Using Expectations to Drive Cognitive Behavior

Unmesh Kurup[1], Christian Lebiere[1], Anthony Stentz[2], Martial Hebert[2]

[1]Psychology, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh PA 15213
[2]Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh PA 15213
{unmeshk|cl+|tony+}@cmu.edu, hebert@ri.cmu.edu

Abstract

Generating future states of the world is an essential component of high level cognitive tasks such as planning. We explore the notion that such future state generation is more widespread and forms an integral part of cognition. We call these generated states *expectations*, and propose that cognitive systems constantly generate expectations, match them to observed behavior and react when a difference exists between the two. We describe an ACT R model that performs expectation driven cognition on two tasks pedestrian tracking and behavior classification. The model generates expectations of pedestrian movements to track them. The model also uses differences in expectations to identify distinctive features that differentiate these tracks. During learning, the model learns the association between these features and the various behaviors. During testing, it classifies pedestrian tracks by recalling the behavior associated with the features of each track. We tested the model on both single and multiple behavior datasets and compared the results against a k NN classifier. The k NN classifier outperformed the model in correct classifications, but the model had fewer incorrect classifications in the multiple behavior case, and both systems had about equal incorrect classifications in the single behavior case.

Introduction

Comparing current and desired states of the world and selecting the operators needed to go from the former to the latter is the essence of AI techniques like means-ends analysis (Simon 1981). We explore the idea that in addition to simply comparing current and desired states, we can drive cognition by generating a future state, called an expectation, at each step and comparing that generated state to the state of the world at the next step. When expectation and perception match, the expected next step is taken and the process is repeated. However, when there is a mismatch, attention is focused on the mismatch to resolve the problem.

The generation of future states and their use to drive the problem solving process has a long history in AI and related fields. As mentioned earlier, future-state generation lies at the core of AI as "search in a state space" and expectations have been used to explain both human reasoning and understanding (Schank 1982) (Schank and Owens 1987). In Neuroscience, expectation generation is central to a number of neural theories from the functionality of the neocortex (Hawkins 2004) (Baars and Gage 2010) to object and scene recognition (Puri and Wojciulik 2008). In Robotics, the Polyscheme cognitive architecture (Cassimatis 2006) has been used to generate expectations to track vehicles (Pless et. al. 2010) and play hide-and-seek games with other agents (2005). In Cognitive Psychology, expectations have been used to successfully model sequence learning (Lebiere and Wallach 2001), game theory (Erev et. al. 2010), imagery-driven planning (Wintermute and Laird 2008), control in dynamical systems (Gonzalez, Lerch and Lebiere 2003), appraisal (Hudlicka 2004) and detecting and recovering from errors in perception (Kurup, Lebiere and Stentz 2011). However, in all these models, the approaches were proposed as solutions to particular problems rather than a general way to structure cognition, or used specific, non-architectural methods to generate and match expectations. Expectation generation and matching is also central to Goal Driven Autonomy (GDA) algorithms (Aha et. al. 2010) especially Learning GDAs (Jaidee, Muñoz-Avila and Aha 2011). The difference between the GDA approach and our proposal lies in the nature of the generation and matching processes. GDA calls for a deliberative strategy of problem solving that applies knowledge to a dynamic environment while expectation-driven cognition calls for these processes to be fast, parallel, non-deliberative and architectural.

In addition, while the behavior classification task is used as an example, it is worthwhile to place our effort within the activity recognition literature. Our classifier is similar to template-matching approaches such as (Zelnik-Manor and Irani 2001) and (Laptev and Lindeberg 2003) and falls under the category of methods that use space-time features in a hierarchy of activity recognition approaches (Aggarwal and Ryoo 2011).

Our model of expectation-driven cognition is based on the ACT-R cognitive architecture, and in particular on two mechanisms of its declarative memory, blending and partial matching. These mechanisms generalize the process of generating and matching expectations and allows the model to construct states that it has never encountered before. We introduce a model of pedestrian tracking and behavior classification that uses expectations to track pedestrians based on object location information. In the learning phase, the model uses mismatches between expectations to generate features and learns to associate these features with particular behaviors. During testing, it uses discovered features to retrieve the associated behavior from memory. We compare the model's performance to that of a k-NN classifier.

ACT-R

The ACT-R cognitive architecture is a modular, neurally-plausible theory of human cognition. ACT-R describes cognition at two levels – the symbolic and the sub-symbolic. At the symbolic level, ACT-R consists of a number of modules each interacting with a central control system (Procedural module) via capacity-limited buffers. Modules represent functional units with the most common ones being the Declarative module for storing declarative pieces of knowledge, the Goal module for storing goal-related information, the Imaginal module which supports storing the current problem state, and the Perceptual (Visual and Aural) and Motor (Manual and Speech) modules that support interaction with the environment. The only way to control a module and access the results of its processing is through that module's buffer. Modules can operate asynchronously, with the flow of information between modules coordinated by the central procedural module.

Declarative memory stores factual information in structures called chunks. Chunks are typed units similar to schemas or frames that include named slots (slot-value pairs). Productions are condition-action rules, where the conditions check for the existence of certain chunks in one or more buffers. If these checks are true, the production is said to match and can be fired (executed). Only one production can fire at a time. In its action part, a production can make changes to existing chunks in buffers or make requests for new chunks. ACT-R also has an underlying sub-symbolic (numerical/statistical) layer that associates values (similar to neural activations) to chunks and productions. These activation (utility in the case of productions) values play a crucial role in deciding which productions are selected to fire and which chunks are retrieved from memory. Only those chunks that have an activation value greater than a threshold (called the retrieval threshold) are retrieved. ACT-R also has a set of learning mechanisms that allow a model to learn new declarative facts and production rules, and to modify existing sub-symbolic values to reflect the statistics of the environment. A full account of ACT-R theory can be found in (Anderson 2007) (Anderson and Lebiere 1998).

In this paper, we restrict further discussion about ACT-R to the declarative module since its performance is most relevant to our current work. As mentioned earlier, the declarative module stores factual information in the form of chunks and makes these available to the rest of the architecture via the retrieval buffer. There are two critical mechanisms for retrieving information in ACT-R's declarative module – partial matching and blending - that are important to generating and matching expectations.

Partial Matching

In ACT-R, productions make requests for chunks in declarative memory by specifying certain constraints on the slot values of chunks. These constraints can range from the very specific where every slot and value of the desired chunk is specified to the very general (akin to free association) where the only specification is the type of the chunk. Request criteria also include negatives specifying that a slot should not have a particular value as well as ranges (in the case of numerical values). The standard request generally specifies the chunk type and one or more slot values but not all. If there are multiple chunks that exactly match the specified constraints, the chunk with the highest activation value is retrieved. The activation value of a chunk (1) is the sum of its base-level activation and its contextual activation. The base-level activation of a chunk is a measure of its frequency and recency of access. The more recently and frequently a chunk has been retrieved, the higher its activation and the higher the chances that it is retrieved reflecting pervasive cognitive phenomena such as the Power Law of Practice and the Power Law of Forgetting. Equation (2) describes how to calculate the base-level activation of a chunk i where t_j is the time elapsed since the jth reference to chunk i, d represents the memory decay rate and L denotes the time since the chunk was created.

$$A_i = B_i + \sum_j W_j S_{ji} + \varepsilon_i \tag{1}$$

$$B_i = ln \sum_{j=1}^{n} t_j^{-d} \approx ln \frac{nL^{-d}}{1-d} \tag{2}$$

The contextual activation of a chunk is determined by the attentional weight given the context element j and the strength of association S_{ji} between an element j and a chunk i. An element j is in context if it is part of a chunk in a buffer (e.g., it is the value of one of the goal chunk's slots). The default assumption is that there is a limited source activation capacity that is shared equally between all context elements. The associative strength S_{ji} is a measure of how often chunk i was retrieved by a production when source j was in context. In addition to the base-level and contextual values, some randomness is introduced into the retrieval process by the addition of noise, which makes retrieval conform to the softmax or Boltzmann distribution.

Without partial matching enabled, the retrieval mechanism only considers those chunks that exactly match the request criteria. However, in real world situations one is seldom exposed to the exact same information, and the mechanism therefore needs to generalize to similar situations. When partial matching is enabled, the retrieval mechanism can select the chunk that matches the retrieval constraints to the greatest degree. It does this by computing a match score for each chunk that scales down the chunk's activation by its degree of mismatch to the specified constraints. Equation (3) specifies how to compute the match score.

$$M_{ip} = A_i - MP \sum_{v,d} (1 - Sim(v, d)) \tag{3}$$

MP is the mismatch penalty, *Sim(v,v')* is the similarity between the desired value v and the actual value v' held in the retrieved chunk. With the use of partial matching, the retrieval mechanism can retrieve chunks that are closest to the specified constraints even if there is no chunk that matches the constraints exactly. This is particularly useful as shown below in situations where values are continuous and dynamic. Since the degree of match is combined with the activation to yield the match score, chunks that have higher activation will also tolerate a greater degree of mismatch. This reflects the interpretation of activation as a measure of likelihood of usefulness (Anderson 1990).

Blending

The second aspect of retrieval is blending (Lebiere 1999), a form of generalization where, instead of retrieving an existing chunk that best matches the request, blending produces a new chunk by combining the relevant chunks. The values of the slots of this blended chunk are the average values for the slots of the relevant chunks weighted by their respective activations, where the average is defined in terms of the similarities between values. For discrete chunk values without similarities, this results in a kind of voting process where chunks proposing the same value pool their strengths. For continuous values such as numbers, a straightforward averaging process is used. For discrete chunk values between which similarities (as used in partial matching) have been defined, a compromise value that minimizes the weighted sum of squared

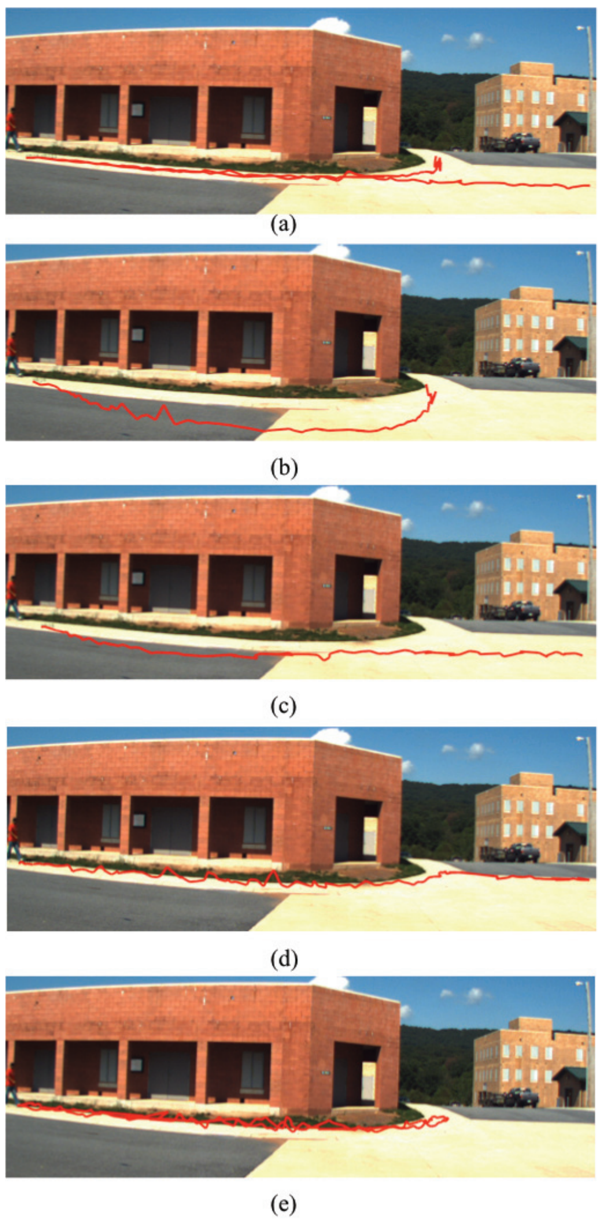

Figure 1. Examples of behaviors (a) normal (Straight & Left), (b) Peek, (c) Detour, (d) Veer and (e) Walkback.

dissimilarities is returned. Formally, the value V obtained by a blended retrieval is determined as follows:

$$V = ArgMin \sum_{i} P_i (1 - Sim(V, V_i))^2 \tag{4}$$

where P_i is the probability of retrieving chunk i and V_i is the value held by that chunk. Blending has been shown to be a convincing explanation for various types of implicit learning (Wallach and Lebiere 2003). Blending of location information in chunks allows the model to predict future locations of objects by giving more weight to recent

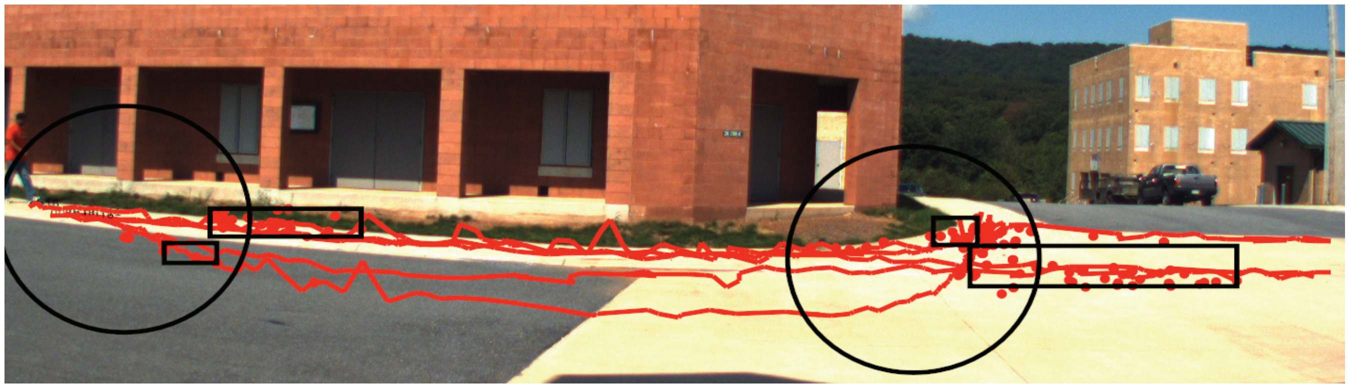

Figure 2: The features learned by the model overlaid on the pedestrian tracks. Each feature is denoted by a rectangular box. The circles denote the point of divergence of tracks that led to the construction of features.

perceptual information while ignoring various individual fluctuations arising from noise. ACT-R's blending mechanism can be thought of as a subset of more general approaches like Conceptual Blending (Fauconnier and Turner 1998) where the structure of the component concepts and the final concept is restricted to a single type and the compositional process for constructing the blended concept is weighted averaging. More comprehensive elements of Conceptual Blending such as non-trivial compositional rules, completion, elaboration and emergent structures are absent in ACT-R blending but could be performed using other cognitive mechanisms after the blending process. In addition, the partial matching and blending mechanisms in ACT-R are meant to capture the fundamental generalization characteristics over similarity-based semantics of modeling paradigms such as neural networks (Rumelhart et. al. 1986) (O'Reilly and Munakata 2000), albeit at a different level of abstraction.

Together, partial matching and blending allow the model to automatically generate expectations of future states from previously observed states.

Classifying Pedestrian Behavior

We tested the idea of using partial matching and blending for expectation-driven cognition by building a model of behavior classification for pedestrians at an intersection with a checkpoint (Fig 1). A pedestrian enters the scene from the left (a convenience for expository purposes) and walks towards the right on the sidewalk. At the intersection, the pedestrian can either go straight or turn left. These two behaviors – walking straight and turning left – represents the two normal behaviors in the scenario. An example of the tracks produced by pedestrians exhibiting these behaviors is shown in Fig 1(a). An additional four behaviors – peek, detour, veer and walkback - are termed suspicious. In the peek behavior (Fig 1(b)), the pedestrian steps out on to the street to look for a checkpoint around the building and finding that it is not present, proceeds to turn left at the intersection. In the detour behavior (Fig 1(c)), the pedestrian steps out on to the street to look for a checkpoint (as in the peek behavior) and finding it present proceeds to walk straight rather than turn left. In the veer behavior (Fig 1(d)), the pedestrian turns left at the intersection, sees the checkpoint, veers off and proceeds to walk straight on instead of continuing along to the left. Finally, in the walkback behavior (Fig 1(e)), the pedestrian turns left at the intersection, spots the checkpoint, reverses course and proceeds to walk back to the starting location. The goal of the model is to track pedestrians across the scene and classify their behavior.

Tracking Pedestrians

A state-of-the-art object detection algorithm (Felzenswalb et. al. 2010) trained to detect people is used to identify the pedestrians in the scene. The output of the algorithm is in the form of bounding boxes for each person in the image, together with a confidence value. The ACT-R model converts this bounding box into a single x,y coordinate that is the mid-point along the base of the bounding box. The model represents this information together with the rate of change (delta) in a chunk in declarative memory. The model tracks pedestrians as follows – at each timestep, the model generates an expected location (using the partially matched and blended location and delta values from declarative memory chunks associated with that person) for each person previously identified. It then assigns each location from perception to a person by picking the person closest to that location. If a location is left over (i.e., all known persons have been assigned to locations), a new person is identified at that location. ACT-R's partial matching and blending ensure that the expected location generated by the model places greater emphasis on the recent history of an object while smoothing out random variations in projecting moving directions. This process can be seen as a spatial version of the general modeling pattern of projecting future system states in dynamic environments (Lebiere, Gonzalez and Warwick 2009).

While there are other general-purpose tracking algorithms, this localized greedy approach is good enough for our goal of studying expectation-driven cognition. More importantly, doing tracking within the architecture gives us a way to use contextual information to influence the process in the future, which is one of the goals of this research program.

Features

The model classifies pedestrian behaviors by abstracting their tracks into a set of features where a feature is simply a rectangular region in the scene. If a pedestrian's track intersects that region, then that track is said to have that particular feature. The directionality of the track is captured by the sequence of features that the model builds for each track.

The model learns the set of features as follows – it builds up a set of expectations for pedestrians as they move across the scene. When an expected track diverges from the observed track by more than a prescribed angle, the model notes the point of divergence and selects a point along each of these tracks (expected and observed) as feature points. When the model is done with the learning set, it clusters these points into features. Fig 2 shows an example of the kinds of features found by the model. The rectangles are the features while the circles indicate the points of divergence (that are used to cluster the features).

Classifying Behaviors

The model works in two stages. In the first stage, the model learns the set of features as described above. In the second stage, for each track the model learns a chunk that captures the association between the features and the pedestrian's behavior. This two-stage breakdown is due to the limited data available since feature-behavior associations cannot be learned before the features themselves have been discovered. With sufficient data, these stages can be combined into a single stage where the initial behaviors are used to discover the features and the later ones to learn the associations.

During testing, the model uses the feature sequence it derives for each pedestrian track to retrieve the appropriate behavior chunk from memory. This model can be seen as a spatial specialization of a more abstract model that recognizes and anticipates behavior by matching action patterns, illustrating the versatility of the cognitive architecture across both symbolic and metric domains (Oltramari & Lebiere, 2011).

Results

The data consisted of four examples of each behavior for a total of 24 behaviors. In addition, we also collected a data set with multiple behaviors. The multiple behavior data had two examples of each behavior (for a total of ten behaviors) in a continuous sequence with up to 3 pedestrians in the scene at any one time.

Learning Model (Single Behavior Set)		Learning Model (Multiple Behavior Set)	
Made	86.1%	Made	82.4%
Correct	68%	Correct	43.8%
Incorrect	18.1%	Incorrect	38.6%

(a)

K-NN Model (Single Behavior Set)		K-NN Model (Multiple Behavior Set)	
Made	100%	Made	100%
Correct	83.3%	Correct	50%
Incorrect	16.7%	Incorrect	50%

(b)

Table 1. The percent of classifications made, percent correct and percent incorrect for (a) model that learns features from the data and (b) a k NN classifier.

We ran monte-carlo simulations (1000 iterations) of classification on both single and multiple behavior sets with ACT-R parameters for *retrieval threshold*, *noise* and *match penalty* set to -8.0, 0.1 and 1.0 respectively. For both single and multiple behavior sets, we randomly selected 3 examples of each behavior (18 examples in total) for the learning set. For single behavior classification, we used the remaining example of each behavior for the testing set (6 examples in total). For multiple behavior classification, the learning set remained the same while the testing set was the multiple behavior set. Table 1(a) shows the results of classification on both sets. The results are good for the single behavior case, with the model making a classification in 86% of the cases. In the remaining 14% of the cases, there were no chunks in declarative memory that had an activation value that was greater than the retrieval threshold leading to the model not producing a classification. The model correctly classified 68% of the total behaviors and incorrectly classified 18%. The results of classification in the multiple behavior case are poorer, with the model incorrectly classifying about as many behaviors as it correctly classifies (~40%). However, this is still above chance, which is at 16.7%.

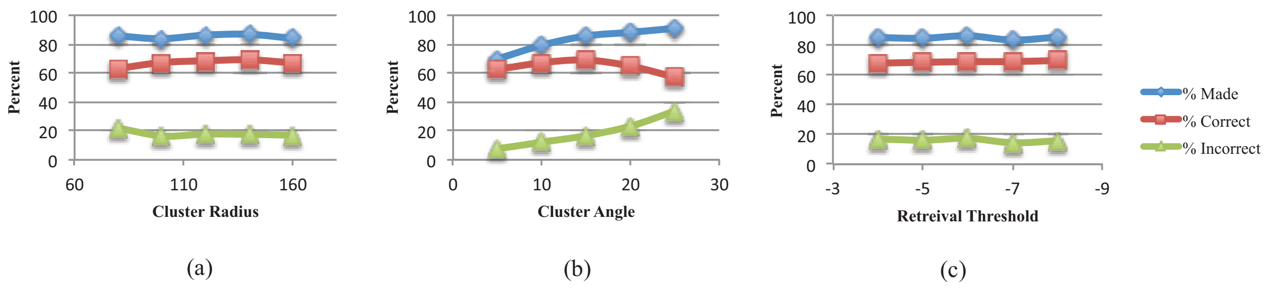

Figure 3. Percentage of classifications made, correct and incorrect vs (a) the radius of the cluster (b) the angle at which features are detected and (c) the retrieval threshold of the ACT R model

For comparison, in Table 1(b) we show the results of a k-NN classifier. This model correctly classifies more behaviors in both the single and multiple behavior cases. Both models incorrectly classify about 17% of the results in the single behavior case, but in the multiple behavior case, the feature-learning model makes fewer mistakes, incorrectly classifying only 38.6% of the behaviors as opposed to 50% for the k-NN classifier.

We also ran a number of simulations to understand the nature of the various parameters involved in the feature-learning model. We tested the effect of three parameters – the cluster radius (the radius of the circles in Fig 2) that determines which cluster a divergence point (the point of deviation from expectation) belongs to, the cluster angle that decides whether a deviation from expectation is large enough to consider it a feature and the retrieval threshold of the model that determines if the activation value of a chunk in ACT-R's declarative memory is high enough for retrieval. Figure 4 shows the results of these tests. We found that the radius of the cluster and the retrieval threshold do not matter at least for a normal range of values. The cluster angle on the other hand has a noticeable effect on all three percentages. The system makes more classifications as the angle increases but that increase comes at the expense of the number of correct classifications. The reason for this is straightforward – as the angle increases, features get aggregated into fewer groups. Once the number of features becomes too low, the model has trouble distinguishing between the different behaviors since they have the same set of features.

Conclusion and Future Work

We investigated the idea of cognition as an active process, driven by the generation and matching of expectations. We built a model that used ACT-R's declarative memory and its partial matching and blending mechanisms to generate expectations, and ACT-R's procedural module to direct cognition based on the match or mismatch between these expectations and observed values.

Our future work is concentrated on understanding how expectations can be used to make perception better by providing context and directing attention. Additionally, we are interested in understanding how expectation-driven cognition can work with deliberative approaches such as Learning Goal Driven Autonomy (Jaidee, Muñoz-Avila and Aha 2011). Finally, the generation of an expectation is still influenced by the model but the theory of expectation-driven cognition, given the rapidity of the generation and matching processes, calls for an architectural solution. Understanding how to achieve this goal with minimal changes to the existing architecture is part of ongoing research.

Acknowledgements

Conducted through collaborative participation in the Robotics Consortium sponsored by the U.S Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement W911NF-10-2-0016.

References

Aggarwal, J. K. and Ryoo, M. S. 2011. Human activity analysis: A review. *ACM Computing Surveys (CSUR)* 43(3).

Aha, D.W., Klenk, M., Muñoz Avila, H., Ram, A., and Shapiro, D. (Eds.) 2010. *Goal Directed Autonomy: Notes from the AAAI Workshop (W4)*. Atlanta, GA.

Anderson, J. R. 1990. *The Adaptive Character of Thought*. Hillsdale, NJ: Erlbaum.

Anderson, J. R. 2007. *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press. New York.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. 2004. An integrated theory of the mind. *Psychological Review* 111 (4):1036 1060.

Anderson, J. R., Lebiere, C. 1998. *The atomic components of thought.* Mahwah, NJ: Erlbaum.

Baars, B. and Gage, N. 2010. *Cognition, Brain and Consciousness: Introduction to Cognitive Neuroscience*. Oxford, UK: Elsevier. 81 83.

Cassimatis, N. 2006. A cognitive substrate for achieving human level intelligence. *AI Magazine* 27(2):71 82.

Erev, I., Ert, E., Roth, A. E., Haruvy, E., Herzog, S., Hau, R., Hertwig, R., Stewart, T., West, R., and Lebiere,, C. 2010. A choice prediction competition, for choices from experience and from description. *Journal of Behavioral Decision Making* 23(1): 15 47.

Fauconnier, G., & Turner, M. 1998. Conceptual Integration Networks. *Cognitive Science* 22(2):133 187.

Felzenswalb, P., Girshick, R., McAllester, D., & Ramanan, D. 2010. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32 (9*)*.

Gonzalez, C., Lerch, J. F., and Lebiere, C. 2003. Instance based learning in dynamic decision making. *Cognitive Science* 27: 591 635.

Hawkins, J. 2004. *On Intelligence*. New York: Henry Holt.

Hudlicka, E. 2004. Two Sides of Appraisal: Implementing Appraisal and Its Consequences within a Cognitive Architecture. Proceedings of the AAAI Spring Symposium: Architectures for Modeling Emotion. AAAI Technical Report SS 04 02. Menlo Park, CA: AAAI Press. 24 31.

Jaidee, U., Munoz Avila, H., and Aha, D.W. 2011. Integrated learning for goal driven autonomy. In Proceedings of the Twenty Second International Joint Conference on Artificial Intelligence. Barcelona, Spain.

Kurup, U., Lebiere, C. & Stentz, A. 2011. Integrating Perception and Cognition for AGI. In Proceedings of the fourth conference on Artificial General Intelligence. Mountaiview, CA.

Laptev, I. and Lindeberg, T. 2003. Space time interest points. In Proceedings of the IEEE Interbational Conference on Computer Vision (ICCV). IEEE, Los Alamitos, CA.

Lebiere, C. The dynamics of cognitive arithmetic. 1999. Wallach, D. and Simon, H. (Eds.) *Kognitionswissenschaft* 8 (1): 5 19.

Lebiere, C., and Wallach, D. 1986. Sequence learning in the ACT R cognitive architecture: Empirical analysis of a hybrid model. In R. Sun, & L. Giles (Eds.), *Sequence Learning: Paradigms, Algorithms, and Applications*. Germany: Spinger LNCS/LNAI.

Oltramari, A., and Lebiere, C. 2011. Extending Cognitive Architectures with Semantic Resources. In Proceedings of the Fourth Conference on Artificial General Intelligence. Mountain View, CA.

O'Reilly, R.C., and Munakata, Y. 2000. *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*. Cambridge: MIT Press.

Pless, R., Dixon, M., Jacobs, N., Baker, P., Cassimatis, N., Brock, D., Hartley, R. and Perzanowski, D. 2009. Persistance and tracking: Putting vehicles and tracjectories in context. In Proceedings of the Applied Imagery Pattern Recognition Workshop. Washington, D. C. 1 8.

Puri A. M, Wojciulik E. 2008. Expectation both helps and hinders object perception. *Vision Research* 48:589 597.

Rumelhart, D.E., McClelland, J. L. and the PDP Research Group. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Cambridge, MA: MIT Press.

Schank, R. C. 1982. *Dynamic memory: A theory of reminding and learning in computers and people*. Cambridge, MA: Cambridge University Press.

Schank, R. C. and Owens, C. C. 1987. Understanding by explaining expectation failures. In R. G. Reilly (Ed.), *Communication failure in dialogue and discourse*. New York: Elsevier Science.

Simon, H. 1981. *The Sciences of the Artificial*. Cambridge, Mass: MIT Press.

Trafton, J. G., Cassimatis, N., Bugajska, M., Brock, D., Mintz, F. E. and Schultz, A. C. 2005. Enabling effective human robot interaction using perspective taking in robots. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* 35(4): 460 470.

Wallach, D., and Lebiere, C. 2003. Conscious and unconscious knowledge: Mapping to the symbolic and subsymbolic levels of a hybrid architecture. In L. Jimenez (Ed.), *Attention and Implicit Learning*. Amsterdam: John Benjamins Publishing Company.

Wintermute, S. and Laird, J. E. 2008. Bimodal Spatial Reasoning with Continuous Motion. Proceedings of the Twenty Third AAAI Conference on Artificial Intelligence (AAAI 08), Chicago, Illinois.

Zelnick Manor, L. and Irani, M. 2001. Event based analysis of video. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Los Alamitos, CA.