The Jean System

Yu-Han Chang, Clayton T. Morrison,
Wesley Kerr, Aram Galstyan,
Paul R. Cohen, Carole Beal
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292
{*ychang,clayton,galstyan,wkerr,cohen,beal*}*@isi.edu*

Robert St. Amant
North Carolina State University
Campus Box 8206
2268 Engineering Building II
890 Oval Drive
Raleigh, NC 27695
stamant@csc.ncsu.edu

Tim Oates
University of Maryland
Baltimore County
1000 Hilltop Circle
Baltimore, MD 21250
oates@cs.umbc.edu

***Abstract*— Jean is a computational account of several aspects of cognitive development, specifically, a kind of Piagetian schema-driven learning. Jean's schemas amalgamate Piagetian sensorimotor schemas with Image Schemas and with Dynamic Maps. These schemas are implemented in our Image Schema Language. Using our Experimental State Splitting algorithm, Jean learns by composing schemas into gists and by differentiating schemas. Jean works in a simple dynamical environment, and we have started to demonstrate transfer of schemas and gists between similar scenarios in the environment, as well as between different environments.**

***Index Terms*— Cognitive development, transfer learning, image schemas, knowledge representation, dynamics.**

I. Introduction

Jean integrates several promising ideas from cognitive science into a model of early cognitive development. From Piaget we borrow the ideas that children learn some of what they know by repeatedly executing schemas, and executing schemas is in a sense rewarding, and some new schemas are modifications or amalgamations of old ones [1], [2]. From the Image Schema theorists [3], [4], [5], [6], [7], [8], [9], [10], [11] we hear that primitive schemas are encodings or redescriptions of sensorimotor information; and these schemas are semantically rich, general, and extend or transfer to new situations, some of which have no salient sensorimotor aspects. Another idea, represented by various authors, is that semantic distinctions sometimes depend on dynamics — how things change over time — and so schemas should have a dynamical aspect [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22]. Much of our algorithmic work has been about learning dynamical models such as maps [23], [24], [25], finite state machines [26], other kinds of Markov chain models [27], [28], and fluents [29], [30].

To build Jean we had to be precise about the form and content of schemas, and about how they are learned. One contribution of Jean is that it realizes schemas and learning methods in computer programs, particularly, our Image Schema Language [31] and the Experimental State Splitting algorithm. The project is less than a year old, so it provides more conjectures than results, but the conjectures are worth stating: It will be possible to provide a relatively small, core set of schemas and a general algorithm to learn others as they are needed or indicated by experience. We are betting on a compositional account of knowledge, in which newly learned stuff is assembled from previously learned and appropriately modified components. Schemas will have to be more than the declarative, logical structures proposed by AI researchers over the decades; they will have to include behavior-generating controllers, dynamic maps, deictic variable bindings, and causal theories; these components will not all develop simultaneously. The *transfer* of schemas between situations will prove to be the most significant research challenge, and some transfer will be initiated by pre-schematic, amodal representations.

II. The Image Schema Language

Image schemas are representations that are "close" to perceptual experience. They are sometimes presented as as re-descriptions of experience. Their popularity is due to their supposed generality and naturalness: So many situations are naturally described in terms of paths, up-down relations, part-whole relationships, bounded spaces, and so on. Even non-physical ideas, such as following an argument, containing political fallout, and feeling "up" or "down," seem only a short step from image-schematic foundations [3], [4], [5].

These ideas are attractive but vague, as we discovered when we tried to build a formal Image Schema Language (ISL) [31]. Published accounts of schemas are often ambiguous; for example, a *path* might be any directional sequence of locations, or a sequence of locations that are marked by some physical attributes, or a sequence of locations one intends to visit, or a sequence of locations another intentional agent has visited, and so on.

We found it necessary to distinguish three kinds of image schema. Static schemas describe unchanging arrangements of physical things; dynamic schemas describe how the environment changes; and action schemas describe intentional aspects of static and dynamic schemas. Thus, the action schema for "approaching" includes a path (a static schema) but gives it the intentional gloss that it is the path one intends

to follow (or is following). Moving might be intentional or it might simply be the result of force acting on an object. Both cases involve a path, but the latter is described by a dynamic schema, not an action schema.

All image schemas in ISL include variables that are bound to objects in the environment. Dynamic schemas also include *maps* which describe how relations (such as distance) change over time [24], [15], [23]. Action schemas also contain *controllers* that control behavior. For instance, Jean's schema for "approaching" binds its variables to the approaching object (typically Jean) and the approached object, a map that shows how distance between the objects changes over time, and a controller that makes Jean move toward the location of the object it is approaching. Eventually, schemas will also be augmented with causal relations (see Sec. V).

The structure of dynamic and action schemas is described in Section III. The rest of this section is primarily about static image schemas.

As represented in ISL, image schemas are objects, in the sense of the object-oriented data model. Each schema has a set of operations that determine its capabilities. For example, operations for a basic container schema include putting material into a container and taking material out. Each schema also has a set of internal slots that function as roles in a case grammar sense [32]. Slots permit image schemas to be related to each other through their slot values. For example, the contents of a container can be other image schemas.

To construct more complex schemas from simpler ones, Jean has a mechanism called *interpretation*, which, in object-oriented terms is like an extended form of delegation. Interpretations map from one or more specifications of a "source" image schema to a "target" schema. For example, we would probably first think to represent a room as a location, or as a bounded space (i.e. a region), but from a fire marshall's perspective it would be useful to interpret a room as a container with a capacity of some number of people. Interpretation gives us flexibility in evaluating the properties of some domain in terms of image schemas; different (even conflicting) interpretations of the same situation can be maintained simultaneously. Interpretation is also critical to metaphorical extension and bears relations to analogical mapping [33].

To illustrate schemas in ISL, it will be helpful to walk through an example, which we take from our work on representing chess patterns. Consider a chess board in which the Black queen has the White king in check. In image schema terms, we say that there exists a path from the queen to the king. In ISL, we generate a path schema, which contains a set of locations, as shown in Figure 1. Representing a path simply as a set of locations gives us generality, but here it's important that the queen can traverse the path in the situation that holds currently on the board. This is captured by an interpretation of the path as a set of directional linkages from each location (a source) to the next on the path (a destination). Another piece of domain information is that no location can be occupied by more than one piece at a time. This is represented by an interpretation of each location as a container with a capacity of 1. When a piece moves to a location, the container reaches capacity and yet another image schema, empty/full, is automatically created, indicating that the location is full.

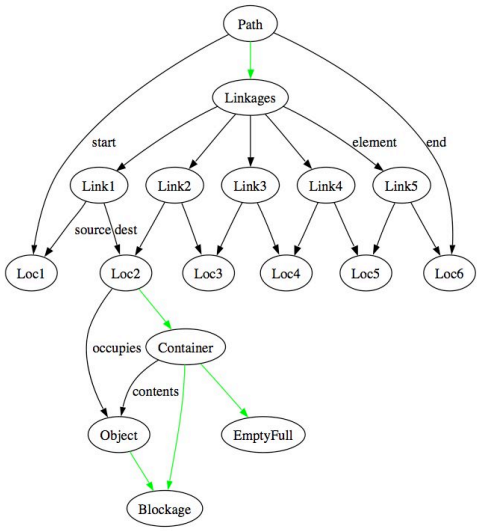

Fig. 1. Representing blockage in ISL.

Given these image schemas, their relationships, and the operations that they support, it becomes possible to reason about the situation and the possible responses White can make to counter the threat of the queen. The check exists because the path from the queen to the king is traversable. Traversability for a path schema is defined, in words, as follows: a path can be traversed when every linkage between successive locations can be traversed. Traversability for a linkage schema, in turn, is allowed when its source can be entered and its destination can be exited. Basic locations have no built-in constraints on entering and exiting, but when a location is interpretable as a container, this changes. One cannot add more to a container that has reached capacity. The interpretation relationships between these schemas cause changes to propagate outward: a full container cannot be added to; its location cannot be entered; a directional linkage cannot be traversed (via its source); a path cannot be traversed (due to a non-traversable linkage). The result is a new image schema, blockage, which is created when a container representing a location that acts as the source of a directional linkage in a path becomes full. The contents of the container constitute the blocker. This structured combination of image schemas—locations, path, linkages, blockage, and so forth—can be stored away in memory for later retrieval, limiting the

need for a complete reconstruction of the combination from scratch.

The ISL representation provides a description of the situation in the form of a structured combination of image schemas. Compare this combination with how we might describe a tactic in chess: "When an opponent's piece puts your king in check, you can counter by moving another piece into its path." The combination of schemas captures the essence of this natural language description. The representation is general, abstracting away the specific positions of the pieces, the existence of other pieces, even the identity of the attacking piece. The generality of the representation can also be seen in that its substructure maps to other basic concepts in chess. By using object schemas that include information about the color of a piece, we can use the path/linkage substructure to represent a threat of one piece on another, when the colors of the pieces are different; if they are the same, we can represent a defense relationship. The representation also supports the ability to reason about emergent structure. White might have a dozen possible moves in the situation given in the example, but few of them will be appropriate (or even legal). One of White's most plausible responses, in terms of image schemas, is to recognize that the situation is a partial match to a blockage schema (which does not yet exist), and that a specific response will lead to the creation of the blockage. Rather than reasoning about the low-level properties of individual pieces, White reasons using tactical abstractions. Other chess concepts similarly lend themselves to abstraction that can be naturally captured by image schemas: application of force on the opponent's king (even if the king is never put in check), balance in the distribution of pieces on the board, control of the center of the board, and so forth. Lower-level descriptions of moves (e.g., based on paths alone) are not inaccurate, but they fail to capture the reasons behind the moves.

III. Learning: Experimental State Splitting

Jean learns new schemas in two ways, by *composing* schemas and by *differentiating* states. Both are accomplished by the Experimental State Splitting (ESS) algorithm. We first describe the data structures ESS produces and operates over, and then describe how they are learned. Given some new environment, ESS tries to construct a finite state machine model of the environment, using schemas as the basic building blocks for describing states and actions. ESS differentiates, or *splits*, states by identifying predictive patterns of schema instantiation observed while interacting with the environment. The agent's model of the world is extended by creating new states that incorporate these schemas. The execution of action schemas transitions the agent between states. The resulting finite state machine can be solved for a policy to attain some chosen goal state in the state space. We call these learned machines for achieving specific goals *gists*. Gists may, in turn, be used as composite actions in future state machines when the agent is faced with a new environment.

Figure 2 illustrates a gist for approaching and contacting a cat. In the scenario this gist was learned, the cat is animate, capable of producing, at will, a number of actions including sitting still, walking or running away. At the same time, the cat does respond to the distance and movement of Jean. In particular, if Jean moves toward the cat rapidly, the cat will run away; if Jean approaches slowly, the cat will tend to keep doing what it is doing. Because of these behaviors, there is uncertainty in Jean's representation of what the cat will do, but the general rule about how to approach the cat is represented in the gist. Namely, the only way to catch the cat is to first get into state s_2, where the cat is nearby and not moving quickly, and then to move fast toward the cat, reaching state s_1. All other patterns of movement leave the robot in states s_3 or s_4. This corresponds to the strategy of slowly sneaking up to the cat and then quickly pouncing on it to catch it.

State s_3 of Figure 2 is comprised of two ISL schemas: an object schema that binds its deictic variable to the cat, and a near-far schema that binds its two deictic variables to the robot (i.e., Jean) and the cat, respectively. The near-far schema also asserts that the cat is more than six units away from the robot. s_3 is associated with two action schemas, fast-approach-object (F) and slow-approach-object (S), represented as arcs leaving s_3. Not pictured in the figure, but part of the representation of the approach action schemas, is the dynamic map that describes the dynamics of distance as one entity approaches another; we will describe the map in a moment.

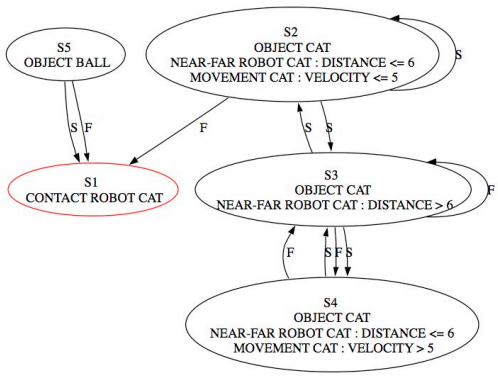

Fig. 2. A learned composite action schema for catching a cat

Jean learns gists by interacting with objects in a simulated world. To date, Jean has learned gists for interacting with

walls (which are inanimate and static), balls (inanimate yet dynamical), and cats (animate, intentional and dynamical). To learn a gist like the one in Figure 2, Jean repeatedly retrieves action schemas from its memory, runs the associated controllers, producing actions, specifically slow and fast movement to a location; assesses the resulting states, and, if the transitions between states are highly unpredictable, Jean splits states to make the resulting states more predictable. In fact, the three states, s_2, s_3 and s_4 were all originally one undifferentiated state in which Jean moved either fast or slowly toward the cat. Its inability to predict whether it could catch the cat drove Jean to differentiate states, resulting in the gist we have been discussing.

The ESS algorithm can be described in familiar policy-learning terms. We assume only that Jean has a goal state and an initial non-goal state. The following outlines the ESS procedure:

- Initialize with two states in our state space, $S_0 = \{s_0, s_g\}$, where s_g is the given goal state.
- While ϵ-optimal policy not found:
  - Gather experience: accumulate data for the transition probabilities $p(s_i, a_j, s_k)$.
  - Find state schema s_i or action schema $a_i(s_i)$ to split that maximizes the entropy score reduction of the split: $H(s_i, a_i) - \min(H(s_{k_1}, a_i), H(s_{k_2}, a_i))$, where s_{k_1} and s_{k_2} result from splitting s_i, or $H(s_i, a_i) - \min(H(s_i, a_{k_1}), H(s_i, a_{k_2}))$, where a_{k_1} and a_{k_2} result from splitting a_i.
  - Split $s_i \in S_t$ into s_{k_1} and s_{k_2}, and replace s_i with new states in S_{t+1}, or split a_i into a_{k_1} and a_{k_2} and replace a_i with new actions in $A(s_i)$.
  - Continue to gather experience, and re-solve for optimal plan in S_{t+1}

S_t is the entire state space at time t. A is the set of all actions, but $A(s) \subset A$ are the actions that are valid for state $s \in S$. $A(s)$ should be much smaller than A. $H(s_i, a_j)$ is the boundary entropy of a state-action pair, where the next observation is one of states in S_t. A small boundary entropy corresponds to a situation where executing action a_j from state s_i is highly predictive of the next observed state. Finally, $p(s_i, a_j, s_k)$ is the probability that taking action a_j from state s_i will lead us to state s_k.

Jean searches for schemas or state variables that predict state transitions, and uses these to split states. Figure 3 shows the distinctions that Jean draws in the process of state splitting. First, the algorithm begins with an undifferentiated non-goal state. Then, it learns that the type of object is an important predictor of whether or not it can catch the object. Balls are easy to catch, whereas cats are hard to catch. From here, the algorithm recognizes that distance is also an important factor in determining whether or not it can catch a cat. When it is near the cat, executing a fast-approach-object (F) will frequently lead to success in catching the cat, whereas when it is fara way from the cat, F does not usually result in catching the cat. Thus, the algorithm splits on distance with a threshold of 6, where $<= 6$ is considered near, and > 6 is far. Finally, the algorithm may notice that even when Jean is near the cat, sometimes it does not succeed in catching the cat. This might be because the cat is already moving away from the agent with some speed. Thus, ESS may do a final split based on the velocity of the cat. This process leads to the states s_2, s_3, s_4 and s_5 that we see in Figure 2.

ESS splits states that have high boundary entropy, given data accumulated by interacting with its environment, always trying to find state descriptions that produce a low-entropy sequence of actions that end in the goal state. Because time is continuous in Jean's environment, Jean needs a way to find states in time series of sensor data. Usually, states are found by *recognizing* them in sensor data, given schemas that describe states, as shown earlier. But sometimes, when Jean enters an unexplored part of its state space, and when it needs to find schemas in a state description that serve as a basis for state splitting, it needs to extract novel state descriptions from sensor data. To do this, Jean uses a simple heuristic: States change when several state variables change more or less simultaneously. This heuristic is illustrated in Figure 4. The upper two graphs show time series of five state variables: headings for the robot and the cat (in radians), distance between the robot and the cat, and their respective velocities. The bottom graph shows the number of state variables that change value (by a set amount) at each tick. When more than a set number of state variables change, Jean concludes that the state has changed. For example, between time period 6.2 and 8, Jean is approaching the cat, and the heuristic identifies this period as one state. Then, at time period 8, several indicators change at once, and we recognize that we are in

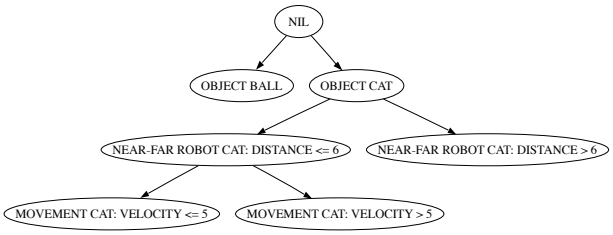

Fig. 3. ESS uses schemas and schema properties to extend state descriptions and create new states. This process "splits" less predictive states into two or more states which better predictive qualities, resulting in the tree structure shown above that describes the space of all the states that are created. Leaf nodes correspond to the entire state space, where each leaf node is a state which a state description that includes all of the schemas from the leaf to the root node of the tree.

a new state, one which corresponds to the cat moving away from Jean. The regions between these changes become the dynamic maps associated with dynamic and action schemas, and the active ISL schemas in these regions are bundled together into composite dynamic and action schemas such as "s_2: Object : Cat ; Near-Far : Robot, Cat :Distance $\leq$ 6 ; Movement Cat : Velocity $\leq$ 5."

IV. The Architecture of Jean

The main functional components of Jean are illustrated in Figure 5. Over time, Jean builds up a repository of schemas and gists, all represented in the Image Schema Language (ISL), as described above. When Jean has a goal, such as catching a cat, it retrieves appropriate gist and runs its controller, which means taking the actions that produce transitions between states. The component of Jean that runs schemas is called the *behavior generator*. Exercising schemas produces sensory data, and lots of it. The job of the *interpreter* is to retrieve and instantiate (i.e., bind the deictic variables of) schemas from the repository. Obviously, Jean will try to interpret the sensory data in terms of the schemas in the gist it is running; for example, if it is trying to catch a cat, then it will prefer to interpret the sensory data as matching the states in Figure 2. Sometimes, though, the fit is poor, and another schema in the repository does a better job of explaining the sensory data. And sometimes, it is necessary to construct a novel static, dynamic or action schema as described in the previous paragraph.

The interpreter produces an *interpretation*, which Jean uses to update the conditional probabilities of state transitions within gists. This cycle of acting, interpreting, and updating state transition probabilities can go on for a long time, but eventually Jean decides that a state in the gist it is running has high enough boundary entropy to warrant splitting. The functional component labeled Experimental State Splitting does this, and the new gist is stored in the gist repository. The other components in Figure 5 are described in the following section.

V. Future Work

Jean is very young and has had limited experience in a single domain, interacting with simulated cats and balls. Our first priority is to give Jean a lot more experience with other kinds of objects in this domain. We also will add structural features, particularly walls, to the environment.

One of the more interesting conjectures about Jean is that it will *transfer* schemas from one domain to another. In our view, transfer means using a gist in a novel situation and, if it doesn't work, modifying it as appropriate. This process is reminiscent of one Piaget called accommodation. We are currently testing transfer in the following protocol: There are two situations, call them A and B, and a gist is learned in A, and another gist, for a particular goal, is required in situation B. The control condition is to learn the gist for B without access to the gist for A. The treatment condition is to allow Jean access to the gist for A, which, if transfer is working, will have the following expected effects: Performance in B may be immediately better (because the gist for A, though not perfect, may suffice to accomplish the goal in situation B, if only sometimes), and the gist for B will be learned faster (because the gist for A is almost right, and so Jean need only learn appropriate modifications). We are testing Jean with this protocol with four A,B pairs: A is catching a ball, B is catching a cat; A is catching a cat, B is catching a cat in an environment that has walls; A is catching a cat, B is ambushing a squad in a completely different domain, a simulation of small-unit tactical warfare. If the experiments show boosts in learning rates due to transfer, this will go some way to providing concrete evidence for the claim that

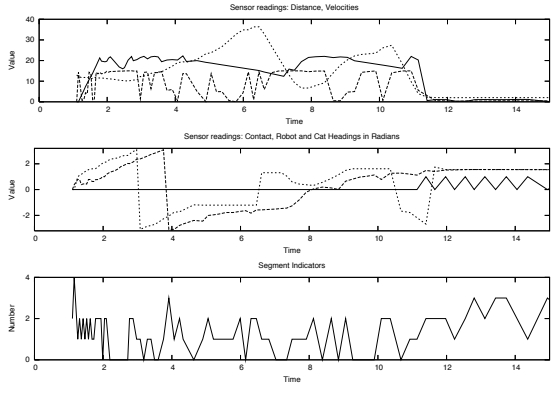

Fig. 4. New states are extracted by cutting multiple time series at places where multiple state variables change simultaneously.

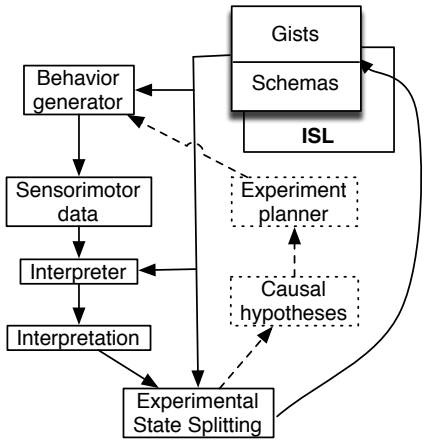

Fig. 5. The main functional components of Jean

image schemas underlie knowledge in many domains.

Another line of development for Jean is intimated by the word "experimental" in Experimental State Splitting, and by the boxes labeled *causal hypotheses* and *experiment planner* in Figure 5. State splitting finds factors that reduce the entropy of state transitions, or conversely, increase the predictability of these transitions. Not all predictive relations are causal, however. While the nature of causal relations is itself a subject for discussion, one account is quite popular and serves our purposes in the Jean project. The *counterfactual theory of causality* says X causes Y iff X precedes Y, and X and Y covary, and X is necessary to affect Y. The necessity condition is framed as a counterfactual: $\neg X \rightarrow \neg Y$. The problem with this theory is that it does not distinguish true causes from mere conditions; for instance, a wire is necessary for electricity to travel from a light switch to a light bulb, but we would not call a wire the cause when we turn on the light. A heuristic to get around this is to assign counterfactually necessary and proximal actions to X's in causal models. Thus flipping a light switch, being the most proximal action to illumination, and counterfactually necessary, is a candidate cause.

It is easy to find actions that are proximal to effects, and to formulate counterfactuals relating these actions to effects. These counterfactuals serve as causal hypotheses for Jean to try to refute. At this point in the project, we have not decided whether Jean will actively plan experiments to test its hypotheses, or will simply match its experiences against active hypotheses, looking opportunistically for refuting evidence. In either case, Jean will develop causal models of its action schemas, and will learn not only what works, but why it works.

ACKNOWLEDGMENT

We would like to thank Wei Mu for help with implementing portions of the Jean playpen scenario in breve.

REFERENCES

[1] J. Piaget. *The Construction of Reality in the Child*. New York: Basic, 1954.

[2] J. Piaget. The role of action in the development of thinking. In W. F. Overton and J. M. Gallagher, editors, *Knowledge and Development*, volume 1, pages 17–42. New York: Plenum, 1977.

[3] G. Lakoff and M. Johnson. *Metaphors We Live By*. University of Chicago Press, Chicago, IL, 1980.

[4] G. Lakoff. *Women, Fire and Dangerous Things*. University of Chicago Press, Chicago, IL, 1987.

[5] M. Johnson. *The Body in the Mind: The Bodily Basis of Meaning, Imagination, and Reason*. University of Chicago Press, Chicago, IL, 1987.

[6] R. W. Langacker. *Foundations of Cognitive Grammar*, volume 1: Theoretical Prerequisites. Stanford University Press, 1987.

[7] R. W. Gibbs and H. L. Colston. The cognitive psychological reality of image schemas and their transformations. *Cognitive Linguistics*, 6(4):347–378, 1995.

[8] J. Mandler. How to build a baby: Ii. conceptual primitives. *Psychological Review*, 99:597–604, 1992.

[9] J. Mandler. *The Foundations of Mind: Origins of Conceptual Thought*. Oxford University Press, 2004.

[10] W. Croft and D. A. Cruse. *Cognitive Linguistics*. Cambridge University Press, 2004.

[11] T. Oakley. Image schema. In D. Geeraerts and H. Cuyckens, editors, *Handbook of Cognitive Linguistics*. Osford University Press, 2006.

[12] F. Heider and M. Simmel. An experimental study of apparent behavior. *Americal Journal of Psychology*, 57(2):243–259, 1944.

[13] E. Thelen and L. Smith. *A Dynamic Systems Approach to the Development of Cognition and Action*. The MIT Press, Cambridge, MA, 1994.

[14] T. Regier. *The Human Semantic Potential: Spatial Language and Constrained Connectionism*. The MIT Press, 1996.

[15] P. R. Cohen. Maps for verbs. In *Proceedings of the Information and Technology Systems Conference, Fifteenth IFIP World Computer Conference*, 1998.

[16] P. W. Blythe, P. M. Todd, and G. F. Miller. How motion reveals intention: Categorizing social interactions. In G. Gigerenzer, P. M. Todd, and the ABC Research Group, editors, *Simple Heuristics That Make Us Smart*, pages 257–285. Oxford University Press, New York, 1999.

[17] S. Intille and A. Bobick. A framework for recognizing multi-agent action from visual envidence. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 518–525, 1999.

[18] A. Bobick and J. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 23(3), 2001.

[19] J. Siskind. Grounding lexical semantics of verbs in visual perception using force dynamics and even logic. *Journal of AI Research*, 15:31–90, 2001.

[20] L. Talmy. *Toward a Cognitive Semantics*, volume 1: Conceptual Structuring Systems (Language, Speech and Communication). The MIT Press, Cambridge, MA, 2003.

[21] C. T. Morrison, E. Cannon, and P. R. Cohen. When push comes to shove: A study of the relation between interaction dynamics and verb use. In *Working Notes of the AAAI Spring Symposium Workshop: Language Learning, an Interdisciplinary Perspective*, 2004.

[22] P. R. Cohen, C. T. Morrison, and E. Cannon. Maps for verbs: The relation between interaction dynamics and verb use. In *Proceedings of the Nineteenth International Conference on Artificial Intelligence (IJCAI 2005)*, 1995.

[23] P. R. Cohen and T. Oates. A dynamical basis for the semantic content of verbs. In *Proceedings of the Grounding of Word Meaning: Data & Models Workshop (AAAI-98)*, pages 5–8, 1998.

[24] M. T. Rosenstein, P. R. Cohen, M. D. Schmill, and M. S. Atkin. Action representation, prediction and concepts. In *AAAI Workshop on Robots, Softbots, Immobots: Theories of Action, Planning and Control*, 1997.

[25] M. T. Rosenstein. Concepts from time series. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 739–745, 1998.

[26] B. Krueger, T. Oates, T. Armstrong, P. R. Cohen, and C. Beal. Transfer in learning by doing. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005.

[27] M. Ramoni, P. Sebastiani, and P. R. Cohen. Bayesian clustering by dynamics. *Machine Learning*, 47(1):91–121, 2002.

[28] L. Firoiu and P. R. Cohen. Segmenting time series with a hybrid neural network - hidden markov model. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, 2002.

[29] P. R. Cohen. Fluent learning: Elucidating the structure of episodes. In *Proceedings of the Fourth Symposium on Intelligent Data Analysis*, volume 2189, pages 268–277, 2001.

[30] P. R. Cohen, C. Sutton, and B. Burns. Learning effects of robot activities using temporal associations. In *The 2nd International Conference on Development and Learning (ICDL 02)*, 2002.

[31] R. St. Amant., C. T. Morrison, Y. Chang, P. R. Cohen, and C. Beal. An image schema language. Submitted to The 7th International Conference on Cognitive Modelling (ICCM 2006), 2006.

[32] C. Fillmore. The case for case. In E. Bach and R. T. Harms, editors, *Universals in Linguistic Theory*. Holt, Rinehart & Winston, London, 1968.

[33] D. Gentner and A. M. Markman. Structure mapping in analogy and similarity. *American Psychologist*, 52(45–56), 1997.