Contentful Mental States for Robot Baby

Paul R. Cohen
Dept. of Computer Science.
University of Massachusetts,
Amherst
cohen@cs.umass.edu

Tim Oates
Dept. of Computer Science.
University of Maryland,
Baltimore County
oates@cs.umbc.edu

Carole R. Beal
Dept. of Psychology,
University of Massachusetts,
Amherst
cbeal@psych.umass.edu

Niall Adams
Dept. of Mathematics.
Imperial College,
London
n.adams@ic.ac.uk

Abstract

In this paper we claim that meaningful representations can be learned by programs, although today they are almost always designed by skilled engineers. We discuss several kinds of meaning that representations might have, and focus on a functional notion of meaning as appropriate for programs to learn. Specifically, a representation is meaningful if it incorporates an indicator of external conditions and if the indicator relation informs action. We survey methods for inducing kinds of representations we call structural abstractions. Prototypes of sensory time series are one kind of structural abstraction, and though they are not denoting or compositional, they do support planning. Deictic representations of objects and prototype representations of words enable a program to learn the denotational meanings of words. Finally, we discuss two algorithms designed to find the macroscopic structure of episodes in a domain-independent way.

Introduction

In artificial intelligence and other cognitive sciences it is taken for granted that mental states are representational. Researchers differ on whether representations must be *symbolic*, but most agree that mental states have *content* — they are *about* something and they *mean* something — irrespective of their form. Researchers differ too on whether the meanings of mental states have any causal relationship to how and what we think, but most agree that these meanings are (mostly) known to us as we think. Of formal representations in computer programs, however, we would say something different: Generally, the meanings of representations have no influence on the operations performed on them (e.g., a program concludes q because it knows $p \rightarrow q$ and p, irrespective of what p and q are about); yet the representations *have* meanings, known to us, the designers and end-users of the programs, and the representations are provided to the programs *because* of what they mean (e.g., if it was not relevant that the patient has a fever, then the proposition `febrile(patient)` would not be provided to the program — programs are designed to operate in domains where meaning matters.). Thus, irrespective of whether the contents of mental states have any causal influence on what and how *we* think, these contents clearly are intended (by us) to influence what and how our programs think. The meanings of representations are not irrelevant but we have to provide them.

This paper addresses philosophical and algorithmic issues in what we might call robot intentionality or a philosophy of mind for robots. We adapt current thinking in philosophy of mind for humans, particularly that of Fred Dretske, reifying in algorithms otherwise abstract concepts, particularly the concept of meaning. If programs could learn the meanings of representations it would save us a great deal of effort. Most of the intellectual work in AI is done not by programs but by their creators, and virtually *all* the work involved in specifying the meanings of representations is done by people, not programs (but see, e.g., (Steels 1999; Pierce & Kuipers 1997; Kuipers 2000)). This paper discusses kinds of meaning that programs might learn and gives examples of such programs.

How do people and computers come to have contentful, i.e., meaningful, mental states? As Dennett (Dennett 1998) points out, there are only three serious answers to the question: Contents are learned, told, or innate. Lines cannot be drawn sharply between these, in either human or artificial intelligence. Culture, including our educational systems, blurs the distinction between learning and being told; and it is impossible methodologically to be sure that the meanings of mental states are innate, especially as some learning occurs in utero (de Boysson-Bardies 2001) and many studies of infant knowledge happen weeks or months after birth.

One might think the distinctions between learning, being told, and innate knowledge are clearer in artificial systems, but the role of engineers is rarely acknowledged (Cohen & Litch 1999; Utgoff & Cohen 1998; Dretske 1988). Most AI systems manipulate representations that mean what engineers intend them to mean; the meanings of representations are exogenous to the systems. It is less clear where the meanings of *learned* representations reside, in the minds of engineers or the "minds of the machines" that run the learning algorithms. We would not say that a linear regression algorithm knows the meanings of data or of induced regression lines. Meanings are assigned by data analysts or their client domain experts. Moreover, these people select data for the algorithms with some prior idea of what they mean. Most work in machine learning, KDD, and AI and statistics are essentially data analysis, with humans, not machines, assigning meanings to regularities found in the data.

We have nothing against data analysis, indeed we think that learning the meanings of representations *is* data analysis, in particular, analysis of sensory and perceptual time series. Our goal, though, is to have the machine do *all* of it: select data, process it, and interpret the results; then iterate to resolve ambiguities, test new hypotheses, refine estimates, and so on. The relationship between domain experts, statistical consultants, and statistical algorithms is essentially identical to the relationship between domain experts, AI researchers, and their programs: In both cases the intermediary translates meaningful domain concepts into representations that programs manipulate, and translates the results back to the domain experts. We want to do away with the domain expert and the engineers/statistical consultants, and have programs learn represention and their meanings, autonomously.

One impediment to learning the meanings of representations is the fuzziness of commonsense notions of meaning. Suppose a regression algorithm induces a strong relationship between two random variables x and y and represents it in the conventional way: $y = 1.31x - .03$, $R^2 = .86$, $F = 108.3$, $p < .0001$. One meaning of this representation is provided by classical inferential statistics: x and y appear linearly related and the relationship between these random variables is very unlikely to be accidental. Now, the statistician might know that x is daily temperature and y is ice-cream sales, and so he or his client domain expert might assign additional meaning to the representation, above. For instance, the statistician might warn the domain expert that the assumptions of linear regression are not well-satisfied by the data. Ignoring these and other cautions, the domain expert might even interpret the representation in causal terms (i.e., hot weather causes people to buy ice-cream). Should he submit the result to an academic journal, the reviews would probably criticize this semantic liberty and would in any case declare the result as meaningless in the sense of being utterly unimportant and unsurprising.

This little example illustrates at least five kinds of meaning for the representation $y = 1.31x - .03$, $R^2 = .86$, $F = 108.3, p < .0001$. There is the *formal* meaning, including the mathematical fact that 86 % of the variance in the random variable y is explained by x. Note that this meaning has nothing to do with the denotations of y and x, and it might be utter nonsense in the domain, but, of course, the formal meaning of the representation is not about weather and ice cream, it is about random variables. Another kind of meaning has to do with the *model* that makes y and x denote ice cream and weather. When the statistician warns that the residuals of the regression have structure, he is saying that a linear model might not summarize the relationship between x and y as well as another kind of model. The domain expert will introduce a third kind of meaning: he will interpret $y = 1.31x - .03$, $R^2 = .86$, $F = 108.3$, $p < .0001$ as a statement about ice cream sales. This is not to say that every aspect of the representation has an interpretation in the domain—the expert might not assign a meaning to the coefficient $-.03$—only that, to the expert, the representation is not a formal object but a statement about his domain. We could call this kind of meaning the *domain* semantics, or the *functional* semantics, to emphasize that the interpretation of a representation has some effect on what the domain expert *does* or thinks about.

Having found a relationship between ice cream sales and the weather, the expert will feel elated, ambitious, or greedy, and this is a fourth, *affective* kind of meaning. Let us suppose, however, that the relationship is not real, it is entirely spurious (an artifact of a poor sampling procedure, say) and is contradicted by solid results in the literature. In this case the representation is meaningless in the sense that it does not *inform* anyone about how the world really works.

To which of these notions of meaning should a program that learns meanings be held responsible? The semantics of classical statistics and regression analysis in particular are sophisticated, and many humans perform adequate analyses without really understanding either. More to the point, what good is an agent that learns *formal* semantics in lieu of *domain* or *functional* semantics? The relationship between x and y can be learned (even without a statistician specifying the form of the relationship), but so long as it is a *formal* relationship between random variables, and the denotations of x and y are unknown to the learner, a more knowledgeable agent will be required to translate the formal relationship into a domain or functional one. The denotations of x and y might be learned, though generally one needs some knowledge to bootstrap the process; for example, when we say, "x denotes daily temperature," we call up considerable amounts of common-sense knowledge to assign this statement meaning. [1] As to affective meanings, we believe artificial agents will benefit from them, but we do not know how to provide them.

This leaves two notions of meaning, one based in the functional roles of representations, the other related to the informativeness of representations. The philosopher Fred Dretske wrestled these notions of meaning into a theory of how meaning can have causal effects on behavior (Dretske 1981; 1988). Dretske's criteria for a state being a meaningful representational state are: the state must *indicate* some condition, have the *function* of indicating that condition, and have this function assigned as the result of a *learning* process. The latter condition is contentious (Dennett 1998; Cohen & Litch 1999), but it will not concern us here as this paper is about learning meaningful representations. The other conditions say that a reliable indicator relationship must exist and be exploited by an agent for some purpose. Thus, the relationship between mean daily temperature (the indicator) and ice-cream sales (the indicated) is apt to be meaningful to ice-cream companies, just as the relationship between sonar readings and imminent collisions is meaningful to mobile robots, because in each case an agent can do something with the relationship. Learning meaningful representations, then, is tantamount to learning reliable relationships between denoting tokens (e.g., random variables) and learning what to do when the tokens take on particular values.

The minimum required of a representation by Dretske's theory is an indicator relationship $\vec{s} \leftarrow I(\vec{S})$ between the external world state $\vec{S}$ and an internal state $\vec{s}$, and a function that exploits the indicator relationship through some kind of action a, presumably changing the world state: $f(\vec{s}, a) \rightarrow \vec{S}$. The problems are to learn representations $\vec{s} \sim \vec{S}$ and the functions f (the relationship $\sim$ is discussed below, but here means "abstraction").

These are familiar problems to researchers in the reinforcement learning community, and we think reinforcement learning is a way to learn meaningful representations (with the reservations we discussed in (Utgoff & Cohen 1998)). We want to up the ante, however, in two ways. First, the world is a dynamic place and we think it is necessary and advantageous for $\vec{s}$ to represent how the world changes. Indeed, most of our work is concerned with learning representations of dynamics.

[1]Projects such as Cyc emphasize the denotational meanings of representations (Lenat & Guha 1990; Lenat 1990). Terms in Cyc are associated with axioms that say what the terms mean. It took a collosal effort to get enough terms and axioms into Cyc to support the easy acquisition of new terms and axioms.

Second, a *policy* of the form $f(\vec{s}, a) \rightarrow \vec{s}$ manifests an intimate relationship between representations $\vec{s}$ and the actions a conditioned on them: $\vec{s}$ contains the "right" information to condition a. The right information is almost always an abstraction of raw state information; indeed, two kinds of abstraction are immediately apparent. Not all state information is causally relevant to action, so one kind of abstraction involves selecting information in $\vec{S}$ to include in $\vec{s}$ (e.g., subsets or weighted combinations or projections of the information in $\vec{S}$). The other kind of abstraction involves the *structure* of states. Consider the sequence AABACAABACAABACAABADAABAC. Its structure can be described many ways, perhaps most simply by saying, "the sequence AABAx repeats five times, and x =C in all but the fourth replication, when x =D." This might be the abstraction an agent needs to act; for example, it might condition action on the distinction between AABAC and AABAD, in which case the "right" representation of the sequence above is something like this $p_1 s_1 p_1 s_1 p_1 s_1 p_1 s_2 p_1 s_1$, where p and s denote *structural* features of the original sequence, such as "prefix" and "suffix'. We call representations that include such structural features *structural abstractions*.

To recap, representations $\vec{s}$ are meaningful if they are related to action by a function $f(\vec{s}, a) \rightarrow \vec{S}$, but f can be stated more or less simply depending on the abstraction $\vec{s} \sim \vec{S}$. One kind of abstraction involves selecting from the information in $\vec{S}$, the other is structural abstraction. The remainder of this paper is concerned with learning structural abstractions, with what AI researchers call "getting the representation right," a creative process that we reserve unto ourselves and to which, if we are honest, we must attribute most of the performance of our programs. Note that, in Dretske's terms, structural abstractions can be indicator functions but not all indicator functions are structural abstractions. Because the world is dynamic, we are particularly concerned with learning structural abstractions of time series.

Structural Abstractions of Time Series

As a robot wanders around its environment, it generates a sequence of values of state variables. At each instant t we get a vector of values $\vec{x_t}$ (our robot samples its sensors at 10Hz, so we get ten such vectors each second). Suppose we have a long sequence of such vectors $X = \vec{x_0}, \vec{x_1}, \ldots$. Within X are subsequences x_{ij} that, when subjected to processes of structural abstraction, give rise to *episode structures* that are meaningful in the sense of informing action. The trick is to find the subsequences x_{ij} and design the abstraction processes that produce episode structures. We have developed numerous methods of this sort and survey some of them briefly, here.

Figure 1 shows four seconds of data from a Pioneer 1 robot as it moves past an object. Prior to moving, the robot establishes a coordinate frame with an x axis perpendicular to its heading and a y axis parallel to its heading. As it begins to move, the robot measures its location in this coordinate frame. Note that the ROBOT-X line is almost constant. This means that the robot did not change its heading as it moved. In contrast, the ROBOT-Y line increases, indicating that the robot does increase its distance along a line parallel to its original heading. Note especially the VIS-A-X and VIS-A-Y lines, which represent the horizontal and vertical locations, respectively, of the centroid of a patch of light on the robot's "retina," a CCD camera. VIS-A-X decreases, meaning that the object drifts to the left on the retina, while VIS-A-Y increases, meaning the object moves toward the top of the retina. Simultaneously, both series jump to constant values. These values are returned by the vision system when nothing is in the field of view.

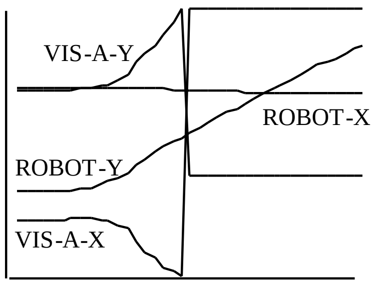

Figure 1: As the robot moves, an object approaches the periphery of its field of view then passes out of sight.

Every time series that corresponds to moving past an object has qualitatively the same structure as the one in Figure 1. It follows that if we had a statistical technique to group the robot's experiences by the characteristic patterns in multivariate time series (where the variables represent sensor readings), then this technique would in effect learn a taxonomy of the robot's experiences. *Clustering by dynamics* (CBD) is such a technique (Tim Oates 2000a): A long multivariate time series is divided into segments, each of which represents an episode such as moving toward an object, avoiding an object, crashing into an object, and so on. The episodes are not labeled in any way. Next, a dynamic time warping algorithm (Kruskall & Liberman 1983) compares every pair of episodes and returns a number that represents the degree of similarity between the time series in the pair. The algorithm returns a degree of mismatch (conversely, similarity) between the series after the best fit between them has been found. Given similarity numbers for every pair of episodes, it is straightforward to cluster episodes by their similarity. Lastly, another algorithm finds the "central member" of each cluster, which we call the *cluster prototype* following Rosch (Rosch & Mervis 1975).

CBD produces structural abstractions (prototypes) of time series, the question is whether these abstractions can be meaningful in the sense of informing action. Schmill shows how to use prototypes as planning operators (Matthew Schmill 2000). The first step is to learn rules of the form, "in state i, action a leads to state j with probability p." These rules are learned by a classical decision-tree induction algorithm, where features of states are decision variables. Given such rules, the robot can plan by means-ends analysis. It plans not to achieve world states specified by exogenous engineers, as in conventional generative planning, but to achieve world states which are preconditions for its actions. This is called "planning to act," and it has the effect of gradually increasing the size of the corpus of prototypes and things the robot can do. In this way, clustering by dynamics yields structural abstractions of time series that are meaningful in the sense of informing action. There is also a strong possibility that CBD prototypes are meaningful in the sense of informing communicative actions. Oates, Schmill and Cohen (2000b) report a very high degree of concordance between the clusters of episodes generated by CBD and clusters generated by a human judge. The prototypes produced by CBD are not weird and unfamiliar to people, but seem to correspond to how humans themselves categorize episodes. Were this not the case, communication would be hard, because the robot would have an ontology of episodes unfamiliar to people.

A Critique of Sensory Prototypes

While the general idea of clustering by dynamics is attractive (it does produce meaningful structural abstractions), the CBD method described above has two limitations. First, it requires someone (or some algorithm) to divide a time series into shorter series that contain instances of the structures we want to find. The technique cannot accept time series of numerous undifferentiated activities (e.g., produced by a robot roaming the lab for an hour). A more serious problem concerns the kinds of prototypes produced by CBD, of which Figure 1 is an example. As noted earlier, this prototype represents the robot moving past an object, say, a cup. Can we find anything in the representation that denotes a cup? We cannot. Consequently, representations of this kind cannot inform actions that depend on individuating the cup; for example, the robot could not respond correctly to the directive, "Turn toward the cup." The abstractions produced by CBD contain sufficient structure to cluster episodes, but still lack much of the structure of the episodes. If one is comfortable with a crude distinction between sensations and concepts, then the structural abstractions produced by CBD are entirely sensory (Paul R. Cohen & Beal 1997). They do not represent objects, spatial relationships, or any other individuated entity.

Learning Word Meanings

Learning the meanings of words in speech clearly requires individuation of elements in episodes. To learn the meaning of the word "cup" the robot must first individuate the word in the speech signal, then individuate the object cup in other sensory series, associate the representations; and perhaps estimate some properties of the object corresponding to the cup, such as its color, or the fact that it participates as a target in a "turn toward" activity. In his PhD dissertation, Oates discusses an algorithm called PERUSE that does all these things (Oates 2001).

To individuate objects in time series PERUSE relies on *deictic markers* — functions that map from raw sensory data to representations of objects (Ballard, Hayhoe, & Pook 1997; Agre & Chapman 1987). A simple deictic marker might construct a representation whenever the area of colored region of the visual field exceeds a threshold. The representation might include attributes such as the color, shape, and area, of the object, as well as the intervals during which it is in view, and so on.

To individuate words in speech, PERUSE requires a corpus of speech in which words occur multiple times (e.g., multiple sentences contain the word "cup"). Spoken words produce similar (but certainly not identical) patterns in the speech signal, as one can see in Figure 2. (In fact, PERUSE's representation of the speech signal is multivariate but the univariate series in Fig. 2 will serve to describe the approach.) If one knew that a segment in Figure 2 corresponded to a word, then one could find other segments like it, and construct a prototype or average representation of these. For instance, if one knew that the segment labeled A in Figure 2 corresponds to a word, then one could search for similar segments in the other sentences, find A', and construct a prototype from them. These problems are by no means trivial, as the boundaries of words are not helpfully marked in speech. Oates treats the boundaries as

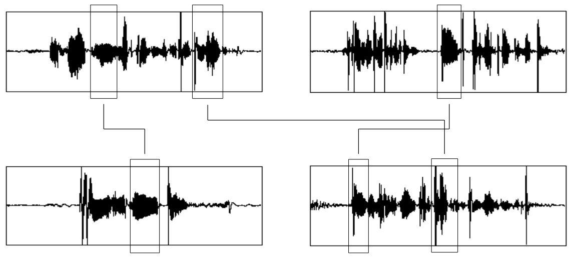
Figure 2: Corresponding words in four sentences. Word boundaries are shown as boxes around segments of the speech signal. Segments that correspond to the same word are linked by connecting lines.

hidden variables and invokes the Expectation Maximization algorithm to learn a model of each word that optimizes the placement of the boundaries. However, it is still necessary to begin with a segment that probably corresponds to a word. This problem is solved with two heuristics: In brief, the entropy of the distribution of the "next tick" spikes at episode (e.g., word) boundaries; and the patterns in windows that contain boundaries tend to be less frequent than patterns in windows that do not. These heuristics, combined with some methods for growing hypothesized word segments, suffice to bootstrap the process of individuating words in speech.

Given prototypical representations of words in speech, and representations of objects and relations, PERUSE learns associatively the *denotations* of the words. Denotation is a common notion of meaning: The meaning of a symbol is what it points to, refers to, selects from a set, etc. However, naive implementations of denotation run into numerous difficulties, especially when one tries to learn denotations. One difficulty is that the denotations of many (perhaps most) words cannot be specified as boolean combinations of properties (this is sometimes called the problem of necessary and sufficient conditions). Consider the word "cup". With repeated exposure, one might learn that the word denotes prismatic objects less than five inches tall. This is wrong because it is a bad description of cups, and it is more seriously wrong because no such description of cups can reliably divide the world into cups and non-cups (see, e.g., (Lakoff 1984; Bloom 2000)).

Another difficulty with naive notions of denotation is referential ambiguity. Does the word "cup" refer to an object, the shape of the object, its color, the actions one performs on it, the spatial relationship between it and another object, or some other feature of the episode in which the word is uttered? How can an algorithm learn the denotation of a word when so many denotations are logically possible?

Let us illustrate Oates' approach to these problems with the word "square," which has a relatively easy denotation. Suppose one's representation of an object includes its apparent height and width, and the ratio of these. An object will appear square if the ratio is near 1.0. Said differently, the word "square" is more likely to be uttered when the ratio is around 1.0 than otherwise. Let ϕ be the group of sensors that measures height, width and their ratio, and let x be the value of the ratio. Let U be an utterance and W be a word in the utterance. Oates defines the denotation of W as follows:

$$denote(W, \phi, x) = Pr(contains(U, W) | about(U, \phi), x) \quad (1)$$

The denotation of the word "square" is the probability that it is uttered given that the utterance is about the ratio of height to width and the value of the ratio. More plainly, when we say "square" we are talking about the ratio of height to width and we are more likely to use the word when the value of the ratio is close to 1.0. This formulation of denotation effectively dismisses the problem of necessary and sufficient conditions, and it brings the problem of referential ambiguity into sharp focus, for when an algorithm tries to *learn* denotations it does not have access to the quantities on the right hand side of Eq. 1, it has access only to the words it hears:

$$hear(W, \phi, x) = Pr(contains(U, W) | x) \quad (2)$$

The problem (for which Oates provides an algorithm) is to get $denote(W, \phi, x)$ from $hear(W, \phi, x)$.

At this juncture, however, we have said enough to make the case that word meanings can be learned from time series of sensor and speech data. We claim that the PERUSE algorithm constructs representations and learns their meanings by itself. PERUSE builds word representations sufficient for a robot to respond to spoken commands and to translate words between English, German and Mandarin Chinese. The denotational meanings of the representations are therefore sufficient to inform some communicative acts.

Although PERUSE is a suite of statistical methods, it is about as far from the data analysis paradigm with which we began this paper as one can imagine. In that example, an analyst and his client domain expert select and provide data to a linear regression algorithm because it means something to them, and the algorithm computes a regression model that (presumably) means something to them. Neither data nor model mean anything to the algorithm. In contrast, PERUSE selects and processes speech data in such a way that the resulting prototypes are likely to be individuated entities (more on this, below), and it assigns meaning to these entities by finding their denotations as described earlier. Structural abstraction of representations and assignment of meaning are all done by PERUSE.

Conclusion

The central claim of this paper is that programs can learn representations and their meanings. We adopted Dretske's definition that a representation is meaningful if it reliably indicates something about the external world and the indicator relationship is exploited to inform action. As this notion of meaning does not constrain what is represented, how it is represented, and how representations inform action, we have considerable freedom in how we gather evidence relevant to the claim. In fact, we imposed additional constraints on learned representations in our empirical work: They should be grounded in sensor data from a robot; the data should have a temporal aspect and time or the ordinal sequence of things should be an explicit part of the learned representations; and the representations should not merely inform action, but should inform two essentially human intellectual accomplishments, language and planning. We have demonstrated that a robot can learn the meanings of words, and construct simple plans, and that both these abilities depend on representations and meanings learned by the robot. In general, we have specified *how* things are to be represented (e.g., as sequences of means and variances, multivariate time series, transition probabilities, etc.) but the *contents* of the representations (i.e., what is represented) and the relationship between the contents and actions have been learned.

Acknowledgments

This research is supported by DARPA under contract number USASMDCDASG60-99-C-0074. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of the DARPA or the U.S. Government.

References

Agre, P. E., and Chapman, D. 1987. Pengi: An implementation of a theory of activity. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, 268–272. American Association for Artificial Intelligence.

Ballard, D. H.; Hayhoe, M. M.; and Pook, P. K. 1997. Deictic codes for the embodiment of cognition. Computer Science Department, University of Rochester.

Bloom, P. 2000. *How Children Learn the Meanings of Words*. MIT Press.

Cohen, P. R., and Litch, M. 1999. What are contentful mental states? dretske's theory of mental content viewed in the light of robot learning and planning algorithms. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*.

de Boysson-Bardies, B. 2001. *How Language Comes to Children*. MIT Press.

Dennett, D. 1998. Do it yourself understanding. In Dennett, D., ed., *Brainchildren, Essays on Designing Minds*. MIT Press and Penguin.

Dretske, F. 1981. *Knowledge and the Flow of Information*. Cambridge University Press. Reprinted by CSLI Publications, Stanford University.

Dretske, F. 1988. *Explaining Behavior: Reasons in a World of Causes*. MIT Press.

Kruskall, J. B., and Liberman, M. 1983. The symmetric time warping problem: From continuous to discrete. In *Time Warps, String Edits and Macromolecules: The Theory and Practice of Sequence Comparison*. Addison-Wesley.

Kuipers, B. 2000. The spatial semantic hierarchy. *Artificial Intelligence* 119:191–233.

Lakoff, G. 1984. *Women, Fire, and Dangerous Things*. University of Chicago Press.

Lenat, D. B., and Guha, R. V. 1990. *Building large knowledge-based systems: Representation and inference in the Cyc project*. Addison Wesley.

Lenat, D. B. 1990. Cyc: Towards programs with common sense. *Communications of the ACM* 33(8).

Matthew Schmill, Tim Oates, P. C. 2000. Learning planning operators in real-world, partially observable environments. In *Pro-*

ceedings Fifth International Conference on Artificial Planning and Scheduling, 246–253. AAAI Press.

Oates, J. T. 2001. *PhD: Grounding Knowledge in Sensors: Unsupervised Learning for Language and Planning*. Ph.D. Dissertation, Affiliation removed for blind review.

Paul R. Cohen, Marc S. Atkin, T. O., and Beal, C. R. 1997. NEO: Learning conceptual knowledge by sensorimotor interaction with an environment. In *Proceedings of the First International Conference on Autonomous Agents*, 170–177.

Pierce, D., and Kuipers, B. 1997. Map learning with uninterpreted sensors and effectors. *Artificial Intelligence Journal* 92:169–229.

Rosch, E., and Mervis, C. B. 1975. Family resemblances: Studies in the internal structure of categories. *Cognitive Psychology* 7:573–605.

Steels, L. 1999. *The Talking Heads Experiment: Volume I. Words and Meanings*. Laboratorium, Antwerpen. This is a museum catalog but is in preparation as a book.

Tim Oates, Matthew Schmill, P. C. 2000a. Identifying qualitatively different outcomes of actions: Gaining autonomy through learning. In *Proceedings Fourth International Conference, pp 110-111*. ACM.

Tim Oates, Matthew Schmill, P. C. 2000b. A method for clustering the experience of a mobile robot that accords with human judgments. In *Proceedings of Seventeenth National Conference, pp. 846-851*. AAAI Press/The MIT Press.

Utgoff, P., and Cohen, P. R. 1998. Applicability of reinforcement learning. In *The Methodology of Applying Machine leraning Problem Definition, Task Decompostion and Technique Selection Workshop, ICML-98*, 37–43.