Modern arguments on this subject often involve actual evidence. But the interpretation of such evidence can be difficult; it is easy to under- or over-attribute knowledge to an infant. The fact that a certain piece of knowledge does not show itself in an infant's behavior until a certain age does not guarantee that it was recently learned. Perhaps the infant had the knowledge sooner, but lacked some further capability needed to act on that knowledge. Or, perhaps the knowledge was recently acquired, but by a nonlearning maturational process. Piaget's strategy of observing infants' activity can give the false impression that learning occurs, by failing to detect the early presence of knowledge in some latent form.

On the other hand, it is also easy to overestimate an infant's knowledge, by presuming more awareness than is actually required to explain an infant's behavior. Consider an infant who sees an object, then reaches out and grasps it. This could be due to the infant's understanding that there are objects, that an object has a spatial location, that it has visual and tactile manifestations, that a certain visual pattern means object A is at position X, and that moving the hand to position X will therefore result in touching the object, which the infant desires. Alternatively, the infant might have no suspicion of the existence of objects, but might have noticed that a certain sensation, followed by a certain action, results in another particular sensation (which the infant desires). A third possibility is that the infant is just exhibiting a reflex consisting of a motor response to a visual stimulus, without specifically desiring the result of that response, without even anticipating what the result will be, indeed without even knowing that there *is* any result.

In the present example, the Piagetian view is that all three interpretations are correct, each at a different stage of development. Mindless reflex activity yields to

learned predictions that can be harnessed to pursue goals. These predictions are at first in drastically subjective form, expressed exclusively in terms of primitive perceptual inputs and motor actions. The predictions are then reformulated in gradually more objective terms of representation, terms that become progressively independent of personal action and perception.

What sort of evidence can be marshaled for or against such an interpretation? In principle, an examination of the infant's neural apparatus could reveal what sort of cognitive event was taking place; but that would require both a technology for monitoring the apparatus, and a theory for understanding what was being monitored, neither of which is forthcoming in the forseeable future. Thus, for now, we must settle for less direct forms of evidence.

- *Pro-Piagetian evidence.* Piaget chronicles a gradual elaboration of abilities, each step incrementally more advanced than the last. The themes of this process correspond to plausible learning methods, which the schema mechanism makes precise. That the incremental elaborations are consistent with the steps taken by a learning mechanism is circumstantial evidence that learning is in fact taking place.

- *Anti-Piagetian evidence.* Many recent experiments reveal infant knowledge that is expressed more subtly than by overt, purposeful action. Often, such expressions occur considerably prior to the first Piagetian manifestations of the corresponding knowledge, casting doubt on the Piagetian interpretation.

Some such evidence suffers from the problem of over-attributing knowledge to an infant. A clear example occurs in T.G.R. Bower's description of a neonate's aversion to a looming object. An infant sits in front of a screen that shows a projected outline of a rapidly approaching object. The infant exhibits an avoidance response: the infant closes its eyes, turns its head away, raises its arms in front of its face, and so on. Bower takes this as evidence that the infant interprets the visual information as

an indication of an approaching object, anticipates that an unpleasant collision could occur, and takes action intended to ward off the collision.

Alternatively, the infant may have no such understanding of the movement and effects of objects, or even of their very existence. Instead, the infant may simply have a reflex that releases a particular motor response to a one simple class of visual stimuli. This more conservative attribution of knowledge is indeed the more plausible, given the obvious benefit of having such a reflex, and the anomolous complexity of the infant's behavior by comparison with any other interactions with objects until several months later.

In other cases, however, Piaget under-attributes the infant's or child's abilities. For example, Piaget demonstrates that a preoperational child, when asked how a given scene (e. g. a model of some terrain) looks to an observer stationed somewhere in the terrain, instead describes how the scene looks from her own vantage point. Piaget infers a general inability to appreciate the difference of another's perspective; but experiments by (Masangkay et al 1974) show that in simpler tasks—eg, asking which of two sides of a card an observer sees when the card is placed between the child and the observer—children as young as two answer correctly. Still, in view of the Piagetian theme of assembling concepts from simpler fragments, it remains plausible that these special-case earlier abilities, missed by Piaget, are precursor components of a more general ability exhibited in the tasks Piaget describes.

Some recent experiments, however, demonstrate knowledge that exists prior to any evident Piagetian explanation. Here, the recent work of Renée Baillargeon is exemplary. In one experiment (Baillargeon 1987), a five-month-old infant (third Piagetian stage) sits opposite a plywood board; the board attatches to a tabletop by hinges on which the board can rotate toward or away from the infant. Initially, the board is rotated flat against the table, tilting toward the infant. Just behind the board is a small toy. As the infant watches, the board rotates up, away from the infant, until it blocks the infant's view of the toy. The experimenter then surreptitiously removes

the now-hidden toy, and the board continues its rotation until it is flat on the table, tilting away from the infant; the board could not have rotated that far if the toy were still in its way.

This seemingly impossible event surprises the infant, as determined by the infant's extended scrutiny of the apparatus, compared to (among other relevant controls) the time spent looking at similar rotation in the absence of an obstructing toy. Moreover, the infant takes into consideration such properties as the hidden toy's size and compressibility, showing surprise only if the board rotates further than those properties should allow.

Baillargeon's evidence thus reveals knowledge of hidden objects in infants who cannot yet retrieve such an object by displacing the barrier (despite being able to grasp and move the barrier when that object is itself of interest). It remains an open question whether such knowledge is innate or learned. Clearly, however, Piaget's explanation for the third-stage obliviousness to hidden objects—that the infant simply does not represent that the object still exists—is contradicted by Baillargeon's evidence.

Nonetheless, a close revision of Piaget's claim remains viable: that the Piagetian story is true, not of the infant's cognition as a whole, but of the infant's central cognitive system. The central system, by this hypothesis, incorporates a general learning mechanism, and uses what it learns to guide its actions to achieve goals. True, the infant's peripheral, perceptual modules enjoy extensive, possibly innate knowledge about physical objects. But the central system, according to the revised Piagetian claim, lacks access to the knowledge embodied in the peripheral modules. Those modules use their knowledge to assemble the perceptual input to the central system, which, by this account, has no initial understanding of that input. The central system must recapitulate for itself much of the peripheral modules' knowledge (such as awareness of hidden objects), according to the Piagetian sequence. Observations of infants' purposeful behavior, in contrast with experiments that elicit subtle indications

of surprise, reflect the Piagetian learning accomplished by the central system.

It may seem implausibly wasteful for the central system to have to recapitulate knowledge already present in other modules. Suppose, however, an evolving learning system suddenly (on an evolutionary timescale) became powerful enough to go far beyond its original function—in particular, powerful and general enough to recapitulate *and transcend* much built-in knowledge in other cognitive modules. The knowledge in other modules would not be suitable for extension by the learning mechanism; there would be no reason for that knowledge to be in a form accessible to the learning mechanism. The built-in knowledge would then be redundant—perhaps, in some cases, even vestigial—as the learned recapitulation gained importance.[3] Built-in, biologically evolved knowledge of the existence of physical objects is (partly) superseded by similar concepts re-invented by each individual; to put it succinctly, *ontology* recapitulates phylogeny.[4]

Postulating the learned recapitulation of apparently built-in competence may seem an unduly contorted effort to salvage Piaget's theory by explaining away the contrary evidence. In my judgement, in the absence of clear positive evidence for constructivism, the defense would be plausible but weak. What makes the view stronger is that it turns out to make sense for infant development to result from a Piagetian learning mechanism, in that, I claim, a reasonably designed mechanism would indeed exhibit the milestones of infant development. This dissertation presents preliminary evidence for that claim.

Baillargeon's own conclusion appears compatible with the recapitulation interpretation, among others. Baillergeon construes the evidence to demonstrate a failure of coordination between an infant's knowledge of hidden objects, and purposeful activity that rests on that knowledge. One possible form of this coordination is the revised

[3]Even if such recapitulation is thus required, might we not expect evolution itself to perform the recapitulation, building the duplicated knowledge directly into the central system? Perhaps, eventually. But would that built-in recapitulation evolve before the central system became powerful enough to ask this very question? If not, the evolved recapitulation has not happened yet.

[4]This adroit pun is due to Ed Hardebeck.

Piagetian view: the perceptual system provides the central system with input, but does not share the knowledge that helped produce the input, which knowledge the central system must recapitulate. Alternatively, such recapitulation may never occur; perhaps the pre-Piagetian awareness of hidden objects is what the infant eventually uses to recover hidden objects, after some yet-unknown impediment to that coordinated activity is overcome, perhaps maturationally. Should this prove true, there may be little to salvage of the constructivist account of sensorimotor development.

The research program presented here takes the revised Piagetian claim as a working hypothesis. The revised claim suffices to support the dual motivation for this research: using human cognition as inspiration for design of an artificial mechanism, and experimenting with an artificial mechanism in order to elaborate and demonstrate the possible workings of human cognition. If, on the other hand, the nativist alternative is correct after all, then these motivations collapse. In that case, aspects of the schema mechanism will still hold interest as artificial learning techniques, but the likelihood of their being prominent in human or humanlike development will be far smaller.

Even if the revised Piagetian account is correct, there remain many possible versions of the claim, with different balances of nativism and constructivism. Consider these illustrative points along a spectrum of possibilities:

- There is an invariant constructivist mechanism, and it is responsible for Piagetian development within the central system.

- The constructivist mechanism is invariant, except for some parameters or resource levels that improve maturationally. This maturational system serves merely to *delay* cognitive development, compared to a system in which the full complement of resources was available from the outset.[5]

[5]Conceivably, though, certain delays of complexity actively help subsequent development by providing useful simplifications to build upon.

- Various auxiliary features are added maturationally, embellishing the the constructivist mechanism but leaving it qualitatively unchanged. As in the case just above, Piagetian development still results from the structures built *by* this mechanism, rather than from the predetermined changes *to* the mechanism.

- There is a preprogrammed succession of fundamentally different developmental mechanisms; for example, one for sensorimotor development, one for the concrete operations phase, one for formal operations.

- Much development is via structures built by a constructivist mechanism, but some major developments (say, instantiating a universal grammar) occur maturationally, due to other, more specialized mechanisms.

- No cognitive development is driven by learning. Acquired knowledge is tightly constrained, for a given domain at a given stage, to be of the sort that that domain's module is preprogrammed to accommodate at that stage.

Only the first of these possibilities is purely constructivist. But only the last three have *significant* maturational, nonlearning aspects, and only the last is entirely nonconstructivist. Any but the last of these possibilities preserves the motivations for building the schema mechanism. The schema mechanism itself, as currently proposed, is at the constructivist extreme of the above spectrum, but that can be regarded as a simplifying assumption. At this distance, the difference is not yet perceptible.

Chapter 3

Inside the schema mechanism

The schema mechanism is engineered to pursue two symbiotic fundamental objectives: to gain knowledge by constructing or revising symbolic assertions about the world, and to use those symbolic constructs to pursue specific goals and to gain further knowledge. The acquisition of symbolic constructs in turn has two principal themes: making discoveries expressed in terms of existing representational elements, and constructing new elements with which to express further discoveries.

This chapter describes the schema mechanism and aspects of its present implementation, the computer program MARCSYST I. The chapter begins with a description of the microworld in which the implementation exists, and the sensorimotor interface between the schema mechanism and the microworld. There follows a specification of the structure and function of the representational elements used by the schema mechanism, and the machinery for constructing and revising new instances of those elements.

A caveat concerning experiments with computer programs is in order. In the hard sciences, experiments must be replicable; published descriptions must be at a level of detail that is adequate to that purpose. Apart from safeguarding against outright error (or fraud), this policy helps determine that observed results indeed follow from the factors described, rather than from extraneous, unnoticed conditions.

Computer experiments in AI, however, are seldom subject to systematic attempts at replication[1] This may be due in part to the nature of computer programs; a deterministic program that produces a given output for a given input will always do so. In this sense, replication is trivially guaranteed.

On the other hand, the slightest change to a program obviates the trivial guarantee. One would like to know what characteristics a program must have—short of being identical—in order to behave similarly to a given prototype. The description that follows is an attempt to so characterize the schema mechanism implementation. As in the hard sciences, this attempt should be validated by independent replication.

3.1 The mechanism's microworld

The computer program MARCSYST I, which implements the schema mechanism, operates in a discrete, two-dimensional microworld (figure 3-1). The program controls an *animaton*—a simulated robot with an animated graphic depiction—that has a body, a single hand, and a visual system. The hand can touch and grasp objects, and move them about. The visual system maps a visual field onto a region of the world in the immediate vicinity of the animaton's body; the visual field provides a bird's-eye view of that region. The animaton can shift its gaze, changing the body-relative orientation of its visual field. (The visual system is designed to provide a bird's-eye view, rather than a projection onto a one-dimensional retina, because of the paucity of information in a one-dimensional projection.) Objects in the microworld (including the animaton body, and the hand) are of uniform size. They move but do not rotate, and their motion occurs in discrete units of the same size as the objects themselves. The microworld is *energetic*, meaning that objects can move spontaneously, not just in response to the animaton's actions.

[1]See (McDermott 1981) for discussion; (Haase 1990)'s reworking of (Lenat 1983)'s Eurisko is an exception.

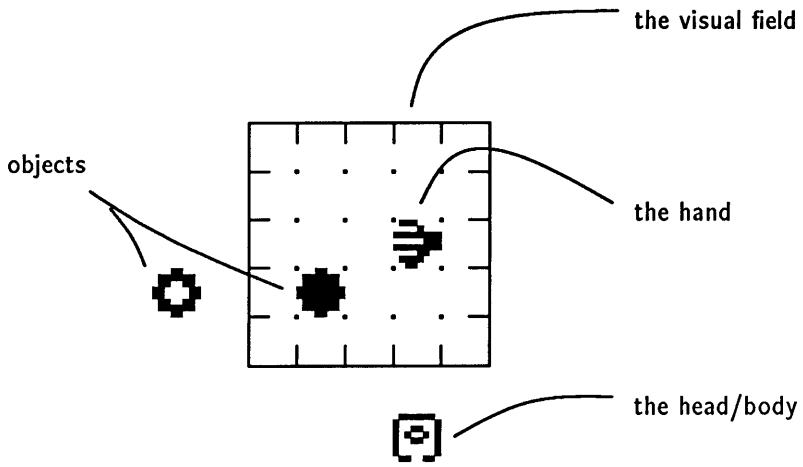


Figure 3-1: The animaton's visual field maps onto a portion of the microworld. Here, the hand and one object are in view.

The schema mechanism's data structures—primitive and constructed—are available to the experimenter for direct examination; in effect, there is a monitoring tool for reading the mechanism's mind. Since the internal representations may be observed directly, we need not try to infer them from the animaton's external behavior. Thus, it is much easier with the schema mechanism than with infants to determine what the system knows, and when it knows it.

The schema mechanism's primitive actions are summarized in table 3.1. The animaton's hand can move about in a particular five-by-five unit region relative to the position of the body, as shown in figure 3-2a (actually, one of the 25 positions is occupied by the body, and thus inaccessible to the hand). Similarly, the visual field can have 25 body-relative orientations, within a range of five units in each dimension (figure 3-2b); the coincidence of the range of eye and hand movements is of no consequence. The potentially visible body-relative region is ten units on a side, since the five-by-five visual field can assume each of five by five orientations. (That the visual field is itself a five-by-five region is also an inconsequential coincidence.)

There are four primitive actions, called incremental hand actions, for moving the hand one unit forward, backward, right, or left. The actions are effective as long as the hand stays within the permitted body-relative range; an action that would move

the hand beyond that range has no effect. Similarly, four incremental eye actions shift the glance orientation one unit in each direction, within the permitted range. Two other actions close or open the hand; these actions can cause objects to be grasped or released. For each of these primitive actions, the mechanism has an initially supplied *bare* schema (empty context and result) with that action.

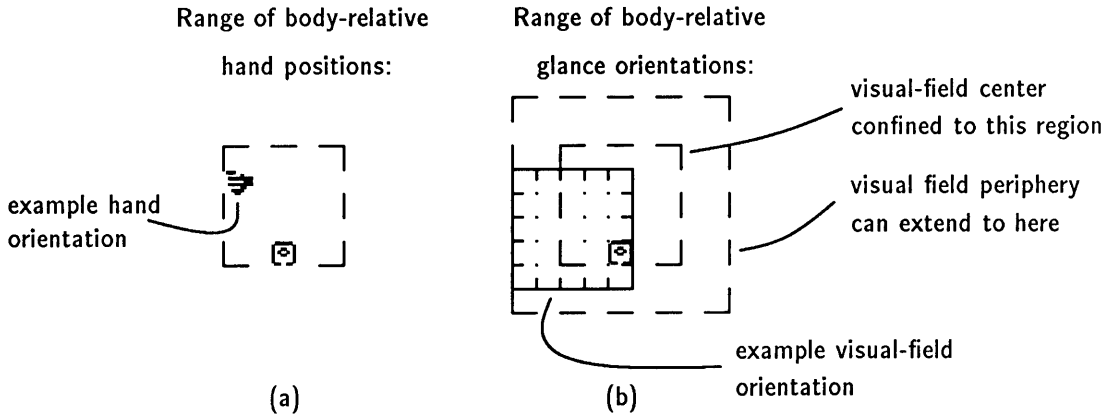


Figure 3-2: The hand and glance each has a 5-by-5 range of possible orientations.

The mechanism's primitive items present visual, tactile, and proprioceptive information (table 3.2). Proprioception is the direct perception of the orientation of limbs or eyes via muscle tension and the like. For each of the 25 body-relative hand positions, there is a proprioceptive item that is On just in case the hand is at that position; similarly, there is a proprioceptive item for each of the 25 visual-field orientations.

There is one coarse tactile item for each of the four sides of the hand, designating contact with that side of the hand. In addition, the left side of the hand (the edge

- **handf, handb, handr, handl:** These actions move the hand incrementally forward, backward, right, or left.
- **eyef, eyeb, eyer, eyel:** These actions shift the glance orientation incrementally forward, backward, right, or left.
- **grasp:** This action closes the hand, grasping any object touching the hand's "fingers" (its left edge) unless the hand was already closed. Once closed or grasping, the hand remains in that state for several (twenty) time units, unless explicitly opened in the interim. Moving the hand moves any grasped object.
- **ungrasp:** This action opens the hand, releasing any object that had been grasped.

Table 3.1: The primitive actions.

with its "fingers") has four tactile detail items that report on the texture of any object that makes contact there; each such item designates an arbitrary, unspecified textural property. There is also an item that is On whenever the hand is closed, and another that is On whenever the hand is grasping something.

Similarly, there are four tactile items designating contact with the four sides of the body; and the front of the body (where the "head" and "mouth" are) has four items that designate arbitrary, unspecified aspects of an object's taste.

There are 25 coarse visual items, one for each of the five by five visual field regions. Each coarse visual item is On just in case an object's image appears at the corresponding region. Coarse visual items thus report only the presence or absence of an object at each region; there is no information as to the specific appearance of the objects. For objects appearing in any of the five *foveal* regions of the visual field (figure 3-3), that information is provided by visual detail items.

There are 16 visual detail items for each foveal region. The items present information intended to be analogous to real-world visual details concerning shape, texture, color, and other aspects of appearance. Rather than being faithful to aspects of real-world appearance, however, the visual detail items, like the texture and taste items,

- **hp00,...,hp44**: Haptic-proprioceptive (hand-position) items, one for each possible hand position. Position (0,0) is the lower left corner of the range; in figure 3-2a, the hand appears at **hp03**.
- **vp00,...vp44**: Visual-proprioceptive items, one for each possible glance orientation. Coordinate designates center of visual field, using same conventions as for hand position; in figure 3-2b, the glance is oriented at **vp01**.
- **tactf,tactb,tactr,tactl**: coarse tactile items, one for each side of the hand (front, back, right, left).
- **text0,...text3**: Detailed tactile (i. e. textural) items, denoting arbitrary textural details of an object touching the "fingers" (left edge of hand).
- **bodyf,bodyb,bodyr,bodyl**: Coarse tactile items, one for each side of the body (front, back, right, left).
- **taste0,...taste3**: Taste items, designating arbitrary surface details of an object touching the "mouth" (front edge of body/head).
- **hcl**: Hand closed.
- **hgr**: Hand closed and grasping something.
- **vf00,...,vf44**: Coarse visual-field items, one for each of 25 regions. Region (0,0) is at the lower left; in figure 3-2b, the body appears at **vf41**.
- **fovf00,...,fovf33, fovb00-33, fovl00-33, fovr00-33, fovx00-33**: Visual details corresponding to each of five foveal regions (figure 3-3): front, back, left, right, and center. Each has 16 arbitrary details: 00,...,03, 10,...,13, ..., 30,...33.

Table 3.2: The primitive items.

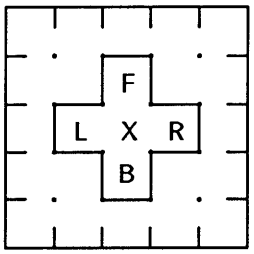


Figure 3-3: Five foveal regions (front, back, right, left, and center) in the center of the visual field provide detailed visual information.

denote arbitrary, unspecified properties. The designation is consistent, in that an object that turns On a given detail item at one foveal region turns On the corresponding detail item at any other such region to which the image shifts. (In fact, the visual detail items are implemented as a low-resolution rendition of the pixel appearance of each object.)

The coarse and detailed visual-region items are entirely unlike pixel-level information from the human retina. Rather than designating something analogous to the intensity of light at a particular spot on the human retina, the visual items convey information similar to the output of sophisticated processing in human vision, involving the detection of edges, the distinction of figure and ground regions, and the formation of $2\frac{1}{2}$-dimensional sketches of objects (Marr 1982). The end product of this processing encodes the existence, appearance, and location of objects in space (three-dimensional space in the real world, or two-dimensional space in the microworld).

Of course, the animaton's visual system does not perform comparably sophisticated computations to arrive at this encoding. Rather, the computation is trivial because of the simplicity of the microworld—in particular, the uniformity of objects' size, units of motion, and mapping onto visual-field regions. Furthermore, because the visual field enjoys a bird's-eye view, it need not recover information from a collapsed representation, as with the interpretation of depth from a real-world two-dimensional projection. Thus, the design of the animaton's visual system deliberately bypasses the difficult problems of real-world vision; the artificial system merely provides information about nearby objects that is roughly similar to real-world visual information.

Notice what is and is not hardwired about the nature of physical objects. The visual system itself may be regarded as being rigged to know much about objects, by virtue of the close correspondnence between the appearance of objects in the visual field and their actual spatial arrangement in the microworld; as just noted, this correspondence stands in for elaborate, visual domain-specific computations in real-world vision. Nonetheless, the schema mechanism, as opposed to its visual subsystem,

does not know about objects. That is, the mechanism does not know that some of its sensory primitives designate visual information, or that there are categories of inputs that correspond to different modalities; the mechanism starts out knowing nothing about what each item or action represents, or how they might be related to one another. It is up to the mechanism itself to derive the meaning of the items and actions by learning about their interrelationships.

From the mechanism's initial point of view, each primitive item and action is a featureless entity, rather like a *gensym* in the computer language LISP (Steele 1948) (a gensym is a symbol with a unique but arbitrary, automatically generated name). Whatever innate, domain-specific knowledge about objects and space the visual system uses to maintain the state of the visual inputs, this knowledge is *encapsulated* to the visual system, and is not available to the schema mechanism. (Similarly, of course, for the tactile and other domains.) This encapsulation corresponds to the working hypothesis put forth in section 2.8, that the Piagetian constructivist account is approximately correct with respect to the acquisition of knowledge by a central, general learning mechanism, although peripheral modules embody more innately specified competence than Piaget acknowledged.

3.2 Representational elements: structure and use

The schema mechanism uses three kinds of representational elements: *schemas*, *actions*, and *items*. This section describes the elements themselves. Then, section 3.4 describes the machinery for building new instance of these elements.

3.2.1 Data structure formats

Schemas

A schema has three main parts: a *context*, an *action*, and a *result*; figure 3-4 shows a schema with context p¬qr, action a, and result xy. By notational convention, a

schema's name is written in the form *context/action/result*; a negated item is preceeded with a ¬, and items conjoined in a context or result are seperated with an & (ampersand) (or, if the items have single-letter names, they are simply concatenated). Thus, the schema in figure 3-4 is p¬qr/a/xy.

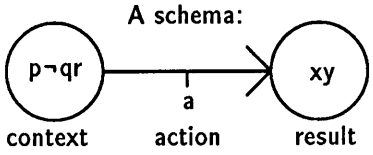


Figure 3-4: The basic components of a schema.

A schema asserts that if its action is taken when its context conditions are all satisfied, then its result conditions will obtain. The assertion is subject to some auxiliary information that the schema maintains, including a reliability factor and a set of known overriding conditions, as discussed below.

Three clarifications may circumvent some easily gained misconceptions about schemas. First, a schema makes no assertion about what happens if its action is taken when its context conditions are not all satisfied. Second, a schema is not a rule that says to take its action when its context is satisfied; the schema just says what would happen if that were done. Third, satisfying a schema's context is not a prerequisite for taking the designated action; the context pertains to the action's result, rather than to the action itself.

As noted in the introduction (section 1.2), a schema serves as a declarative, procedural, and experimental unit of representation. Declaratively, a schema asserts a prediction about what would happen if a given action were taken. Procedurally, a schema directs activity, often to pursue a designated goal. Experimentally, a schema performs certain controlled experiments to ascertain which events are relevant to one another, as explained immediately below and in section 3.4.1.

A schema's context is a set of zero or more *items* (discussed in the next section), each included in either positive or negative form; a schema's result is another such set. An item can be in the state of being On or Off. (A synthetic item can also be

in an Unknown state; see section 3.4.3.) A schema's context is *satisfied* when all the positively included items are On and all the negatively included items Off.

A schema is said to be *applicable* when its context is satisfied and no known overriding conditions obtain. To *activate* a schema is to initiate its action when the schema is applicable. A schema asserts that its activation culminates in turning On those items that are positively included in the result, and turning Off those items that are negatively included. An activated schema is said to *succeed* if its predicted results all in fact obtain, and to *fail* otherwise.

Schemas compete for activation on two bases: a schema may be activated for the sake of its own exercise, giving the mechanism a chance to test its validity and to extend or revise it; or it may be activated to help achieve a goal. When a reliable schema whose context is satisfied includes what the mechanism considers a goal in its result, the value of that goal contributes to the mechanism's incentive to activate the schema. More generally, as shown in figure 3-5, there may exist a *chain* of schemas from a current state to a goal. Such a chain has an initial schema whose context is satisfied. Its result conditions are a superset of the context conditions of the next schema in the chain, and so on to the final schema, whose results include a goal. If the chained schemas are reliable, activating each in succession should achieve the context conditions of the next one, which can then be activated in turn, until the goal is achieved.

There are two kinds of activation: explicit and implicit. To explicitly activate an applicable schema is to select it for activation and initiate its action. As a side-effect of an explicit activation, other applicable schemas, not themselves selected for activation, may have their actions initiated (if they share the same action as the schema that was explicitly activated). Such schemas are said to be implicitly activated. As documented in section 3.4.1, schemas maintain some statistics that depend on activation, but do not distinguish between implicit and explicit activation. Implicit activation also helps maintain an estimate of a given schema's *cost* of explicit

activation; its cost is the minimum (i. e. the greatest magnitude) of any negative-valued results of schemas that are implicitly activated as a side-effect of the given schema's activation.

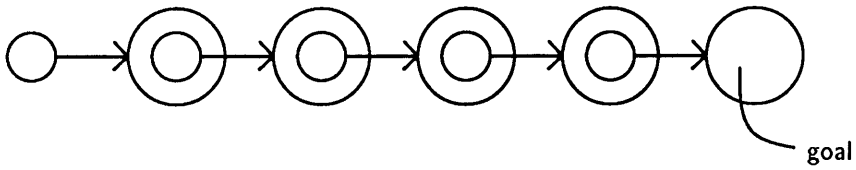


Figure 3-5: Schemas chain from a current state to a goal state.

A schema maintains various auxiliary data, documented for reference in table 3.3, and discussed in this and subsequent sections. The data include a *reliability* measure and a *correlation* measure.

- A schema's reliability is the probability with which the schema succeeds when activated. Each schema keeps track of its success rate when activated (biased toward more recent activations), which is taken to measure its reliability.

- A schema's correlation is the ratio of the probability with which a transition to the schema's result state obtains when the schema is activated to the frequency with which that transition obtains when the schema is applicable, but not activated (here again, a tabulation of actual frequency serves as a presumptive probability). Thus, a schema's correlation indicates the extent to which the result depends on the action.

Activating a schema for the sake of its result makes most sense when the schema's reliability and correlation are both high, so that the action is likely to be both sufficient and necessary.

In addition to its three main parts, each schema has two large ancillary structures, an extended context and an extended result (figure 3-6). Each has a slot for every item in the schema mechanism—not just the items appearing in that schema. (Each extended result also has a slot for certain context-like sets of items, as explained below

Correlation.	Ratio of frequency of result transition with vs. without activation.
Reliability.	Rate of successful activation.
Duration.	Average time from activation to completion of action.
Cost.	Average cost (negative-value side-effects) of activating the schema.

Table 3.3: Schema data.

in sections 3.3.1 and 3.4.1). Each such slot maintains some data about correlations between the schema and that item, and also, based on that data, specifies whether that item's being On (or being Off) overrides the schema; if so, the schema is inapplicable whenever the overriding item is On (or Off, as specified), even if the schema's context is satisfied.

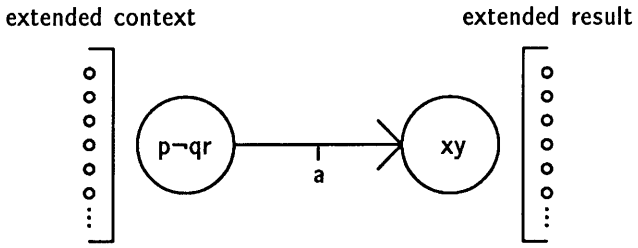


Figure 3-6: A schema has an extended context and extended result.

A schema's auxiliary data (including the content of the extended-context and extended-result slots) is subject to revision, as documented in section 3.4.1. But a schema's context, action, and result uniquely identify that schema, and do not change.

Although schemas maintain some statistical information, such as the reliability factor and correlations just mentioned, schemas are designed to provide symbolic, qualitative representations of the world. The schema mechanism endeavors to build schemas that are of near-certain reliablility; there is no attempt to make accurate or sophisticated models of the probabilities of less certain events. Quantitative reliability measures serve mainly to exclude schemas that fall far short of the ideal. Extended-context and -result correlations have a different purpose: to guide the construction of reliable schemas, as explained in section 3.4.1.

Items

An *item* is a binary state element. Each item represents some condition in the world, and has a state of On or Off to assert respectively that the condition does or does not currently obtain. An item also maintains some auxiliary data, documented for reference in table 3.4, and described in the sections to follow.

Generality.	Rate of being On rather than Off.
Accessibility.	Rate of being at the end of some chain of schemas that starts with an applicable schema.
Primitive value.	Built-in Positive or negative desirability measure.
Delegated value.	Acquired positive or negative desirability measure.

Table 3.4: Item data.

There are two kinds of items, primitive and synthetic. Primitive items are built in to the schema mechanism—they are part of its initial endowment. Each primitive item corresponds to some sensory input; for the current implementation, the inputs are as shown above in table 3.2. The state of a primitive item—that is, whether the item is On or Off—is maintained by the sensory apparatus.

Synthetic items are constructed by the mechanism itself. Each such item designates the validity conditions of a particular unreliable schema, called the item's *host schema* (figure 3-7); the synthetic item is called its host schema's *reifier*, because constructing the item treats the attainment of those conditions as a thing or state in its own right, thus reifying the validity conditions of the host schema. By notational convention, a synthetic item's name is its host schema's name, surrounded by square brackets; thus, the item in figure 3-7 is [p/a/x].

Primitive items are hardwired to sensory inputs that maintain their state. For each synthetic item, however, the schema mechanism itself must discover the conditions under which the item should be On or Off. Section 3.4.3 describes the machinery for this discovery process.

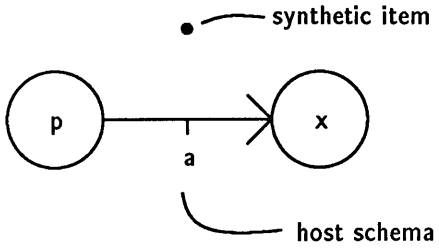


Figure 3-7: A synthetic item and the host schema it reifies.

Actions

There are two kinds of actions, primitive and composite. Primitive actions, like primitive items, are part of the schema mechanism's built-in endowment. Just as each primitive item is wired to a sensory input device, each primitive action is wired to a device that carries out a particular motor action. Table 3.1 above documents the primitive actions used in the current implementation. Initiating a primitive action (by activating a schema which has that action) initiates the corresponding motor device.

A composite action is defined with respect to some *goal state*. Like a schema's context or result, a composite action's goal state is a set of (positively or negatively included) items. A composite action is essentially a subroutine: it is the action of achieving its goal state, by whatever means available. The means are given by chains of schemas that lead to the goal state from various other states (figure 3-8); such schemas are said to serve as *components* of the composite action. (A given schema may be a component of any number of composite actions, or of none at all.)

Each composite action has an associated *controller*. Just as a schema's extended context and extended result have a slot for every extant item, a composite action's controller has a slot for every schema. Each slot contains data about whether the schema lies along some chain to the goal state, and, if so, the *proximity* to the goal that will be achieved if the schema is activated. Proximity is a inversely proportionate to the expected time to reach the goal state, derived from the expected activation time of the schemas in the relevant chain; proximity is also proportionate to those

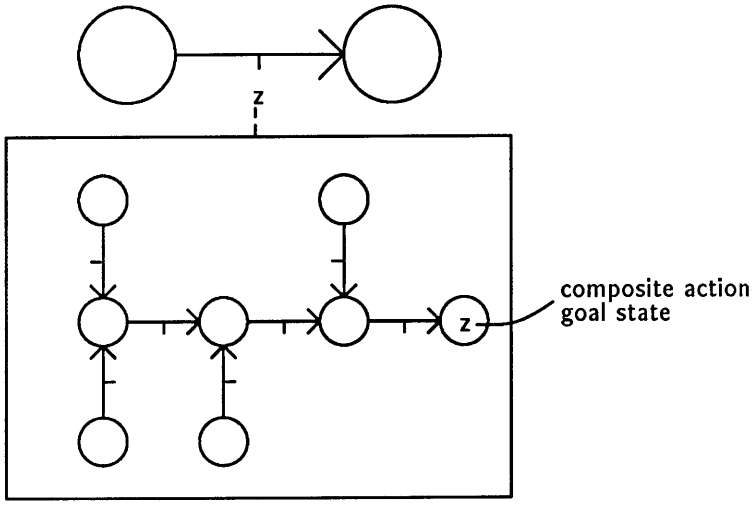


Figure 3-8: A schema whose composite action has goal state z.

schemas' reliability, and inversely proportionate to their cost of activation.

Initiating a composite action (again, by activating a schema which has that action) causes the controller to identify a schema (among those currently applicable) with greatest proximity to that action's goal state; that schema is then activated. This process iterates until either the goal state obtains, or the action fails. The action is considered to have failed if either it has greatly exceeded its expected execution duration (a statistic that each action maintains, based on prior performance) without making much progress—that is, without much increase in proximity to the goal; or if a brief interval passes during which no component schema is applicable.

The iterative selection of the most proximal component permits a kind of *opportunism* (Agre 1988) in composite action execution: control may pass from one chain of schemas to another, if a more proximal schema along a different chain unexpectedly becomes applicable. The controller does not notice this shift as such; the shift is just a consequence of always selecting next the most proximal applicable component.[2]

[2]Recent work on *universal plans* (Schoppers 1987) describes a planning scheme similar in this regard to composite-action control.

3.2.2 Control

Schemas compete for activation. At top level, the schema mechanism selects a schema for activation at each next time unit (in the current, discrete-time implementation; a continuous-time version might perform this selection at regular, frequent intervals—perhaps a few times per second). In the present implementation, only one schema is activated at a time. However, the activation of a schema that has a composite action entails the immediate activation of some component schema; thus, the mechanism supports nested activations, but not parallel activations.

The top-level selection process activates the applicable schema that asserts the greatest activation importance. The importance of activating a given schema is based on two criteria: explicit goal-pursuit, and exploration. The goal-pursuit criterion contributes to a schema's importance to the extent that the schema's activation helps chain to some item that is of positive value; the exploration criterion boosts the importance of a schema to promote its activation for the sake of what might be learned by that activation. (Of course, the exploration criterion also serves a kind of goal, the goal of acquiring knowledge; but explicit goal-pursuit refers to achieving some explicitly represented state for the sake of its explicitly represented value.)

To strike a balance, the mechanism alternates between emphasizing goal-pursuit critera for a time, then emphasizing exploration criteria. Also, rather than merely selecting the schema asserting the highest activation value, the mechanism chooses at random among those schemas with value within a certain factor of the maximum value then asserted; each such schema's probability of selection is proportionate to its share of the total such value. This process prevents a small advantage from disproportionately favoring some schemas' activation over others nearly as good; but limiting the selection to schemas close to the maximum value prevents large differences from being passed over.

A new activation selection occurs at each time unit. Even if a chain of schemas leading to some goal is still in progress, each next link in the chain must compete

for activation. Thus, as with the execution of a composite action, control may shift to an unexpected, new, better path to the same goal. Top-level selection carries this opportunism one step further: here, control may even shift to a chain that leads instead to a different, more important goal.

The mechanism also permits an executing composite action to be interrupted. A schema with a composite action, of course, may take arbitrarily long to complete, depending on the length of the chain of schemas used to reach the action's goal state, and on the duration of the activation of each schema in the chain. Even if a schema with a composite action is in progress, the cycle of schema selection continues at each next time unit. If the pending schema is re-selected, its composite action proceeds to select and activate the next component schema (which may recursively invoke yet another composite action, etc). If, on the other hand, a schema other than the pending schema is selected, the pending schema is *aborted*, its composite action terminated prematurely. A pending schema is given enhanced importance for selection, so it will likely be re-selected until its completion, unless some far more important possibility arises. Hence, there is a kind of focus of attention that deters wild thrashing from one never-completed action to another, while still allowing interruption for a good enough reason.

Explicit goal-pursuit

Three kinds of value may be associated with an item: primitive, instrumental, or delegated value. Each is a positive or negative quantity associated with an item or set of (possibly negated) items.

- Primitive value is associated with certain primitive items. In biological systems, for example, representations of events beneficial to the organism or species (e. g. taste of food, sexual stimulation) ought to have built-in positive value, and designations of deleterious events (hunger, pain) should be negative. Correspondingly, the present schema mechanism implementation assigns positive

primitive value to certain tastes, and negative primtive value to certain tactile sensations ("sharpness").

- A state is of instrumental value if its attainment is a specific prerequisite for achieving something else of value. When the schema mechanism activates a schema as a link in some chain to a positively valued state, then that schema's result (or rather, the part of it that include's the next link's context) is said to have instrumental value.

  Instrumental value, unlike primitive (and delegated) value, is transient rather than persistent. As the state of the world changes, a given state may lie along a chain from the current state to a goal at one moment but not the next.

- Delegated value combines aspects of primitive and instrumental value. As with instrumental value, an item's delegated value derives from other things of value that that item helps achieve. But delegated value, like primitive value, is persistent. Delegated value is assigned as follows.

  At each time unit, the schema mechanism computes the value explicitly *accessible* from the current state—that is, the maximum value of any items that can be reached by a reliable chain of schemas starting with an applicable schema. (Section 3.3.1 discusses the machinery for identifying such chains efficiently.) The mechanism also keeps track of the average accessible value over an extended period of time.

  For each item, the mechanism keeps track of the average accessible value when the item is On, compared to when the item is Off. If the accessible value when On tends to exceed the unconditional accessible value, the item receives positive delegated value; if the accessible value when On is less than the unconditional average, the item receives negative delegated value. (Each item also keeps track how much of the time it is accessible; this information determines composite action formation, as explained in section 3.4.2.)

For purposes of the value-delegation comparison, accessible items of zero value count as having slight positive value, thus delegating more value to states that tend to offer a greater variety of accessible options.

Delegated and instrumental value serve complementary functions: value is delegated to states that generally tend to facilitate other things of value; instrumental value is for states that currently facilitate other things of value, and by a specifically forseen chain of events. Thus, delegated value may be said to be *strategic* whereas instrumental value is *tactical*.

An item does not (and should not) receive delegated value just by virtue of receiving frequent instrumental value. The state of, say, being in a standing position is often of instrumental value (as a prerequisite for walking somewhere, for instance); but it would be foolish (under most circumstances) to make a point of remaining standing just in case a contingency arose that required walking somewhere. If a frequently instrumental state is itself readily accessible, then the things it facilitates are accessible even before the instrumental state itself has been achieved. Consequently, the value accessible when the state obtains does not exceed the value accessible when it does not; therefore, no value is delegated to that state.

Delegated value arises, and is useful, when a state is that is *not* readily accessible facilitates other things of value under circumstances that are likely to occur while the state still obtains. To an infant, for example, the presence of a parent may receive delegated value, even when there is no specific goal for which the infant needs the parent at the moment, because such a need arises often enough that it is good to have the parent nearby just in case. When a given state does not facilitate a specific goal at the moment, there is no chain of schemas to impart instrumental value to that state; consequently, delegated value is needed to promote the strategic pursuit of that state.

Furthermore, to designate goals only in terms of the mechanism's primitive lexicon would be as burdensome as having to represent all predictions and plans at that

bottommost level of abstraction. The delegation machinery allows higher-level, constructed concepts to acquire lasting value as well. At the same time, this machinery must ensure consistency between original and delegated value, so that pursuing the delegated value will continue to promote the top-level goals that the preassigned, primitive values are designed to coincide with. Arbitrary, unconstrained revision of the system's goals would be disastrous.

In particular, the mechanism must avert the danger of positive feedback in value delegation when two or more states are of mutual strategic value. Depending on how much value is delegated, each state's increase in delegated value could cause a similar increase in the other's, and so on without bound. To dampen such feedback, the value delegated to an item is only *half* of the difference between the unconditional average attainable value, and the value attainable when the item is On.

Of course, despite such safeguards, the delegation of value not only facilitates prior goals, but also changes the goal structure for the future. Thus changes that locally do a better job of pursuing what is already sought may eventually culminate in additional goals far removed from what was originally pursued. This is not unlike biological evolution, in which the implicit goal of perpetuating what is aready there is most effectively achieved by making slight improvements, thereby perpetuating inexact copies that are more robust than the original design; eventually, what is being perpetuated may bear little resemblance to its ancestors. (Indeed, biological cognitive systems' built-in values for certain primitive sensations may be regarded as having been delegated by evolution to various explicitly represented states and processes—eating, copulating, etc—whose attainment strategically facilitates the implicit goal of perpetuating the genome.)

Numeric values are used to adjudicate the selection of a schema for activation. Yet this selection makes a qualitative decision: which of several eligible schemas to prefer. Basing this qualitative choice on a quantitative measure may seem inappropriate, particularly in light of the schema mechanism's presumption in favor of symbolic,

nonnumeric representations. But I maintain that numeric values are appropriate to the selection task: given n explicit goals, n numeric values allow the derivation of n^2 preferences that might arise in pairwise choices between goals; they also permit the derivation of exponentially many possible choices between sets of multiple goals.[3] Just as monetary exchange, as opposed to bartering, prevents having to trade one commodity directly for a preferred one, using quantitative values prevents having to make a direct qualitative comparison of each pair of results that the system can choose between. Still, delegated value derives from facilitating the accessibility of other things of value, a qualititative relation.

Although the current schema mechanism implementation includes primitive, instrumental, and delegated value, the mechanism's acquired skills to date are so unsophisticated that primitive and delegated value have little effect on the mechanism's activity; there simply are not any interesting things of value that the mechanism knows how to achieve. The mechanism's activity is influenced instead by instrumental value (in that the initiation of a composite action involves chaining to its goal state), and by exploration value, described below. Thus, in particular, the utility of delegated value remains to be demonstrated.

Exploration value

The schema mechanism maintains a cyclic balance between emphasizing goal-directed value and exploration value. The emphasis is achieved by changing the weights of the relative contributions of these components to the importance asserted by each schema. Goal-directed value is emphasized most of the time; but a significant part of the time, goal-directed value is diluted so that only very important goals take precedence over exploration criteria.

[3]The constraints of nonreflexivity, asymmetry, and transitivity imposed by numeric values ought to be respected by a preference system: it makes no sense to prefer A to itself (reflexivity), or to prefer A to B and B to A (symmetry), or to prefer A to B and B to C but not A to C (nontransitivity), given those pairwise choices all in the same situation.

A schema's exploratory value is calculated to promote useful learning by the schema mechanism, rather than to pursue explicitly represented goals. Two chief components of exploration value are *hysteresis* and *habituation*: a recently activated schema is favored for activation (hysteresis), providing a kind of focus of attention that promotes repetition of a small number of schemas; but a schema that has recently been activated many times becomes partly suppressed (habituation), preventing a small number of schemas from persistently dominating the mechanism's activity.

A schema records its usage rate—its frequency of being selected for activation. Other factors being equal, a more frequently used schema is favored for selection over a less used schema. This factor mitigates possible redundancy among structures. Suppose there is some set of nearly-identical schemas—schemas which differ, say, by including different infrequently arising context conditions that only slightly affect reliability; or schemas that use different, effectively synonomous items to designate the same condition (see section 3.4.3). If one of these schemas, by chance, is used slightly more than the others, it accumulates greater usage—which, in turn, promotes its further usage (relative to those others), further increasing its value relative to those others. This deliberate instability carves out a situational niche in which only a few schemas, among all the similar ones, will dominate.[4] The instability is controlled by subordinating the usage factor to other components of a schema's value.

Another component of exploration value is designed to share activation among different actions. Without such a component, actions that appear in relatively many schemas tend to be initiated more often than others, which in turn promotes the construction of more schemas for those actions, leading to instability. To circument this problem, schemas with underrepresented actions receive enhanced exploration value. Similarly, a component of exploration value promotes underrepresented *levels* of actions, where a structure's level is defined as follows: primitive items and actions are of level zero; any structure defined in terms of other structures is of one greater

[4]This trick also appears in (Holland et al 1986).

level than the maximum of those structres' levels.

3.3 Architecture

To be a viable partial theory of theory of human intelligence, or a viable model for artificial intelligence, the schema mechanism must not be intractibly inefficient. A mechanism's efficiency depends in part on its presumed architecture; the schema mechanism's dual citizenship, in psychology and A.I., requires two such substrates, which this section presents: a loosely envisioned neural architecture, and the actual architecture of MARCSYST I, the existing software implementation of the schema mechanism.

3.3.1 Neural architecture

The schema mechanism is intended to explain aspects of human learning. The mechanism's design must therefore respect the constraints of neurophysiological plausibility; there ought to be a conceivable neural implementation of the schema mechanism, one that does not violate what is known about the human brain.

This section outlines a neurally plausible architecture for the schema mechanism. The outline is coarse—it is nothing more than a characterization of the sheer number of computational units involved, the required connectivity among them, and the time complexity of the required computations. I argue that these are plausibly within human-brainlike bounds. In contrast with connectionist (McClelland et al 1986) or neural-net models (Anderson and Rosenfeld 1988), the proposed architecture makes no attempt to indicate how specific functions performed by the schema mechanism might be implemented by vaguely neuron-like computational elements, such as linear-threshold units (Minsky and Papert 1969).

The presumed architecture supports one million computational units that are exhaustively cross-connected; that is, a seperate physical pathway exists between each

pair of units. (Some units, corresponding to sensorimotor primitives, also connect to peripheral modules.) The connections transmit data, both numeric and symbolic, the latter consisting only of a small number of discrete tokens (perhaps a few dozen); unlike, say, the letters of the alphabet, these tokens do not combine productively to form long composite structures (such as words or sentences). The connection points between units, and the units themselves, can each store some data, again consisting of numeric quantities and a small number of tokens. The connections between units, and the units themselves, operate in parallel.

Each unit and each connection point performs some simple, constant-time computations perhaps a few times per second; the results of the computation can affect the stored values, and can be output along the connection lines. The computation at each connection point is a function of the stored values there, and of data input from the two units that connect there. The computation at each unit is a symmetric function of the connection-line inputs to the unit, and of the stored values at the unit and at its connection points; the function might be the conjunction or disjunction of a binary value from each input, or to compute the sum, average, or maximum of a numeric value from each input, or from those inputs flagged by a particular token stored at or input to the corresponding connection point (figure 3-9) (such computations can be performed in time logarithmic to the number of inputs). A unit can output numbers and tokens along its connection lines, as well as receiving such data. There is also centrally coordinated global communication, such as broadcasting a message to every schema, every action, or every item.

The above assumptions are similar to those of standard connectionist architectures, except for the presumption here of exhuastive cross-connectivity, and for the absence here of any attempt to reduce the computation performed by each unit to the behavior of neuron-like elements.

Let us presume, for the sake of this analysis, that on the order of a few million schemas, items, and actions might suffice to implement adult-level intelligence. (At

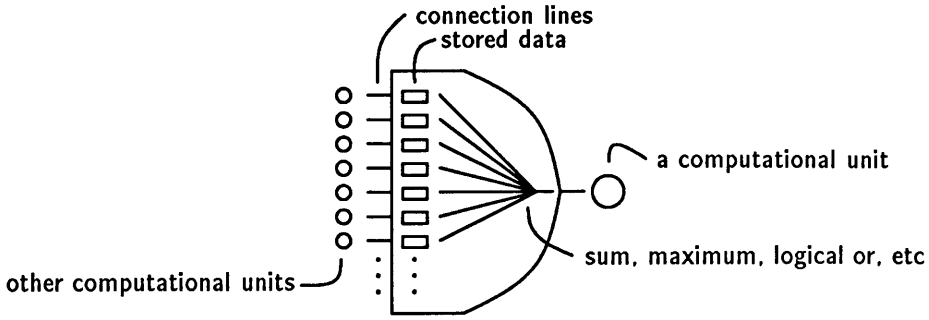


Figure 3-9: Each unit connects exhaustively to the others.

least on the order of a million cognitive cognitive units of some sort must be needed for human intelligence; that is the smallest round number that isn't clearly wrong. For comparison, the average vocabulary of an English-speaking adult is a few tens of thousands of words; and presumably there are many non-linguistic concepts for each one named by a word.) Assuming that each schema, action, and item is implemented by one of the computational units just discussed, I argue that something like the schema mechanism, with at least one million representational units, could fit in the neocortex of the human brain.

I have little to say about the computational units themselves, except that the 10,000 or so neurons available per unit are, intuitively, more than enough for the assigned computations. The more difficult matter is to account for: the cross-connectivity that supports extended contexts and results, and action controllers; identifying chains of schemas from current states to goals, and noting the accessibility of goals from current states; and defining new schemas in terms of actions and items.

The postulated crossbar connecting every unit to every other supports all four of these capabilities. (In fact, a slightly smaller crossbar, connecting all schemas to all items, and all actions to all schemas, would suffice; but the order of size of the required structures would not be much less.) The remainder of this section first describes how the crossbar supports these capabilities, and then argues for the neurophysiological plausibility of the crossbar itself.

Extended contexts, extended results, and action controllers

The exhaustive crossbar straightforwardly supports extended contexts, extended results, and action controllers. Each slot in an extended context or result, for example, is a connection point between a schema and an item; the slot computes and stores correlations between a schema's activation and the state of some item (as detailed in 3.4.1). Each action controller slot connects a composite action and a schema. The connection point stores proximity information (described below), and receives data from the connecting schema as to whether that schema is currently applicable. The lines connecting to a given composite action collectively compute the maximum stored proximity among slots for schemas that are currently applicable.

Chaining

Identifying chains of schemas serves two functions: it propagates instrumental value to intermediate states between a current state and a goal; and it is part of the assessment of schemas' proximity to a composite action's goal state.

Finding a chain that leads to a particular item works by a parallel *broadcast*. The item sends a message to each schema asserting that that item is a goal; the item's value is also transmitted. A schema ignores this message unless the schema includes the item in its result; each connection point between a schema and item includes that information, stored when the schema is created (see below). If the schema does include the item that sent the message, and if the schema is reliable, then the schema broadcasts a message in turn to its context, making its context a goal; the value information is broadcast as well. Also broadcast is a proximity measure that takes account of the schema's reliability, expected duration of activation, and cost.

This process iterates, tracing backward along various chains in parallel, each schema along the way storing its proximity to the original goal, and the goal's value. The proximity measures computed by each link of the chain combine as the broadcast proceeds, diminishing the proximity at each step. When two or more items send

converging messages to the same schema, the largest proximity measure is stored and propagated further; the others are ignored. The backwards iteration proceeds to some maximum depth of search; the time required is propotionate to this depth.

A schema's context designate a conjunction of items, rather than just one item. Broadcasting a message to each item individually would not work, since arbitrarily many schemas might do so simultaneously. It is necessary to distinguish, say, between broadcasting to the items **a** and **b** from a schema whose context includes both, and broadcasting to those items from two distinct schemas, one with just **a** in its context, the other with just **b**. A chaining schema's result must include the entire context of the next schema in the chain; hence, in the first case, a schema whose result included only **a** or only **b** would not be a link in the chain, but it would be in the second case.

This problem is solved by broadcasting to the context set as a whole. As mentioned above in section 3.2.1, certain context conjunctions—specifically, those that are contexts of schemas of nonnegligible reliability—have extended result slots, just as individual items have. As with the slots for individual items, each such slot is set up, when a schema is created, to store a bit that says whether the schema's result items include all of that slot's conjunct items; a second bit indicates whether the result negates an included item.

When chaining is used to propagate instrumental value to help find the next schema to activate, the process proceeds not just from one goal item, but simultaneously from all items that have positive primitive or delegated value. When two or more goals' broadcasts converge to the same schema, the one with greatest value is stored and propagated; among converging broadcasts that have the same value, the one with greatest proximity is used, as above. A schema whose context is satisfied does not broadcast further, but rather competes for activation based in part of the instrumental value received from the broadcast. There is no need for the mechanism to keep track of which goal a particular broadcast is in aid of; keeping track of value and proximity provides the information needed by the selection for activation.

Chaining also serves a distinct but related purpose: determining each schema's proximity to a given composite action's goal, so that that information can be recorded in the slots of the action's controller. The broadcast process for this purpose proceeds as above, but for just one composite action's goal at a time, since in this case it is necessary to know what goal each schema is helping to chain to; broadcasts converging to a given schema from multiple actions' goal states would not all be able to propagate further back in the chain. (Of course, the mechanism could be extended to support simultaneous broadcasts for some fixed number of composite actions, but not for arbitrarily many, given the assumption that each computational unit can receive and store just a small number of distinct tokens.) Section 3.4.2 elaborates on the use of chaining information by composite action controllers.

Finally, chaining is used to determine what states are accessible from the current state. (Accessibility is discussed above in connection with delegated value in section 3.2.2, and also below in section 3.4.2, in connection with composite action construction.) To determine accessibility, the mechanism broadcasts messages forward along chains of reliable schemas (in contrast with propagating instrumental value and finding goal proximity, for which the broadcast goes backwards). To begin, each schema that is currently applicable broadcasts a message via its extended result to the items and conjunctions that are included in the schema's result. Any schema that has such an item or conjunction as its context braodcasts in turn via its own extended result, and so on, to some maximum depth of search. Any item or conjunction of items that receives a message by this process is currently accessible.

Composition

By the above architectural assumptions, the computational units that implement the schema mechanism do not support productive composition of symbolic tokens, as letters compose to form arbitrarily many words or sentences. Yet the schema mechanism presumes the ability to compose schemas from actions and items in just

such a fashion. The exhaustive crossbar between schemas and items, and between schemas and actions, reconciles these two assumptions.

A schema designates its context and result simply by storing, at each connection point to an included item, the data that that item is included; whether it is included in the context, result, or both; and whether each such inclusion is positive or negative. Similarly, the schema designates its action at the connection point between the schema and the action.

Creating a schema with a specified context, action, and result has two steps: checking whether such a schema already exists (in which case it is not duplicated); and allocating an unused computational unit, setting up the connection-point designations of the context, action, and result, and also the extended-result data designating items and some sets of items that are included in the result proper.

- To check if a specified schema already exists, the mechanism broadcasts to the action, and to each included item, its designation in the specified schema. The items and actions transmit this data along their connection lines to all schemas. If any schema finds a match, at all its item and action connection points, between the status of those items and actions for that schema, and their status according to the broadcast, then the specified schema already exists.

- Each reliable schema's context, if it is a set of more than one item, is allocated a computational unit that connects to every item; an item's membership in such a set is recorded at the set's connection point to that item, along with a designation of positive or negative inclusion. The unit also connects to every schema, as part of each schema's extended result.

  - When a unit is allocated for a new such set, each schema's extended result must record, at the connection point to the new set, whether the new set is included in the schema's result. To this end, every schema is informed of the number of items in the new set. For each schema, each connection

point between the schema and an item stores whether the item belongs to the schema's result (and, if so, its sign), and is told by the item whether the item is included in the new set (and, if so, its sign). On this basis, the schema counts how many of its result items also belong to the new set, with the appropriate sign. If that number is greater than or equal to the size of the new set, the schema's result includes the new set, and this fact is recorded at the connection point between the schema and the new set.

- When a new schema is built, its extended result must record, at each connection point, whether the item or set of items at that connection point is included in the schema's result. For individual items, that information provides the very specification of the schema's result, as noted above. For sets of items, each unit designating such a set determines (by a process similar to the one just described) whether it includes all the items in the new schema's result. If so, the unit thus informs the new schema via the extended-result connection line between that set and the schema; that connection point then records the information.

Neural crossbars

An exhaustive crossbar between units of one type and units of another might be implemented as in figure 3-10a. Each unit of the first type connects to the input side of a *fanout element*; the output side of the fanout element has a separate connection for every unit of the second type.

A row of adjacent fanout elements, shown in figure 3-10b, forms a sheet that extends from all elements of the first type to all elements of the second. The connection is accomplished by taking a similar sheet of fanin elements (figure 3-11a) rotated ninety degrees from the fanout sheet and facing in the opposite direction, and placing the two sheets together (figure 3-11b). A similar pair of sheets implements exhaustive cross-communication in the other direction. Perhaps several such pairs of sheets

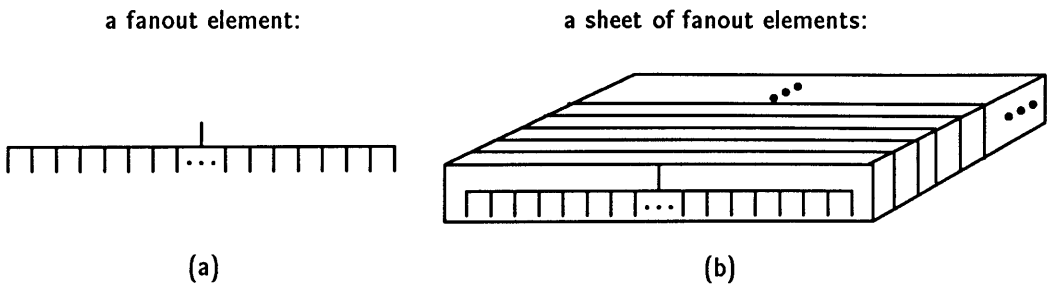


Figure 3-10: Each fanout element connects one input unit to all output units.

would be needed to implement various different crossbar computations.

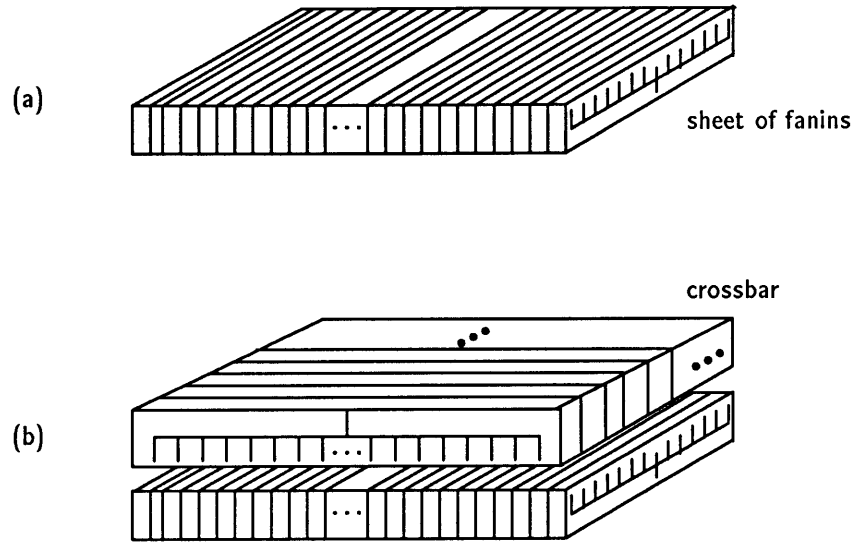


Figure 3-11: A fanout sheet atop a fanin sheet forms an exhaustive crossbar.

It remains to consider the implementation of the fanin and fanout elements themselves. If there are one million units being cross-connected, then the branching factor for the fanout and fanin elements far exceeds the 1,000-10,000 factor for neurons (see e. g. Crick and Asanuma 1986 for this and other neurophysiological data cited just below). However, each element can be constructed as a two-stage device comprised of elements that have a neurally plausible branching factor, as shown in figure 3-12.

The total number of neurons required is about 10^{10}, which lies within the bounds set by the size of the human neocortex. Furthermore, almost all the neurons comprising the crossbar are second-stage neurons, each of which needs to reach only a small

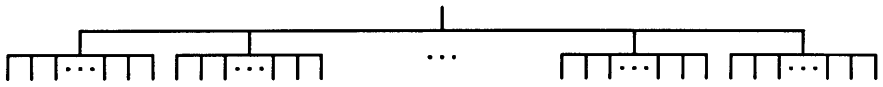


Figure 3-12: A two-stage fanout element.

fraction (1/1,000) of the target volume. Consequently, although the crossbar crossbar implements exhaustive cross-connectivity with regard to the million computational units, it exhibits almost exclusively local connectivity with respect to individual neurons. Only the first stage neurons—one crossbar neuron in a thousand—makes a distant connection. This locality accords with observations of the wiring of the human cortex.

3.3.2 Digital implementation architecture

MARCSYST I, a computer program that is the current digital implementation of the schema mechanism, runs on a Connection Machine (CM) (Hillis 1985), using a dedicated Lisp machine (Symbolics 3650) as front end. The CM's salient architectural features are as follows:

- There are up to 64,000 physical processors that operate in parallel. The machine portion available for this research had 4,000 processors. Each processor has 64,000 bits of memory.
- The CM is a SIMD machine (Single Instruction, Multiple Data streams), which means that all processors execute the same instruction at once, each on its own data.
- Some instructions operate locally to each processor, affected by or affecting that processor's data alone. Other instructions act globally, computing, for example, the sum or maximum of some specified processors' values for a given numeric datum, or the logical conjunction or disjunction of some specified processors' values for some logical datum. Some global instructions act in the other direction, sending a value to all processors.

- Finally, there is a class of *communications* instructions, which send messages from one or more source procesors to one or more target processors per source processor. A source processor may designate its target by address, or by coordinates in an n-dimensional grid into which virtual processors can be organized. A message may be sent to an entire row (or even hyperplane) of such a grid at once.

Most of MARCSYST I is written in *LISP(Thinking Machines 1988), a parallel extension of LISP (Steele 1984). Some inner-loop code is written in ParIS (Thinking Machines 1988), an assembly-language-like instruction set for the CM.

MARCSYST I allocates virtual processors for each schema, action, and item, and for each connection point in the schemas-items crossbar and the actions-schemas crossbar; the connection-point virtual processors are organized into two-dimensional grids for purposes of the communications instructions. The basic compuations performed at each time unit by each schema, action, item, and by each crossbar connection point, are cycled through in sequence, all instances of the same kind of structure performing their computation in together. The crossbar connectivity is simulated by using CM communications instructions to transmit data from one kind of structure to another.

Each of MARCSYST I's 4,096 CM processors can designate one schema or composite action; these are large structures because of the extended contexts and results, and controllers. Ninety percent of the processors are reserved for schemas, the remainder for composite actions. Available memory for schemas is the limiting factor in the implementation's performance.

3.4 Constructing and revising schemas, actions, and items

Above all, the design of the schema mechanism reflects its need to learn, to build its own structures for its own use, to come to represent the world in a way that is both practical and informative. As mentioned in chapter 1, the processes of constructing new schemas, actions, and items correspond roughly and respectively to learning by induction, abstraction, and conceptual invention. Schemas express discoveries about the relations among existing actions and items; composite actions designate the achievement of particular goals, abstracting above the details of how those goals are reached, permitting the goal itself to be seen as a cause of further results; and, especially, synthetic items represent aspects of the state of the world of which (some) previously represented states were mere manifestations, such that there the new concept may be inexpressible, as a practical matter, as any fixed function of its manifestations.

3.4.1 Marginal attribution: spinning off new schemas

Piagetian development is rife with examples of generalizations and specializations of schemas. These examples involve the discovery of consequences of actions, and the discovery of the conditions that these consequences depend on. The schema mechanism tries to capture this sort of discovery with the process of marginal attribution, which constructs new schemas.

The chicken-and-egg problem: which comes first, the context or result?

As noted in section 1.2, the task of constructing reliable schemas poses a chicken-and-egg problem. Even if some result very reliably follows a given action under certain conditions, the result may follow arbitrarily rarely in general. For example, the action of moving my hand incrementally backward reliably causes a tactile sensation

on my chin—provided that the hand was just in front of the chin before that action occurred. In general, though, that particular result seldom follows; and similarly for other results that the same action has reliably under various other conditions.

Not only might each of an action's many reliable results (subject to the right conditions) follow seldom in general; in addition, when such a result does follow, it may be accompanied by dozens, perhaps thousands, of entirely coincidental state transitions at various levels of description. Thus, identifying an action's result as such, before knowing the corresponding context conditions, is not a mere matter of noting that the result typically, or occasionally, follows the action.

Thus, the chicken-and-egg problem: a result does not look like one except with respect to the appropriate context. Until the context is known, finding the result is difficult; but finding the context is impossible without knowning what result it is the context for.

Another, related chicken-and-egg problem arises even after a result has somehow been identified, if a conjunction of several conditions is required for the result to follow the action; if less than all of those conditions is satisfied, the result does not follow. Consequently, the relevance of any one of those conditions is difficult to discern until the others have been identified—only when the last conjunct is added does the schema become reliable. More generally, if the required context is a disjunction of many conjunctions, the same problem arises for each of the conjunctions.

There is a brute-force solution to the conjunctive-context problem: express all possible conjunctions of items, and for each one, tabulate the probability of the result following the action when that conjunction is satisfied. In fact, this approach solves the context-result problem too, if all context-result pairs are similarly tabulated for each action. However, these approaches are clearly intractible; the number of expressible context conjunctions, or of context-result pairs, is exponential in the number of items. If the conjunctions are limited in size to k conjuncts, only polynomially many (n^k, where n is the number of items) need be monitored, as (Littlestone 1987) points

out; still, if n is on the order of a million or more, even a limit of, say, five conjuncts puts n^k vastly beyond the number of neurons or synapses in the brain.

The combinatorial problem would be eased if there were *a priori* constraints on which items might be relevant to which schemas or actions. But it is impossible, in a constructivist learning mechanism, to supply such constraints. Certainly, among the primitive sensory items and motor actions, there are no natural partitions—hand motions, for example, can have tactile, visual, and auditory effects; further, the effects might be contingent on conditions in any of those domains. Similarly, vocal actions can have diverse effects—especially via people as intermediary agents (and the discovery of such effects needn't entail awareness of that agency). As for constructed items and actions, as opposed to primitive ones, it is even harder to impose *a priori* constraints on relations among elements when the elements are not *a priori* themselves, when those elements derive from primitive underpinnings which, as just argued, are also without such constraints. (The need to be able learn without *a priori* relevance constraints does not, however, preclude the possibility of other machinery that uses acquired constraints on relevance enable more powerful forms of learning.)

Fortunately, both chicken-and-egg problems—the context-result problem, and conjunctive context problem—have a solution that does not presuppose *a priori* constraints on relevance. This solution is the marginal attribution algorithm, which requires only the brute force of an crossbar between schemas and items. For n schemas and items, only n^2 computational units are needed for an exhaustive crossbar (rather than an exponential or larger-order polynomial number); as argued in section 3.3.1, a crossbar of that size is neurophysiologically plausible.

Marginal attribution: the basic machinery

Marginal attribution works as follows. Initially, for each primitive action, the schema mechanism has a *bare* schema, with empty context and result (e. g. at left in figure 3-

13). Similarly, when a new composite action is defined, the mechanism constructs a bare schema that uses that action. Bare schemas make no assertion in their own right, but they are the points of departure for the discovery of the effects of their respective actions. Marginal attribution initially identifies relevant but unreliable effects of a schema's activation, then searches for context conditions with respect to which those effects obtain more reliably. A series of intermediate, unreliable schemas serves as a scaffold for the construction of an eventual, reliable schema (when the process succeeds). Each schema keeps track of its own reliability, so the intermediate constructs are not mistaken for reliable assertions.

A bare schema's extended result discovers effects of the schema's action. The discovery proceeds by way of two statistics maintained by each extended result slot. One statistic, the positive-transition correlation, is the ratio of the probability of the slot's item turning On when the schema's action has just been taken to the probability of its turning On when the schema's action is not being taken. The other statistic, the negative-transition correlation, is a similar ratio, but with respect to turning Off instead of On. These statistics are tabulated over a number of trials in which the action is taken, and trials in which it is not; the more trials there have been, and the more discrepancy there is between the two probabilities, the sooner the machinery will detect the difference (see section 3.4.1). The sampling is weighted toward the most recent trials.

Since the machinery seeks transitions to the result state, a trial for which the result was already satisfied before the action was taken does not count as a positive-transition trial; and one for which the result was already unsatisfied does not count as a negative-transition trial. Arguably, the mechanism should also look for a result that is kept constant by an action, when it would otherwise have changed. The present implementation does not do this—looking for transitions is more important, and memory is limited—but it could easily be extended to maintain such statistics as well.

If some extended result slot for a given schema shows that an item is significantly more likely to turn On (or Off) when the schema's action is taken, that item is deemed *relevant* to the action. A relevant item is a candidate for positive inclusion (if it turns On) or negative inclusion (if Off) in a schema that is said to *spin off* from the given schema. A spinoff schema copies the given schema's context, action, and result, but with the designated item included in the copy's result (or context, as discussed below). For example, in figure 3-13, the extended result of the schema /a/ discovers the relevance of items x, y, and z. Correspondingly, the schemas /a/x, /a/y, and /a/z spin off from the bare schema /a/.

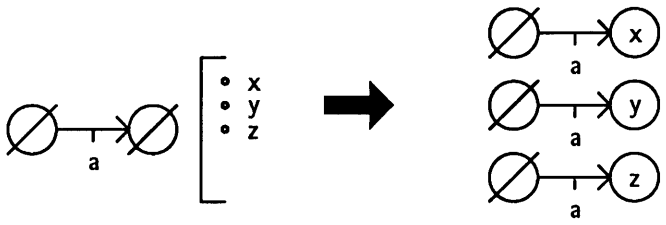


Figure 3-13: A bare schema discovers some results of an action, spawning spinoff schemas.

A relevant result need not follow an action reliably. In fact, its occurrence following the action may be arbitrarily unlikely, provided that it is even less likely in the action's absence. The relevance criterion uses the schema to specifiy a controlled experiment, comparing what happens with activation to what happens without (the control). Subtle but significant statistical differences then serve to identify a relevant but arbitrarily unreliable result, solving the context-result chicken-and-egg problem.

The machinery's sensitivity to relevant results is amplified by an embellishment of marginal attribution: considering only *unexplained* transitions, that is, transitions that were not predicted by the activation of a reliable schema; when an item's (or conjunction's) state transition is explained, schemas' extended results do not update their slots for that item (or conjunction). Consequently, a given schema whose activation is a less frequent cause of some result needn't compete with other, more frequent causes, once those causes have been identified; for the result to be deemed relevant

to the given schema, that schema need only bring about the result more often than its other *unexplained* occurrences.

Once an relevant result has been so designated by the spawning of a spinoff schema, the induction machinery looks for context conditions with respect to which the result follows more reliably than in general; the spinoff schema's extended-context slots maintain statistics that identify such conditions. In particular, each extended-context slot records the ratio of the probability that the schema will succeed (i. e. , that its result will obtain) if the schema is activated when the slot's item is On, to the probability of success if that item is Off when the schema is activated. As with extended-result statistics, these are weighted toward more recent trials; also, the more trials there have been, and the greater the difference between the two probabilities, the sooner the machinery can detect the difference.

As mentioned previously, for purposes of the statistics maintained by marginal attribution, a schema is considered to have been activated (*implicitly* activated) any time its action is taken when its context is satisfied, even if that schema was not selected for activation (*explicit* activation). Thus, many schemas' extended-context data may be updated at once. (In fact, all activation-dependent schema data equates implicit and explicit activation; and the explainedness of a state transition, invoked just above, is also with respect to either kind of activation.) Since implicit and explicit activation both require a schema's context to have been satisfied, each extended-context slot measures the corresponding item's contribution *together with* the context proper to the schema's reliability.

If the first (or second) of the extended-context probabilities is significantly higher than the other, the item is deemed a relevant condition for the schema's success, and is a candidate for positive inclusion (if the schema is more reliable with it On) or negative inclusion (more reliable when Off) in the context of a spinoff schema. In figure 3-14, the extended context of /a/x discovers that p boosts the schema's reliability, spinning off p/a/x; similarly, the relevance of i to /a/y spins off the schema

i/a/y.

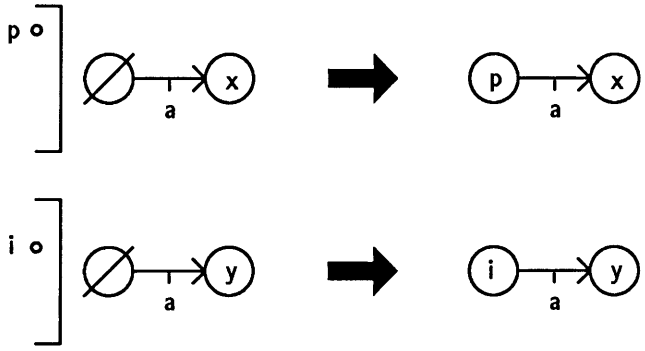


Figure 3-14: Each of two schemas discovers a relevant context item, spawning a spinoff schema.

Context spinoff schemas, like result spinoffs, need not be reliable. For an item to be a relevant condition for a given schema, the schema need only be significantly more reliable for one state of the item than for the other, but even the greater of these reliability levels can be arbitrarily small. A context spinoff's own extended context seeks conditions that further improve reliability; the discovery of such conditions spawns additional context spinoffs, as in figure 3-15. In this fashion, marginal attribution can build up to some conjunction of conditions that does make the schema reliable.

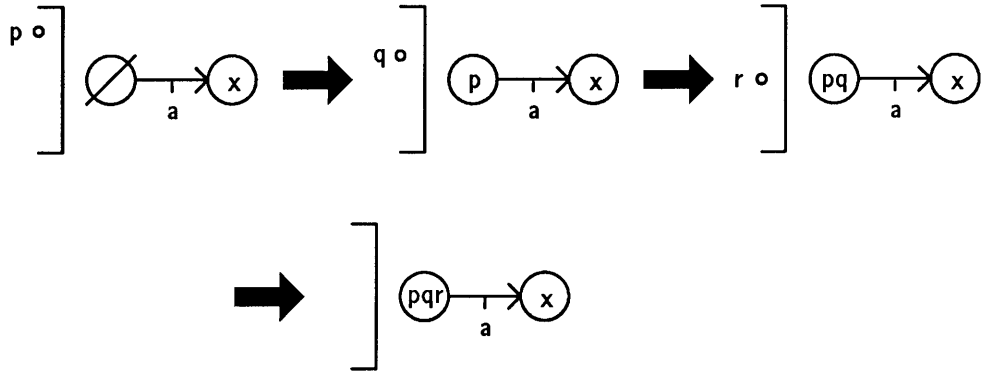


Figure 3-15: Successive spinoffs build up to a conjunction of context conditions.

Here again, distinguishing relevance from reliability solves a chicken-and-egg problem. If, say, items p, q, and r must all be On for result x to follow from action a, then the probability that the result follows if, say, p was On when the action initiated—call

that P(win|a)—is just the probability that q and r were On then too. If p, q, and r are statistically independent of one another, then P(win|a) is the product of the individual probabilities of q and r being On; if p, q, and r are positively correlated, then P(win|a) is even larger than that product. However small this probability may be, it is significantly larger of the zero likelihood that the result follows the action if p is Off (given the above assumption that p, q, and r are all required); hence, the relevance of p is detectable (and similarly for the other conjuncts; the one that makes the biggest difference will be detected first). Note that discovering the context-relevance of p does *not* depend on there being any nonzero chance that the schema succeeds when only p (but not q and r) is on.

Most generally, there may be a disjunction of conjunctions of conditions under which the result x follows the action a. The schema mechanism does not represent disjunctive contexts as such; however, it may construct several reliable schemas that all have the same action and the same result, but with different contexts. This effectively expresses a disjunctive condition for the result to follow the action.

In that case, however, p's relevance is detected only if P(win|a) exceeds the probability that some disjunct that does not include p is satisfied. If, on the other hand, some alternative disjunct—say, the conjunction of items d, e, f—is more likely to be satisfied (when p is On) than is the conjunction of q and r, then the relevance of p will be obscured.

Suppressing redundant attribution

There is an embellishment of the marginal attribution algorithm—deferring to a more specific applicable schema—that can enable the discovery of an item whose relevance has been obscured. Suppose, in the above example, that the context-relevance of d to schema /a/x is not obscured; the schema's extended context discovers this relvance, leading to the construction of the schema d/a/x. The extended-context slot for d in /a/x records that a schema has been spun off from that schema for that (positively

included) item. The following embellishment then occurs:

- All correlation data in all extended context slots of the schema /a/x are reset to zero.
- Subsequently, whenever /a/x is activated and d is On, the updating of all extended context data for that trial of /a/x is suppressed.

The effect of this embellishment is that the extended context of /a/x now maintains correlation data only for trials for which d is not On (resetting the data erases correlations that had been tabulated without this condition). Thus, when d is on, attribution is deferred from /a/x to the more specific applicable schema d/a/x. That schema, of course, can update its own extended context data for the trial, leading to the eventual construction of def/a/x.)

Once the relevance of d has been thus recorded, the probability of /a/x succeeding when a is On no longer has to compete with the probability of its success when d is On. The embellishment of deferring to a more specific applicable schema ensures that as some (conjuncts of) disjuncts of a disjunctive condition are identified, it becomes easier to detect the relevance of (conjuncts of) other disjuncts—the other disjuncts need only compete against the "background" probability of the schema's success due to yet-unidentified conditions.

Deferring to a more specific applicable schema performs a second vital function. Consider again the sequence of constructions shown in figure 3-15, in which /a/x spins off p/a/x, which spins off pq/a/x, which spins off pqr/a/x. If not for the provision for deferring to more specific applicable schemas, /a/x would also spin off q/a/x and r/a/x; schema p/a/x would also spin off pr/a/x, and so on.

With just three items in the eventual realiable context, such a proliferation of intermediate constructs poses no crisis. In general, though, the number of such intermediate constructs is exponential in the size of the eventual context (the set of intermediate constructs corresponds to the powerset—the set of all subsets—of the

eventual context). Fortunately, deferring to more specific applicable schemas prevents this exponential proliferation. If /a/x has already spun off, say, p/a/x, then /a/x's extended context slot for q will no longer be updated on trials when p is On; hence, /a/x will not redundantly discover the relevance of q.

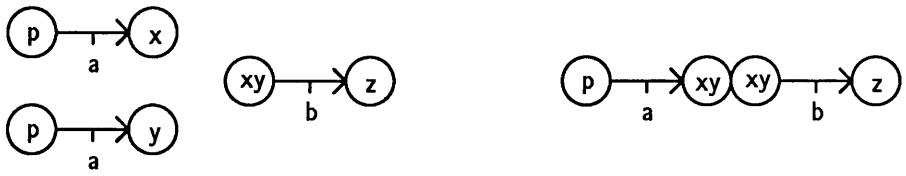
A second embellishment also reduces redundancy: when a schema's extended context simultaneously detects the relevance of several items—that is, their statistics pass the significance threshold on the same trial—the most specific is chosen as the one for inclusion in a spinoff from that schema. Thus, if i is a special case of j (that is, i is On only when j is On), and the extended context of /b/z discovers the relevance of both simultaneously, i/b/z will spin off. (Both conditions' relevance will be discovered simultaneously if all encountered trials of /b/z when j is On also have i On.) If the more general condition j actually suffices, then /b/z will eventually spawn j/b/z as well, due to trials when j is On and i is Off. If, on the other hand, the more specific condition is necessary, j/b/z will not be built. (See section 4.1.4 for an example from the implementation's performance.)

Without this specific-priority embellishment, /b/z might first spawn j/b/z. Then, if the more specific condition i were actually necessary, /b/z would defer attribution to j/b/z, which would spawn ij/b/z. The unnecessary conjunction ij, appearing as the context of a reliable schema, would then be eligible for inclusion in the results of other schemas. The specific-first embellishment avoids this unnecessary proliferation.

An item is considered more specific if it is On less frequently. Although the specific-first embellishment is intended for situations in which the more specific item is a special case of the more general (as opposed to occurring disjointly), the machinery does assure that this is so. When it is not so, the specific-first criterion makes an arbitrary choice among relevant items.

Result conjunctions

In order for one schema to chain to another, its result items must include all the context items of the other (and with the same signs). For purposes of chaining, the schemas in figure 3-16a are not equivalent to those in figure 3-16b; the chaining broadcast described in section 3.3.1 identifies a chain to x in the second case, but not the first. Consequently, the marginal attribution machinery must be able to build schemas with conjunctive results, as well as conjunctive contexts.



(a) No chain to xy. (b) These schemas chain together.

Figure 3-16: Predicting two items separately does not chain to a context that requires their conjunction.

The mechanism might be designed to build conjunctive results incrementally, as with contexts. However, this approach would create a powerset proliferation problem, as above. And the above solution to that problem for conjunctive contexts—deferring to more-specific applicable schemas—does not suffice for conjunctive results; it fails to block a different exponential proliferation, as illustrated in figure 3-17. Suppose there exist reliable schemas p/a/x and q/a/y. If p/a/x sometimes activates when q is On, then if p/a/x could have its own result spinoffs, it would discover the relevance of y as a further result, and would spin off the schema p/a/xy; similarly, q/a/y could spawn q/a/xy. Either of these schemas, in turn, could spawn the reliable schema pq/a/xy, which combines the assertions of p/a/x and q/a/y.

A combination of two such schemas is acceptable. But, here again, if n schemas thus combine, the number of such combinations is exponential in n. To prevent the explosive proliferation of such combinations, the schema mechanism does not build conjunctive results incrementally; a schema with more than one item in its result is

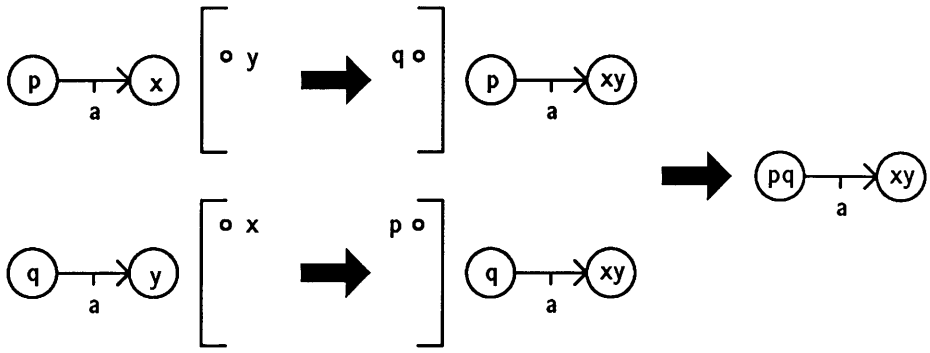


Figure 3-17: Incrementally extending results would proliferate combinations of schemas.

prevented from spinning off a schema with an addition to that result. Chaining to contexts that have more than two items is made possible by permitting a schema to spawn a multiple-item result spinoff all at once, as follows.

As noted in section 3.2.1, a schema's extended result has a slot for every conjunction of items that appears as a context of a reliable schema (as well as a slot for every individual item). Marginal attribution treats each such conjunction just like an individual item with respect to maintaining extended-result statistics about the correlation between its transition and the schema's activation, and with respect to including a relevant conjunction in the result of a spinoff schema. Thus, when a conjunctive result is actually needed—to chain to a reliable schma's context—the marginal attribution machinery will permit that result.

Overriding conditions

Extended contexts, like extended results, identify relevant items for inclusion in spinoff schemas. Extended contexts serve a second function: identifying *overriding* conditions, that is, conditions under which an ordinarily reliable schema is invalid. A schema whose context is satisfied is nevertheless inapplicable when its extended context reports that a known overriding condition obtains.

The example in figure 3-18 illustrates the need to recognize overriding conditions. The schema p/a/x is very reliable, but fails when the (unusual) condition w

obtains. The extended context of p/a/x duly discovers the relevance of w being Off—the schema has a much higher probability of suceeding if activated then than if w is On. Consequently, the schema ¬wp/a/x is spun off.

But merely creating the more specific schema ¬wp/a/x does nothing to suppress p/a/x when w is On. Permanently suppressing p/a/x, and relying instead on ¬wp/a/x, would solve that problem, but at an unacceptable cost: schemas chaining to x via a would now have to include ¬w in their results—and similarly for all other overriding conditions that may be discovered. But if these conditions arise rarely, the overhead of having to build new chains of schemas that explicitly include the negations of the overriding conditions is unacceptable.

Instead, the suppression of p/a/x when w is On is accomplished by the extended context's override machinery, which notes that item w is not in the state which makes more reliable than otherwise by a significant factor; hence the mechanism deems the schema unreliable at the moment, and does not regard it as applicable. At other times, however, p/a/x may applicable and useful.

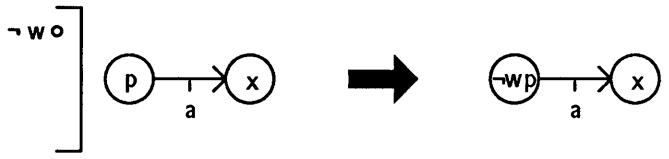


Figure 3-18: Condition w overrides schema p/a/x.

Sustained context conditions

Actions have variable execution times. In the present implementation, each primitive action takes one time unit to execute (in principle, this need not be the case). The time between a composite action's initiation and completion can vary considerably, even for different invocations of the same action, depending on the number of steps in the shortest chain to the action's goal state.

Some context conditions need only be satisfied when an action is initiated. Others

need to be satisfied throughout the action's execution. The primary extended-context slot correlation, described above, compares two probabilities of a schema's success that are conditional on an item's state at the time that the schema's action is initiated. A second correlation, also maintained by each extended context slot, compares similar probabilities defined with respect to an item's state at the conclusion of the action. If both the initiation-time and completion-time correlations are significant, the mechanism presumes that the corresponding condition needs to be sustained throughout the action's execution.

If a context condition needs to be sustained until completion of a (composite) action, the mechanism obliges this requirement in two ways:

- When components of the composite action are selected for execution, actions whose results assert the negation of that condition are thereby suppressed. First, the activated schema informs its sustained context items of that status. Then, the mechanism identifies every schema whose result would negate a sustained item. Any such schema that is of applicable, of nonnegligible reliability, and is not superseded by a more specific applicable schema informs its action of its status. The action then suppresses the activation of all schemas that have that action.

- If such a condition becomes negated anyway (due to external events or to unanticipated side-effects of the mechanism's actions), the pending schema is aborted. (In that case, some chain of schemas that reestablishes the violated context condition and proceeds to the same goal may well be the basis for the next activation, effectively repairing the problem.)

Except for conditions that need only be satisfied initially, the mechanism does not seek context conditions that need to be satisfied for only part of the action-execution interval. This is in keeping with the use of schemas to chain to a goal—each prior link establishes the conditions needed for the next link to be applicable. A condition

which is only necessary at, say, the completion of an action can be designated as a condition to be sustained thoughout the action. Finer specification can always be achieved by spawning spinoffs from the components of a composite action, rather than designating conditions to be sustained throughout the composite action's execution.

Compactly storing correlation data

Each schema has an extended-context and extended-result slot for every item (and, in the case of the extended context, for certain conjunctions of items). Almost all of the memory required by the schema mechanism's data structures is devoted to the correlation statistics in schemas' extended contexts and extended results. A naive representation of these statistics would be so bulky that the schema mechanism could not be implemented on present-day hardware. This section digresses from loftier theoretical concerns to describe a low-level scheme for compactly representing such statistics.

Every extended-context or extended-result slot maintains two correlation statistics, each of which is the ratio of two probabilities, a *with-probability* and a *without-probability*. For extended-result slots, these are respectively the probabilities of particular state transition with or without activation of the schema; for extended-context slots, these are the probabilities of successful activation with or without a pariticular item being on. *Positive* trials are the events whose probability is tabulated. For extended results, positive trials are ones for which the state transition does occur; for extended contexts, positive trials are ones for which the activation is successful.

Naively, each correlation statistic could be represented by a pair of probabilities (figure 3-19), each represented as a rational number, with fixed-length numerator (corresponding to the number of positive trials) and denominator (the total number of trials).[5] The size of the smallest detectable probability then depends on the number

[5] A floating-point representation might be used instead. However, incrementing the number of trials by one then becomes impossible when the exponent is larger than zero, so that the least significant bit of the mantissa is greater than one.

of bits used for each probability; for example, sensitivity to about one event in a million requires a numerator and denominator of twenty bits each, for each of the two probabilities per correlation statistic.

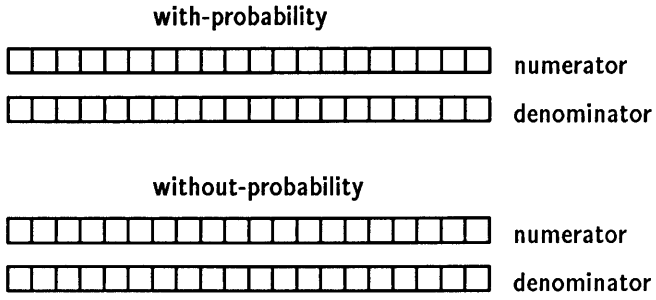


Figure 3-19: Representing a correlation as a pair of probabilities, each with a 20-bit numerator and denominator, has a resolution of $1/1,000,000$.

The representation can be made more compact by alternating between one with-sample and one without-sample (figure 3-20); an alternation bit is added to the representation to indicate whether the next sample should be a with-trial—that is, a trial that contributs to the with-probability—or a without-trial. If the next trial is not of the indicated type, it is ignored. Alternating between the two types of samples assures that the two probabilities have the same denominator. Then, since only the ratio of the two is of interest, the denominators needn't be stored.

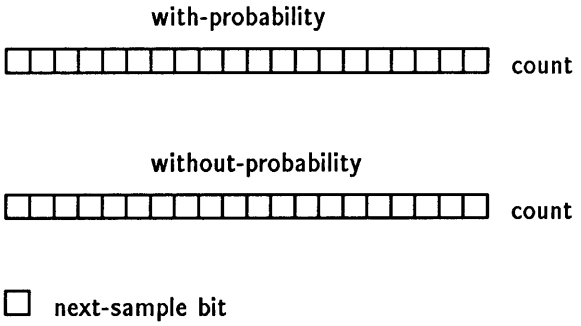


Figure 3-20: Alternating between the two samples obviates the need for the denominators.

A provision for overflow offers further improvement. If either numerator reaches its maximum value, both numerators shift right by one bit (that is, divide by two);

this operation preserves their ratio. Moreover, the representation now has sensitivity to arbitrarily small probabilities—not merely, say, one in a million—since precision is no longer limited by the size of the numerator. The number of bits per numerator can be reduced sharply (figure 3-21). As information about earlier trials vanishes when the numerators shift, the sample is biased toward more recent trials. This bias is arguably desirable: circumstances change, and if the two probabilities significantly differ in the course of recent trials, it is likely that the item in question is indeed relevant now.

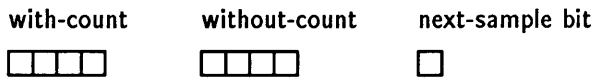


Figure 3-21: Right-shifting to prevent overflow requires fewer bits per count.

A final improvement, due to Rivest (personal communication), compresses both counts into one signed count (MARCSYST I uses a four-bit count, plus sign bit, as in figure 3-22). The count increments for a with-trial, and decrements for a without-trial. To attenuate random drift (which otherwise would soon bring the count to one of its two extrema even if the two probabilities were equal), increments (and decrements) are of different sizes. In particular, if the count is positive, then a with-trial increments the count by a larger amount (three, in the present implementation), but a without-trial decrements by a smaller amount (presently two). Similarly, if the count is negative, with-trials increment the count by the larger amount, and without-trials decrement it by the smaller amount. This disparity exerts pressure toward zero, so that the with-probability must exceed the without-probability by the ratio of the two increments for it to be likely that the value steadily diverges from zero. If the value reaches either extreme, the corresponding item is deemed relevant.

The actual ratio of the two probabilities cannot be recovered from the single-count representation; that representation indicates only whether the ratio's magnitude exceeds a threshold determined by the ratio of the two increments. For a given probability ratio, the ratio of the larger increment to the maximum count magnitude

Figure 3-22: Two counts collapse into one signed count which gets incremented and decremented, with a bias towards zero.

determines how many positive trials are required before relevance is detected. When two probabilities are actually equal, the likelihood of a false indication of relevance decreases exponentially in the number of trials required (double-exponentially in the number of bits in the count). Hence, although the parameters used by MARCSYST I are no doubt adequate only because of the unrealistic simplicity of the microworld—requiring fewer trials to distinguish real correlations from coincidence—scaling up does not strain computational storage or time resources.

3.4.2 Composite actions

Even for sensorimotor-stage schemas, primitive actions alone are insufficient, for two reasons. The schema mechanism needs to express actions at higher levels of abstraction; and it needs to discover the results of external events as well as of its own actions.

Composite actions facilitate the abstraction and externalization of actions. A composite action is defined by its *goal state*; it is the action of bringing about that state. A composite action is implemented by schemas that chain to its goal state. When a primitive action is initiated (by the activation of a schema which has that action), the action triggers a particular hardwired mechanism. When a composite action is initiated, the schema mechanism identifies a chain of schemas leading from the current state to the goal state, and activates those schemas in sequence. (More accurately, the information to identify such chains is stored in advance in the action's *controller*, and is deployed when the action is initiated, as discussed below.) Schemas that lie along some chain to a composite action's goal state are called *component schemas* of that action.

Consider, for example, the action of turning on a lightswitch. On a given occasion, that action might be accomplished by a particular low-level motor action, occurring in just the right context at the end of some chain of schemas that prepares for the final flick of the switch. Rather than (or in addition to) such a representation, it is valuable for the schema mechanism to designate turning on the lightswitch as an action in itself.[6] Such a designation offers three advantages:

- By abstracting above the action's implementation, the mechanism can learn about the result of turning on the lightswitch per se (eg, that a light turns on), rather than just learning about the result of some particular lowlevel action, which lesson would not generalize to the next instance of turning on the lightswitch, if accomplished then by a different lowlevel action.

- Also, by abstracting above the action's implementation, the mechanism is able to organize activity hierarchically. A chain of schemas may incorporate the action of turning on the lightswitch—or much higher-level actions than that—as a single step, the details of which needn't be accounted for as part of that chain; the details may depend in part on circumstances that are yet unknown when that action is initiated.

- Finally, representing lightswitch-on as an action enables the schema mechanism to learn about the effects of that action (eg, a light going on) even when the action occurs as an external event, not under the mechanism's own control (as explained immediately below). Thus, the schema mechanism's composite-action facility brings about a transition from representing the result of some action, to representing the external result as an action in itself—and in turn finding its own results. This facilitates the Piagetian progression from schemas of physical activity, to schemas that are independent of personal action, via intermediate

[6]This hypothetical example is considerably beyond the implementation's actual achievements. Lower-level examples of the same principle appear in the synopsis of chapter 4.

schemas that involve the effects of personally-caused external events.

Special properties of schemas that have composite actions

As noted in section 3.4.1, the marginal attribution facility considers a schema to have been implicitly activated if the schema's action is initiated when the schema is applicable, even if that schema was not selected for activation, and thus was not responsible for the action's initiation. Composite actions carry implicit activation one step further. A composite action is considered to have been implicitly taken whenever its goal state becomes satisfied—that is, makes a transition from Off to On—even if that composite action was never initiated by an activated schema—in fact, even if the goal state's achievement is due to external events entirely uninfluenced by the mechanism. Consequently, a schema whose action is composite is implicitly activated each time its action's goal state becomes satisfied when the schema is applicable. Marginal attribution can thereby detect results caused by the goal state, even if the goal state obtains due to external events.

Designating external events as actions combines with activation hysteresis (section 3.2.2) to promote *imitation* by the schema mechanism of external events that correspond to extant schemas. Hysteresis applies even to implicitly activated schemas, so if a schema is implicitly activated due to an external event, the schema receives enhanced value for an explicit activation, which would repeat the external event.

A schema with a composite action is restricted from collecting extended result data except on trials for which the action was explained—its goal state predicted by some reliable schema that just finished its (perhaps implicit) activation. And on such a trial, data is collected only at those extended-result slots whose items' transition was unexplained. This restriction mitigates an undesirable proliferation of schemas when several state transitions that have been made goal states of composite actions tend to co-occur. Without the restriction, each of the co-occurring actions would come to regard the others' goal states as its own results, spawning a spinoff

schema for each. But with the restriction, if the co-occurring transitions become explicable to the mechanism at about the same time, the mutual attribution and consequent proliferation is blocked. (In particular, the restriction prevents the useless construction for each composite action of a useless schema whose result is just the action's goal state.) Examples of this restriction's usefulness occur in section 4.1.7.

A composite action is *enabled* when one of its components is applicable. If a schema is applicable but its action is not enabled, activating the schema is ineffective, and the schema most likely fails. If failures due to a disabled action are frequent, a schema's extended context may be able to learn override conditions that designate situations in which the action does not work. For such conditions to percolate up into a schema's context violates the abstraction barrier that is supposed to insulate the schema from the details of executing its action. But if the action fails under identifiable circumstances, the abstraction barrier is partly misplaced, and the schema's context needs to take note of the problem.

Constructing and maintaining composite actions

As discussed in section 3.2.2, the schema mechanism keeps track of the average accessibility of each item and result-conjunction of items. An item or conjunction is accessible when there is a chain of schemas that leads to it from the current state; its average accessiblity is the probability that the item is accessible at a given moment. When the accessibility of an item or conjunction is high, the schema mechanism defines a composite action with that item or conjunction as its goal state. The schema mechanism also constructs a bare schema which has that action; that schema's extended result discovers effects of achieving the action's goal state.

When a new composite action forms, the mechanism also allocates and initializes the new action's controller, which, as discussed in section 3.3.1, connects to all schemas, with a slot for each schema that records the schema's proximity to the action's goal state. To initialize the controller, the mechanism broadcasts a message

backwards in parallel through chains of schemas that lead to the goal state (section 3.3.1). Occasionally thereafter, when the composite action is taken, the mechanism performs another such broadcast to update the controller information. Usually, though, the action executes on the basis of the already recorded controller data.

Recording proximity information in an action's controller is similar to *chunking* in SOAR (Laird, Newell and Rosenbloom 1987); both involve searching through a state-space, recording the points of departure, so that the path from those points to the goal is subsequently known without having to recapitulate the search. But the nature of the search that is thus abbreviated is different here; see sections 6.1.3 and 6.1.4 for elaboration.

Using controller data has several advantages over performing a broadcast. The most straightforward advantage is that it is faster: a broadcast takes time proportionate to the maximum chain length searched for, whereas finding the closest applicable schema based via the controller only takes time logarithmic in the number of schemas.

Controller data facilitates the concurrent activity of several composite actions. (The current implementation only activates one toplevel schema at a time, but many nested composite actions may run simultaneously; furthermore, the mechanism could be extended to permit several toplevel activations.) As noted in section 3.3.1, concurrent broadcasts would interfere with one another. Using prerecorded controller data circumvents such interference.

Using controller data also extends the length of chains that can be found by a broadcast, by means of an embellishment to the broadcast process. When a broadcast updates the information in a previously initialized composite action, the existing data serves as a point of departure. That is, rather than beginning the broadcast only from schemas whose results include the goal state, the broadcast also starts with schemas of already-known proximity to the goal. Schemas that had been at the fringe of prior broadcasts can now discover predecessor links in chains to the goal.

A second embellishment creates still other advantages. A composite action con-

troller does not only record proximity information from broadcasts. It also averages in data from actual executions of the action. That is, each time a composite action is explicitly initiated, the controller keeps track of which component schemas are actually activated and when. (The present implementation only keeps track of the initial such component for each time an action is initiated; this lets the data be kept globally, instead of commanding space in each controller slot.) If the action sucessfully culminates in its goal state, the actual cost and duration of execution from each entry point are compared with the proximity information stored in the slot of each component actually activated; in case of discrepancy, the stored information is adjusted in the direction of the actual data. If the action fails to reach its goal state, the proximity measures for the utilized components are degraded.

Most straightforwardly, this empirical revision of controller data serves to correct false predictions based on proximity broadcasts. More subtly, the revision might foster the discovery of certain kinds of reliable paths that a proximity broadcast cannot identify as such (although such discovery is thus far undemonstrated by the implementation). In particular, it might be expected to foster the discovery of diverging and reconverging paths, of paths that require the repetition of a particular component, and of paths that involve on-the-fly repair of broken links in a chain.

- *Divergence and reconvergence.* Consider a set of three chains of schemas to a common goal, as shown in figure figure 3-23a. The three paths diverge, via three schemas with the same context and action as one another, but different results. Suppose it is reliably the case that one of these three results follows, but no particular one follows reliably (for example, there may be a 1/3 chance of each occurring). Since each of the three results lies on a path that reliably reconverges to the goal, a chain that passes through the area of divergence and reconvergence is reliable.

  However, the broadcast process misses this reliability. The cumulative proximity measure broadcast along each of the three chains is attenuated by the low

reliability of each of the three diverging schemas. Thus, the broadcast proximity at and before those links in the chain underestimates the actual proximity of those links.

An underestimated component might nonetheless be selected by the controller, if no component schema with greater proximity is applicable. Each time such a selection culminated in reaching the goal state, the proximity measure for that component increases, until the estimate became accurrate.

- *Repetition.* There may be a component schema that needs to be repeated several times until its result obtains successfully, enabling further progress along a chain (figure 3-23b). As in the previous example, the schema that is unreliable at each repetition (but that is, by assumption, reliable within several repetitions) attenuates the proximity measure that broadcasts backward through that link in the chain. Also as in the previous example, the empirical success of paths to the goal that pass through the underestimated link tends eventually to correct the underestimate.

- *On-the-fly repair.* Suppose a particular component schema is unreliable, and often fails when there is no other applicable component to shift to, thus interrupting the composite aciton. It may so happen that certain schemas that tend to be applicable and to get activated at that point have the side-effect of making applicable some component of the interrupted action. It may even be the case that those schemas tend to create some new component schema, and create circumstances that make it applicable. The break in the original chain is thus repaired. If such repair follows reliably, the controller again comes to recognize empirically that the unreliable component, and its predecessor links, reliably leads to the goal state.

  By counting on this repair taking place, the machinery effectively invokes the system's overall intelligence, used to effect the repair, as a subroutine. But

this invocation is not explicit; it is a consequence of the empirically derived high proximity value for an unreliable component which nonetheless leads to situations in which repair is possible.

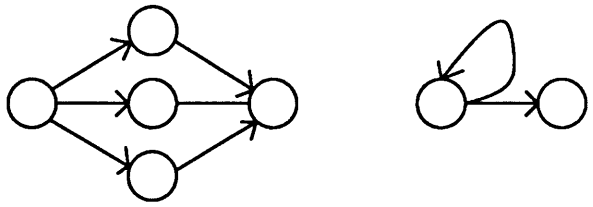


Figure 3-23: Action controllers make possible the discovery of paths that diverge and reconverge, or that involve repetition.

3.4.3 Synthetic items

It is plainly inadequate to represent states of the world directly in terms of primitive sensory elements. Even if, say, statements about physics, baseball, or politics could in principle be reduced to statements about the sensory manifestations of those domains, the reduction would be impossibly cumbersome. If a learning system's initial conceptual repertoire is indeed limited to sensorimotor terms, then a necessary condition for the system's eventual attainment of humanlike intelligence is the ability to synthesize much higher-level concepts. Such concepts must somehow be defined in terms of the primitives; the question is how.

One possibility is to use boolean combinations of prior concepts. For example, the contexts of schemas express conditions for action-result pairs, the conditions expressed

as arbitrary boolean combinations of items. (An individual context does not have disjunctive form, but several schemas that share the same action and result effectively express a disjunctive condition.) In the standard AI sense of *concept learning*, a new target concept is just such a boolean combination, and the system's task is to converge to the correct one. I argue, however, that new concepts need to differ from old ones more radically than being novel boolean combinations of old ones.

A concept's *verification conditions* are the possible indications of whether or not a given situation presents an instance of that concept. When a concept is defined as a boolean function of other concepts, the arguments to that function serve as verification conditions for that concept (except for any arguments that the function ignores, such as the argument *a* in the function *(a or not-a) and b*; but all items in a schema's context should make a difference, since only relevant items are added to spinoffs). The designation of a persistent object at a particular location, for example, might include the verification condition of having recently seen the object there. (This designation would require being able to define new concepts or conditions as boolean combinations of past as well as present values of other conditions, effectively treating different time-values as different conditions.)

But it is well known that few concepts can be defined by a fixed set of verification conditions. It is logically possible that the world will present evidence that some additional condition is relevant, or that what had been considered a conclusive indication of the concept in question actually admits previously unsuspected exceptions. In the case of a persistent object, for example, one may discover a new theory of physics which predicts that an object will materialize at a particular position under certain circumstances; observing those circumstances then becomes a verification condition for there being an object at that position. Or one may be told of an object's presence by a person whom one considers trustworthy. Being so informed qualifies as a verification condition; but discovering that that person is a pathological liar may rescind that verification condition.

As these examples illustrate, the acquisition of new verification conditions, or the revision of old ones, for a given concept may involve discoveries in domains that are arbitrarily abstruse (e. g. theoretical physics), or arbitrarily remote from that concept (e. g. judging people's personalities). Thus, verification conditions should always admit extension and revision in the course of learning more about the world; and since the verification conditions for a given concept can always change, the concept cannot be *defined* by a fixed set of such conditions. Moreover, nothing in this analysis depends on the functions in question being boolean (as opposed to, say, functions of first-order logic). More generally, then, it does not work to define a concept as *any* fixed function of its verification conditions.

However, any concept entertained by a physically realized mechanism *must* be expressible as some function of the system's (past and present) inputs, because the (full or partial) state of any physical system computes some function of the system's inputs. This expressibility in principle may seem to contradict the fact that the verification conditions might always need revision depending on yet-undiscovered aspects of the world. However, a concept might—in principle—be defined to incorporate the very criteria for judging the appropriateness of particular verification conditions, based on the totality of knowledge available to the system. That construal of a concept's definition produces *meaning holism* (Fodor 1987), whereby each concept's meaning must incorporate all other concepts and beliefs.

However possible in principle such a definition might be, it is prohibitively unwieldy. Correspondingly, although the schema mechanism does maintain verification conditions for each synthetic item (as discussed below), those conditions are not fixed or exhaustive. Rather, they are always subject to change, and thus do not define the synthetic item. Section 3.4.3 describes what does do so.

Intension, extension, and verification conditions

Using synthetic items to define concepts offers a perspective on the relations among the intension, extension, and verification conditions of a concept. A concept's *extension* is the set of possible circumstances under which the concept holds. (This definition presumes that the concept is propositional; if, instead, it designates, say, a particular object, it can be converted into a related proposition, such as *Such-and-such object is present.*) The *intension* of a concept is a particular designation or representation of that concept; in the schema mechanism, the intension of the concept represented by a given synthetic item is the set of validity conditions of the item's host schema. Finally, the verification conditions are what an agent uses to determine whether the concept does or does not obtain (under actual or hypothetical circumstances).

On the face of it, it may seem arbitrary to ascribe to an agent a concept whose extension differs from the extension of the verification conditions that the agent uses for the concept. After all, the verification conditions specify precisely when the agent deems the concept applicable; in what sense might the agent actually have in mind a different concept than that? Synthetic items suggest an answer: an item's verification conditions change, and change systematically in the direction of corresponding to the item's intension. That intension, then, is the concept that the verification conditions try to match (though they may never fully converge to it).

The relation among an item's intension, extension, and verification conditions helps solve the puzzle of how a concept's extension can have psychological reality. It is well known that two concepts can have the same extension but different intensions (e. g. Fodor 1981). For example, section 4.3.1 discusses the formation of a synthetic item that designates a palpable-object-at-position-X, and another designating a visible-object-at-position-X. In a world lacking invisible or intangible objects, the two concepts are coextensive: each holds exactly when the other does. But they have different intensions: one is defined with respect to a host schema for reaching and

touching something, the other for looking and seeing. At first, the schema mechanism does not represent their mutual equivalence; indeed, it sometimes recognizes the applicability of one of the coextensive concepts, but not the other. In what sense, then, is their coextensivity psychologically real, that is, a property of the schema mechanism rather than of the world external to the mechanism?

Again, the answer concerns the verification conditions of the two concepts, which converge to the same extension, which—at an imaginary limit—matches the two synonymous intensions. Even before this convergence occurs, the verification conditions, as maintained by the schema mechanism, can be said to be disposed to so convergence, given suitable experience in the world; and this dispositional property is reasonably regarded as a property of the mechanism, even though—like other dispositional properties, such as solubility—it depends as well on external conditions.

Constructing synthetic items

Creating new state elements involves a more radical sense of novelty than building new schemas and actions. Spinoff schemas and composite actions are merely reorganizations of existing structures. But a synthetic item is a new element of the system's ontology—an element fundamentally different from the prior contents of the system's conceptual vocabulary, to the extent of being practically inexpressible as any function of those prior concepts, as just discussed.

Sometimes, as with conservation of object or of mass, what's required is the conception of some underlying physical reality. In contrast, conservation of number, for example, involves the conception of an underlying nonphysical abstraction. The synthetic-item machinery is designed to promote conservation discoveries of both kinds by creating new items to represent newly-conceived aspects of reality.

The schema mechanism constructs a synthetic item to reify the validity conditions of an unreliable schema. That is, a new synthetic item is defined to represent whatever unknown aspect of the world governs the schema's validity. This is best explained

with an example.

Consider the schema in figure 3-24, which asserts that moving the hand to some body-relative position X results in a tactile sensation at the hand. This schema is unreliable; it only succeeds when there happens to be an object at that position, waiting to be touched.

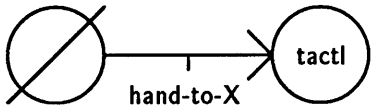


Figure 3-24: Sometimes, moving the hand to X brings tactile contact.

Significantly, however, the schema is *locally consistent*, meaning that if it happens to succeed when activated on some occasion, it is likely to succeed again if activated again within, say, the next several seconds. This consistency follows from the tendency of objects in our environment to stay put for a while. The schema mechanism, of course has no appreciation of this explanation; but it does keep track, empirically, of each schema's local consistency, the probability of its success when its last activation was successful; and, for a schema with high local consistency, the mechanism also tabulates the expected duration of the schema's consistency, the average interval during which the schema is observed to remain valid.

When a schema is found to be unreliable but locally consistent, the mechanism constructs a new synthetic item, called that schema's *reifier*; the schema is the new item's *host schema.* The host schema's reifier designates whatever condition makes the schema valid—in this case, roughly the condition that a paplable object is present at body-relative position X.

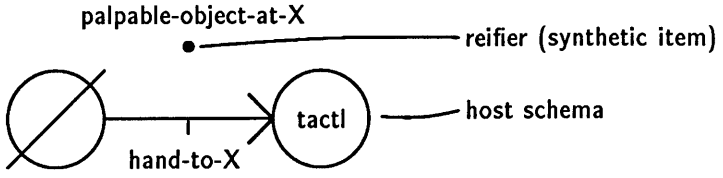


Figure 3-25: A synthetic item reifies the conditions under which this schema is valid.

The host schema associates its reifier with a *probe*—the host schema's action—and a *manifestation*—the host schema's result. A synthetic item thus works backward from a thing's manifestation, and a way of probing for that manifestation, to define the very thing manifested. In the present example, an object at a given position is manifested by a tactile sensation when probed by putting the hand there. But the concept a palpable object being there says more than that that probe in fact yielded the manifestation—the concept further entails that, even when the probe is not actually carried out, it *would* yield the manifestation if it *were* carried out (a so-called *counterfactual* assertion (Lewis 1973), based on a hypothetical premise—that the probe is carried out—that is contrary to fact).

What persists in between probes and in between manifestations is the fact that the probe *would* yield the manifestation. This persistence is not merely the recency or recurrence of the manifestation; many states recur without there being any underlying entity which persists between recurrences and which the recurrent state manifests. In the present example, the condition that persists is a rudimentary fragment of the concept of there being a physical object (at a particular position). As section 4.3.2 illustrates, the development and intercoordination of many such fragments implements progressively better approximations to the concept of physical objects.

Looking for persistence is certainly built into the schema mechanism's synthetic item facility; it is innate, rather than acquired. When the mechanism constructs a synthetic item, what is novel and learned is not persistence per se, but rather the very thing whose persistence is noticed—not the manifestation, but rather the state of the world such that the probe would yield the manifestation.

Maintaining verification conditions

The state of a primitive item is set directly by some input module. In contrast, the state of a synthetic item must be maintained according to learned criteria for distinguishing whether the represented state currently obtains or not—that is, according

to learned verification conditions.

The schema mechanism recognizes four kinds of verification conditions:

- *Host schema trial.* Each time the host schema completes its activation, it turns its reifier On or Off according to whether the schema succeeded or failed.

- *Context conditions.* The host schema may spin off other schemas that include context conditions under which the result follows the action reliably. For example, as illustrated in figure 3-26, the palpable-object schema shown above may spin off a schema whose context designates reliable visual evidence for the presence of an object at position X.

  A reliable schema reports its applicability to its *parent* schema (the schema that spun it off); in this example, when the visual-evidence schema is applicable, it reports that fact to the palpable-object schema. Thus, the palpable-object schema knows, without actually having to try, that its activation would succeed at the moment. Accordingly, the palpable-object synthetic item is turned On.

- *Predictions.* A synthetic item, like a primitive item, may come to be included in the results and contexts of many schemas; indeed, synthetic items would not otherwise be useful. If a synthetic item appears in the result of a reliable schema, and that schema is activated, then in the absence of any evidence to the contrary, the mechanism presumes that that schema succeeded; thus the item is turned On (if positively included in the result, or Off if negatively included).

- *Local consistency.* When a synthetic item turns On, it stays in that state (unless turned Off by one of the above conditions) for a period of time equal to the empirically determined expected duration of the host schema's local consistency; then it turns Off. When a synthetic item turns Off, it remains Off until turned On by one of the above conditions. Thus, local-consistency evidence is just the memory of the most recent host-trial, context-condition, or prediction-based evidence for the state of an item.

If several times the expected duration of local consistency passes without any evidence about a given synthetic item's state, its state becomes **Unknown**. An item also becomes **Unknown** if there is contradictory evidence as to its state—except that host-trial evidence simply overrides any conflicting evidence, since host-schema validity is the very definition of a synthetic item's referent; also, local-consistency evidence simply yields to any newer evidence that comes along.

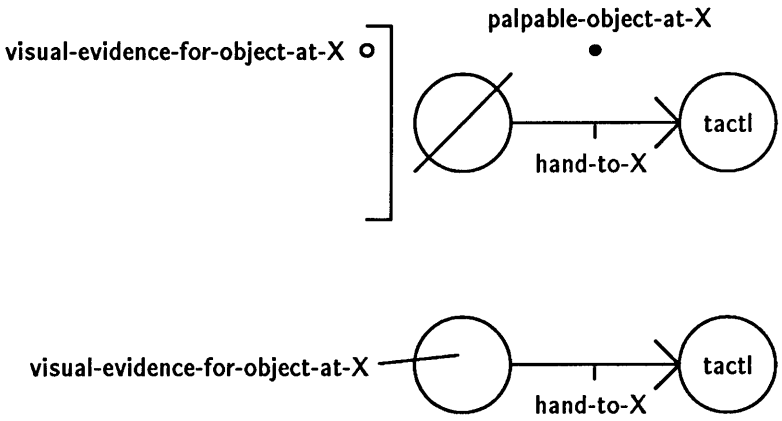


Figure 3-26: Context spinoffs specify evidence that helps maintain a synthetic item's state.

Chapter 4

Synopsis of schema mechanism development

The schema mechanism is designed to recapitulate significant milestones of the Piagetian developmental sequence in infancy, in a manner consistent with the Piagetian developmental themes, and thus to explain how that development might in fact come about. This chapter presents a synopsis of the actual developmental progression exhibited by the extant computer implementation of the schema mechanism.

The schema mechanism has been run from scratch on several dozen occasions, with at least minor variations of the mechanism from one run to the next. The synopsis describes typical results from two modes of operation: a *full* mode (using the microworld as described in section 3.1, and a *restricted* mode (using a restricted version of the microworld, described below). Informally, the results described here are similar to the results from several runs of similar versions of the mechanism, or with different randomizing factors; thus, the results are unlikely to be a fluke. But there is no attempt here to quantify the consistency and variation of results from different runs. The synopsis may be regarded as a pilot study in preparation for a more rigorous analysis, which should be carried out in conjunction with independent replication efforts, as discussed at the start of chapter 3.

Running in the full mode exhausts the available Connection Machine memory before constructing synthetic items that designate persistent objects. Consequently, there is also a restricted mode, for which the microworld is curtailed in two respects: of the primitive eye and hand actions, only forward and backward ones are enabled (left and right are disabled); and there is only one object (other than the body and hand) that comes into view, and it stays within the body-relative region indicated in figure 4-1. Except where otherwise noted, the descriptions through section 4.2.1 refer to the full mode, and descriptions thereafter to the restricted mode.

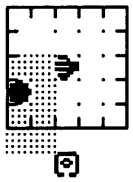


Figure 4-1: In the restricted microworld, hand and eye motions are vertical, and a single external object is confined to the shaded region.

4.1 Spatial substrates

4.1.1 Initial schemas

As noted in section 3.1, there is an initial, bare schema for each primitive action. Figure 4-2 shows the initial schemas for the full mode and for the restricted mode. (The reader may wish to refer back to tables 3.1 and 3.2 in section 3.1 for the names and descriptions of the primitive actions and items.)

4.1.2 Grasping

The first schema built is /grasp/hcl (figure 4-3), which asserts that grasping results in the sensation of the hand being closed. This schema is unusual in that its result follows from its action unconditionally; hence, the schema is reliable despite an empty

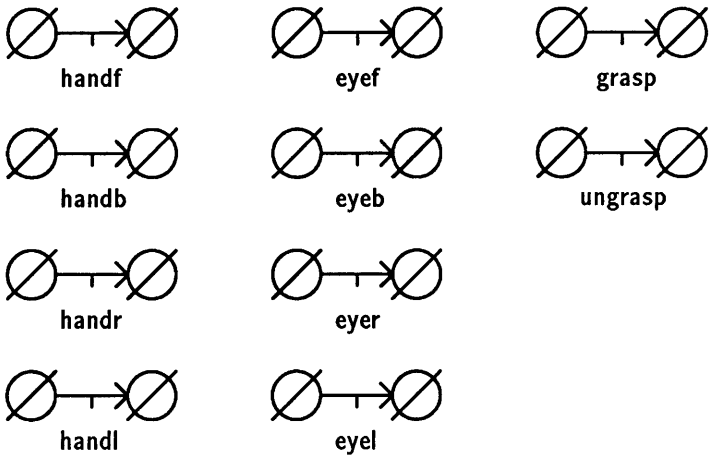


Figure 4-2: The initially supplied schemas.

context. That the result follows unconditionally also makes the schema easy to discover quickly, since every occurrence of the action produces a state transition to the result (unless the hand is already closed, in which case the action's occurrence does not count as a trial); and a transition to the result occurs only when that action is taken. Thus, the significant difference between the result's occurrence with and without the action becomes apparent more quickly than in the case of a relevant result that follows only infrequently, and that occurs under other circumstances as well.

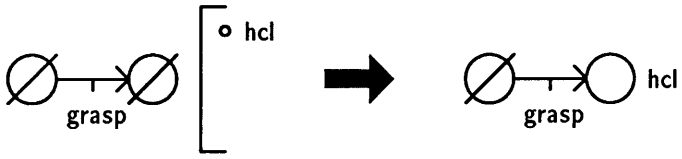


Figure 4-3: The grasp action closes the hand.

Similar schemas describe the ability to close the hand and grasp an object that touches the hand's "fingers", provided that the hand wasn't already closed (figure 4-4). The unreliable schema /grasp/hgr designates the relevance of the grasp action to the sensation of grasping. The (fairly reliable) schema tactl/grasp/hgr denotes the necessity of being in appropriate contact with an object; and tactl&¬hcl/grasp/hgr notes that the hand must not already be closed in order for grasping to follow.

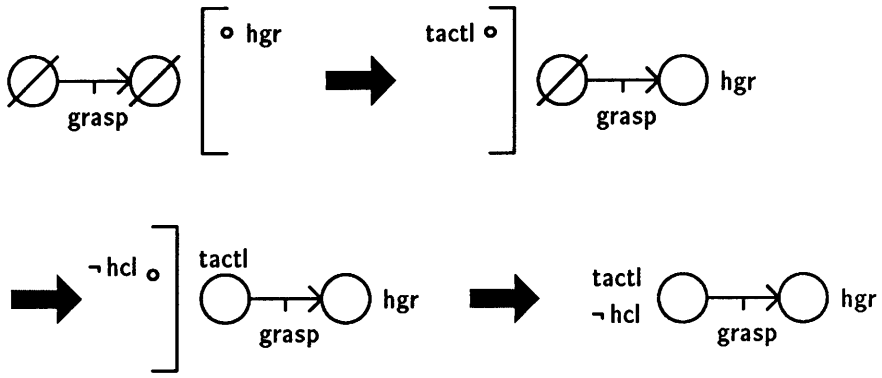


Figure 4-4: The grasp action grasps an object in contact with the hand (unless the hand was already closed.)

4.1.3 Elaborating the visual field

Often, it happens that an object is in the visual field when an incremental glance action occurs. Suppose, for example, that on several occasions, an object appears at vf22 when the action eyel is taken (figure 4-5). As a result of the action, the image shifts to the adjoining visual region to the right, and vf32 turns On.

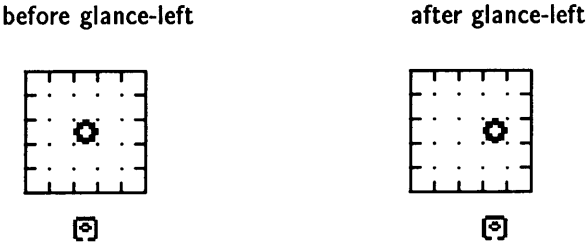


Figure 4-5: A glance action shifts a visual image to an adjoining region.

The transition to vf32 is an infrequent result of the action of glancing leftward; it results only if an object happens to be within view, and at just the right region of the visual field, when the action occurs. Moreover, that transition also happens, on occasion, in the absence of the action in question—if, say, a downward glance brings an image from vf31 to vf32, or if a moving object's image passes through that region while the glance is stationary.

Nonetheless, the transition to vf32 happens more often when the action eyel is taken than when not.

- When **eyel** is taken, a transition to **vf32** follows if:
  - A stationary object appears at **vf22** before the action starts, and the glance is not already at its leftmost orientation; or
  - A moving object arrives at the projection of **vf32** as the action concludes (regardless of whether the glance orientation changed, or was already at its leftmost extreme).
- When **eyel** is not taken, a transition to **vf32** follows if:
  - Some other glance action moves the image of a stationary object to **vf32**; or
  - A moving object arrives at the projection of **vf32**, regardless of whether a glance action was just taken.

Transitions to **vf32** brought about by moving objects happen about as often when the **eyel** action is taken as when not; in either case, what is required is that the object's image move to wherever **vf32** ends up being mapped. Since objects are stationary most of the time, the comparison between the likeihood of transition with and without the action is dominated by the case in which the object does not move.

Transitions due to a stationary object require that some incremental glance action be taken, that visual field is not already in its most extreme orientation in the direction of that action, and that the object's image is in the appropriate adjoining region just before the action. The glance-orientation and image-position requirements are as likely to be met in the case of the **eyel** action as in the case of any of the other three incremental glance actions; therefore, these factors attenuate the probability of the **vf32** transition equally whether or not the **eyel** action occurs. The only remaining factor is whether a glance action occurs, and this occurrence is significantly more likely (in fact, certain) if the action **eyel** is taken than if not. Thus, the transition to **vf32** is significantly more likely when **eyel** occurs.

As indicated in figure 4-6, the extended result of /eyel/ discovers the relevance of vf32, spinning off the schema /eyel/vf32. Of course, the relevance of other visual-field items is similarly discovered by the extended result, leading to spinoffs for those items as well.

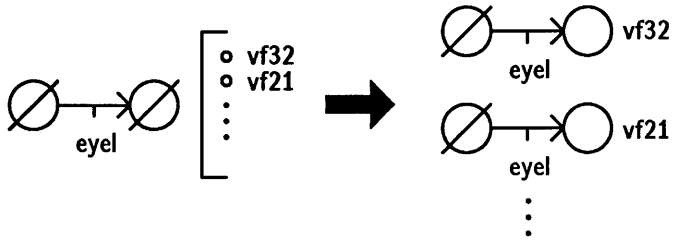


Figure 4-6: A glance-action schema discovers visual-field results.

These schemas, with empty contexts, are all unreliable. But their extended contexts each identify the appropriate context condition, designating the visual-field region immediately to the right of the result item (glancing left shifts an image to the right). So, for example, /eyel/vf32 spins off the reliable schema vf22/eyel/vf32, and similarly for the other schemas showing results of glancing left (figure 4-7), except for those glance-left schemas that result in a visual appearnce at the leftmost edge of the retina.

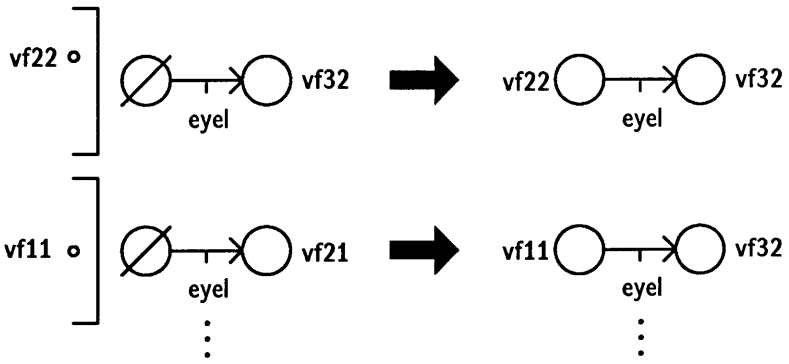


Figure 4-7: Schemas expressing visual results identify corresponding context conditions.

Similar schemas form for each of the other three incremental glance actions. Eventually, these schemas link together to form a network that elaborates the spatial

structure of the visual field (figure 4-8). The spatial elaboration is practical; the adjacency of visual-field regions is designated by their connection via an incremental glance action. The network comprises chains of schemas that say how to shift an image from one visual-field region to another by a series of incremtal glance actions.

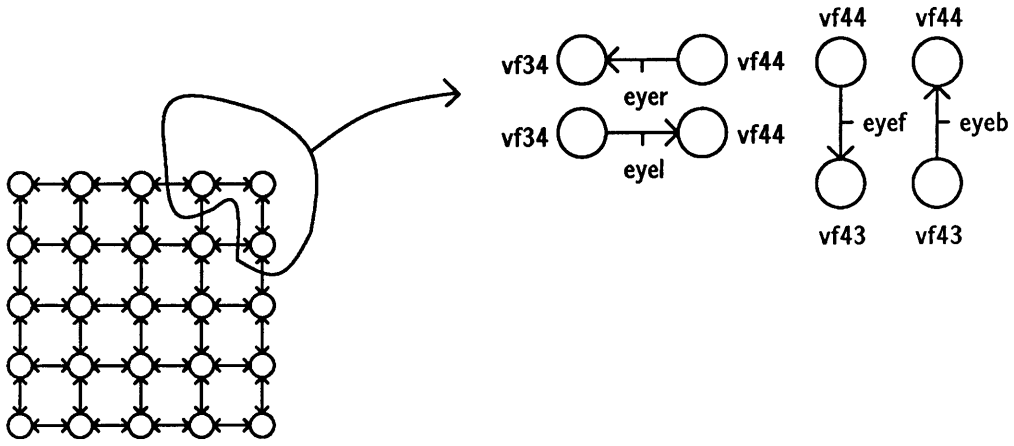


Figure 4-8: Schemas with incremental glance actions link adjacent visual-field items.

In the full mode, the schema mechanism constructs most of the schemas shown in figure 4-8, but it does not realize the entire network.

4.1.4 Foveal relations

The visual-detail items in the fovea also have adjacency relations; when an image shifts from one foveal region to another, the details of its appearance shift correspndingly. The extended result of the bare schema for each incremental glance action (such as /eyer/ in figure 4-9) notes the relevance of each visual detail item, spinning off schemas such as /eyer/fovx12 and /eyer/fovf30.

The extended context of each such schema seeks conditions that make the schema's result follow reliably. For some schemas, such as /eyer/fovx12, the corresponding visual-detail item in an adjoining retinal region serves as such a condition; thus, for example, the schema fovr12/eyer/fovx12 spins off (figure 4-10a), and similarly for other actions, regions, and details.

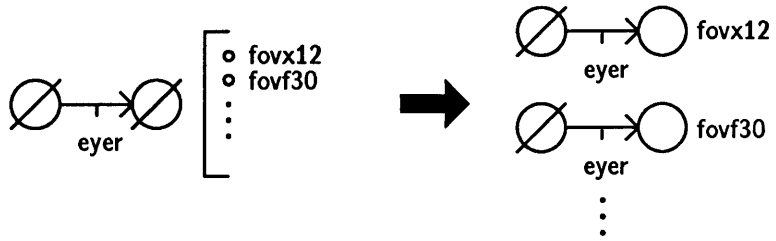


Figure 4-9: A glance-action schema discovers visual-detail results.

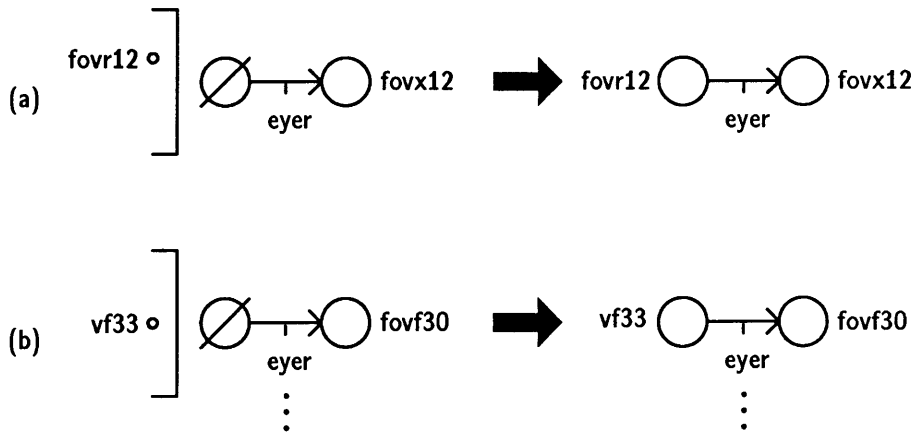


Figure 4-10: Glance-action schemas discover contexts for visual-detail results.

For other schemas, such as /eyer/fovf30, there is no visual-detail item to confer reliability, since vf33, the region immediately to the right the forward foveal region, is not itself a foveal region, and thus conveys no visual detail. The extended context of /eyer/fovf30 does identify the coarse item vf33 as a relevant condition, leading to the construction of vf33/eyer/fovf30 (figure 4-10b). This schema, though still unreliable, is much more reliable than the unconditional /eyer/fovf30.

The extended context of /eyer/fovx12 also identifies the result's adjoining coarse item (in this case vf32) as a relevant context condition. If vf32 spawned a spinoff schema vf32/eyer/fovx12 before fovr12 spins off fovr12/eyer/fovx12, then the extended context of vf32/eyer/fovx12 would itself discover the necessity of the condition fovr12, constructing the schema vf32&fovr12/eyer/fovx12. Since fovr12 is never On unless vf32 is On, /eyer/fovx12 would not, in this case, ever spawn fovr12/eyer/fovx12, due to the provision, discussed in section 3.4.1, for suppressing redundant attribution by deferring to a more-specific applicable schema.

Two factors make it likely that /eyer/fovx12 will spawn fovr12/eyer/fovx12 before spawning vf32/eyer/fovx12. First, the mechanism may encounter an object lacking the visual feature 12 whose image passes from the right-foveal region to the central-foveal region when the action eyer is taken. In that case, fovx12 makes a larger observable contribution to the reliability of /eyer/fovx12 than does vf32, so the extended context will detect the relevance of fovx12 sooner. Second, if instead the relevance of the two items is detected concurrently, the mechanism prefers to create a spinoff for the more specific context condition (section 3.4.1 again), again favoring fovx12.

Once fovr12/eyer/fovx12 exists, the context condition vf32 does not give rise to further spinoffs for the action eyer and result fovx12: /eyer/fovx12 cannot spawn vf32/eyer/fovx12, because of redundant-attribution suppression; and fovr12/eyer/fovx12 does not spawn vf32&fovr12/eyer/fovx12, because there is no measured improvement in the reliability of fovr12/eyer/fovx12 when vf32 is On rather than Off (indeed, fovr12/eyer/fovx12 cannot even be tested when vf32 is Off, since fovr12 must then

be Off too, making the schema inapplicable). Thus, the mechanism avoids the inefficiency of building a conjunctive context here when a single item suffices.

That inefficiency does arise in another way, however. If, in the trials encountered so far, every object with feature 12 also has, say, feature 40, then from the point of view of the extended context of /eyer/fovx12, there is no distinction between the relevance of fovr12 and fovr40. The extended context discovers both conditions simultaneously, and it is a matter of chance which then spawns a spinoff schema. If fovr12 happens to be chosen, then the other condition becomes extraneous (like the condition vf32 in the exmple above). If fovr40 happens to be chosen instead, spawning the schema fovx40/eyer/fovx12, then fovr12 instead appears to be extraneous—until the mechanism encounters an object that has feature 12 but not 40. When, on some occasion, glancing left shifts that object's image from the rightmost foveal region to the central foveal region, /eyer/fovx12 can spawn fovr12/eyer/fovx12; since fovr40 was Off, it does not suppress the attribution of relevance on this trial to fovr12.

4.1.5 Elaborating the proprioceptive fields

Incremental glance actions affect visual proprioceptive items as well as visual-field items. Schemas such as vp20/eyef/vp21 express the adjacency of visual proprioceptive items by designating their connectivity with respect to incremental glance actions. (I omit the details of this schema's derivation, which is similar to the examples above.)

Such schemas link the visual proprioceptive items into a network (similar to the visual-field network in section 4.1.3) that elaborates their spatial structure (figure 4-11). This network provides a chain of schemas from any given eye orientation to any other, conferring the ability to shift from any orientation to any other.

Similarly, incremental hand actions affect haptic (hand) proprioceptive items; for example, hp23/handl/hp13 shows the adjacency of hp23 and hp13. Such schemas form yet another network (figure 4-12) that implements a practical description of the spatial arrangement of the haptic proprioceptive items. (This network has a hole at

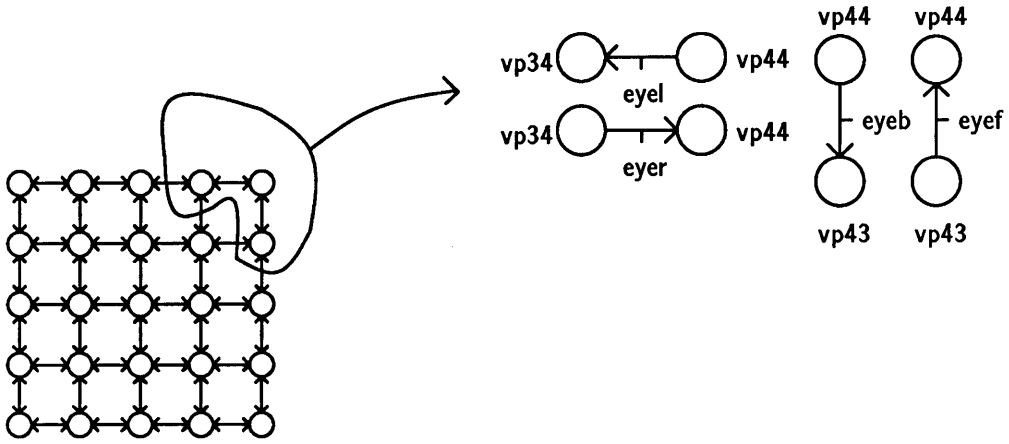


Figure 4-11: Schemas with incremental glance actions link adjacent visual proprioceptive items.

hp02; that body-relative position is inaccessibile because the body is there.)

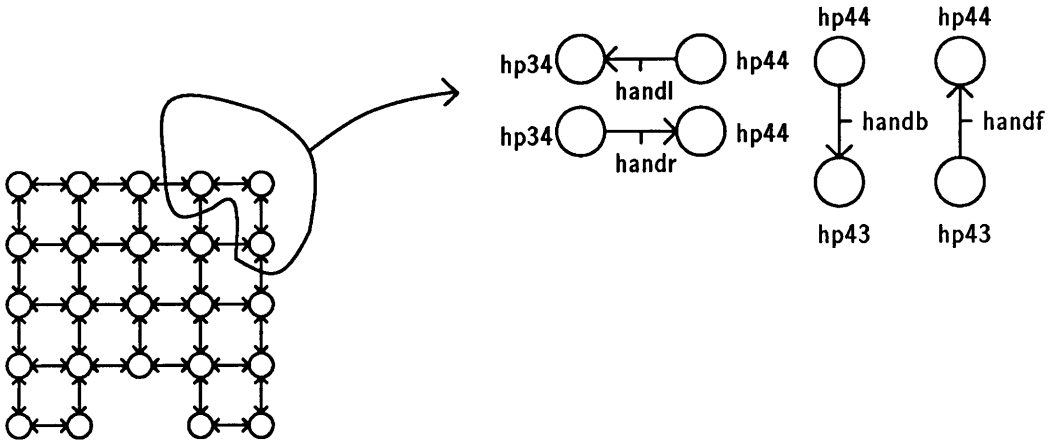


Figure 4-12: Schemas with incremental hand actions link adjacent haptic propriceptive items.

As with the visual-field netwok, the full mode realizes most, but not all, of the schemas in these networks.

4.1.6 Negative consequences

Shifting the position of the hand, the glance, or a visual image not only establishes a new position, but also eradicates the prior position. Schemas like those in figure 4-13 designate such consequences.

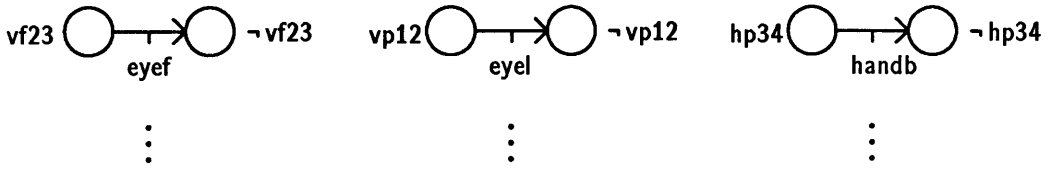


Figure 4-13: Moving to a new position eradicates the old one.

4.1.7 Positional actions

Each of the proprioceptive items linked in the above networks is readily accessible—there is a chain of schemas from any glance orientation to any other, and from any hand orientation to any other. Each such item can thus become the goal state for a composite action—the action of achieving that glance or hand orientation. As stated in section 3.4.2, for each newly defined composite action, the mechanism also builds a bare schema which has that action. Figure 4-14a, for example, shows the bare schema with a composite action whose goal state is **hp22**; the action's component schemas are those of the network in figure 4-12 above. The action **hp34**, shown in figure 4-14b, has the same component schemas—as do all the other composite actions with haptic proprioceptive goal states.

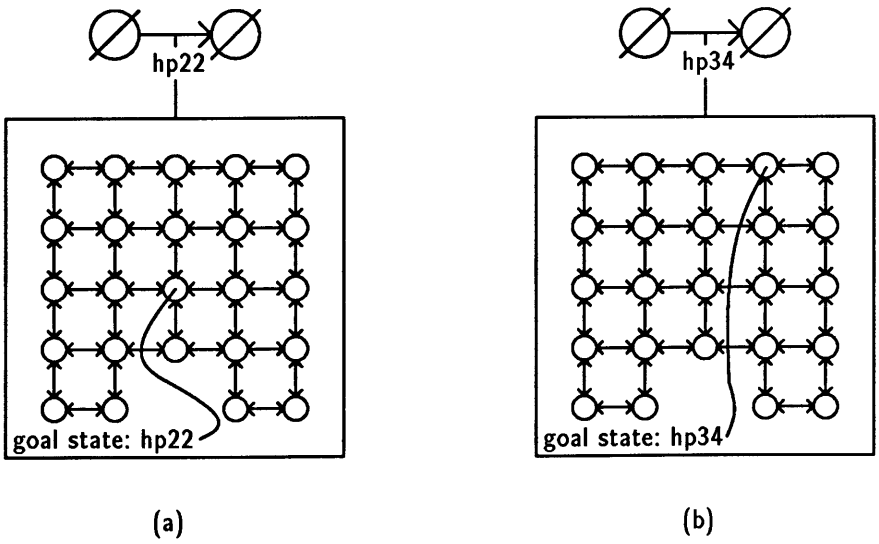


Figure 4-14: Composite actions form for various hand positions. Each defines the action of bringing the hand to that position.

These proprioceptive-goal hand actions are *positional*, in contrast with the primitive hand actions, which are incremental. Activating a given positional hand action moves the hand to a particular position, regardless of where the hand started.

Similarly, the schemas that link together the visual proprioceptive items make each of those states accessible, enabling each to be the goal of a composite action, as illustrated in figure 4-15; the actions' component schemas are not shown. These composite actions are positional glance actions, again in contrast with the incremental primitives.

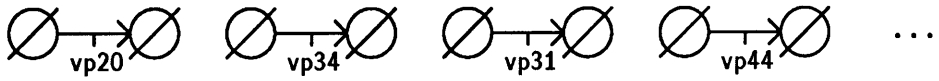


Figure 4-15: A composite action forms for various glance orientations. Each defines the action of shifting the glance to that orientation.

Finally, the schemas that link adjacent visual-field items also provide a basis for the definition of composite actions with those items as goal states (figure 4-16). Each such composite action is the action of shifting an image to a particular region of the visual field.

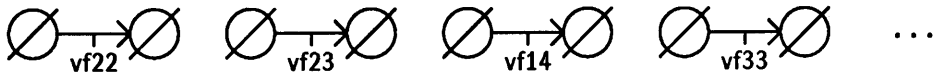


Figure 4-16: A visual-field action shifts an image to a particular region. The first two actions shown are foveation actions.

Of particular interest are the *foveation* composite actions: a foveal action shifts an image to one of the foveal regions of the visual field. Foveal actions permit the visual details of an object to become apparent.

The visual-detail items themselves become goal states of composite actions (figure 4-17). Most objects exhibit a number of visual details, which therefore tend to co-occur when the object's image appears at some foveal region. This could lead to an n^2 proliferation of schemas, in which each visual-detail action claimed each co-occurring visual detail as a result. The restriction discussed in section 3.4.2—

preventing result spinoffs for explained result transitions or unexplained actions—substantially mitigates this proliferation. Still, many such schemas do form, because the various visual-detail events do not become explicable simultaneously; some are designated the goals of composite actions before others are explained. (Section 6.2 raises the possibility of purging such schemas.)

Figure 4-17: Visual-detail items also define foveation actions.

4.2 Steps toward intermodal coordination

The schemas documented above set forth a substrate for the practical representation of visual and proprioceptive spatial knowledge. Other schemas begin to describe the relationships among these domains—in particular, the visual manifestations of moving the hand, and coordination between sight and touch.

4.2.1 Visual effects of incremental hand motions

Moving the hand while it is in view affects where the hand's image appears in the visual field. If the motion occurs within the foveal region, the visual change can be reliably predicted (if the hand is only peripherally visible, its identity as the hand is uncertain). As shown in figure 4-18, a bare incremental hand-motion schema such as /handl/ spawns schemas that show the hand appearing at various visual-field regions (eg, /handl/vf12, /handl/vf30; similarly, of course, for other hand actions and visual regions).

Some of these schemas denote the motion of the hand from a foveal visual region, others from a peripheral region. In the case of a peripheral origin, the best that the extended context can do is to discover that an image at the appropriate adjoining region is a relevant context condition; for example, /handl/vf30 spawns vf40/handl/vf30

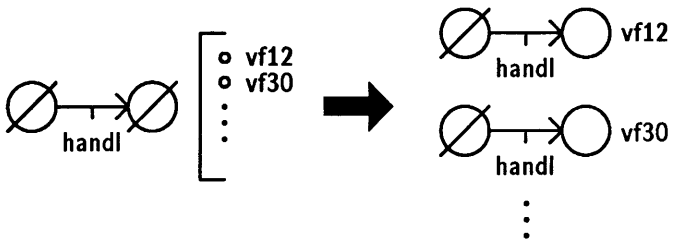


Figure 4-18: These are initial schemas for depicting the visual effects of hand motions.

(figure 4-19). This schema is unreliable, since the object seen at **vf40** need not be the hand.

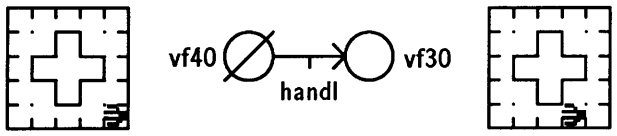


Figure 4-19: The motion of the hand's visual image is unreliably predicted from its peripheral appearance.

In contrast, the extended context of, say, **/handl/vf12** discovers the relevance of various visual features of the hand when it appears at the adjoining foveal region **vf22** (figure 4-20a). The relevance of the item **vf22** is also noted, but only after some of the details that are unique to the hand are found relevant; **vf22**, and details less specific to the hand, also obtain when objects other than the hand are at **vf22**, and thus make a smaller difference to the with-without comparison made by the extended context data. Therefore, **/handl/vf22** spawns a context that best distinguishes the hand from other objects seen at **vf22** just before moving the hand to the left.

This process culminates in the schema **SeeHand@22/handl/vf12** (figure 4-20b), where **SeeHand@22** is shorthand for a conjunction of visual features that suffices here to distinguish the hand from other objects seen at **vf22** before moving the hand left; additional visual details of the hand do not further measurably increase the reliability with which the action follows the result, and are not relevant context candidates—though they would gain this status if other objects were encountered that share with the hand the details noted so far. (In the restricted mode, the hand and the body

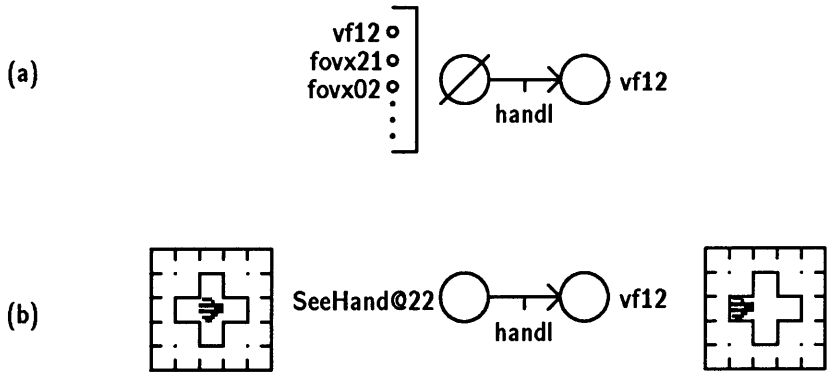


Figure 4-20: When the hand appears in the fovea, the destination of its image is reliably anticipatible.

are the only objects that appear in some positions, and a single visual detail suffices to distinguish them from one another.)

Similarly, schemas such as SeeHand@23/handb/SeeHand@22 in figure 4-21 chain together to say how to move the hand so as to move the hand's image among the foveal regions. (SeeHand@23 is shorthand notation like SeeHand@22.)

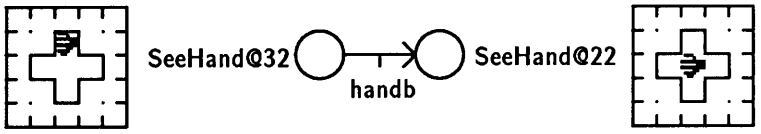


Figure 4-21: Moving the hand moves its image acros the fovea.

The schemas just described are among the last built in the full mode before the implementation runs out of memory. Except where otherwise noted, the remainder of the synopsis describes the restricted mode.

4.2.2 Touching what's seen beside the hand

Sometimes, moving the hand not only shifts its visual image, but also results in tactile contact. The schema in figure 4-21, for example, discovers this additional result, spawning the schema in figure 4-22a. That schema is able to discover a condition that confers reliability on the tactile result: that an object be seen next to where the hand will move. The schema in figure 4-22b incorporates that condition in its

context. (In fact, the restricted-mode runs out of memory just before building this schema; by the end of the run, the prior schema's extended context has taken note of the condition's relevance, but the threshold required to spawn a new schema has not been reached.)

This schema would tell the mechanism how to touch what is seen (at a particular place in the visual field)—provided that the hand is in view at a foveal region from which there is a chain of schemas for moving the hand to be seen beside the object. Thus, the ability to grasp the object is only applicable when the object and the hand are both in view. This limitation corresponds to an early form of grasping observed by Piaget (section 2.2). Section 4.4.1 shows how the schema mechanism might overcome this limitation.

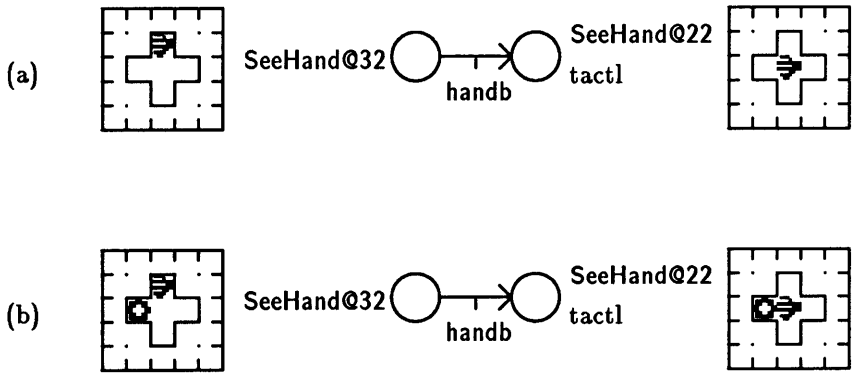


Figure 4-22: Moving the hand, and its image, results in tactile contact if an object is present next to the hand's destination.

There is another route that starts in the direction of developing of a schema for touching what's seen near the hand, but turns out to be a dead end. The schema /handb/, for example, spawns the unreliable schema /handb/tactl. That schema, however, is unable to make progress toward identifying visual conditions that confer reliability on the schema. The problem is that the object being touched could appear anywhere in the visual field (or could fail to appear at all); its appearance at a given visual region makes tactile contact no more likely than its appearance at any other region (and not measurably more likely than when it does not appear, given that an

object is likely to be nearby even if not in view). Similarly, the hand's appearance at a given region makes tactile contact no more likely than for another hand position.

Any *conjunction* of an image's appearance at one region and the hand's appearance at the appropriate nearby region would confer reliability on the schema /handb/tactl. But, in this case, the marginal attribution facility is unable to build incrementally to any of the required conjunctions, because the conjuncts, taken individually, do not enhance reliability. The schema mechanism breaks this impasse by discovering that tactile contact results from the activation of a schema whose context already designates the appearance of the hand at a particular visual-field region; given that context, the mechanism can discover where an object must appear to be touched by moving the hand forward.

4.2.3 Bringing the hand into view

The schema mechanism learns how to bring its hand into view. Being able to do so extends the ability to touch what is seen near the hand to the general ability to touch what is seen.

Unreliable schemas such as /hp23/SeeHand@22 (figure 4-23a) reflect the fact that seeing the hand at the center of the fovea sometimes results from that positional hand action. This schemas is reliable for a particular glance orientation (which must be sustained through the positional hand action's execution); thus, the above schemas spawns vp23/hp23/SeeHand@22. By constructing a number of such schemas, the mechanism in effect builds a dispatch table that says, for each of several glance orientations, where the hand must be put (relative to the body) to appear in the visual field when the glance is in that orientation.

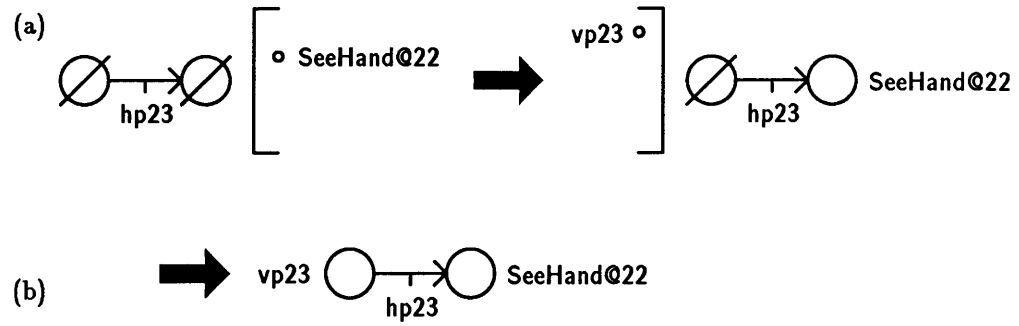


Figure 4-23: Schemas that dispatch from glance orientation move the hand into view.

4.3 Beginnings of the persistent-object concept

At this point, schemas are structured so as to provide rudimentary representations of the spatial relationships of both external and proprioceptive sensory data, both within and between the various sensory modes. This knowledge was acquired through action, and its embodiment is practical: it is knowledge of how to act and what to expect to happen. But the content of these schemas is not only procedural: the coordination of hand motions and eye motions, of seeing and feeling, begins to describe the nature of objects and space; sight and touch begin to be known as coordinated properties of external objects.

All this boasts respectable progress from the schema mechanism's initial endowment of knowledge, in which all actions and items were devoid of any meaning to the mechanism. Still, it remains to transcend the rendering of reality only in terms of sensory and motor primitives. If an object is not perceived, then as far as schema mechanism is concerned, it has ceased to be—there are no items whose state signifies the thing's continued existence. And similarly, an object's specific identity is immediately forgotten when its distinguishing features (e. g. visual details) cease to be perceived, even when some (partial) perception of the object persists. Next, the schema mechanism begins to synthesize items to represent these persistent states in their own right.

4.3.1 Palpable and visible persistent objects

Various positional hand actions, e. g. hp22, sometimes result in tactile contact, e. g. tactl. The schema /hp22/tactl reflects this occasional result (figure 4-24). As in the example of section 3.4.3, this schema is unreliable—it only succeeds when an object happens to be present at position hp12. But the schema is locally consistent—if it succeeds on some occasion, it probably will succeed again if activated again soon, since nearby objects tend to stay put for a while. Without understanding that reason, of course, the schema nonetheless discovers, over a number of trials, that it is indeed locally consistent; and the schema determines the expected duration of its local consistency, the average time that the schema remains valid (in this case, the average time that an object stays put there).

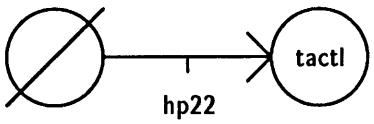


Figure 4-24: Moving the hand to a particular position sometimes results in tactile contact.

Such schemas develop for other hand positions as well. Each such schema serves as host to a synthetic item that designates a persistent palpable object at a particular body-relative position. For example, the host schema /hp22/tactl acquires a reifying synthetic item that I'll call PalpableObj@12 (figure 4-25). The host schema's positional hand action serves as probe, the schema's tactile result as manifestation, of the condition reified by the synthetic item—the condition of there being a palpable object beside that hand position.

Analogous synthetic items designate persistent visible objects. For example, the unreliable, locally consistent schema /vp21/vf13 (figure 4-26) reveals a manifestation of a visible object—seeing it at visual-field center—by the probing action of glancing at a particular body-relative position. The reifying synthetic item, which I'll call VisibleObj@12, designates a persistent visible object at that position.

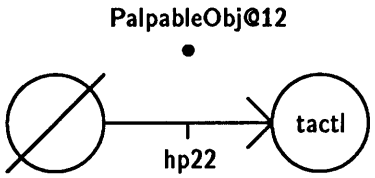


Figure 4-25: This synthetic item designates a persistent palpable object at a particular position.

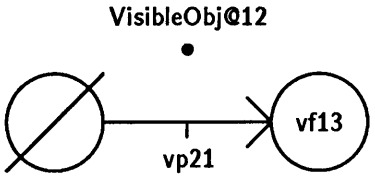


Figure 4-26: This synthetic item designates a persistent visible object at a particular position.

Tactile and visual details also serve as manifestations of conditions representable by synthetic items. Figure 4-27 shows some synthetic items that designate the persistence of the particular details associated with objects at particular locations. Such items let the mechanism represent more than the continued existence of an unperceived object—its persistent identity is now representable as well, at least to the extent that its apparent details specify its identity.

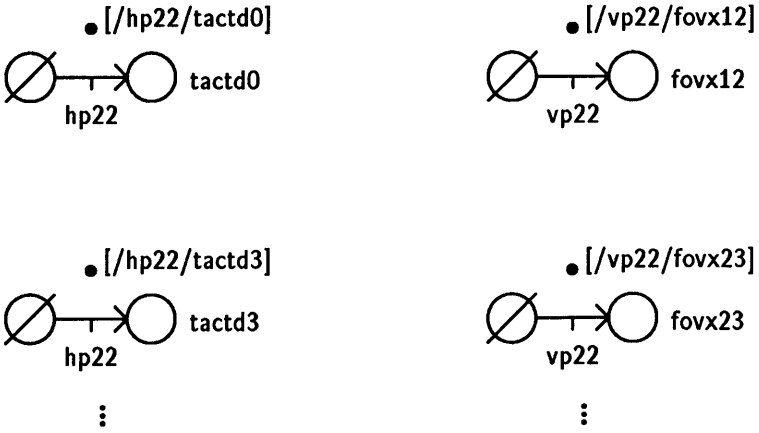


Figure 4-27: Some synthetic items correspond to a persistent object's specific identity.

In a similar vein, some items designate the identity of an object whose image has moved to the visual periphery, rendering its details inaccessible. For example, in

the full mode, the schema vf33/eyel/fovf21 is unreliable but locally consistent, since on occasion the only visible object has visual detail 21; shifting that object's image back to the fovea recovers the detail. The item [vf33/eyel/fovf21] (figure 4-28) reifies this state of affairs; when that item is On, the mechanism effectively remembers which object is now seen peripherally, even though that information is no longer perceptually apparent. In contrast with the previous examples of remembered details, this representation has the advantage of being independent of the (body-relative) position of the object in question; but it has the disadvantage of being easily confused when several objects are present, since it might be a different object whose visual image makes the host schema applicable.

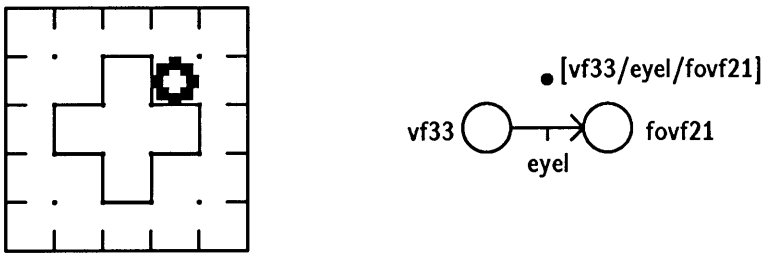


Figure 4-28: This synthetic item remembers a visual detail of an object no longer visible at the fovea.

A more realistic visual system would persent some degree of visual detail even at the periphery. Schemas incorporating such detail in their contexts could host synthetic items that remember additional detail; this would be more reliable, since the included details might distinguish among several objects in view at once.

4.3.2 Coordinating visible- and palpable-object representations

The synthetic items PalpableObj@12 and VisibleObj@12 in the previous section actually designate the same state as one another—the state of there being an object at body-relative position 12. This is not true in all possible worlds; in principle, invisible or intangible objects could exist. Were there such objects, a palpable object at some

location would not assure the presence of a visible one, or vice versa. But since such objects do not exist, PalpableObj@12 and VisibleObj@12 are in fact *coextensive*—whenever one of those states obtains, so does the other.

At first, the schema mechanism is unaware of these items' coextension. What turns On PalpableObj@12 need not affect VisibleObj@12, or vice versa; initially, in fact, each is turned On only by the successful activation of its own host schema. Thus, the two items can be in opposite states at the same time; like a third-stage Piagetian infant, the mechanism may know that it can touch a currently unperceived object, but not know that it can look at it—or vice versa, depending on the modality by which the object recently manifested itself to the mechanism.

But the mechanism begins to learn that these items are effectively synonymous. As described in section 3.4.3, a synthetic item's state is maintained in part by reliable context spinoffs spawned by the host schema. Such schemas' contexts specify conditions under which the host schema is reliable; hence, its reifier turns On. In this case, the host schema /vp21/vf13 spawns the reliable spinoff schema PalpableObj@12/vp21/vf13; conversely, /hp22/tactl VisibleObj@12/hp22/tactl *almost* spawns (figure 4-29) (the context correlation is noticed, but the limit of memory is reached before the correlation reaches the threshold for spawning a spinoff schema). Other pairs of synonymous synthetic items designating objects at other body-relative positions can likewise be coordinated by their host schemas' extended contexts.

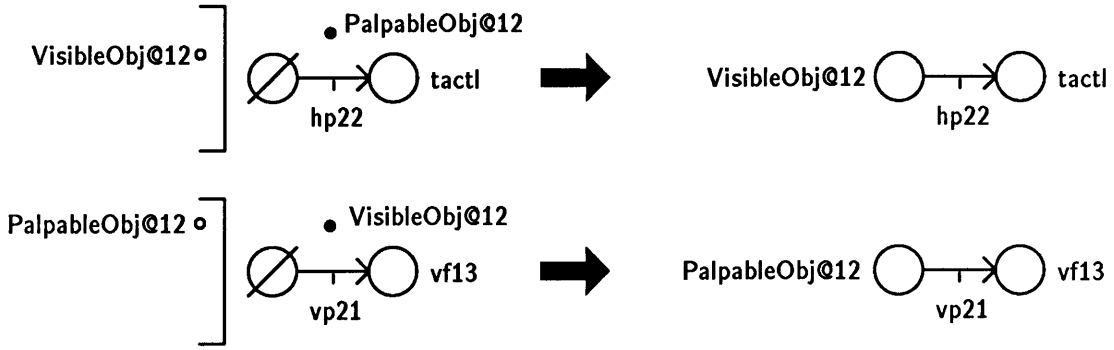


Figure 4-29: palpable-object representations admit visual evidence, and vice versa.

When this coordination is achieved, each host schema's extended context also determines that when the other synonymous item is Off, that host schema is unreliable; hence, its reifier turns Off. When two items thus help maintain one another's state, there is a danger of oscillation when their states differ (and if their states never differed, neither would ever be in a position to turn the other On or Off—the other would already be in that state). Two factors prevent such oscillation: 1) that each item is in an Unknown state when it has received no recent evidence; and 2) as discussed in section 3.4.3, host-trial evidence takes precedence over other verification conditions, and memory of previous evidence yields to current evidence; hence, seeing or touching an object—or failing to when looking or reaching for it—definitively sets the state of one of the two synonymous items, which then sets the state of the other.

Intermodal persistent-object schemas are among the last structures built during the restricted mode before the implementation runs out of memory.

4.4 Hypothetical scenario of further developments

This section describes some hypothetical further achievements of the schema mechanism—developments that build directly on the substrate of knowledge that the implementation has in fact constructed, and that would perhaps be exhibited if the same software were to run on a larger machine. I present these hypothetical developments both to call attention to what the implementation has not yet achieved, and to specify part of a target scenario for future work.

4.4.1 Touching what's seen, and vice versa

Schemas that chain to the state of seeing the hand at a particular region of the visual field—for example, SeeHand@22—lead to the construction of a composite action with that goal state. The bare schema that has that action (figure 4-30a) discovers that various tactile events, e. g. tactl (touching the left side of the hand), occasionally

result from this action, as expressed by the schema /SeeHand@22/tactl (figure 4-30b). This schema's extended context discovers the condition needed for the schema to be reliable: the condition vf12, which designates an image seen beside vf22, the place to which the action will bring the hand.

Figure 4-30: Moving the hand, and its image, sometimes results in tactile contact.

The schema vf12/SeeHand@22/tactl then forms (figure 4-31). This schema enables the mechanism to touch what is seen (at a particular place in the visual field). Unlike the schema in figure 4-22, this schema can be used to touch an object even if the hand is not seen beside it—because the action SeeHand@22 includes components for bringinf the hand into view (such as the schema shown in figure 4-23b).

Figure 4-31: This schema tells the mechanism how to touch an object seen at vf12.

The composite action SeeHand@22 acquires other component schemas besides hand-motion schemas. In particular, eye-motion schemas can serve instead to shift the hand's image. It would be unhelpful for the action's controller to select eye-motion components in this case; shifting the gaze to see the hand at vf22 brings the hand no closer to the object that had been seen at vf21. But the context-preservation feature, discussed in section 3.4.1, suppresses such components: the extended context of vf21/SeeHand@22/tactl discovers that the condition vf21 must be sustained throughout the schema's execution; and the schema's activation thereafter suppresses any action that would negate that condition. The applicable schema vf21/eyer/$\neg$vf21, for example, indicates that eyer would negate that condition, and so suppresses the action eyer, which in turn suppresses the activation of any schema that has that action.

(Similarly, of course for the other incremental eye actions.)

Alternatively, when the goal state SeeHand@22 is realized by glance actions rather than hand actions, it occasionally has the result vf21. The unreliable schema /SeeHand@22/vf21 spawns the reliable schema tactl/SeeHand@22/vf21 (figure 4-32), which sustains its context condition tactl throughout its execution, assuring that the action's goal state is pursued by eye movements rather than hand movements.

Figure 4-32: This schema facilitates looking to see what the hand is touching.

Thus, the mechanism (hypothetically) achieves bidirectional visual-tactile coordination: the ability to touch what is seen and to look at what is touched.

Some coordination between visual and tactile details also arises. That is, an object's appearance sometimes predicts its texture, and vice versa. Figure 4-33 shows schemas that express such predictions.

Figure 4-33: Some schemas coordinate visual and tactile details.

4.4.2 Grasping and moving objects

Touching what is seen is a useful precursor for grasping and manipulating what is seen. Figure 4-34a shows a schema for moving a grasped object incrementally, as expressed in terms of the visual manifestation of that motion. Figure 4-34b shows a schema that expresses similar knowledge in terms of persistent objects.

Moving an object not only puts it in a new place, but also removes it from its previous place. Schemas such as those in figure 4-35 express such knowledge. These are similar to, but more sophisticated than, the schemas in section 4.1.6 that show

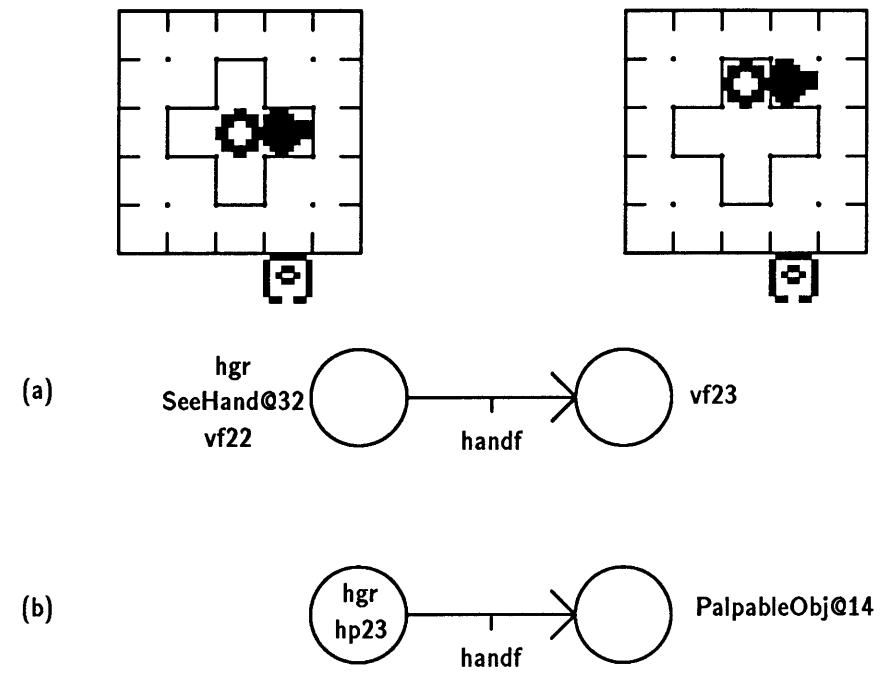


Figure 4-34: These schemas depict object motion, visually or in terms of persistant-object representations.

the negation of previous hand and glance orientations, and of visual images, following hand or glance actions. The earlier schemas expressed direct, nearly unconditional results of primitive actions on primitive items; the present schemas, in contrast, express effects on grasped objects, and are therefore subject to context conditions that describe the appropriate graspedness.

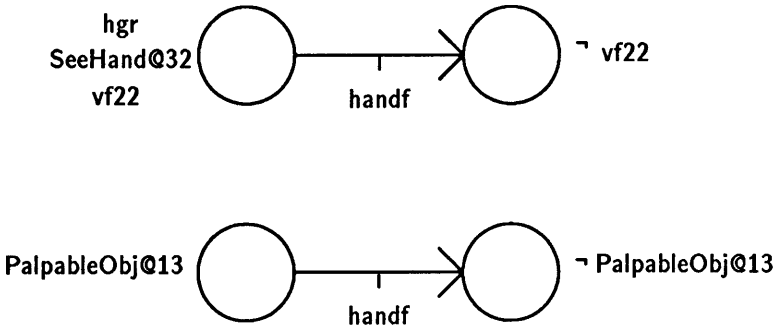


Figure 4-35: Moving an object removes it from its previous position.

4.4.3 Hidden objects

Suppose the microworld were modified so that an object is obscured from the animaton's view if some other object (other than the animaton's own body) lies directly behind it (figure 4-36). (This ad hoc modification clashes with thinking of the visual field as providing a bird's-eye view.) This modification introduces the problem of hidden objects.

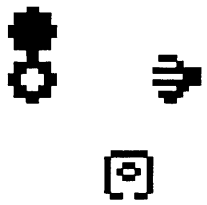


Figure 4-36: The hollow object hides the solid object from view.

If the animaton centers its gaze on the location of the hidden object (figure 4-37), the schema /vp23/vf22, host to the synthetic item VisibleObj@23, implicitly activates, but its result fails to obtain. The synthetic item thus turns Off. Due to the coordination described above in section 4.3.2, the synonymous item PalpableObj@23 also turns Off—incorrectly, since the hidden object is still palpable. Thus, the mechanism, like a third-stage Piagetian infant, is ignorant of the possibility of touching the hidden object.

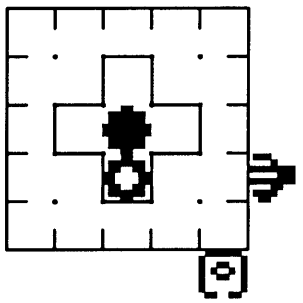


Figure 4-37: The animaton looks directly at the solid object, but does not see it.

One expression of an object's persistence while hidden appears in figure 4-38. A schema for moving the hand while grasping the obstacle, thus displacing the obstacle,

has the unreliable but locally consistent result of causing the manifestation to reappear. This schema could serve as host to a synthetic item that designates an object hidden behind the obstacle.

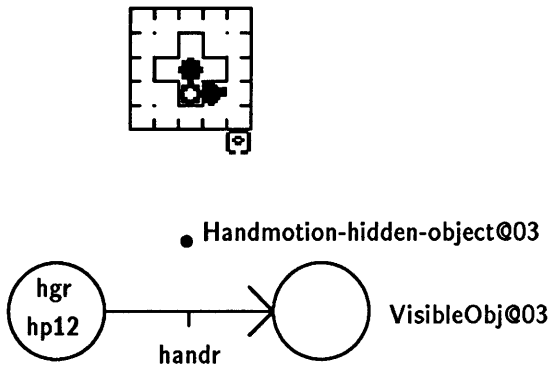


Figure 4-38: Displacing the obstacle reveals a hidden object, locally consistently.

This representation would be vulnerable to making a curious mistake. Suppose the host schema activates successfully, turning the hidden-object item On. Next, the previously hidden object moves to a new location, in full view of the mechanism. However, the hidden-object item remains On; there has been no unsuccessful activation of the host schema to turn it Off. If the object were now hidden behind another obstacle at its new position, and the first obstacle returned to its original position, the mechanism could exhibit a Piagetian fourth-stage place error (section 2.4) by still expecting to be able to find the object behind the first obstacle.

This place error can be corrected by representing the displacement of the obstacle on a less subjective level of abstraction. The schema in figure 4-39a has a composite action designating the very displacement of the obstacle, rather than using the primitive hand action of the previous host schema. Now, if the object moves to another location after its original hidden position is uncovered, the continuing uncoveredness of that position entails the continuing implicit activation of the schema whose action is that there be no obstacle covering that position. As soon as the object moves away from the uncovered position, that implicit activation is an *unsuccessful* activation, which turns Off the associated synthetic item. (To eradicate the place error, the

more subjective host schema must come to designate the new item as a synonym, as shown in figure 4-39b.)

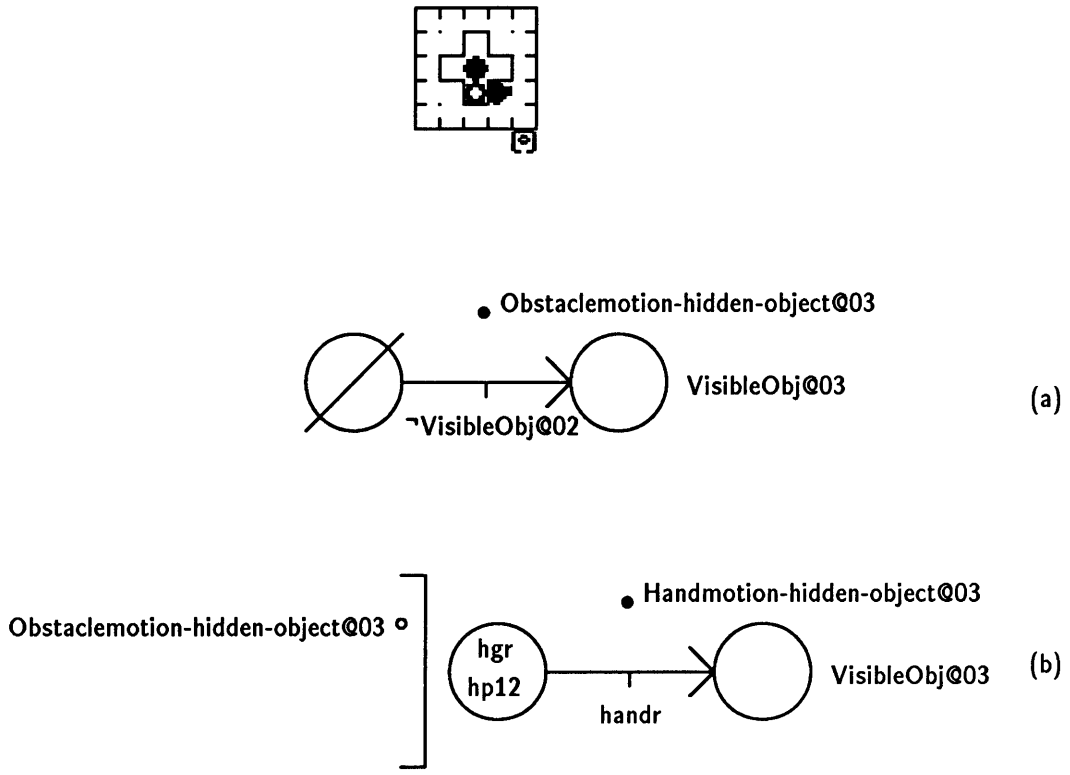


Figure 4-39: A more objective representation of the action fixes fourth-stage place error.

4.4.4 Large-scale Space

The spatial framework developed in the scenario is centered on the mechanism's own body; proprioceptive inputs serve as spatial coordinates. This framework suffices for a stationary infant. But after a while, the infant begins to crawl and then walk. By displacing herself, the infant moves *all* the objects in her body-relative space (by moving that space itself). But in externally-based, large-scale space, it is the infant that moves, while the other objects remain still. Coordinates in large-scale space are not given proprioceptively, but can be expressed in terms of fixed landmarks. The infant herself is just one of many objects that can move about in large-scale space.

It is plausible to imagine the schema mechanism going on to build a representation of large-scale space much the way it constructs its "personal" spatial framework (though I have worked out no details for this). The accessor condition for an object at some landmarked position in large-scale space is to move oneself to the landmark there; the manifestation is for the object then to be present in body-relative space. As in the scenario, extended views could be coordinated together, allowing an object's position also to be recognized by seeing the object from a remote landmark, or from a position between landmarks.[1]

One might regard the entire schema mechanism as essentially a large-scale-space facility, in which the terms of representing positions have been generalized—any primitive or constructed item can be used, not just views of landmarks—and the actions that connect places in the space have been generalized to arbitrary state-achievements, not just moving among landmarks. (Something like the schema mechanism may even have arisen, in the course of evolution, as a variation of a large-scale-space facility.) The scenario shows how this generalized large-scale-space facility can be "retrofitted" to the reconstruction of personal space; the extrapolations below speculate about the extension of this facility to the representation of more-abstract "spaces".

[1]Anecdote: I have a bad sense of direction. Riding up in the Tech Square elevator, I don't know which way I'm facing with respect to the building's surrounds, or with respect to the floor I'm heading to (though I know the orientation of that floor with respect to the building's surrounds; e. g. I know which way my office faces outside). For many years, I didn't realize there was an easy way to keep track of such things. At some point, it occurred to me that whenever I enter an area from which familiar landmarks are hidden, I can first take a moment to imagine a bird's-eye, x-ray view of the area, and use this to designate landmarks (usually walls) in the new area according to the familiar landmarks that lie beyond them. I then use the internal landmarks to keep track of my orientation, rather than trying to remember how many turns I've made since leaving the familiar surrounds.

The bird's-eye imaging has the flavor of a bridging schema that sets up an extended view (in terms of internal landmarks) of orientation with respect to external landmarks. Perhaps people with a good sense of direction have bridging schemas that get activated reliably in such situations; mine, alas, seem to require special urging.

4.4.5 Reality and beyond

One can imagine the schema mechanism constructing arbitrarily elaborate models of the current state of the world. But can such a mechanism possibly move beyond sensorimotor-level representations, to construct *episodic* memories that designate the state of propositions that concern things other than immediate physical reality? Consider, for example, the assertion that a certain object was in a given location *yesterday*, rather than *now*. Its position *now* can be expressed as the coordinated reification of the validity of various schemas, as discussed in the scenario. But *past* state cannot be similarly represented, unless there is some accessor condition by which some manifestation of the state can be revealed.

Sometimes this is the case, as when a seagull walking along the beach leaves tracks in the sand that indicate its past presence. But it is unusual for past events to be so obliging; or is it? One very general way for a past event to "leave tracks" is to be *remembered by a person.* In principle, one's memory of a past event can serve, for purposes of maintaining a synthetic-item state, as one "view" (among many intercoordinated ones) of that past state. Other possible views include:

- Physical remnants of the event, such as tracks in the sand.
- One's cognitive remnants of the event, other than explicit memories; for example, new abilities or insights acquired as a result of the event.
- *Other people's* cognitive remnants of the event, in the form of memories or new abilities, as manifested verbally and by other behavior.

Such views serve as fragments of a representation of past state; the real representation is their coordinated ensemble.

This may seem circular. After all, the representation of a past state cannot arise from a representation of a memory of the past state, since that memory requires that the past state is already representable. But then, in the same way, it is circular

to say that the representation of an object's presence at some location when hidden arises from a representation of the object's presence there when not hidden, since that assumes there is already a representation of the object's presence at that location. On closer examination, though, the circle becomes a spiral: the new representation is not stated in terms of *itself*, but rather in terms of a cruder approximation to itself. The scenario proposes the details of a few turns of that spiral for the development of the physical-object concept; with regard to the speculation about representing past states, I have no such details to offer. So, rather than giving a plausibility argument here, I am just presenting a bare-possibility argument. Still, the thought seems intriguing.[2]

This idea generalizes to the representation of abstractions. At a given moment, the state of any item in the schema mechanism is always some function of the past and present state of primitive, sensory items; hence, it would seem that an item can only represent some physical reality, as reflected in the sensory data on which the item's state (solely) depends. But whenever one's cognitive apparatus includes machinery to perform a certain computational task, it may be possible, by representing that machinery, to make (indirect) statements about the abstraction that the computation embodies. For example:

- A thing's *name* is accessed by (something like) holding it up to an adult and manifested by the adult's saying "That's an x".

- *Classification* is like naming, except that a class names only one of a thing's many attributes, so the same thing can belong to many classes. Piaget shows that the child gradually coordinates an extensive view of a class—defined by the actual set of members—with an intensive view—defined by a distinguishing attribute of all members of the class.

[2]Another possibility is to have a distinct episodic-memory module accessible to the central system. Traces of memories stored there could serve as one kind of manifestation of past events, to be coordinated with the others mentioned above. Something resembling Minsky's *k-lines* (Minsky 1986) might provide an interface for storing and retrieving central-system states as memories.

- The *cardinality* of a group of things is the number one arrives at by counting them—that is, by reciting a number sequence in synchronization with touching the objects, touching each exactly once. That process is the accessor, and the final number the manifestation, of the cardinality of the collection. The individual must discover that this is a persistent property of a collection—that if the process is repeated, one gets the same number (conservation of number). Note that this proposes an inductive basis for the discovery—that the individual notices, that cardinality is persistent, without understanding why. On the other hand, the induction can be carried out over a number of subactivated trials, so the generalization can be arrived at merely by thinking; it need not depend on actual events in the environment, as an extreme of the empiricist tradition has held.

- In formal reasoning, the *validity* of an argument (as opposed to the truth of its conclusion) has the accessor of inducing belief in the argument's premesis; the manifestation is believing the conclusion.

In each of these examples, a new abstraction is conceived in terms of how a person's computational machinery—one's own, or another's—behaves in some situation. The new conception reifies the set of circumstances under which a piece of one's computational machinery behaves a certain way.[3] Even more than with the physical-object concept, each such conception eventually requires an ensemble of many fragments of representation. As with the physical-object concept, Piaget presents snapshots of various incomplete versions of the eventual coordinations. In these, one sees bizarre bugs in a child's behavior that would be inexplicable if a more "appropriate" representation were in use—appropriate, that is, to the given representation considered

[3]This is not to say that one explicitly or introspectively thinks of one's conception of abstractions as being the representation of properties of certain machines; it need not "feel like" that is what is going on. One's explicit ideas about one's representations are implemented by other structures that express a theory about those representations; and that theory, like any other that one holds, can be arbitrarily far off base.

in isolation, rather than as part of the developmental system—just as the spatial nonlocality of hidden objects is a bizarre property of the infant's fourth-stage conception. The conservation protocol in section 2.7 is one striking illustration of this phenomenon.

In general, then, self-modelling might provide a route to the representation of abstractions, allowing an intelligent system to move beyond representing only the state of the physical world. It is interesting to compare this with Papert's (1980) speculation that access to the abstractions embodied in computers will profoundly change the way people think, once intellectually-accessible computer systems become widely available, especially to children; by working with the concrete embodiment of a computational abstraction, a person may appropriate a model of that abstraction for her own internal use. In effect, what I suggest here is that (much of) the necessary access to sophisticated computer systems has long been provided to people—in the form of people themselves. And the resulting cognitive revolution was quite as spectacular as what Papert predicted.

Additionally, modelling one's own mind, and others', is important in its own right. Understanding other people makes them more predictable and easier to interact with in beneficial ways. Understanding oneself provides the opportunity to better exploit one's own abilities, by forming model of strengths and weaknesses and means of improvement. And what we call consciousness requires a memory of the occurrence of a thought or experience, understood as such. Nothing in the schema mechanism's sensorimotor-level development, for example, qualifies as conscious. One might say metaphorically that the mechanism is aware of the things that it represents; but to take that awareness literally, in the sense of humanlike consciousness, would be to indulge in a kind of animism. Consciousness requires knowledge (and hence representation) of one's own mental experiences; this the schema mechanism does not come close to demonstrating.

Chapter 5

Virtual structures and mechanisms

The schema mechanism implementation has made rudimentary but encouraging progress in the direction of the Piagetian infant's development of the concept of physical object. Replicating this development is of special interest because of the possibility that it is just the earliest achievement of a learning mechanism with far-reaching capabilities. The present results certainly do not establish that the schema mechanism is capable of going far beyond its achievements so far; but the mechanism's arguable similarity to what is arguably a powerful human learning mechanism warrants at least the speculation that extended achievements are possible.

This chapter elaborates the speculation by exploring some hypothetical further activity of the schema mechanism—activity that is tantamount to the development by the schema mechanism of virtual structures and mechanisms. Some of the hypothetical developments presented here depend on two proposed (i. e. unimplemented) extenstions to the schema mechanism, subactivation and reciprocal-action identification, which are introduced below.

5.1 Virtual generalizations

Many conventional formal systems make it easy to express generalizations. In the predicate calculus, for example, one writes *For all x, $P(X)$ implies $Q(x)$.*, where P and Q are predicates that apply to some arbitrary object. From the foregoing proposition, and the proposition $P(a)$, $Q(a)$ follows; it can be deduced that a particular object a that satisfies P must also satisfy Q. Other systems, such as semantic networks and knowledge-representation languages, provide analogous ways to perform a deduction that instantiates a generalization, that is, that applies the generalization to a particular instance.

The schema mechanism has no comparable facility for expressing generalizations. Disconcertingly, the mechanism must re-learn essentially the same fact in numerous different guises, rather than learning it in a general form and deducing the instantiations. For example, learning about persistent palpable objects at a given body-relative position is independent of learning about them at other positions. Similarly, the effect of grasping a persistent object and then moving the hand incrementally must be learned seperately for each body-relative position; there is no automatic generalization from one position to another, and no way to parameterize position in order to express a more general, position-independent principle (such as *Moving an object at (x,y) incrementally right brings it to $(x+1,y)$*) which could then be instantiated for various particular positions (e. g. *Moving an object at* $(3,2)$ *incremetally to the right brings it to* $(4,2)$).

One approach would be to augment the schema mechanism with parameterized representations. But it is unclear how the mechanism itself might devise appropriate parameterizations. A parameterization scheme limited to a few built-in special cases would be of little use or interest.

Instead, I suggest a way that the schema mechanism might behave *as though* it expressed and instantiated generalizations; the mechanism might then be said to embody *virtual generalizations.* This capability is speculative; the scenario to follow has

not actually been demonstrated by the schema mechanism, and even the speculation relies in part on currently unimplemented extensions to the mechanism, as described below.

The realization of virtual generalizations relies on representations of the same event in different frames of reference—for example, representing an event visually, relative to the visual field, and also in terms of visible-object synthetic items, in body-relative terms. A representation with respect to a particular position in one field of reference—say, the visual field—applies to a number of different positions in the other frame of reference—in this example, the body-relative frame. Call these the *source* and *target* reference frames, respectively. A specific, fixed-position representation in the source frame thus implies a general, position-independent statement about the target frame. Shifting the glance orientation changes the mapping from source to target, instantiating the generalization at a different target position, as shown in figure 5-1.

Figure 5-1a shows a schema for moving a grasped object incrementally forward; the schema expresses the visual manifestation of this event (vf23 turning On). In each of the two examples in figure 5-1b, the glance orientation is such that a grasped object appears in the same part of the visual field as in the context of the schema in figure 5-1a. The schemas in figure 5-1b describe the same event as the schema in figure 5-1a, but in terms of persistent objects at particular body-relative positions, rather than in terms of visual appearance.

The visual-field view shown in figure 5-1a serves as a *canonical perspective* of the event that is represented. Suppose the mechanism learns that it is interesting to orient the glance so as to bring about a canonical perspective; in effect, the mechanism learns the heuristic, the rule of thumb, that achieving that perspective is a good idea. Given the opportunity, the mechanism will then tend to bring about that perspective. Doing so serves to instantiate the generalization with respect to whatever target position is mapped onto. If a canonical perspective is achieved by foveation, as in

this example, the primitive value associated with the visual-detail items promotes the canonical perspective: trying to turn On the foveal items achieves the canonical perspective. In other examples, the items designating the canonical perspective may achieve delegated value by virtue of the things of value that are made accessible by achieving the canonical perspective.

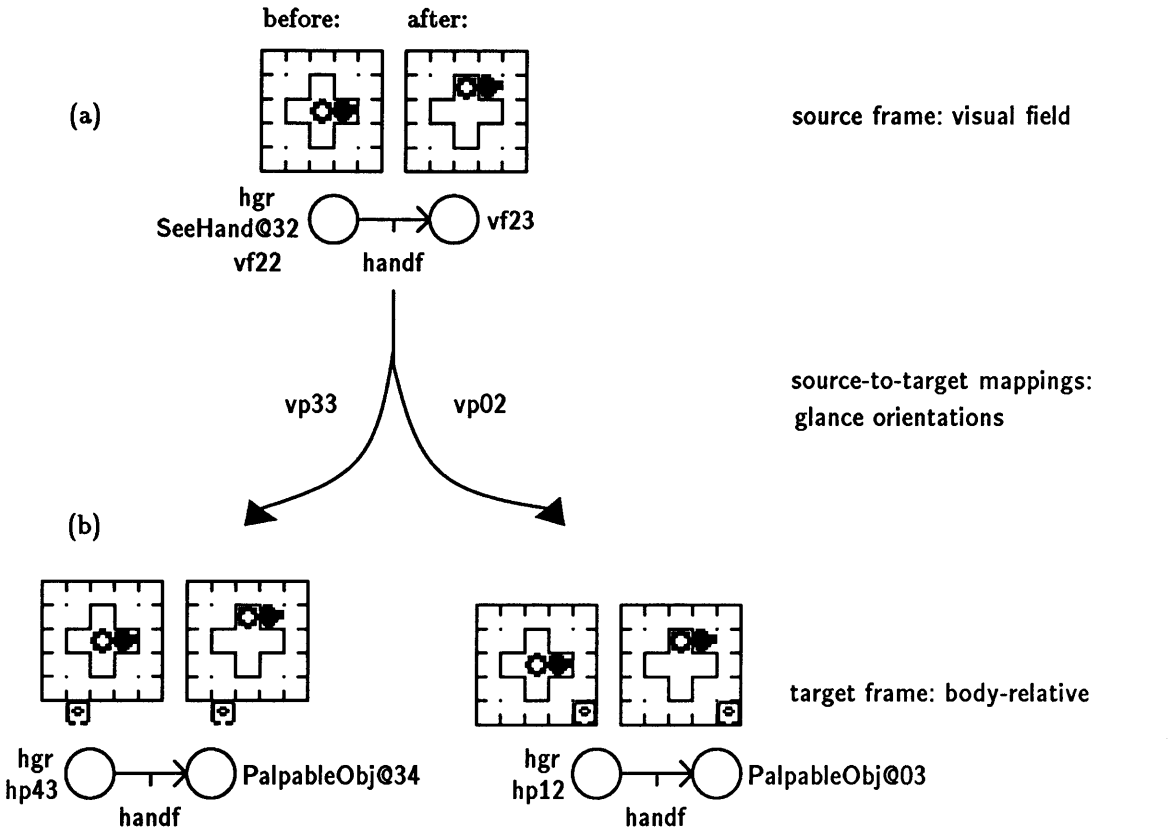


Figure 5-1: Virtual generalizations are instantiated by mapping one reference frame to another.

5.1.1 Implicit and explicit instantiation

A virtual generalization can be instantiated either implicitly or explicitly. *Implicit* instantiation merely consists of achieving the canonical perspective, making the source-frame schema applicable. *Explicit* instantiation consists of building a target-frame schema for that pertains to the current target-frame position. This requires no

special machinery; after a number of trials in the given target-frame position, the marginal attribution facility builds a schema describing the event in terms of the target frame—presuming, of course, the availability of target-frame representational elements (in this case, visible-object synthetic items). (Furthermore, such elements can themselves be formed by the instantiation of virtual generalizations, as discussed below.) Activation of the source-frame schema brings about the event that serves as the basis for building the target-frame representation, thus explicitly instantiating the virtual generalization for a particular target-frame position.[1]

To repeat, virtual generalizations and their implicit and explicit instantiations do not correspond to particular built-in features of the schema mechanism. Rather, they are epiphenomena, higher-level emergent tendencies of the mechanism. (They are also hypothetical, that is, not yet demonstrated by the implementation.)

Once a generalization has been explicitly instantiated in a target-frame schema, that schema can participate in a chain of schemas leading to some goal. Thus, the target-frame schema, like any other schema, can be identified by a rapid, parallel process as being of use for a given purpose at a given moment. In the case of implicit instantiation, there is no target schema to be so identified. Thus, the mechanism must heuristically perform an action to achieve a canonical perspective *before* the mechanism can recognize the generalization's pertinence. This requirement makes generalization by implicit instantiation is an inherently serial process, since it is not possible to adopt arbitrarily many perspectives simultaneously. But, since implicit generalization promotes explicit generalization, the initial slow serial process gives rise with repetition to the fast parallel process.

The computational space requirements of stamping out explicit instantiations of

[1]The idea of virtual generalization via canonical perspectives appears in (Drescher 1985), and in Agre and Chapman's notion of *indexical-functional* representations (Agre and Chapman 1987); see section 6.1.7. (More specifically, indexical-functional representations correspond to implicit, but not explicit, instantiation of virtual generalizations.) Agre and Chapman have implemented a system that plays a video game and that demonstrates the effectiveness of this alternative to conventional expressions of generalization. However, their system is entirely hardwired; it does not learn its own representations or skills.

virtual generalizations may prove burdensome. But the burden could be offset by encouraging the garbage collection of unimportant target-frame schemas. The mechanism could identify unimportant target-frame schemas by a conjunction of two criteria:

- *Rederivability.* The marginal attribution facility records each spinoff schema in the extended context or result of the spinoff's parent schema. This record suppresses subsequent, redundant attempts to spin off the same schema. The process could be modified to keep track of the frequency with which such redundant attempts are thwarted. A schema that is a target-frame instantation of a virtual generalization would tend to be the subject of frequent such attempts, promoted by implicit instantiation of the generalization. There could be a presumption in favor of garbage collecting readily rederivable schemas, on the grounds that they will tend to reappear as needed.

- *Importance.* A schema that serves as an explicit target-frame instantiation is important in proportion to its frequency of activation, and the value of the result in aid of which it is activated. If such a schema is used more frequently than rederivation attempts arise, then its garbage collection based on rederivability should be suppressed.

5.1.2 Generalizing to other positions in the same reference frame

Virtual generalizations depend on a mapping between source and target frames of reference. This mapping may be bidirectional; that is, the reference frame that sometimes serves as the source frame, used to express a virtual generalization, may at other times serve as the target frame, used to express an instantiation of a generalization from the other frame. For example, in figure 5-2, a body-relative schema now expresses a generalization with respect to the visual field; the body-relative source

frame here maps to the visual-field-relative target frame, rather than vice versa.

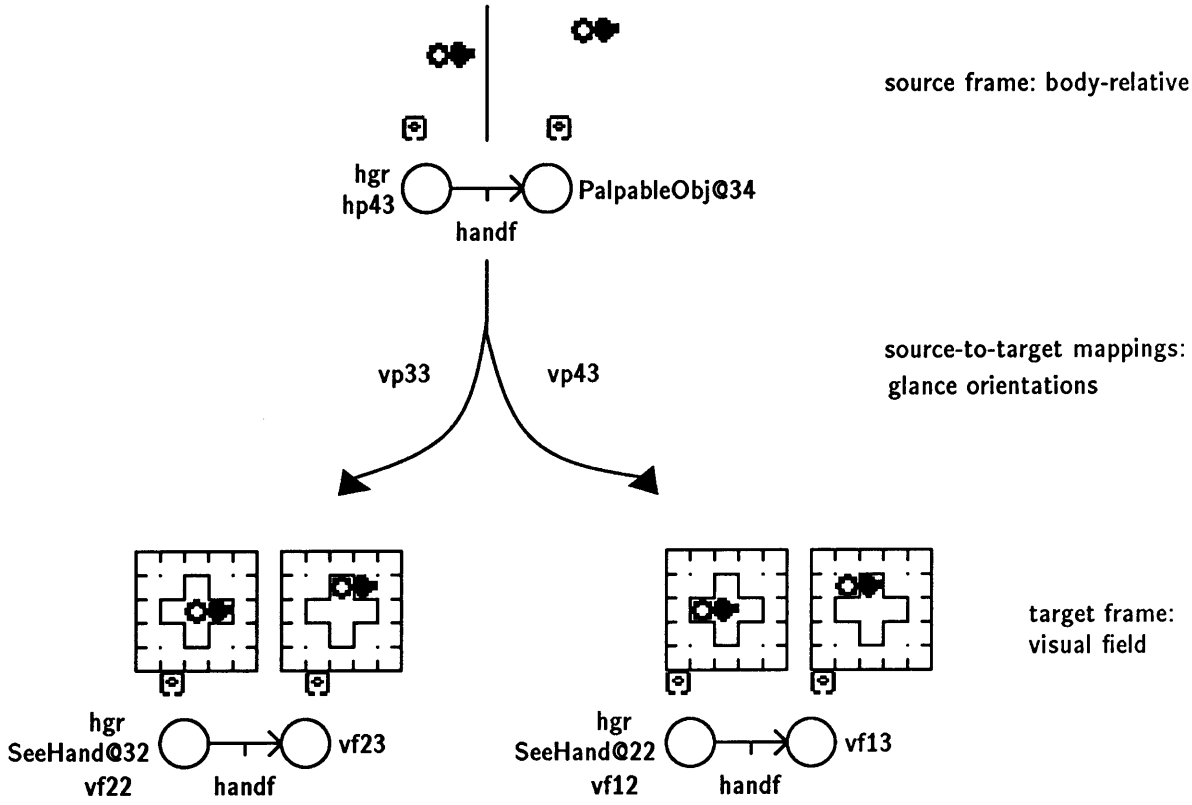


Figure 5-2: Here, the body-relative perspective is the source frame; the visual field is the target frame.

A bidirectional mapping between two frames of reference makes possible a virtual generalization that is instantiated at other positions in the same frame of reference that serves to express the generalization. The instantiation occurs in two steps: first, an explicit instantiation is made at some position in the other frame; then, that instantiation serves as a generalization, which is instantiated at various positions in the first reference frame, thus applying the original generalization to other positions in the same reference frame.

Of course, the extrapolation of a schema from one position to another in the same frame of reference does not follow deductively; the refrence frame might be inhomogeneous, so that objects behaved differently at different positions. (Indeed, even the extrapolation to multiple positions in a distinct target frame of reference does not

follow deductively.) Consequently, the processes described here may be regarded as inductive. But it is equally reasonable to say that a given schema represents a virtual generalization over all positions; that generalization itself is arrived at inductively, but its instantiation at various positions is then a matter of deduction. These views are not contradictory; they are just two ways of interpreting the same schema mechanism phenomenon.

5.1.3 Subactivation

Empirical evidence from real-world events is not the only source of knowledge; much can be learned from detached reflection and deliberation as well. A plausible cognitive mechanism must be able to imagine events, as well as participate in actual events. This section sketches a proposed extension to the schema mechanism that would enable it to do this.

To activate an applicable schema is to take its action. To *subactivate* an applicable schema is to *simulate* taking its action, by forcing its result items into a simulated-On state (or, if negated, a simulated-Off state). In addition, any other applicable schemas which share the subactivated schema's action are considered to be implicitly subactivated; their results of their activation are also simulated by giving the appropriate items a simulate-On or simulated-Off state. If a subactivated schema's action is composite, the mechanism may elect to subactivate the action's components, or simply to treat the action as atomic.

An item's simulated state is distinct from its actual state (though if no simulated state has been designated for a given item, its actual state is used instead). If a schema's context conditions are all satisfied with respect to those items' simulated state, then that schema is deemed applicable for subactivation (but not necessarily for actual activation, which requires that the actual states satisfy the context). In other words, the simulated state from a prior subactivation serves as a point of departure for the next simulated action (though real actions, of course, only proceed from real

states). The mechanism thus engages in a multi-step "thought experiment".

Such an experiment would be useless if the mechanism could not learn from it. But if the marginal attribution machinery took notice of simulated states as well as of actual states, then learning could proceed from imaginary as well as actual events. It might seem that there would be nothing new to learn from a subactivation, which only involves the re-enactment of results already represented by extant schemas. In fact, however, the side-effect of implicitly subactivating some schemas when others are explicitly subactivated can bring about novel sequences of events, leading to new knowledge—or at least newly-expressed knowledge which, like any deduction, was implicitly present all along. This re-expression is especially promising when explicit subactivation at one level of abstraction has side-effects on another level; the following section outlines some examples of this form.

5.1.4 Subactivation and virtual generalizations

As noted above, implicit generalization is a serial process; an action must first bring about a canonical perspective in order for the generalization's applicability to the present situation to become apparent. In addition to being serial, this instantiation process also relies on a physical action (to bring about the perspective), rather than just involving some internal calculation. Although being serial is inherent to implicit generalization, relying on physical action is not. This section discusses the hypothetical use of subactivation in lieu of physical action to achieve the implicit or explicit instantiation of virtual generalizations.

Not surprisingly, the idea is for the schema mechanism to *imagine* achieving a canonical perspective—that is, to subactivate rather than activate a schema which achieves the perspective. Given an adequate substrate of schemas that describe the source and target frames of reference, subactivation can accurately simulate what it would be like to bring about the canonical perspective. Figure 5-3 illustrates the subactivation of the schema /vf22/ to shift an image from vf11 to a canonical perspec-

tive at **vf22**. The chain of schemas in figure 5-3a implement the action /**vf22**/. The schemas in figure 5-3b are (implicitly) subactivated in succession as side effects of the schemas which implement the composite action. In this illustration, the initial visual orientation is **vp22**. The orientation after foveation would be **vp11**; the subactivation simulation shows this new orientation, due to the schemas in figure 5-3b, which are implicitly subactivated as a side-effect of the explicit subactivation of the schemas in figure 5-3a. (Other implicitly subactivated schemas, not shown, turn **Off** the original visual-field item **vf11** and proprioceptive item **vp22** in the subactivation simulation.)

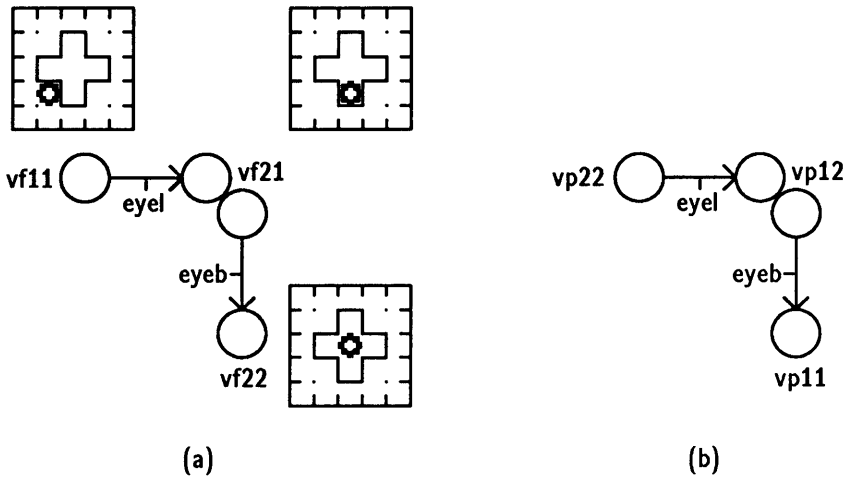


Figure 5-3: Explicitly subactivating the sequence in (a) implicitly subactivates the sequence in (b), changing the subactivation-simulated glance orientation.

Correctly simulating the new visual orientation is crucial, since maintaining the subactivation-state of the relevant visible-object synthetic items depends on that orientation. In this example, suppose the subactivation next simulates moving the hand beside the object and grasping it; suppose the schema in figure 5-1a is then subactivated, showing the visual effects of moving the grasped object forward. In consequence, and because the subactivation now shows **vp11 On**, **VisibleObj@11** and **PalpableObj@11** turn **Off** and **VisibleObj@12** and **PalpableObj@12** turn **On** in the subactivation simulation; on the basis of such subactivated trials, the mechanism can spin off a target-frame schema similar to those in figure 5-1b. Thus, a virtual generaliza-

tion is explicitly instantiated, just as though the entire experiment had been carried out in reality, rather than by subactivation.

Thus, when schemas exist that supply enough information about the source and target frames of reference, implicit instantiation (leading to explicit instantiation) can take place by subactivation; it suffices for the mechanism to imagine assuming the canonical perspective, rather than having to do so physically. The implicit instantiation is still a serial process, however, since the mechanism cannot simultaneously carry out arbitrarily many distinct subactivation simulations at once. But, as usual, when a target-frame schema is built, making the instantiation explicit, that schema can subsequently participate in fast, parallel chaining searches.

5.1.5 Conservation by instantiation of reciprocal-pair generalizations

The ability to repeatedly touch and then withdraw from an object appears early in Piagetian development (section 2.1). As primitive as this capability is, it is an important precursor to the concept of object persistence. It is itself a special case of persistence; it repeatedly recovers the tactile manifestation of an object that is—briefly—unperceived. However, this recovery can only be accomplished immediately after the manifestation ceases, and only by a particular action which is the reciprocal of the action that canceled the manifestation.

Suppose the schema mechanism were extended to identify pairs of reciprocal actions—pairs such that the first action turns Off some item that reliably but unexpectedly (with respect to any schema's prediction) turns back On if the second action immediately follows the first. An action's reciprocal promotes the recovery of a manifestation that the first action just caused to cease. For example, the action of moving the hand left is the reciprocal of moving the hand right, when the latter action removes the hand from an object to the left (figure 5-4a); and the action of glancing left is the reciprocal of glancing right, when the latter averts the gaze from

an object to the left (figure 5-4b).

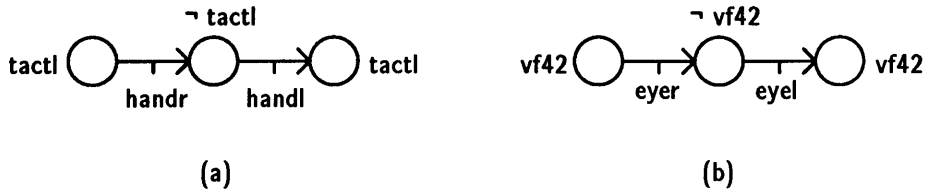


Figure 5-4: Reciprocal actions recover lost manifestations.

Recognizing the effect of such reciprocal actions is a limited recognition of persistence, as just noted. Moreover, if the mechanism specifically promotes the successive activation of reciprocal actions, this promotion will serve as a catalyst to develop the more profound sense of persistence embodied by synthetic items that designate persistent palpable or visible objects.

The successive activation of reciprocal actions promotes the formation of synthetic items by demonstrating the local consistency of their host schemas. Consider, for example, the reciprocal hand actions of figure 5-4a. Their successive activation when the hand is at, say, hp22 implicitly activates the schema /hp22/tactl. The implicit activation is successful—the schema's result does obtain; several repetitions of the pair of successive activations thus amount to successive succesful activations of /hp22/tactl, exhibiting that schema's local consistency, and spurring the construction of a synthetic item for that schema (if none exists already). Similarly, of course, for other hand positions (and for visible-object items; for those, reciprocal eye actions, rather than hand actions, give aid.)

We may regard this synthetic item formation as the explicit, position-specific instantiation of a position-independent generalization expressed in terms of the successive reciprocal actions. Furthermore, this explicit instantiation can even be accomplished by subactivation of the relevant schema and its reciprocal action, as illustrated in figure 5-5. Due to the schemas that designate the adjacency of hand-position items with respect to incremental hand actions, subactivating the sequence of reciprocal actions has the side-effect of simulating (in this example) the repeated achievement of

hp23 simultaneously with the repitition of Tactl, demonstrating the local consistency of /hp23/tactl and promoting the reification of that schema's validity by the synthetic item PalpableObj@2,3.

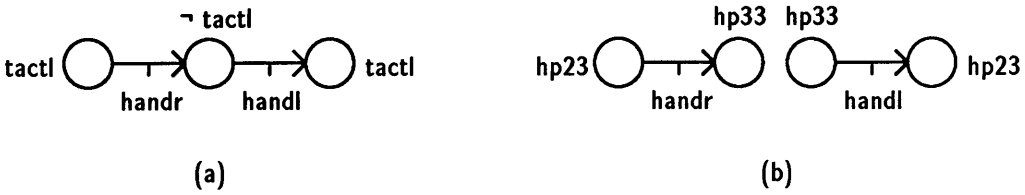


Figure 5-5: Subactivating the reciprocal actions in (a) implicitly subactivates the schemas in (b), showing the side-effect on hand position.

Thus, the need to replicate the discovery of persistent objects at different positions is mitigated by systematically promoting that replication, as the instantiation of a virtual generalization.

5.2 Deductive overriding of default generalizations

Commonsense reasoning is *nonmonotonic*; we may believe the generalization that *For all* X, $P(X)$ *implies* $Q(X)$, and then learn that for some A, $P(A)$ is true but $Q(A)$ is false, contradicting the generalization. Typically, we retain the generalization as a default assertion, which we can override in special situations in which the default is known not to hold. (Such reasoning is called nonmonotonic, in reference to the fact that the set of statements believed to be true does not just increase with additional knowledge; sometimes, additional knowledge forces the retraction of a prior view held by default.) Use of extended-context information to override an (imperfectly) reliable schema (section 3.2.1) implements a kind of nonmonotonic reasoning: the reliable schema makes a default assertion, which is trusted except when some specific overriding condition obtains.

Overriding conditions pose a special problem for virtual generalizations. If an ordinarily reliable schema that expresses a virtual generalization is overridden by

some particular condition, then all instantiations of the generalization ought to be overridden by that condition too. In figure 5-6a, for example, the schema SeeDisplace expresses the visual effect of moving the hand while grasping an object. The schema exhibits an overriding condition for a particular object that (we stipulate) is too heavy to move; thus, the object remains in place when the hand moves. Another schema, SeeHeavy (figure 5-6b), asserts that the heavy grasped object stays in place when the hand moves.

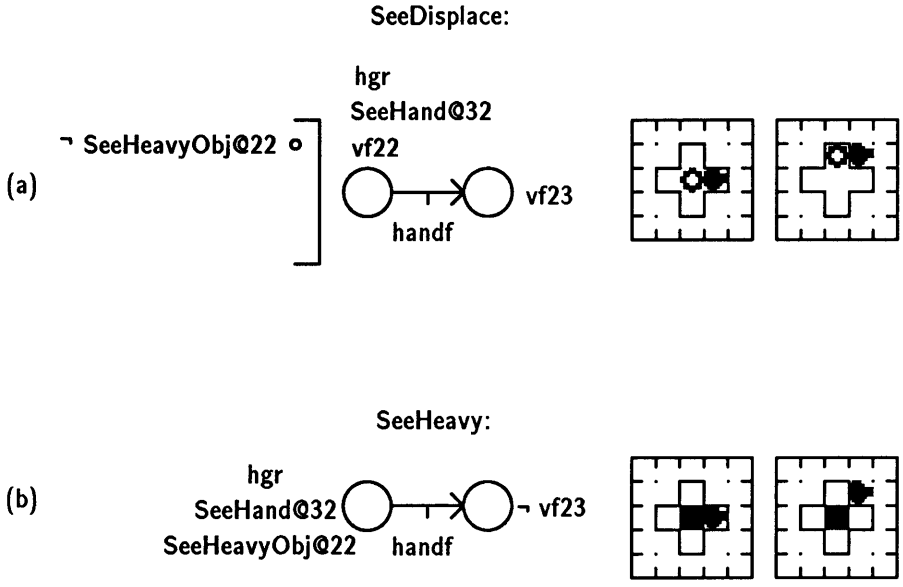


Figure 5-6: Displacing an object fails if it is too heavy.

Suppose a heavy object is present, and the schema mechanism uses subactivation to simulate achieving the canonical perspective that makes SeeDisplace applicable. That schema's override condition then also turns On, suppressing the schema; and furthermore, SeeHeavy becomes applicable. Thus, this implicit instantiation appropriately gives SeeHeavy precedence over SeeDisplace.

However, suppose SeeDisplace has been explicitly instantiated at some body-relative position—say (3,4); call the schema that expresses this instantiation ObjDisplace34 (figure 5-7). Suppose further that the overriding heavy-object schema has not been explicitly instantiated at that position. Now, when the canonical per-

spective has been achieved, there is a conflict between the prediction made by the now-applicable SeeHeavy schema—which asserts that the grasped object will remain in place if the hand moves—and the target-frame schema ObjDisplace34, which predicts that the grasped object will move when the hand does, contradicting SeeHeavy. The contradiction arises at the item VisibleObj@34; ObjDisplace34 predicts that that item should turn On, while SeeHeavy predicts a visual scene that doesn't show the object there.

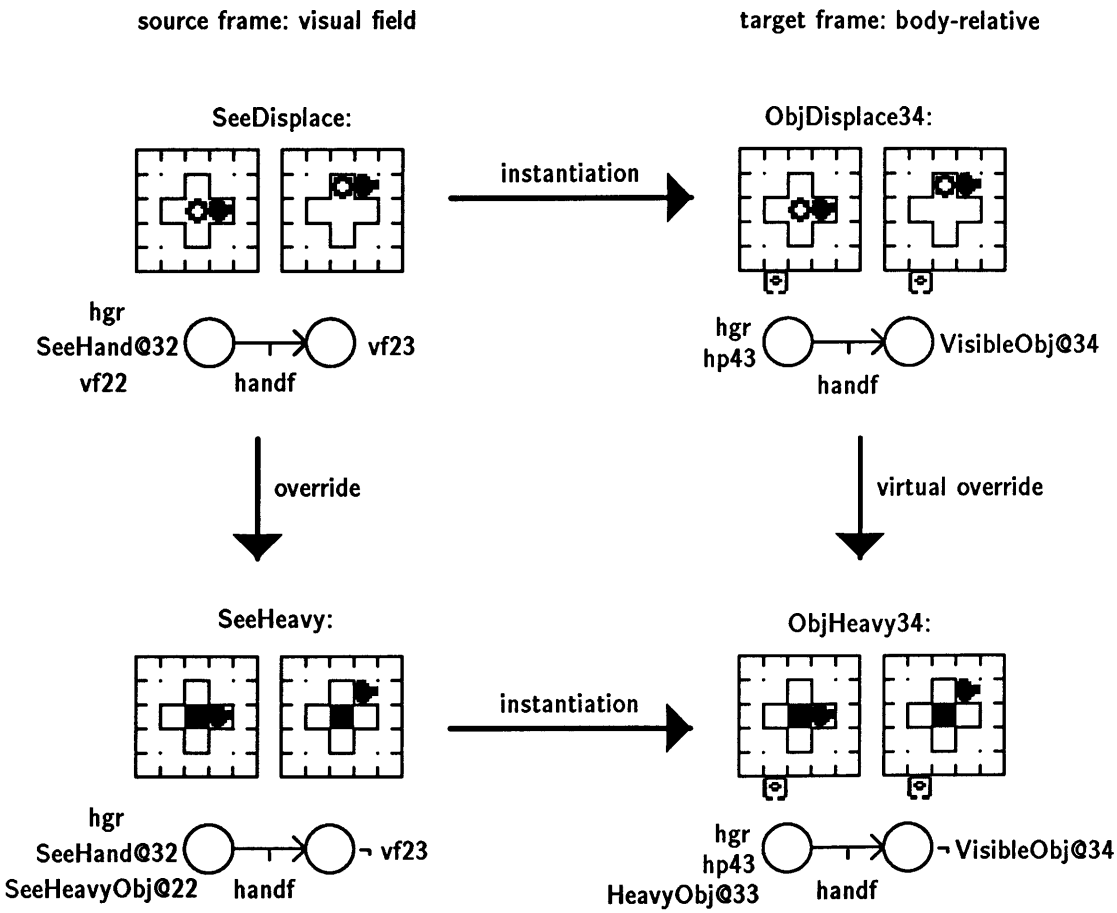


Figure 5-7: The schema ObjHeavy34 should override ObjDisplace34.

Of course, if the hand action actually occurs in this position on several occasions, the overriding generalization expressed by SeeHeavy will be explicitly instantiated for that position; that is, the schema ObjHeavy34 will be created, and the schema ObjDisplace34 will come to recognize an override condition (the condition of there

being specifically a heavy object at that position). But until then, there are conflicting predictions; worse still, if the canonical perspective has *not* been achieved, then the schema SeeDisplace will assert, wrongly but without opposition, that the object will move—even though the mechanism should know better than that, because the visual virtual generalization already properly takes note of the exceptional condition.

It is desirable that the mechanism be able to apply an overriding condition to each target-frame position, rather than having to physically try the relevant action several times at each such position in order to learn the exception there. In the absence of this ability, the utility of virtual generalizations would be severely curtailed, given the prevalence of imperfect generalizations that admit specific exceptions.

The ability to project a general schema's overriding conditions onto target-frame instantiations can be achieved if it is possible to appropriately resolve the conflict just noted between an overriding source-frame prediction (here, SeeHeavy), and a non-overridden target-frame prediction (SeeDisplace). Given a proper resolution of that conflict in the course of a subactivation, the exceptional event would be correctly simulated—the heavy object would be shown to remain stationary. On that basis, the ObjHeavy34 schema would be built, explicitly instantiating the overriding condition at that position.

I speculate that the required conflict resolution might be achieved by augmenting the schema mechanism to be able to tell that the basic target-frame schema (here, ObjDisplace34) was derivable by subactivation from another schema or schemas (SeeDisplace), and to suppress the prediction made by a derivable schema when the schemas from which it is derivable are applicable (even if overridden, as SeeDisplace is here). Intuitively, SeeDisplace *accounts for* ObjDisplace34 (and for other instantiations, in other positions); so something that supersedes SeeDisplace (e. g. SeeHeavy) should also supersede what is accounted for by SeeDisplace, e. g. ObjDisplace34. The target-frame override is thus *deduced* from the source-frame override, even though the overriding condition may never have been encountered at the target-frame position

in question.

The crux of this approach to deduced overrides is the detection of the derivability of one schema from others by subactivation. The schema mechanism could recognize derivability by detecting what is in fact derived (or rederived) from what during subactivation.

Here is a sketch of how this detection might be accomplished. Suppose that the mechanism kept track of the schemas used to maintain each item's state in the course of a subactivation, and that it also kept track of which items' states were relied on for the creation of a new schema in the course of that subactivation. Then, for each schema derived (or rederived, as defined above in section 5.1.1) by subactivation, the mechanism could note which schemas' result items caused the simulation of state-transitions of the items that appear in the result of the derived schema. Those are schemas from which that schema is derivable. (Counting rederivation, as well as derivation, ensures the recognizability of a schema's derivability even if it arose independently, empirically, before being derived from other schemas.)

As mentioned above, there are two ways to view the instantiation of virtual generalizations. From one standpoint, a virtual generalization quantifies over positions in some space (either physical positions, as in the above examples, or positions in some abstract space); the generalization is arrived at inductively, and its instantiation by subactivation may be regarded as a deduction. Alternatively, each application of the original schema to a new position may be regarded as an inductive generalization. From this standpoint, the projection of overriding conditions onto new instantiations may be seen as resolving a conflict between two inductive generalizations at different levels of description. (In the above example, one level of description is in terms of visual images, the other in terms of objects that persist at body-relative positions.)

5.3 Virtual mechanisms

An individual's intelligence develops; an adult's thought is more advanced than an infant's. Piagetian development involves not only more elaborate representations of the world—the focus of this thesis—but also more advanced forms of reasoning, understanding, and problem solving.

5.3.1 Virtual mechanisms and Piagetian development

Some of the stages of sensorimotor development chronicled by Piaget follow directly from representational advances. For example, Piaget's fourth stage brings the ability to coordinate schemas so as to use one object to act upon another (section 2.4). This has the prerequisites of representing the behavior of the acting-on and the acted-upon objects individually (as the results of schemas; first, there must be items capable of expressing those results), and being able attribute the latter to the former (by having composite actions whose goal states correspond to the behavior of the acting-on object—for example, in the hypothetical schema of figure 4-39[2]).

Other advances in intelligence, however, require more than representational advances. For example, fifth-stage tertiary circular reactions (section 2.5) involve the on-the-fly development of new techniques for acting upon an object. Or, to take an example from much later development, the stage of formal operations brings, among myriad new intellectual powers, the ability to systematically consider hypothetical explanations for an event by exhaustively generating all possible permutations of candidate factors. (For example, an individual may be asked to devise a series of experiments to discover which subset of a group of combined chemicals was responsible for a particular reaction.) Such a capability may well depend in part on new representations—of an abstract space that organizes permutations, for example—but

[2]If the acting-on object is the hand, then less representational sophistication is required, since hand motions are primitively represented as actions.

it requires more than that too. It requires the ability to deploy the representation as needed to generate explanations that depend on such permutations.

The schema mechanism's built-in problem-solving behavior is crude, consisting of finding an explicit chain of extant schemas leading from a current state to a goal state. Moreover, the schema mechanism itself does not develop; the mechanism remains constant as its constructs—schemas, actions, and items—evolve. Superficially, this constancy is at odds with the need for intelligence to grow. But just as virtual structures (e. g. the virtual generalizations of the previous section) can overcome some of the limitations of the actual data format, so *virtual mechanisms* can develop and improve despite the schema mechanism's own invariance.

At a given moment, what action the schema mechanism inititates, and what internal structures it creates or alters, are a function of the extant data structures (and, of course, of the mechanism's inputs). This function is invariant; it is in that sense that the mechanism itself does not change. But the schema mechanism, taken together with its acquired structures, operates according to some function of its inputs—and that function can change, as the structures themselves change. What I call a virtual mechanism is simply the operation of the invariant schema mechanism in concert with some or all of its evolving structures. Thus, the invariant schema mechanism can support virtual mechanisms that change.

In fact, the schema mechanism's hypothetical manipulation of virtual generalizations, discussed in the previous section, is an example of a virtual mechanism as well as of virtual structures; the tendency to activate a source-to-target mapping schema implements a virtual mechanism for instantiating virtual generalizations. But even this rudimentary example, although sketched here in some detail, remains undemonstrated by the implementation; the further-reaching virtual mechanisms required for even sensorimotor-level Piagetian development are, at this point, no more than a bare possibility for a system like the schema mechanism.

5.3.2 Virtual mechanisms and the mind's expressibility

Perhaps human cognitive development culminates in some virtual mechanism that is fixed from then on. In this case, adult intelligence can be explained at that level, without reference to its development. But there is an alternative possibility which I think more likely. It may well turn out that the work of the developmental process is never complete—that the elusive human attributes of "creativity" and "common sense" (a kind of routine, practical creativity) depend in part on continual revision and extension of the constructed virtual mechanism. Then, I see no reason to expect that the precise rules of revision are expressible on any level of abstraction higher than that which describes the developmental system.[3] If this is so, then an artificial intelligence designed on a higher level of abstraction is sure to exhibit some degree of stereotypical mechanical rigidity in the face of certain unanticipated contingencies; if this is so, humanlike flexibility must be explained in terms of a long-term developmental system, rather than as a later, static, higher-level virtual system.

The prospect of there being no precise virtual mechanism is related to arguments by some authors (e. g. Dreyfus 1979, Winograd and Flores 1986) that intelligence is inexpressible as a rule-like system. But if the human mind is a mechanism, and if the Church-Turing thesis is correct, then a formal (hence rule-like) description of the mind is surely possible. Nonetheless, if we think of rules in the sense of *consciously followed* prescriptive steps—such as those of a recipe or other explicit plan of action—then indeed there may be no precise description of intelligence at that level.[4]

[3]This is related to the argument in (Hofstadter 1982) for describing the mind at what he calls the *subcognitive* level, the details of which are inaccessible to conscious cognition. However, descriptions of the mind at the developmental-mechanism level need not be subcognitive.

[4]Of course, one might imagine a person explicitly following rules that prescribe a hand-simulation of a mechanism of intelligence; the person would thereby be acting intelligently by consciously following explicit rules. But this approach is not just impossibly cumbersome; it is useless, in principle. Either the hand-simulated intelligence will try the same trick, setting up another level of simulation in turn, and so on to infinite regress; or else a different reasonable approach is taken at some level of simulation. In the second case, all intervening levels of simulation are superfluous, including the first. Whatever non-rule-like train of thought leads the last level of simulation to any action or belief, that train of thought might just as well have been used in the first place, in lieu of

any simulations.

Chapter 6

Conclusion

6.1 Situating the schema mechanism in A.I.-space

This section relates the schema mechanism to other A.I. efforts. First come some broad observations of where this research fits along several dimensions that characterize A.I. research programs There follows a more detailed comparison of the schema mechanism with some proximally related work.

6.1.1 The schema mechanism and production systems

Superficially, a schema resembles a production rule (eg Newell and Simon 1972). A production rule has two parts: an *antecedent* and *consequent*, also called the *left side* and *right side*, respectively. The antecedent specifies conditions for the rule's applicability; the consequent specifies what happens when the rule is invoked. Some production systems invoke every rule whose antecedent is satisfied; others arbitrate among such rules to invoke just one, or a small number of them.

Three parts or two: modularity for learning

A schema differs most obviously from a production rule by having three main parts rather than two. One way to assess the consequence of this difference is to compare

the use of schemas and productions for achieving a goal. Productions can serve as *situation-action* rules, in which the left side specifies conditions for taking the action designated by the right side; the conditions may include a specification of a current goal, so that the rule is invoked only when that goal is asserted. Alternatively, productions can represent *situation-result* rules, in which the left side includes an action and some preconditions for taking it, and the right side specifies a result. The production system must identify a sequence of rule invocations that leads to a goal; each rule's result may contribute to the conditions needed for the next one's invocation. Identifying chains of schemas is a similar process.

For purposes of chaining to goals, then, there is little difference between schemas and productions; the information in schemas could be converted to two-part pro duc tion-rule syntax and used in that form. However, for purposes of learning such rules in the first place, the three-way distinction, I argue, is crucial.[1]

One way to learn how to act is to discover *what would happen* if an action were taken, and to use that piece of knowledge (perhaps in concert with other such units) to decide which action to take, based on the desirability of the outcomes of various possible actions. The schema mechanism takes this approach. Another possibility is to try to learn directly what action is best in a given situation (rather than deriving that from a representation of what would happen); Holland's *bucket brigade* algorithm exemplifies this approach. First, the system learns the desirability of actions that lead immediately to goals, in certain situations; it then learns the desirability of actions that lead to those precursor situations, in other situations; and so on, extending backward from the goals. The *credit-assignment* problem—attributing an eventual outcome to earlier events or actions—is addressed by passing credit incrementally backward from the goal.

The end result is much the same as with tripartate schemas. In the situation-

[1]Situation-result rules could be annotated to make a three-way distinction by dividing the situation into action and preconditions; but that annotation would amount to having a three-part schema.

action paradigm, results are not explicitly represented; still, the actions were learned on the basis of the usefulness of the results obtained. When the rules are invoked, the actions occur in turn, each enabling the next rule in the sequence. Thus, once formed, such rules are as useful as schemas for reaching goals—although situation-action rules would not support subactivation, for which results must be explicitly simulated, hence explicitly predicted.

However, situation-action learning is intrinsically, infeasibly slow. One reason has to do with the fact that such learning only takes place along the fringe of the state-space that has already been connected to the goal. Encountering a situation several steps back from the goal is of no use—even if the right action happens to be taken then—if subsequent steps do not lead to the recognized fringe. In contrast, with context-action-result structures, various islands of the state-space can be learned as encountered, with no forseen applicability to any goal, then quickly chained to reach a goal when the necessary pieces have been assembled.

Human beings—especially infants and children at play—clearly do seek and obtain knowledge for its own sake, not just to apply to specific goals. Metaphorically speaking, bucket-brigade-style situation-action learning does only applied research, whereas schema learning does basic research as well. For infants and technological cultures alike, it is imperative to be able to acquire knowledge without first having to be able to specify the use to which that knowledge can be put.

Empirical learning of situation-action rules is slow for a second reason as well. Different goals arise in different circumstances. For any kind of state-space, it may sometimes be desirable to be at a given point in that space, sometimes elsewhere; for example, at different times, one might want to assume a variety of different positions in a room (going to the door, sitting on a chair, etc). In order for the bucket-brigade to deal with this variability of goals, rules' situations must include either the very fact that a certain position is now a goal, or a reference to some current circumstance that bears the information that that positions is desirable (the doorbell has rung, dinner

is served, etc).

The number of situation-action rules that have to be learned is then proportionate the product of the size of the state-space and the number of goal positions. In contrast, each piece of a schema-implemented state-space network says what the result of some action would be, and is acquired independently of the system's goals; these pieces are then are used to chain to various goals in order to determine what action to take. Thus, decomposing the question of what action to take into the questions of what the result would be, and what results are desirable, has the right modularity for learning.

If goals were relatively constant—as strategic goals are, more than tactical goals—then the problem of multiplying positions with goals would not arise. In my view, bucket-brigade-style credit-assignment misapplies a strategic learning algorithm to tactical learning. The schema mechanism instead distinguishes between instrumental value, which facilitates tactical planning, and delegated value, which promotes strtgegic pursuits (section 3.2.2).

The foregoing considerations—of basic vs. applied learning, and tactical vs. strategic learning—establish the need to represent the result of an action. This, in turn, requires a designation of appropriate context conditions, since, as discussed in section 3.4.1, a given action may have a variety of distinct results in different situations. Finally, for purposes of learning, the context and action cannot combine to form an undifferentiated antecedent of a two-part rule; the marginal attribution machinery, needed to solve the context-result chicken-and-egg problem (section 3.4.1), compares what happens with vs. without the action, given satisfaction of the context condition, and thus requires an explicit distinction between context and action.

Constants and variables

Many production systems allow variables to appear in production rules. A rule's antecedent is checked for satsifaction with respect to any instantiation of those variables; if some instantiation matches, the consequent is asserted using the same variable val-

ues as resulted in an antecedent match.

Checking for antecedent satisfaction is much slower when variables are permitted, because many different instantiations may have to be tried. Still, an intelligent system needs to express and instantiate generalizations, and it is unclear that the variable-matching method for doing so cannot be made as efficient as any other (see section 6.1.7 below).

The schema mechanism does not support variables or matching for the elements of schemas. Therefore, some other method is needed to support generalizations. Section 5.1 raises the possibility that the mechanism might maintain virtual generalizations, together with virtual machinery for their instantiation. The reason to rely on this hope, rather than building in a variable-matching implementation of generalizations, is just that there is no apparent way to support such an implementation without abandoning the constructivist working hypothesis by including domain-specific build-in structure. For example, if each proprioceptive item were structured as, say (Prop Hand 3 2), with components that designate spatial coordinates, then the mechanism might be augmented to express generalizations of the form Prop Hand x y, where x and y can be matched to particular coordinates. Atomic elements, however, do not lend themselves to such generalization.

Perhaps the system itself could be made to devise structured representations to support variablized generalizations. If virtual generalization turns out to work, the inclusion of such machinery might be gratuitous, even if feasible. But if virtual generalization fails, devising such machinery may be vital to the schema mechanism.

6.1.2 The schema mechanism and connectionism

Schemas, although different from production rules, have in common with productions that they are a kind of qualitative, symbolic construct. This contrasts with connectionist systems, which pass numeric values through networks that have adjustable weights.

Yet the schema mechanism's architecture (section 3.3) is connectionist—symbolic structures are composed by setting bits at connection points; data paths transmit only nonsymbolic information, consisting of numbers, truth values, or a small number of atomic tokens (which could just as well be numbers). In fact, as the next section argues, a schema's extended context is essentially a connectionist network solving a classifier problem. The schema mechanism might be viewed as a kind of hybrid system, in which symbolic structures are created and maintained with the help of a connectionist substrate.

Extended context as connectionist network

A connectionist network divides a global computation into numerous simple, local computations. A single-layer, single-output connectionist network has a processing unit which computes a simple function—typically a weighted sum—of the network's numeric input values. A multi-layer network includes *hidden* processing units whose inputs are other units' outputs.

If the inputs are restricted to the values 0 and 1, we can regard a connectionist network as computing a boolean function of its inputs; the function's value is taken to be 0 if the output value is below a specified threshold, else 1. Equivalently, the network classifies all possible input combinations into one of two sets, corresponding to the two boolean outputs.

A classifying network can be *trained* by starting the network with arbitrary weights, presenting a series of example input combinations, and adjusting the weights according to the correctness of the network's classification for each example. There are various algorithms for this adjustment; all share the property that, on each example, each unit's weight is adjusted according to 1) the sign of the unit's contribution to the weighted sum; and 2) whether the network's comptation for that example gave the right answer. A positive contribution to a correct answer may be rewarded by increasing the weight's magnitude; a negative contribution to a correct answer may

be punished by decreasing the weight's magnitude.

A single-layer network can compute some boolean functions, but not all. For example:

- If a function is a conjunction of several inputs (e. g. $a(-b)c$, a network can realize that function by having a positive threshold k, and dividing that threshold among the weights for the non-negated conjuncts. Negated conjuncts receive negative weights; all other weights are set to zero. Then, only if the non-negated conjuncts are all 1, and the negated ones all 0, can the threshold be reached.

- If a function is a disjunction of non-negated inputs (e. g. $a + b + c$), then each disjunct can be given a weight whose magnitude exceeds that of the positive threshold k.

- If a function is a disjunction of possibly negated inputs (e. g. $a + (-b) + c$), then the threshold is set to $-k$, where k is the number of negated inputs. Each negated input receives weight $-(1 + 1/k)$, so that even if all non-negated inputs are 0, the threshold will still be met, unless all negated inputs are 1. Each non-negated input receives weight 1, so that even if all negated inputs are 1, any non-negated input will cause the threshold to be reached if that input is 1.

If there exists a set of weights to compute a given function, a *convergence* theorem (Minsky and Papert 1969) shows that a connectionist network can be trained to adjust its weights so as to compute that function. This fact is noteworthy; it means that a series of incremental adjustments to local computing elements culminate in computing the appropriate overall computation.

But single-layer networks cannot compute arbitrary boolean formulae. This is made apparent by considering DNF (disjunctive-normal-form) formulae; a DNF formula is a disjunction of clauses, each a conjunction of (possibly negated) atomic terms. Consider, for example, the formula $ab + cd$. If ab and bc each exceed the threshold, then a's weight or b's must be at least half the threshold, as must eithr

c's or d's. But then the larger weight from one conjunction, plus the larger from the other, also exceeds the threshold; no assignment of weights to a, b, c and d can allow ab and cd to exceed the threshold, while preventing both ac and ad (or bc and bd) from doing so.

The problem is that inputs that satisfy the formula are not linearly seperable from those that do not. Multi-layer connectionist networks solve this problem by having hidden units that compute functions in terms of which the formula is linear. For example, two internal units might compute the conjunctions ab and cd; an output unit then computes the disjunction of those internal units' outputs. However, there is no demonstration that such networks converge to the appropriate weights within a practical number of training examples, if the inputs number hundreds or more, and if there may be many (say, dozens) of conjunctive clauses of several terms each.

Marginal attribution in the schema mechanism takes a different approach. A schema's extended context resembles a first-order connectionist network; it faces the classification problem of distinguishing input combinations (i. e. , items' states) that correspond to successful activations from those that correspond to failures. (Of course, extant items aren't always adequate to make that distinction.) Each extended-context slot's correlation measure is roughly like a connectionist weight; it adjusts after each trial to reflect the corresponding item's contribution to the overall classification. An item's relevance is identified quickly; the identification needs only a handful of successful trials to demonstrate a significant difference in the schema's success rate as a function of the item's state.

Rather than using intermediate processing units to compute conjunctions, the schema mechanism builds spinoff schemas, whose contexts compute conjunctions. Each such schema has its own extended context—in effect, its own entire connectionist network. Having an entire such network support each small, symbolic unit of representation is expensive, though arguably (section 3.3.1) within neurophysiolocially plausible bounds.

Back-propagation and empirical credit assignment

Section 6.1.1's remarks about the modularity of learning, and about credit assignment, also apply to much connectionist work. (Holland's bucket-brigade alogrithim, in fact, has dual citizenship as a production system and a connectionist system; since Holland's rules' antecedents require no variable-matching, and since all applicable rules are invoked in parallel, a network of such rules is isomorphic to a connectionist circuit.) Sutton's *temporal difference* methods (Sutton 1988) generalize the bucket-brigade algorithm, and introduce an important distinction between rewarding that which leads to eventual success vs. rewarding that which leads to a normally reliable precursor of eventual success. Nontheless, temporal difference methods fall within the scope of the foregoing discussion.

6.1.3 The schema mechanism and search algorithms

The schema mechanism broadcasts messages in parallel through chains of schemas (section 3.3.1). Backward broadcasts from a goal state find paths to that goal; forward broadcasts from the current state find accessible states. Such broadcasts implement a breadth-first traversal of the state-space described by schemas. Such searches are prominent in conventional A.I.; they appear, for example, in classic game-playing programs (eg Samuel 1963), which do a minimax search (eg Winston 1984) in a heuristically limited portion of state space; and in SOAR (Laird, Newell, and Rosenbloom 1987), which uses productions that chain to describe series of transformations.

Compared to these other state-space searches, schema mechanism broadcasts are more efficient and more limited. Both properties derive from the fact that a broacast merely propagates messages in parallel through existing structures, whereas general state-space searching requires computing at each step a new total world state from which the next step can proceed.

Consider, for example, a goal of placing two toy blocks on a table. It is not enough for there to be a schema for placing each of the blocks (figure 6-1a). Even though

activating them in succession would indeed reach the goal state, those schemas do not chain to the goal state, because neither schema's result shows there being two blocks on the table; each shows only one. This matters, since, for example, any other schema whose context required two blocks on the table could not be chained to by either of these schemas. Chaining to the goal is possible only if one of the schemas has a result that designates both blocks on the table; if the schema's action is to place a single block, then a two-block result requires the context condition that the other block already be there. Such a schema is indeed chained to by a single-block-placement schema (figure 6-1b), thus chaining to the goal.

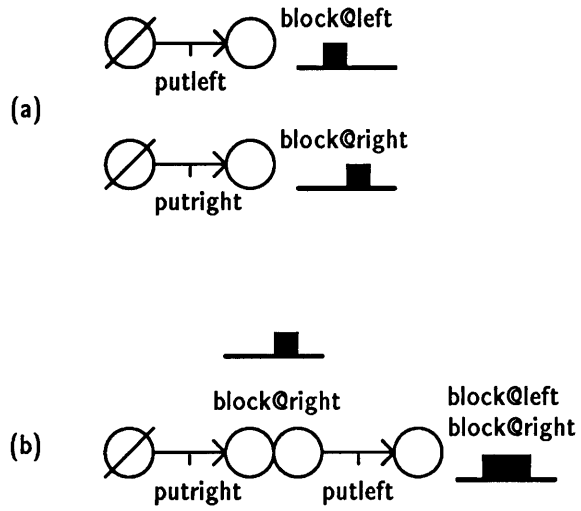


Figure 6-1: Achieving each result separately does not chain to a conjunctive goal.

In short, each link along the chain must explicitly designate all aspects of the world state that are still needed further down the chain. In contrast, conventional state-space searches can generate that information on the fly by applying successive transformations each of which designates only that part of the world that is locally relevant to that transformation. This difference makes the conventional transformations far more general, because the links in a schema chain must involve not only a widely applicable transformation (e. g. placing a block), but also a specific version of it that is relevant to the goal of the particular sitation (e. g. placing a block when another has already been placed). But such on-the-fly generation cannot be done in

parallel by propagating a fixed set of tokens through pre-established links; hence, the greater expense of the conventional paradigm.

Subactivation, the extension to the schema mechanism proposed in section 5.1.3, offers the possibility of performing the more general and more expensive kind of search. Subactivating, say, one of the block-placement schema in figure 6-1a produces a simulated world state in which the block is on the table; subactivating the other placement schema then yields a simulated state in which both blocks are there. From such subactivations (or from actual activations of the same schemas), the mechanism can derive the dual-block-placement schema of figure 6-1b (either by marginal attribution, or perhaps by a faster process suggested in the following section); that schema is thereafter available for rapid, parallel chaining to the goal.

As with conventional state-space search, subactivation steps are serial. Also, as with conventional search, there needs to be some systematic or heuristic basis for selecting the next subactivation step, since the action is not yet known to chain to the goal; rather, that discovery will be a consequence of the subactivation. What is needed is the development of a virtual mechanism (section 5.3) implemented with the aid of schemas that promote the appropriate search steps in an appropriate sequence. Such a development is, at this point, entirely speculative and undemonstrated.

6.1.4 The schema mechanism and explanation-based learning

As noted in section 5.1.3, having multiple levels of representation means that schemas at a given level might combine to predict results at another level, which can then be described at that level. Were there but one level of representation, subactivation would just re-enact what is already known, without deriving anything new.

One way for the schema mechanism to learn from subactivated events is by using the marginal attribution machinery just as for actual events. But this is needlessly inefficient, for it requires several trials, despite the fact that repeating a sequence

of subactivations with the same structures will have the same result; the repeated subactivated trials convey no more information than a single such trial.

Conceivably (though I propose no details here) the mechanism could keep track of the aspects of the world state that the subactivated result depended on, and build a schema based on a single subactivated trial, with the depended-on state elements in the context of the new schema. Such a technique would resemble chunking in SOAR (Rosenbloom and Newell 1983), which, like other explanation-based learning mechanisms, identifies and records the dependencies in a search process, and abbreviates subsequent searches by recording what follows from those dependencies, so that that search need not be recapitulated. (The recording of proximity information in a composite action's controller was also likened to SOAR broadcasting—section 3.4.2—but with respect to the more primitive search process carried out by parallel chaining.) Or, in lieu of explicit dependency tracking, the mechanism might simply grant a subactivated trial exaggerated impact on correlation statistics, having the same effect as several actual trials.

Section 3.2.2 raised the possibility that the hysteresis of schemas' activation value—a tendency to reactivate schemas recently activated—might promote imitation, by inducing the mechanism to explicitly activate a schema that had just activated implicitly. Explanation-based learning by means of subactivation might exploit a tendency toward imitation: a subactivated imitation of an event just observed might include a derivation of an explanation of the event, by deriving schemas that predict the observed outcome.

The mechanism might safeguard against spurious subactivation acquisitions by flagging subactivation-derived schemas that have received no actual test. Such schemas can be viewed with some suspicion, particularly in the event of conflict with more concretely supported schemas.

6.1.5 The schema mechanism and rational learning

Biological entities are self-constructing and self-maintaining. To be sure, suitably nuturing environments are essential. But the eventual structure of a newly conceived organism is determined primarily from within, by its genetic endowment; likewise, repair of injuries is guided from within.

Artificial machines, in contrast, are built and repaired from without—for example, by technicians at a factory. It is not clear that the biological approach is advantageous if factories (or the like) are readily available to provide synthesis and maintenance. But if they are not available, self-organization is the only possible choice; systems that bootstrap themselves from scratch, as in the case of biological evolution, must therefore organize from within.

In the cognitive domain, the dichotomy empirical vs. rational learning (Kodratoff 1988) parallels the distinction between building from within vs. from without. Rational learning analyzes representations in light of their internal structure and their relations to other structures—forming analogies, noticing dependencies, and so on. The bulding or modifying of a particular structure is determined mostly by the knowledge implemented by myriad other structures; in that sense, the construction and elaboration is directed from without, not from within the structure that is affected.

In contrast, empirical learning proceeds locally; each structure (for example, a schema) maintains its own statistics, and spawns variations of itself. True, the schema's behavior depends in part on its connections to other structures (for example, via the extended context and result). But those structures do not implement an understanding of the subject matter of the schema they connect to; the connections do not allow those structures to analyze the schema, but rather pass data to the schema, which process the data (e. g. by maintaining correlation statistics), again without having any representation of the meaning of the structures.

Doing empirical learning—having structures that learn from within, rather than from without—is crucial to constructivist bootstrapping, just as self-replication is to

evolutionary bootstrapping. Just as evolution had no factories to set it in motion, constructivist systems have no built-in domain-specific knowledge to guide earliest learning. This is most evident with regard to the invention of new concepts by the assembly of autonomously developed precursor fragments of that concept (recall the discussion of this Piagetian theme in section 2.8.1).

For example, in conventional systems of representation, an object's presence at some location is designated by some such notation as (AT-POSITION OBJ259 25 16); the system's domain knowledge and deductive resources combine with this notation to derive, for example, that to grasp OBJ259, the hand should move to (25,16) (and that to see it, the glance should be directed there). In contrast, the schema mechanism constructs the very concept of *at-position* from what other systems would regard as derived fragments of the concept. This is not merely a matter of reasoning in the other direction; the concept is invented, rather than just having its applicability deduced, from the underlying fragments. This having been done, the system indeed reasons both from the fragments to the concept (by using verification conditions to judge the concept's applicability), and vice versa (by the concept's designation in schemas' contexts and results).

These considerations do not argue for empirical learning to the exclusion of rational learning. Explanation-based learning is a form of rational learning; and, as the previous section suggests, subactivation may allow the schema mechanism to support such learning—after an adequate substrate of knowledge has been laid down by another process. (Analogously, biologically evolved systems that build from within eventually do implement factories, which build from without; the two systems are thereafter symbiotic.) Empirical rather than rational learning is needed to get off the ground, both at the very origin of individual development, and upon introduction to drastically novel domains; conceivably, even routine situations often require some on-the-fly fine-tuning of skills at levels for which no rational analysis is available. Conventional A.I. proclaims that "in the knowledge lies the power"; the constructivist

reply, to paraphrase a well-known adage, is that learning will get you through times of no knowledge better than knowledge will get you through times of no learning.

6.1.6 Virtual mechanisms and self-modification

There are two ways that a mechanism might change as it learns. As discussed in section 5.3, a mechanism might operate in conjunction with its acquired data structures to form a virtual mechanism that evolves as the structures evolve. Alternatively, the mechanism might actually modify itself. For example, Lenat's learning system Eurisko (Lenat 1983) represents its own implementation in a format that the system itself can modify (though not with great usefulness; see Haase 1989). Similarly, in SOAR, aspects of the mechanism's control structure are represented in a format that the system can change, making the system partially self-modifying.[2]

In my view, it is implausible for a constructivist system to modify its own implementation by the same principles it uses to modify its other data structures. Representing the system's implementation in structures amenable to elaboration by the system itself is vastly more difficult than thus representing, say, the rudiments of physical objects. It would make no sense to design a system that starts with the far more sophisticated built-in knowledge, but has to reinvent the much more basic knowledge.

[2]Even a self-modifying mechanism can be described as an invariant mechanism operating in conjunction with variable data to produce a variable mechanism at a higher level of abstraction. For example, any computer implementation of Eurisko runs on digital hardware that remains constant; the hardware maintains data which describe the implementation, and which change. But the level of abstraction at which there is an invariant mechanism is one that describes a general-purpose computer, not one that describes anything specific to Eurisko's learning apparatus. In contrast, the schema mechanism separates into a fixed mechanism and mutable data at an abstraction level that does correspond to the substance of the learning mechanism.

6.1.7 The schema mechanism and situated activity

The schema mechanism follows Piaget in emphasizing that an individual's physical activity is the foundation for acquired knowledge—even eventual abstract knowledge far removed from physical domains. Recent trends in A.I., highlighted by the work of Suchman (1987), Agre and Chapman (1987) and Agre (1988) and Brooks (1986a), also address activity that is said to be *situated* in the physical world.

Brooks offers an intriguing methodological rationale for his line of research. Brooks designs robot systems with roughly insect-like abilities; he argues that, on the scale of biological evolution, insects are most of the way to humans, so artificial replication of humanlike intelligence might arise from gradual elaboration of artificial insectlike intelligence. In my view, infants are a more fruitful point of departure than insects; but this, like many methodological disagreements, is probably most quickly resolved by investing the necessary years of work on both approaches, and seeing which (if either) succeeds.

A central theme of situated-activity work is the use of what Agre and Chapman call *leaning on the world* in lieu of requiring explicit representations. In particular, actions can be specified relative to perceptual states rather than with respect to an explicit internal model of the state of the world. This is similar to the Piagetian view, but with a different emphasis. Sensorimotor schemas, which do not represent an object apart from its perceptual manifestations, indeed provide a basis for the infant's early activity. But, even more importantly, such schemas and activity also form a scaffold for the creation of explicit representations apart from perception, which in turn support more sophisticated activity; the emphasis on perception-based activity's role in bootstrapping up to explicit, perception-independent representations distinguishes constructivist A.I. from situated-activity A.I.

Agre and Chapman, and Brooks, present innovative architectures to support situated activity. Agre and Chapman's system is organized around the use of *visual routines* (Ullman 1984) to direct action; Brooks's system is organized in a *subsump-*

tion hierarchy (Brooks 1986b) that allows progressive elaboration of the system's behavior. In keeping with the philosophy of representation avoidance, both systems have machinery that falls under the situation-action paradigm, in that there is no designation of expected results of actions. This, I believe, will prove to be an impediment to automatic learning in such systems, for the modularity reasons discussed above in section 6.1.1. (Also, not representing results prevents being able to learn from thought experiments as well as from real activity.) Agre and Chapman's primary argument against using world models that make results explicit is the intractability of conventional planning (Chapman 1987) based on maintaining world models (particularly in the presence of uncertainty); but, as discussed in sections 3.2.1 and 3.2.2, the schema mechanism's control structure is efficient, and, like some systems without explicit world models, supports the deferral of lowlevel aspects of a plan that depend on yet-unknown details of the world, and can respond seamlessly to unanticipated problems and opportunities.

Agre and Chapman's *indexical-functional* representation has much in common with virtual generalization in the schema mechanism. Like virtual generalization, indexical-functional representation expresses a general rule by mapping an instance to a particular perceptual view, and then having a routine that applies to that view (for example, moving the hand to the position of object X can be expressed as: look at object X; then move the hand to where-looking). But the schema mechanism, in keeping with its philosophy of *transcending* leaning on the world, can go on to build explicit instantiations of virtual generalizations, obviating the subsequent need to physically enact the mapping step (section 5.1.1).

Agre and Chapman propose indexical-functional representation as an efficient alternative to variable binding. I am unpersuaded, and do not make a similar claim for the schema mechanism's virtual generalizations. Variable binding is intractible when done exhaustively; but it need not be so when good heuristics guide the matching of variables to constants. The mapping step for virtual generalizations, and for

indexical-functional representations, effectively embodies such heuristics; the techniques will stand or fall on the development of heuristics that indeed converge to the correct sliver of an exponential search space.

6.1.8 Other Piagetian learning systems

Cunningham

Cunningham's work (Cunningham 1972) was the direct inspiration of my own effort; that work first suggested to me the idea of trying recapitulate sensorimotor development, and the mechanism Cunningham proposed served as a point of departure for the schema mechanism. Cunningham presents a hypothetical sensorimotor-level scenario for his unimplemented mechanism. His scenario emphasizes the development of the typical stages of intellegent strategy (the various circular reactions, etc), rather than the development of object concepts. Cunningham does not propose a viable mechanism for empirical learning; his bipartite schemas simply tie together all simultaneously active elements. There is no other provision for creating new representations.

Becker

Becker (1973) proposes a mechanism and microworld for sensorimotor-level learning (though he does not explicitly cast this in a Piagetian context, and he presents no scenario of expected development). Becker's mechanism examines an exhaustive record of serial primitive events. An event designated as a goal is found in the sequence. A sequence starting with this event is taken to be a result, and a sequence of events preceeding it is proposed as its cause. Different event sequences leading to a common result are compared, and irrelevent events (and irrelevant ordering-constraints on the events) are discarded. This creates schemas such as

```
[a] -> [b][c] -> [d] -> [e][f][g] => [h][i] -> [j]
```

the sequence to the left of the double arrow is a cause, the sequence to the right a result; a single arrow designates an ordering constraint, while events not separated by an single arrow are mutually unordered. The double arrow is place so that no actions lie in the result sequence. Elements of the cause sequence include actions, and non-actions that serve serve as context conditions, which, as in the schema mechanism, assure that the result will follow the action. Becker argues for this context-action-result structure on the grounds of being able to chain schemas to lead to a goal (although, as argued in section 6.1.1, two-part rules can be used for that purpose as well).

The state elements that appear in event sequences are structured rather than atomic. His system includes machinery for comparing and generalizing over parts of these structures, but the structuring itself is built in; there is no provision for the system to acquire such structuring of its own. There is no abstraction facility apart from discarding irrelevant components (or orderings) from a compound structure.

Becker's system does not address the chicken-and-egg problem of empirical learning. Rather, the combinatorial problem of associating events is glossed over by considering only serial events. A variant of Becker's unimplemented mechanism was implemented by (Bond and Mott). Their system used a simple robot, which learned to turn towards and approach a light source to recharge its battery when it ran low. Despite its being situated in the real world, the robot's trivial sensorimotor interface ensured that events were serial, and were typically related when contiguous.

BAIRN

BAIRN, a program by Wallace, Klahr, and Bluff (Wallace et al 1987), is a production-system model of cognitive development. BAIRN organizes its declarative and procedural knowledge in structures called *nodes*. A node comprises a set of productions, some of which express procedural knowledge—what action to take given particular circumstances and goals; others express declarative knowledge—what follows from

current facts.

Insofar as nodes compete for activation, resulting in the invocation of their constituent productions, a node is somewhat like a schema with a composite action (though a node's productions need not converge to a goal state). In addition, a token corresponding node's activation can appear as a condition in a production rule. The node's activation thus effectively defines a state element; in this regard, nodes are like synthetic items.

But nodes and synthetic items represent differently. A schema designates a specific assertion, the counterfactual proposition that a given action, under specified circumstances, would have a particular effect; and a synthetic item represents the validity conditions of a schema, the conditions under which the schema's assertion is true. In contrast, a node need not correspond to a succinct assertion, though it might, depending on the productions in the node.

Wallace et al report an impressive synopsis of BAIRN's aqcuisition of conservation of number; the developmental progression closely follows the sequence shown in children (Gelman and Gallistel 1978). The progression culminates in BAIRN's construction of nodes designating the cardinality of collections, with productions that embody the understanding that a collection of a number of objects keeps its cardinality, despite any rearrangement of the objects, unless something is added or removed.

The construction of these nodes is particularly striking in view of the amorphous nature of what a node represents. The key to BAIRN's ability to build such nodes is the presence of highly structured built-in nodes that serve as predecessors to the eventual number-representing nodes. In particular, Wallace et al postulate built-in subitizing nodes, which perceive the numerosity of a small collection of objects that are in the system's focus of attention. When BAIRN counts actual objects, its differentiation and generalization machinery builds variants of the primitive nodes, appopriately modifying the variants' constituent production rules. A complicated derivation leads eventually to the number nodes.

Wallace et al cite compelling evidence for the existence of innate subitizing abilities in infants (e. g. Strausss and Curtis 1981); hence, their built-in subitizing nodes are not at all ad hoc. Still, it is an open question how such innate competence might be embodied with respect to the central system. In the spirit of the schema mechanism, for example, there might be a primitive item whose meaning is *There are two objects in view that resemble the object I'm focusing on*, another item that means *There are three of them*, etc. These primitive items could enable a system's behavior to give evidence of subitizing abilities before the system recapitulates any actual understanding of number. In contrast, BAIRN's built-in subitizing nodes have extensive internal structure that is in the same format that BAIRN itself uses, and that is fully accessible to BAIRN; without this accessibility, the construction of number-representing nodes could not proceed. Thus, BAIRN's invention of the number concept does not accord with a radically constructivist account of human development; whether it accords with the actuality of human development remains to be seen.

6.2 Future work

Suggestions for extensions of this work are scattered through this chapter and the preceeding ones. First, as noted in the introduction to chapter 4, the present implementation results are best viewed as a pilot effort; validation of the implementation results presented here requires replication and quantitative characterization of those results. Secondly, some further through the Piagetian sequence might be achieved just by moving the existing implementation to a larger machine, and making trivial microworld extensions, such as providing for visually obscured objects.

Thirdly, several extensions to the basic mechanism appear worthy of exploration, some of which have been mentioned above:

- *Subactivation.* As discussed in section 5.1.3, subactivation would allow the

mechanism to learn from thought experiments, as well as from actual physical events. Furthermore, the learning mechanism might be augmented (section 6.1.4) to learn from a single subactivated trial, rather than having to do statistical learning based on several identical repetitions.

- *Combinatorics and garbage collection.* Marginal attribution does a creditable job of picking out reliable schemas (and their precursors) from the exponential space of expressible schemas. But even among such schemas, there can be combinatorial proliferations of schemas that are useless variations of one another (eg, schemas expressing the co-occurence of foveal events, as discussed in section 4.1.7); there may also be many schemas that explore useless and sterile corners of state-space. The schema mechanism may benefit from being able to recognize and purge such structures—that is, to *garbage-collect* them. (I borrow the term garbage collection from programming languages that feature automatic reclamation of memory used by inaccessible and therefore unusable structures. The metaphor is used loosely here; in the present sense, the reclaimed structures are estimated to be of lesser value, but need not be flatly unusable.)

  - The most straightforward garbage collection technique is to purge the least useful schemas, where usefulness increases with the frequency of a schema's activation, and with the result-value achieved by activating it (section 3.2.2). Depending on actual activation makes the usefulness measure responsive to all factors that contribute to a schema's selection for activation (including the availability of other, competing schemas in the situations that make a given schema applicable).

    The value of actions and items might derive from the value of the schemas in which they appear. Care must be taken when purging a structure to either purge those structures that contain it as a component, or to change such them to remove their reference to that structure.

Garbage collection based on actual use is infeasible until a large enough set of structures has ammassed. The threshold is determined by the amount of structure needed to do interesting and useful things; the schema mechanism's basic rather than applied learning (section 6.1.1) causes it to build fragments of skills that will not become useful until the rest of the necessary fragments also arise. Only then can the usefulness of the useful structures become apparent, supporting an informed choice of which structures to purge.

- Other possible garbage collection criteria involve recognizing particular kinds of unnecessary proliferations, and purging the proliferating structures.
  * As pointed out in section 4.1.7, some schemas such as /fovx23/fovx12 manage to form despite the requirement for attribution that an action be explained and a result not explained (section 3.4.2). The happens because fovx23 might happen to become explicable much earlier than fovx12. In consequence, many merely co-occurring events are deemed to cause one another. A possible mitigation of the problem would be to discover belatedly that the result fovx12 is otherwise explained, and regard /fovx23/fovx12 as less useful.
  * Building up to a context conjunction one item at a time leaves behind a trail of precursor structures with incomplete contexts. If these structures' extended contexts fail over a number of trials to make any progress toward spawning further spinoffs, they might be purged.
  * The mechanism might keep track of how often a given schema would be spun off, if it didn't already exist. If re-creating circumstances arise much more frequently than the schema's activation, the schema might be purged, in the expectation that it is likely to be re-created before it is next needed (or at least before the next several times). This

reclamation would have the plausible consequence that the mechanism might become rusty at unused skills, due to the need to re-create some components on the fly when the skill is resurrected.

- A schema's extended context data proposes spinoff schemas, and also maintains override conditions for the schema. The latter function could be generalized by allowing the extended context to act as a conventional connectionist network (recall section 6.1.2), adjusting its weights to compute some function of its inputs (the state of all items) that corresponds to the validity conditions of the schema. Such a function might usefully complement the validity conditions computed by the contexts of spinoff schemas.

In addition to further extending and experimenting with the schema mechanism itself, the schema mechanism's developmental progression might suggest experiments to perform with actual infants, to find evidence for or against corresponding details of their development.

6.3 Evaluation and summary

It is important that at attempt to engineer a constructivist mechanism be guided by a plausible theory of constructivism in humans; this provides both a point of departure for the mechanism's design, and a roadmap of target abilities by which the mechanism can be appraised and revised. Moreover, taking such a theory as a working hypothesis for the design of an artificial mechanism provides an elaboration and appraisal of the original theory. The schema mechanism is built upon Piaget's theory of cognitive development; the focus is on sensorimotor-period development, since the underlying mechanism is easiest to discern when acquired structure is still simple.

The schema mechanism is a self-extensible system that constructs schemas, actions, and items, and uses these to represent the state of the world, to assert predictions about the world, and to make plans in the pursuit of goals.

The achievements of the schema mechanism implementation are on target, but preliminary. The mechanism does use plausibly designed domain-indpendent learning machinery to recapitulate some early milestones of Piagetian development, including the anticipation of visual effects of hand motions, learning how to bring the hand into view, discovering intermodal coordination (e. g. touching what's seen, and vice versa), conceiving of persistent visible and palpable objects, and discovering their coextension. However, the mechanism just barely reaches the point of constructing some such representations, and does not go so far as to put them to practical use (say, to grasp an object in order to do something with it). And even its rudimentary abilities are aquired in the context of a microworld and sensorimotor interface that are far simpler than what the human environment provides.

Prior to its most advanced acquisitions, the schema mechanism weaves networks of spatial knowledge (the visual and proprioceptive networks) that are not predicted by Piaget. Nor do these acquisitions contradict Piaget; there is no conspicuous external manifestation of their presence or absence, so their development is not externally evident. Nonetheless, their development proceeds according to the same themes as explicitly Piagetian acquisitions, and, in the schema mechanism, obtains from the same machinery. Doubtless much of this micro-Piagetian knowledge (as we might call it) is also built in to innate cognitive modules; this may be so of much Piagetian knowledge as well, as discussed in section 2.8.3. But, as argued there, a general learning mechanism may need to recapitulate what is built in elsewhere in order to represent that knowledge in the format that the learning mechanism can operate on, to transcend what was built in.[3]

Indeed, such eventual trasncendence is the only apparent reason for any system,

[3]The design of the schema mechanism makes no attempt to incorporate in its peripheral modules the sort of innate competence that exists in those modules in human beings. For one thing, not enough is known of this innate competence to support a reasonable job of replicating it. Also, it is unclear what influence, if any, the peripheral competence has on central-system development. If the peripheral competence does help, then not having it in the schema mechanism makes the mechanism's task more formidable.

natural or artificial, to be designed to bother with recapitulating what is built in. Thus, if infant cognitive development is in fact a form of learning, there is good reason to expect that later development involves similar learning, so that the principles of the early learning extrapolate to more advanced performance.

I think this argument offers the strongest reason to suspect that something along the lines of the schema mechanism may be capable of more advanced achievements. As just noted, the mechanism's performance so far, while on the right track, is far too rudimentary to suggest an extrapolation to adultlike (or even childlike) intellectual capabilities. (I think this true of all artificial intelligence systems to date; extant A.I. programs either have expert-like abilities in very narrow domains, general methods that work on toy problems, or are virtual programming languages whose generality and power derive from their specific programming.)

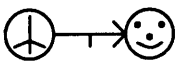
In particular, the schema mechanism—again, like other A.I. systems—faces combinatorial problems that threaten its ability to scale up to more advanced abilities. Although the schema mechanism incorporates a number of features designed to mitigate the combinatorial assault, there is no theoretical argument or practical demonstration that these features are necessary or sufficient; these features are the weakest and most tenative part of the mechanism's design. But the core features of the schema mechanism—the machinery for induction (marginal attribution), abstraction (composite actions), and conceptual invention (synthetic items)—arguably might help explain Piagetian development. To the extent that an artificial system resembles a natural system, the natural system is an existence proof that something resembling the artificial system can indeed work. And if the natural system has a way to keep its combinatorics in check—as it must, if it functions—then such an ability can be built into the artificial system as well.

Without Piaget's elaborate observations of actual development, one might react to the schema mechanism by saying Yes, that looks potentially powerful, but the mechanism's early object representations (for example) seem too strange for us to be

comfortable with the idea of an intelligence having to pass through that. Similarly, given Piaget alone, one might acknowledge the force of his description of major developmental themes, but be skeptical of how or why a sensibly engineered mechanism would exhibit (for example) the odd bugs evident in early object-understanding. The schema mechanism and Piaget's theory, taken together, provide significant (albeit circumstantial) evidence for one another.

When a mechanism figures out for itself that there are objects "out there", that is a dramatic demonstration of an ability to invent new concepts. The schema mechanism implementation has taken preliminary but promising steps in that direction. If this success continues—if it is shown that more of the Piagetian sequence of achievements, and mistakes, would indeed follow from this machinery designed to construct and use novel representations—then I think it likely that such machinery is actually involved in the infant's development.

In sum, to the extent that the schema mechanism might approximate the actual mechanism of early Piagetian development, and to the extent that the mechanism of early Piagetian development might be responsible for later development as well, it is plausible that something like the schema mechanism can account for aspects of later development as well. The most ambitious hope for the present research is that it may be an early step towards an eventual such account.



The end.

Bibliography

Harold Abelson and Gerald J. Sussman. (1985) *Structure and Interpretation of Computer Programs.* M.I.T. Press, Cambridge Ma.

James A. Anderson and Edward Rosenfeld (eds.). (1988) *Neurocomputing.* M.I.T. Press, Cambridge Ma.

Philip E. Agre. (1988) *The Dynamic Structure of Everyday Life.* M.I.T. A.I. Technical Report 1085. M.I.T. A.I. Laboratory, Cambridge Ma.

Philip E. Agre and David Chapman. (1987) Pengi: An Implementation of a Theory of Activity. *Proceedings of the Sixth National Conference on Artificial Intelligence.* pp. 196-201.

Renée Baillergeon. (1987) Young Infants' Reasoning About the Physical and Spatial Properties of a Hidden Object. *Cognitive Development.* 2, 179-200.

Joseph Becker. (1973) A model for the encoding of experiential information. in Roger Schank and Kennith Colby (eds.) *Computer Models of Thought and Language.* pp 396-434. Freeman, San Francisco.

T.G.R. Bower. (1977) *A Primer of Infant Development.* Freeman, San Francisco.

Rodney A. Brooks. (1986a) Achieving Artificial Intelligence through Building Robots. M.I.T. A.I. Memo 899. M.I.T. A.I. Laboratory, Cambridge Ma.

Rodney A. Brooks. (1986b) A Robust Layered Control System for a Mobile Robot. in *IEEE Journal of Robotics and Automation.* **2**(1), pp. 14-23.

David Chapman. (1987) Planning for Conjunctive Goals. *Artificial Intelligence.* **323**, pp 333-377.

Noam Chomsky. (1988) *The Managua Lectures.* M.I.T. Press, Cambridge, Ma.

F. Crick and C. Asanuma. (1986) Certain aspects of the anatomy and physiology of the cerebral cortex. in James L. McClelland and David E. Rumelhard et al (eds.)

Parallel Distributed Processing: Explorations in the Microstructure of Cognition. pp. 334-371. M.I.T. Press, Cambridge Ma.

Michael Cunningham. (1972) *Intelligence: It Origins and Development.* Academic Press, N.Y.

Gary L. Drescher. (1985) *The Schema Mechanism: A Conception of Constructivist Intelligencs.* M.I.T. M.S. thesis.

Hubert Dreyfus. (1979) *What Computers Can't Do.* 2nd ed. Harper Row, N.Y.

Jerry A. Fodor. (1975) *The Language of Thought.* Harvard University Press, Cambridge Ma.

Jerry A. Fodor. (1981) Methodological solipsism considered as a research strategy in cognitive science. in Jerry A. Fodor, *Representations: Philosophical Essays on the Foundations of Cognitive Science.* M.I.T. Press, Cambridge Ma.

Jerry A. Fodor. (1987) *Psychosemantics.* M.I.T. Press, Cambridge Ma.

Rochel Gelman and C. R. Gallistel. (1978) *The Child's Understanding of Number.* Harvard University Press, Cambridge Ma.

Kenneth W. Haase. (1989) Automated discovery. in Richard Forsyth (ed.), *Machine Learning: Principles and Techniques.* pp. 127-155. Chapman and Hall, N.Y.

Kenneth W. Haase. (1990) *Programs Which Invent.* forthcoming PhD dissertation, M.I.T.

W. Daniel Hillis. (1985) *The Connection Machine.* M.I.T. Press, Cambridge Ma.

Douglas Hofstadter. (1982) Artificial Intelligence: Subcognition as computation. Indiana Universtiy Computer Science Dept. Technical Report 132.

John H. Holland, Keith J. Holyoak, Richard E. Nisbett, and Paul R. Thagard. (1986) *Induction.* M.I.T. Press, Cambridge Ma.

Yves Kodratoff. (1988) *Introduction to Machine Learning.* Morgan Kaufmann, San Mateo Ca.

John E. Laird, Allen Newell, and Paul S. Rosenbloom. (1987) Soar: An architecture for general intellgence. *Artificial Intelligence.* **33**

Douglas B. Lenat. (1983) The role of heuristics in learning by discovery: three case studies. in Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell (eds.) *Machine Learning.* Vol I. pp. 243-306. Tioga, Palo Alto Ca.

David Lewis. (1973) *Counterfactuals.* Harvard University Press, Cambridge Ma.

David Marr. (1982) *Vision.* Freeman, San Francisco.

Z.S. Masangkay, K.A. McCluskey, C.W. McIntyre, J. Sims-Knight, B.E. Vaughn, and J.H. Flavell. (1974) The early development of inferences about the visual percepts of others. *Child Development* **45** pp. 357-366.

James L. McClelland, David E. Rumelhard, and the PDP Research Group. (1986) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* M.I.T. Press, Cambridge Ma.

Marvin Minsky. (1986) *The Society of Mind.* Simon and Schuster, N.Y.

Marvin Minsky and Seymour Papert. (1969) *Perceptrons.* M.I.T. Press, Cambridge Ma.

Alan Newell and Herbert A. Simon. (1972) *Human Problem Solving.* Perntice-Hall, Englewood Cliffs N.J.

Seymour Papert. (1980) *Mindstorms.* Basic Books, N.Y.

Jean Piaget. (1952a) *The Origins of Intelligence in Children.* Norton, N.Y.

Jean Piaget. (1952b) *The Child's Conception of Number.* Norton, N.Y.

Jean Piaget. (1954) *The Construction of Reality in the Child.* Ballentine, N.Y.

Jean Piaget and Bärbel Inhelder. (1969) *The Psychology of the Child.* Basic Books, N.Y.

M. Piattelli-Palmarini. (1980) *Language and Learning: The Debate between Jean Piaget and Noam Chomsky.* Harvard University Press, Cambridge Ma.

Paul S. Rosenbloom and Allen Newell. (1983) The chunking of goal hierarchies: a generalized model of practice. in Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell (eds.) *Machine Learning.* Vol II. pp. 247-288. Tioga, Palo Alto Ca.

Bertrand Russell. (1945) *A History of Western Philosophy.* Simon and Schuster, N.Y.

A.L. Samuel. (1963) Some studies in machine learning using the game of checkers. in E.A. Feigenbaum and J. Feldman (eds.) *Computers and Thought.* pp. 71-105. McGraw-Hill, N.Y.

Marcel J. Schoppers. (1987) Universal Plans for Reactive Robots in Unpredictable Environments. *Proceedings of the Tenth International Joint Conference on Artificial Intelligence.* pp 89-104.

Guy L. Steele. (1984) *Common LISP: The Language.* Digital Press, Burlington Ma.

M. S. Strauss and L. E. Curtis. (1981) Development of numerical concepts in infancy. in C. Sophian (ed.), *The Origins of Cognitive Skills.* Lawrence Erlbaum, Hillsdale N.J.

Lucy Suchman. (1987) *Plans and Situated Action.* Cambridge University Press, Cambridge.

Richard S. Sutton. (1988) Learning to predict by the methods of temporal differences. *Machine Learning.* **3** pp. 9-44.

Thinking Machines Corporation. (1988) **Lisp Reference Manual.* Cambridge Ma.

Thinking Machines Corporation. (1988) *Paris Reference Manual.* Cambridge Ma.

Shimon Ullman. (1984) Visual Routines. *Cognition.* **18,** pp. 97-159.

Ivan Wallace, David Klahr and Kevin Bluff. (1987) A self-modifying production system model of cognitive development. in David Klahr, Pat Langley and Robert Neches (eds.) *Production System Models of Learning and Development.* M.I.T. Press, Cambridge Ma.

B. White and R. Held. (1966) Plasticity of sensory-motor development in the human infant. in W. Allinsmith and J. F. Rosenblith (eds.) *The Causes of Behavior.* Allyn and Bacon, Boston.

Terry Winograd and Fernando Flores. (1986) *Understanding Computers and Cognition.* Ablex, Norwood N.J.

Pactrick H. Winston. (1984) *Artificial Intelligence.* (2nd edition) Addison-Wesley, Reading Ma.