

# Integrating Perception and Cognition for AGI

Unmesh Kurup<sup>1</sup>, Christian Lebiere<sup>1</sup>, and Anthony Stentz<sup>2</sup>

<sup>1</sup> Department of Psychology, Carnegie Mellon University,  
5000 Forbes Ave, Pittsburgh, PA 15213

<sup>2</sup> National Robotics Engineering Consortium, Robotics Institute,  
Carnegie Mellon University, Pittsburgh, PA 15201  
{unmeshk, cl, tony}@andrew.cmu.edu

**Abstract.** Current perceptual algorithms are error-prone and require the use of additional ad hoc heuristic methods that detect and recover from these errors. In this paper we explore how existing architectural mechanisms in a high-level cognitive architecture like ACT-R can be used instead of such ad hoc measures. In particular, we describe how implicit learning that results from ACT-R's architectural features of partial matching and blending can be used to recover from errors in object identification, tracking and action prediction. We demonstrate its effectiveness by building a model that can identify and track objects as well as predict their actions in a simple checkpoint scenario.

**Keywords:** Cognitive Architectures, Integrating Perception & Action, Object Tracking, Instance-based Learning.

## 1 Introduction

Perception is a key component of any system with claims to AGI [1]. While cognition can affect perception, much of perception remains bottom-up with parallel processes that implement functions from figure-ground separation to object identification and tracking. The parallel, bottom-up nature of perception is important for an agent interacting with the external world since it allows the agent to quickly understand and categorize what it perceives and, consequently, take appropriate action. However, even with the best current algorithms, information from perception is also likely to be error-prone and probabilistic. It then falls to cognition to take this information and refine it using additional knowledge.

Approaches that deal with errors in perceptual processing are usually ad hoc and based on specialized heuristics that take advantage of the domain of interest [2]. For example, in person detection, it is often the case that people are assumed to be “on the ground” and so any data that points to a person floating above the ground can be considered as unlikely. In this paper, we take a more cognitively plausible approach by using ACT-R [3], to model a system that learns to predict the result of perception. This prediction is then used to supplement perceptual inputs in order to overcome any errors. In general, this predictive ability is not restricted to the information from perception. It can be used at the cognitive level for predicting or anticipating actions,

goals and cognitive states. However, in this paper, we limit ourselves to using predictions to refine perceptual input.

At the level of cognitive theory, the ACT-R model of perceptual refinement is an example of instance-based learning in humans where information from multiple instances are generalized and used to set up expectations about future situations. Generating expectations based on past experience is key to a number of cognitive endeavors. Expectation-based models have been developed and validated in a number of domains, including expectations of future perceptual events in sequence learning [4]; expectations of other players' moves in games [5][6]; expectations for the outcome of actions such as probabilistic payoffs [7][8]; expectations of dynamical system behavior in control problems [9][10]; mental imagery for general problem solving [11] and perspective taking [12][13]. Indeed, theories have proposed [14] [15] that the fundamental computational property of the human cortex is the completion of spatiotemporal patterns to generate expectations. Thus it seems fitting that we would explore the role of expectations in how cognition oversees perception and compensates for its shortcomings.

The current model (as well as some of the other models of instance-based learning mentioned above) is built upon two features of ACT-R's declarative memory and retrieval mechanism – partial matching and blending. In the following sections we describe the ACT-R architecture, demonstrate the effects of partial matching and blending, and describe how these effects are useful in recovering from perceptual errors in a simple checkpoint scenario.

## 2 ACT-R

The ACT-R cognitive architecture is a modular, neurally-plausible theory of human cognition. The ACT-R architecture 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 inference 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 modules that support interaction with the environment. The only way to access the content stored in a module is through that module's buffer. Modules can operate asynchronously, with the flow of information between modules synchronized 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). In its action part, a production can make changes to existing chunks in buffers or make requests for new chunks. ACT-R has a second underlying sub-symbolic (numerical) 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. ACT-R also has a set of learning mechanisms that allow an ACT-R model to learn new declarative facts and production rules as well as modify existing sub-symbolic values. A full account of ACT-R theory can be found in [16] and [3].

In this paper, we restrict further discussion about ACT-R to the declarative module since its performance is implicated in 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 this current integration.

## 2.1 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 where you can specify 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. (2) describes the equation for calculating the base-level activation of a chunk  $i$  where  $t_j$  is the time elapsed since the  $j^{\text{th}}$  reference to chunk  $i$ ,  $d$  represents the memory decay rate and  $L$  denotes the time since the chunk was created.

$$A_i = B_i + \sum_j W_j S_{ji} \quad (1)$$

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

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 (i.e., 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 chunk 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 Gaussian noise.

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

Without partial matching enabled, the retrieval mechanism only considers those chunks that match the request criteria. When partial matching is enabled, the retrieval mechanism can retrieve the chunk that matches the retrieval constraints to the greatest degree. It does this by computing a match score for each chunk that is a function of the chunk's activation and its degree of mismatch to the specified constraints. (3) is the formula for computing the match score.  $MP$  is the mismatch penalty,  $Sim(v,d)$  is the similarity between the desired value  $v$  and the actual value  $d$  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 [17].

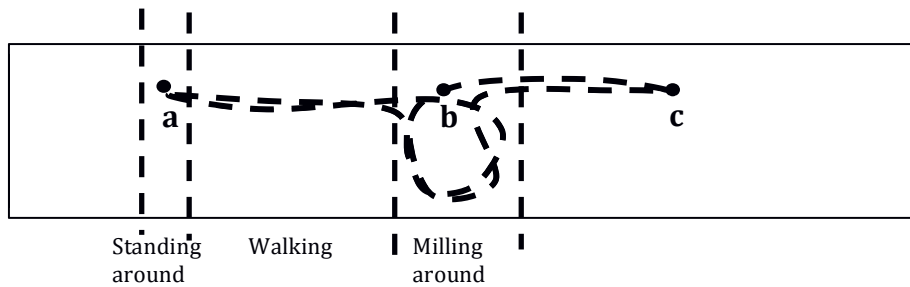
## 2.2 Blending

The second aspect of retrieval is blending [18] [19], 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 dissimilarities is returned. Formally, the value obtained by a blended retrieval is determined as follows:

$$V = Min \sum_i P_i (1 - Sim(V, V_i))^2 \quad (4)$$

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 [20] [9]. Blending of location information in chunks allows the model to predict future locations of objects by giving more weight to recent 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 [21] 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. 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 [22][23], albeit at a different level of abstraction.

Together, partial matching and blending allows the model to overcome errors in object identification and tracking. Blending also allows the model to predict the possible action an object may follow based on its past actions. This reflects the fact that, as discussed above, blending is applicable to all types of values, from discrete to continuous and including intermediate domains such as discrete values over a semantic space (e.g. words).

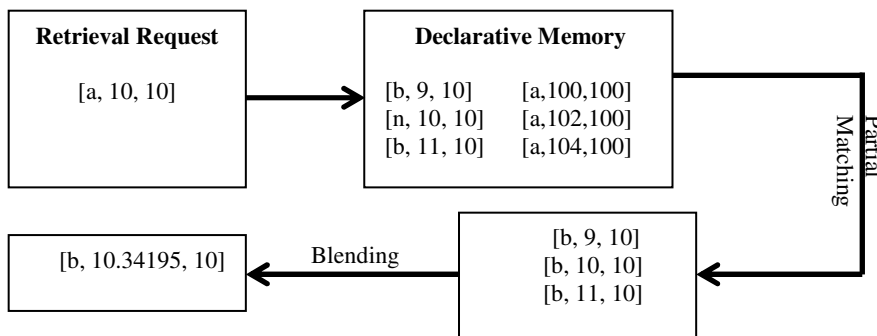


**Fig. 1.** Checkpoint scenario showing the movements of objects *a* and *c* and the division of *a*'s route by type of action

### 3 Modeling Object Identification, Tracking and Action Prediction in ACT-R

#### 3.1 Checkpoint Scenario

The checkpoint scenario consists of a robot patrolling a checkpoint looking for people and objects of interest. In the current version of the scenario, it is assumed that the robot is stationary at a location that affords it a complete view of the checkpoint. Its goal is to identify and track the objects in its view and classify their actions. Fig 1 shows the example scenario consisting of three people performing one of three actions – standing around, walking or milling around. Initially, all actors (“a”, “b” and “c”) are



**Fig. 2.** Effects of partial matching and blending on retrieval

standing around. After 30 seconds, “a” and “c” start walking towards “b”. After 60 seconds, they start milling around (walking in a circle). After 90 seconds, they start walking back towards their starting locations, going back to standing around after 120 seconds. For this scenario, perception provides an identification vector (which is currently limited to an id but might be eventually expanded to a multi-variable vector that includes additional information such as height, color, etc) and location information. However both location and object id information have errors associated with them. For now the output of perception is simulated by adding an error term to the data. Location error generated by a logistic distribution is added to the location information that is made available, while objects have a fixed 1/10 chance of being mis-identified by perception (which translates to an average of 15 errors in every trial).

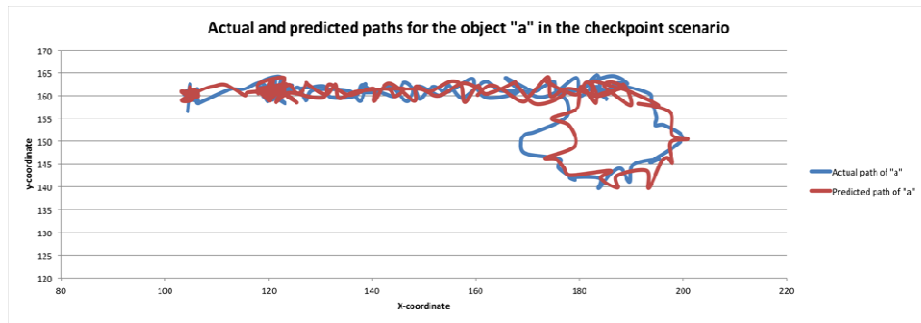
### 3.2 ACT-R Model

The ACT-R model represents information about an object in a chunk, called a visual-memory chunk for convenience, that contains the object’s id, current location, rate of change of location (delta), and current action. Every time an object id and its location are reported by perception, the model uses this information to retrieve a chunk in order to interpret that information against its recent experience. The retrieved chunk is a result of both partial matching and blending by the retrieval mechanism. The value contained in the delta slot is used to predict the object’s future location and the value in the action slot is used to predict the object’s future action. If the retrieved blended chunk does not exactly match the information from perception (which it rarely does), a new chunk is learned whose slot values contain the information from perception. Over the course of the experiment, the model learns a number of such chunks that effectively improve its ability to alleviate id errors, predict object location and predict object action. In the following sections we describe how the use of partial matching and blending accomplish this goal.

### 3.3 Recovering from Object Identification Errors

There are a number of errors that can happen during object identification including failure to identify a known (previously encountered) object, identifying non-existent objects and identifying a known object as another known object. In this work we model all three errors. In the first type of error, perception fails to identify a previously seen (and identified) object. Instead, it either identifies it as a new object or fails to identify it at all. The second type of error is similar to a false-positive, where perception identifies an object even though no such object exists in the scene. Finally, perception can be confused between different objects in the scene and identify one object incorrectly as another. In all cases, partial matching provides a way for the system to recover the right identification. To see how this works, consider the following example (shown in Fig 2) where there are 6 chunks in declarative memory, three chunks with “b” in the name slot and values (9,10) (10,10) and (11,10) in the respective current location slot and three chunks with “a” in the name slot and values (100,100), (102,100) and (104,100) in the respective (x,y) slots. When a chunk is requested with the value “a” in the name slot and (10,10) in the (x,y) slot, partial matching produces the first 3 chunks as better matches for the request even though the

value in the name slot is different. The values of each slot of the three chunks are then combined to create the new blended chunk. The same process can be used to recover from the other types of id errors. If a non-existent object was proposed instead of “a”, partial matching would retrieve the chunks with “a” in the name slot because the current locations would be most similar to the locations for chunks with “a” in the name slot. Similarly, if an object was not identified at all, its location will provide enough similarity to retrieve the correct chunks. It is important to note that this method is not simply a nearest-neighbor approach. Instead, it is architectural and takes into account contextual elements like time. For example, if an object “c” used to be at the same location as “b” but in the past, there would be corresponding chunks in memory with location slot values similar to the location slot values of object “b”. However, these chunks would not be retrieved because their activations would be low due to the fact that they were created when “c” used to be at that location (and time from creation is an important part of a chunk’s activation).

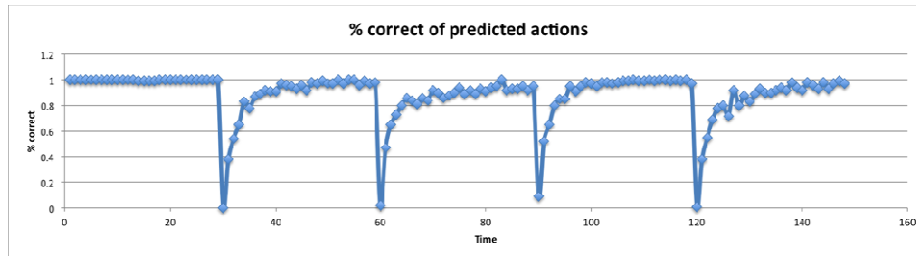


**Fig. 3.** The actual and predicted paths for object “a”

We were able to recover from most errors in the many trials we ran. However, there are other scenarios where this approach sometimes fails. For example, as two objects get closer to each other, their location values become more similar making it difficult for partial matching to retrieve the right chunks. Even so, the delta values provide another contextual cue to use if the two objects are moving at different rates or in different directions.

### 3.4 Recovering from Object Location Errors

The model predicts an object’s future location by adding a delta value to its current location where the delta value is the difference between the current and previous location. In an error-free system, this would work very well except for the cases when there is a sudden change in delta. When location errors are present, the last delta value is not enough. With ACT-R’s blending mechanism, when a visual-memory chunk is retrieved, the value for the delta slot is blended over the existing relevant chunks. As mentioned earlier, in calculating the blended value for the delta slot, the blending mechanism takes into account the activations of the chunks. One major contributor to activation strength is the time of creation of the chunk. That is, the more recently the



**Fig. 4.** Percent correct of predicted actions for object “a”

chunk was created, the higher its activation. This means that more recent chunks, and hence more recent values of delta, contribute more to the retrieved blended value making the prediction more accurate. Figure 3 shows the actual and predicted paths for object “a” from the scenario averaged over 1000 trials.

### 3.5 Predicting Actions

Finally, blending can also be used to provide robust action prediction. Recall that the scenario has 3 different actions. We concentrate on the actions performed by object “a” which includes “standing”, “walking” and “milling around”. Figure 4 shows the percentage of correct predictions for each time step in the scenario. The drop-offs correspond to the changes in action by “a” such as going from “standing” to “walking” or from “walking” to “milling around” and so on. Anytime the action changes, it takes some time for the model to get back to a high-level of prediction, as it has to wait for enough recent chunks to accumulate with the correct value in the action slot. A system that simply uses the last action to predict the next one would do better (except for the same drop-offs) but would do poorly when there are errors in the system. In general, recent experience should be combined with patterns stored from long-term experience (e.g., which action was performed by people in various circumstances) to yield predictions that reflect both short-term and long-term knowledge.

## 4 Conclusion and Future Work

The best perceptual routines available today for object identification, tracking and action recognition are often error-prone and current approaches to handling these errors are usually ad hoc. In contrast, a cognitive approach like the use of ACT-R builds on the principles of human cognition to provide a way to recover from such errors. ACT-R’s partial matching and blending features in its memory representation and retrieval provides one such cognitively plausible way. The most encouraging outcome of using ACT-R is the clear effect of existing architectural mechanisms on the problem of handling perceptual errors. However, there are problems that remain to be addressed. The drop-offs in action prediction are glaring and learning to predict these changes is a focus of on-going research. Even though the checkpoint scenario concentrated on the use of partial matching and blending to make predictions of the

immediate state of the world, these processes can just as easily be leveraged to retrieve and construct predictions about more abstract and/or long-term actions. We are currently testing a number of approaches including contextual attributes like time (actions change at approximately similar intervals of time, for example, a person starts walking after standing around for a while) and spatial relationships (a person who is walking will tend to start milling around when he/she is near to another person). We have had some success with using spatial contextual cues but problems still remain. In addition, current predictions are limited to one or a few time steps in advance. An ongoing goal is to expand this ability to predict behaviors and actions longer into the future.

Finally, in this paper, we have focused solely on leveraging existing architectural mechanisms without considering the computational cost of calculating expectations and evaluating them during problem solving. It is very likely that such processes require additional architectural support and the nature and design of such support is part of our future work.

**Acknowledgements.** This work was 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

1. Laird, J.E.: Towards Cognitive Robotics. In: SPIE Defense and Sensing Conferences, Orlando, FL (2009)
2. Wojek, C., Walk, S., Schiele, S.: Multi-Cue Onboard Pedestrian Detection. In: CVPR, pp. 1–8 (2009)
3. Anderson, J.R.: How Can the Human Mind Occur in the Physical Universe? Oxford University Press, New York (2007)
4. Lebiere, C., Wallach, D.: Sequence learning in the ACT-R cognitive architecture: Empirical analysis of a hybrid model. In: Sun, R., Giles, L. (eds.) Sequence Learning: Paradigms, Algorithms, and Applications. LNCS/LNAI. Springer, Germany (2001)
5. Lebiere, C., West, R.L.: A dynamic ACT-R model of simple games. In: Proceedings of the Twenty-First Conference of the Cognitive Science Society, pp. 296–301. Erlbaum, Mahwah (1999)
6. West, R.L., Lebiere, C.: Simple games as dynamic, coupled systems: Randomness and other emergent properties. *Journal of Cognitive Systems Research* 1(4), 221–239 (2001)
7. Lebiere, C., Gonzalez, C., Martin, M.: Instance-based decision-making model of repeated binary choice. In: Proceedings of the 8th International Conference on Cognitive Modeling, Ann Arbor (2007)
8. Erev, I., Ert, E., Roth, A.E., Haruvy, E., Herzog, S., Hau, R., Hertwig, R., Stewart, T., West, R., Lebiere, C.: A choice prediction competition, for choices from experience and from description. *Journal of Behavioral Decision Making* 23(1), 15–47 (2010)
9. Gonzalez, C., Lerch, J.F., Lebiere, C.: Instance-based learning in dynamic decision making. *Cognitive Science* 27, 591–635 (2003)

10. Lebiere, C., Gonzalez, C., Warwick, W.: Convergence and constraints revealed in a qualitative model comparison. *Journal of Cognitive Engineering and Decision Making* 3(2), 131–155 (2009)
11. Wintermute, S., Laird, J.E.: Bimodal Spatial Reasoning with Continuous Motion. In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 2008)*, Chicago, Illinois (2008)
12. Trafton, J.G., Harrison, A.M., Fransen, B.: An embodied model of infant gaze-following. *International Conference of Cognitive Modeling* (2009)
13. Trafton, J.G., Schultz, A.C., Perzanowski, D., Bugajska, M.D., Adams, W., Cassimatis, N.L., Brock, D.P.: Children and robots learning to play hide and seek. *Human Robot Interaction* (2006)
14. Granger, R.: Engines of the brain: The computational instruction set of human cognition. *AI Magazine* 27(2), 15–31 (2006)
15. Hawkins, J., Blakeslee, S.: *On Intelligence*. Times Books, NY (2004)
16. Anderson, J.R., Bothell, D., Byrne, M.D., Douglass, S., Lebiere, C., Qin, Y.: An integrated theory of the mind. *Psychological Review* 111(4), 1036–1060 (2004)
17. Anderson, J.R.: *The Adaptive Character of Thought*. Erlbaum, Hillsdale (1990)
18. Lebiere, C.: The dynamics of cognitive arithmetic. In: Wallach, D., Simon, H.A. (eds.) *Kognitionswissenschaft*, vol. 8 (1), pp. 5–19 (1999)
19. Gonzalez, C., Lebiere, C.: Instance-based cognitive models of decision-making. In: Zizzo, D., Courakis, A. (eds.) *Transfer of knowledge in economic decision making*, Palgrave MacMillan, New York (2005)
20. Wallach, D., Lebiere, C.: Conscious and unconscious knowledge: Mapping to the symbolic and subsymbolic levels of a hybrid architecture. In: Jimenez, L. (ed.) *Attention and Implicit Learning*, John Benjamins Publishing Company, Amsterdam (2003)
21. Fauconnier, G., Turner, M.: Conceptual Integration Networks. *Cognitive Science* 22(2), 133–187 (1998)
22. Rumelhart, D.E., McClelland, J.L., the PDP Research Group: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, Cambridge (1986)
23. O'Reilly, R.C., Munakata, Y.: *Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain*. MIT Press, Cambridge (2000)