***** in press, News Ideas in Psychology ****

Autonomous Learning in Psychologically-Oriented Cognitive Architectures: A Survey

Sébastien Hélie

Purdue University

Ron Sun

Rensselaer Polytechnic Institute

Running head: Autonomous learning in cognitive architectures

For correspondence,

Sébastien Hélie

Department of Psychological Sciences

Purdue University

703 Third Street

West Lafayette, IN 47907-2081

Phone: (765) 496-2692

E-mail: shelie@purdue.edu

Version R1, last modified March 6th 2014.

Abstract

This survey paper discusses the topic of autonomous learning in psychologically-oriented cognitive architectures and reviews some of the most popular cognitive architectures used in psychology, namely ACT-R, Soar, and Clarion. Autonomous learning is critical in the development of cognitive agents, and several learning-related desiderata useful for 'psychological' cognitive architectures are proposed. This article shows that all the reviewed cognitive architectures include some form of explicit ('symbolic') and implicit ('subsymbolic') learning. Additionally, ACT-R and Clarion are shown to include a top-down learning algorithm (from explicit to implicit), and Clarion also includes a bottom-up learning process (from implicit to explicit). Two simulation examples are presented with each cognitive architecture to illustrate the autonomous learning capacities of each modeling paradigm. While Clarion is more autonomous (requiring less a priori knowledge), Soar and ACT-R have so far been used in more complex tasks. The presentation concludes with some general considerations for future work.

Keywords: Cognitive Architectures, Autonomous Learning, Psychology, ACT-R, Clarion, Soar.

1 Introduction

This article surveys existing cognitive architectures in relation with autonomous learning for psychologically-realistic applications. A cognitive architecture is the essential structures and processes of a domain-generic computational cognitive model used for a broad, multiple-level, multiple-domain, analysis of cognition and behavior (Sun, 2004). Specifically, cognitive architectures deal with componential processes of cognition in a structurally and mechanistically well-defined way. Its function is to provide an essential framework to facilitate more detailed exploration and understanding of various components and processes of the mind. In this way, a cognitive architecture serves as an initial set of assumptions to be used for further development. These assumptions may be based on available empirical data (e.g., psychological or biological), philosophical thoughts and arguments, or computationally-inspired hypotheses concerning psychological processes (Sun, 2002). A cognitive architecture is useful and important precisely because it provides a comprehensive initial framework for further modeling and simulation in many task domains.

In order to achieve generality in a psychologically-realistic way, cognitive architectures should include only minimal initial structures and independently learn from their own experiences – that is, autonomous learning (Sun, 2000). Autonomous learning is an important way of developing additional structure, bootstrapping all the way to a full-fledged cognitive model (Sun, 2004). However, it is important to be careful and devise only minimal initial learning capabilities that are capable of “bootstrapping” relevant capacities for whatever phenomenon is modeled (e.g., Sun, Merrill, & Peterson, 2001). By doing so, many structures can be placed back into the world, instead of placing

them in the head of the agent (Bickard, 1993; Hutchins, 1995; Sun, 2000). The avoidance of overly complicated initial structures, and thus the inevitable use of autonomous learning, may often help to avoid overly representational models that are designed specifically for the task to be achieved (Sun, 2000). This flexibility is essential in achieving general intelligence (Newell, 1990).

The remainder of this article is organized as follows. Section 2 discusses the importance of psychologically-oriented cognitive architectures for cognitive science. Section 3 discusses some learning-related considerations (i.e., desiderata) in the development of psychologically-realistic cognitive architectures. Section 4 introduces some of the most popular cognitive architectures that are currently used in psychology and cognitive science. Section 5 presents a discussion of autonomous learning in the cognitive architectures presented in Section 4. Section 6 describes some simulation examples to illustrate the learning capabilities of the architectures reviewed in Section 4. Section 7 presents a short summary and prescriptions for future work.

2 Why are psychologically-oriented cognitive architectures important?

While there are all kinds of cognitive architectures in existence, in this survey we are concerned specifically with psychologically-oriented cognitive architectures.[1] Psychologically-oriented cognitive architectures are particularly important because they shed new light on human cognition and therefore they are useful tools for advancing the

[1] As opposed to software engineering-oriented cognitive architectures (e.g., LIDA; Franklin & Patterson, 2006), cognitive robotic architectures (e.g., SS-RICS; Kelley & Avery, 2010), or neurally-oriented cognitive architectures (e.g., ART; Carpenter & Grossberg, 1987).

understanding of cognition (Newell, 1990). In understanding cognitive phenomena, the use of computational simulation on the basis of cognitive architectures forces one to think in terms of processes, and in terms of details (Sun, 2002). Instead of using vague, purely conceptual theories, cognitive architectures force theoreticians to think clearly. They are therefore critical tools in the study of the mind (Newell, 1990). Cognitive psychologists who use cognitive architectures must specify a cognitive mechanism in sufficient detail to allow the resulting models to be implemented on computers and run as simulations. This approach requires that important elements of the models be spelled out explicitly, thus aiding in developing better, conceptually clearer theories. It is certainly true that more specialized, narrowly-scoped models may also serve this purpose, but they are not as generic and as comprehensive and thus may not be as useful to the goal of producing general intelligence (Sun, 2002, 2004).

It is also worth noting that psychologically-oriented cognitive architectures are the antithesis of “expert systems”: Instead of focusing on capturing performance in narrow domains, they are aimed to provide broad coverage of a wide variety of domains in a way that mimics human performance (Langley, Laird, & Rogers, 2009). While they may not always perform as well as expert systems, business and industrial applications of intelligent systems increasingly require broadly-scoped systems that are capable of a wide range of intelligent behaviors, not just isolated systems of narrow functionalities. For example, one application may require the inclusion of capabilities for raw image processing, pattern recognition, categorization, reasoning, decision-making, and natural language communications. It may even require planning, control of robotic devices, and interactions with other systems and devices. Such requirements accentuate the

importance of research on broadly-scoped cognitive architectures that perform a wide range of cognitive functionalities across a variety of task domains.

3 Learning-related desiderata of psychologically-realistic cognitive architectures

This section presents a short list of desiderata for the development of psychologically realistic cognitive architectures (extended from Sun, 2004). Such desiderata are considered here because they are crucially important for understanding and modeling autonomous learning in a psychologically-realistic way (Sun, 2000, 2002, 2004; Sun et al., 2001). Here, we focus on desiderata of learning processes as well as on what needs to be learned. Other non-learning-related desiderata (e.g., compositionality, universal computation) can be found in Anderson and Lebiere (2003).

3.1 Dichotomy of implicit and explicit processes

Implicit processes are often described as inaccessible and imprecise, while explicit processes are contrasted as accessible and precise (Dreyfus & Dreyfus, 1987; Helie & Sun, 2010; Reber, 1989; Smolensky, 1988; Sun, 1994; Sun, Slusarz, & Terry, 2005). These differences are related to some other well-known dichotomies such as ‘symbolic’ versus ‘subsymbolic’ processing (Sun, 1994), conceptual versus subconceptual processing (Smolensky, 1988), and conscious versus unconscious processing (Sun, 2002). It can also be justified psychologically by the voluminous empirical studies of implicit and explicit learning (Reber, 1989), implicit and explicit memory (Schacter, Wagner, & Buckner, 2000), implicit and explicit perception (Bornstein & Pittman, 1992), and so on. These empirical dichotomies are closely related,

and thus they can all serve as justification for a more general distinction between implicit and explicit cognitive processes.

3.2 Explicit learning and memory

Explicit memories are those accessible to conscious awareness (Eichenbaum, 1997). Typically this includes working memory, episodic memory, and often also (explicit) semantic memory (Anderson, Bothell, Byrne, Douglass, Lebiere, & Qin, 2004). There are several types of explicit learning, such as memorization and hypothesis testing (Ashby & O'Brien, 2005). Memorization is useful mostly in situations where no further elaboration is required (Craik & Tulving, 1975) whereas hypothesis testing (Evans, 2006) can be used not only to learn new explicit memories (e.g., rules) but also elaborate using e.g., specialization or generalization (Helie & Ashby, 2012; Sun et al., 2001). Unlike implicit learning, explicit learning can be 'one-shot' and may not require immediate or consistent feedback (Maddox, Ashby, & Bohil, 2003).

3.3 Implicit learning and memory

Implicit memories often refer to the memories of skills that are learned through practice (Willingham, 1998). Traditionally these have been motor skills, such as those used when playing golf or tennis (Beilock et al., 2002; Wilson, Sun, & Matthews, 2009), but conceptual priming and other implicit semantic memories may also be included in implicit memories (Schacter, Chiu, & Ochsner, 1993). There are several signatures of implicit learning concerning implicit memories that makes it qualitatively different from learning that is mediated by explicit memory. First, there typically is little conscious

recollection or even awareness of the details of implicit memories. Second, implicit learning is often slow and incremental (Schacter et al., 2000; Willingham, 1998).

3.4 Synergistic interaction

Recently, there have been some emerging indications of synergy between implicit and explicit cognition. Sun and colleagues (Helie & Sun, 2010; Sun, 1994, 2002; Sun et al., 2005; Wilson et al., 2009) hypothesized that the reason for having the two separate components, the implicit and the explicit, or any other similar combination of components, was that these different systems could work together synergistically, supplementing and complementing each other in a variety of different ways (Sun et al., 2005). This is because these two components have qualitatively different characteristics, thus often generating better overall results when they are combined (Breiman, 1996; Dreyfus & Dreyfus, 1987; Helie & Sun, 2010; Mathews, Buss, Stanley, Blanchard-Fields, Cho, & Druhan, 1989; Sun et al., 2005).

3.5 Bottom-up and top-down learning

The interaction between the two sides of the dichotomy allows for a gradual transfer of memories between the explicit and implicit modules (functional memory stores), in addition to explicit and implicit, within module, learning. Learning resulting from module interaction includes top–down (explicit learning first and implicit learning later) and bottom–up (implicit learning first and explicit learning later). Recent research and data suggest that bottom-up learning is one of the most important types of learning in human everyday activities (Helie, Proulx, & Lefebvre, 2011; Karmiloff-Smith, 1986; Merleau-Ponty, 1963; Stanley et al., 1989; Sun, 2002; Sun et al., 2001, 2005).

3.6 Discussion

The five desiderata listed above are necessary if autonomous learning is to be modeled in a psychologically-realistic way. Humans display all of the above mentioned types of learning (as described by the referenced papers above), and this may be the key to human adaptability and general intelligence. For example, one may encounter a completely new situation, and need to learn using trial-and-error learning (either explicit or implicit). If the new skill was developed implicitly, then bottom-up learning may be useful in transferring the skill to explicit memory. Once explicit, the new skill can be generalized, and possibly shared with the community (Karmiloff-Smith, 1986). Likewise, one may collaborate with someone who possesses a new skill that needs to be acquired. If the collaborator has an explicit representation of the skill, the new skill may be communicated to the learner, who may then acquire the skill without going through a trial-and-error process (explicit learning) (Cangelosi, Greco, & Harnad, 2002).

While there are advantages to explicitly representing skills (e.g., generality, communication), using explicit skills can be costly in attentional resources (Helie, Waldschmidt, & Ashby, 2010). It is thus useful to recode the skill implicitly (top-down learning) (Helie & Cousineau, 2011). One common example is driving. Once driving has become an implicit skill, one may be able to hold a conversation with a passenger while driving without much impairment. Thus each kind of learning described in this section allows humans to learn autonomously in some situations and may be required for general adaptability and intelligence.

4 Example cognitive architectures

Below, we first provide an overview of some popular cognitive architectures used extensively in psychology, namely ACT-R, Soar, and Clarion. Following this presentation, we provide a shorter, more high-level description of other cognitive architectures that are not used as extensively in psychology to model human learning. In Section 5, each of the psychologically-oriented cognitive architecture is evaluated in relation to the learning-related desiderata described in Section 3.

4.1 Adaptive Control of Thought-Rational (ACT-R)[2]

ACT-R is based on three key ideas (Taatgen & Anderson, 2008): (a) rational analysis, (b) the distinction between procedural and declarative memories and, (c) a modular structure linked with communication buffers (see Figure 1). According to the rational analysis of cognition (first key idea; Anderson, 1990), the cognitive architecture is optimally tuned to its environment within its computational limits. Hence, the functioning of the architecture can be understood by investigating how optimal behavior in a particular environment would be implemented. According to Anderson (1990), such optimal adaptation is achieved through evolution.

Insert Figure 1 about here

The second key idea, the distinction between declarative and procedural memories, is implemented using different modules in ACT-R, each with its own representational format and learning rule. Procedural memory is represented by production rules that can act on the environment (through the initiation of motor actions). In contrast, declarative memory is passive and uses chunks to represent world knowledge

[2] ACT-R papers and software are available at: http://act-r.psy.cmu.edu/.

that can be accessed by the procedural memory but does not interact directly with the environment.

The last key idea in ACT-R is modularity. As seen in Figure 1, procedural memory (i.e., the production system) cannot directly access information from the other modules: the information has to go through dedicated buffers. Each buffer can hold a single chunk at any time. Hence, buffers serve as information processing 'bottlenecks' in ACT-R. This restricts the amount of information available to the production system, which in turn limits the processing that can be done by this module at any given time. Processing within each module is encapsulated (Fodor, 1983). Hence, all the modules can operate in parallel without much interference. The following subsections describe the different ACT-R modules in more details.

4.1.1 The perceptual-motor modules

The Perceptual-Motor modules in ACT-R have been a recent addition (Anderson et al., 2004). The modules include a detailed representation of the output of perceptual systems, and the input of motor systems. In essence, these ACT-R modules are adaptations from more detailed EPIC (Meyer & Kieras, 1997) perceptual-motor modules (Taatgen & Anderson, 2008). The manual module is in charge of transforming action chunks from the production system into motor output to act on the environment. However, the functioning of this module has not been detailed at a level similar to that of the visual modules (Anderson et al., 2004; Taatgen & Anderson, 2008) and thus will not be further discussed.

The visual module has been further divided into the well-established ventral (what) and dorsal (where) visual streams in the human brain (Ungerleider & Haxby,

1994). The main function of the dorsal stream module is to find the location of features in a display (e.g., red colored items, curved shapes) without identifying the objects. The output from this module is a location chunk, which can be sent back to the central production system. The function of the ventral stream module is to identify the object at a particular location. For instance, the central production system could send a request to the dorsal stream module to find a red object in a display. The dorsal stream module would search the display and return a chunk representing the location of a red object. If the central production system needs to know the identity of that object, the location chunk would be sent to the ventral stream module. A chunk containing the object identity (e.g., a fire engine) would be returned to the production system. As such, the visual modules in ACT-R can be more accurately described as 'visual attention' modules rather than actual visual modules (Anderson et al., 2004).

4.1.2 The goal module

The goal module serves as the context for keeping track of cognitive operations and supplement environmental stimulations (Anderson et al., 2004). For instance, one can do many different operations with a pen picked up from a desk (e.g., write a note, store in a drawer, etc.). What operation is selected depends primarily on the goal that needs to be achieved. If the current goal is to clean the desk, the appropriate action is to store the pen in a drawer. If the current goal is to write a note, putting the pen in a drawer is not a useful action.

In addition to providing a mental context to select appropriate production rules, the goal module can be used in more complex problem solving tasks that need subgoaling (Anderson et al., 2004). For instance, if the goal is to play a game of tennis, one first

needs to find an opponent. The goal module must create this subgoal that needs to be achieved before moving back to the original goal (i.e., playing a game of tennis). Note that goals are centralized in a unique module in ACT-R and that production rules only have access to the goal buffer. The current goal to be achieved is the one in the buffer, while later goals stored in the goal module are not accessible to the production. Hence, the 'play a game of tennis' goal is not accessible to the production rules while the 'find an opponent' subgoal is being pursued. This restriction on goal processing is currently being investigated by ACT-R researchers and is not a central tenet (Anderson et al., 2004).

4.1.3 The declarative module

The declarative memory module contains knowledge about the world in the form of chunks (Anderson et al., 2004). Each chunk represents a piece of knowledge or a concept (e.g., fireman, bank, etc). Chunks can be accessed effortfully by the central production system, and the probability of retrieving a chunk depends on the chunk activation. The chunk activation is the sum of a base-level activation (which represents the odds of needing that chunk in the future) and contextual activation (as a function of the current goal and the environment state). The retrieval time of a chunk is negatively related to its activation. It is important to note that the knowledge chunks in the declarative module are passive and do not do anything on their own. The function of this module is to store information so that it can be retrieved by the central production system (which corresponds to procedural memory). Only in the central production system can the knowledge be used for further reasoning or to produce actions.

Finally, the ACT-R theory of declarative knowledge includes the distinction between explicit and implicit memories. Explicit memory corresponds to the chunks that are present in the declarative module (i.e., the memory structure) while implicit memory corresponds to the "subsymbolic" activation that govern the availability of the chunks in the declarative module (Anderson et al., 2004).

4.1.4 The procedural memory

The procedural memory is captured by the central production rule module and fills the role of a central executive processor. It contains a set of rules in which the conditions can be matched by the chunks in all the peripheral buffers and the output is a chunk that can be placed in one of the buffers. The production rules are chained serially, and each rule application takes a fixed amount of psychological time. The serial nature of central processing constitutes another information processing bottleneck in ACT-R (Anderson et al., 2004).

Because only one production rule can be fired at any given time, its selection is crucial. In ACT-R, each production rule has a utility value that depends on: a) its probability of achieving the current goal, b) the value (importance) of the goal and, c) the cost of using the production rule (Anderson et al., 2004). The first two factors are positively related to the rule utility while the last is negatively related to the rule utility. The most useful rule is always chosen on every processing cycle, but the utility values are noisy, which can result in the selection of sub-optimal rules (Taatgen, Lebiere, & Anderson, 2006).

Rule utilities are learned online by counting the number of times that applying a rule has achieved the goal. Anderson (1990) has shown that the selection according to

these counts is optimal in the Bayesian sense. Also, production rules can be made more efficient by using a process called *production compilation* (Taatgen & Anderson, 2004; Taatgen et al., 2006). Specifically, if two production rules are often fired in succession and the result is positive, a new production rule is created which directly links the conditions from the first production rule to the action following the application of the second production rule. Hence, the processing time is cut in half by applying only one rule instead of two (because rule application is serial).

4.2 Soar[3]

According to the Soar theory of intelligence (Laird, Newell, & Rosenbloom, 1987), human intelligence is an approximation of a knowledge system (Taatgen & Anderson, 2008). Hence, the most important aspect of intelligence (natural or artificial) is the use of all the available knowledge (Lehman, Laird, & Rosenbloom, 2006), and failures of intelligence are failures of knowledge (Wray & Jones, 2006).

All intelligent behaviors can be understood in terms of problem solving in Soar (Taatgen & Anderson, 2008). As such, Soar is implemented as a set of *problem-space computational models* (PSCM) that partition the knowledge in goal relevant ways (Lehman et al., 2006). Each PSCM implicitly contains the representation of a problem space defined by a set of states and a set of operators that can be understood using a decision tree (Wray & Jones, 2006). In a decision tree representation, the nodes represent the states, and one moves around from state to state using operators (the branches/connections in the decision tree). The objective of a Soar agent is to move from

[3] A simulation environment, papers, and details can be found on the Soar homepage: http://sitemaker.umich.edu/soar/home.

an initial state to one of the goal states, and the best operator is always selected at every time step (Lehman et al., 2006). If the knowledge in the model is insufficient to select a single best operator at a particular time step, an impasse is reached, and a new goal is created to resolve the impasse. This new goal defines its own problem space and set of operators.

4.2.1 Architectural representation

The general architecture of Soar is shown in Figure 2. The main structures are a working memory and a long-term memory. Working memory is a blackboard where all the relevant information for the current decision cycle is stored (Wray & Jones, 2006). It contains a goal representation, perceptual information, and relevant knowledge that can be used as conditions to fire rules. The outcome of rule firing can also be added to the working memory to cause more rules to fire. The long-term memory contains associative rules representing the knowledge in the system in the form of “IF --> THEN” rules. The rules in long-term memory can be grouped/organized to form operators.

Insert Figure 2 about here

4.2.2 The Soar decision cycle

In every time step, Soar goes through a six-step decision cycle (Wray & Jones, 2006). The first step in Soar is to receive an input from the environment. This input is inserted into working memory. The second step is called the *elaboration phase*. During this phase, all the rules matching the content of working memory fire in parallel, and the result is put into working memory. This in turn can create a new round of parallel rule

firing. The elaboration phase ends when the content of working memory is stable, and no new knowledge can be added in working memory by firing rules.

The third step is the proposal of operators that are applicable to the content of working memory. If no operator is applicable to the content of working memory, an impasse is reached. Otherwise, the potential operators are evaluated and ordered according to a preference metric. Step four is the selection of a single operator. If the knowledge does not allow for the selection of a single operator, an impasse is reached. The fifth step is to apply the operator. If the operator does not result is a change of state, an impasse is reached. Finally, step six is the output of the model, which can be an external (e.g., motor) or an internal (e.g., more reasoning) action.

4.2.3 Impasses

When the immediate knowledge is insufficient to reach a goal, an impasse is reached and a new goal is created to resolve the impasse. Note that this subgoal produces its own problem space with its own set of states and operators. If the subgoal reaches an impasse, another subgoal is recursively created to resolve the second impasse, and so on. There are four main types of impasses in Soar (Wray & Jones, 2006).

The first type of impasses happens when no operator is available in the current state (i.e., the third step in the decision cycle returns zero operator recommendation). In this case, a new goal is created to find an operator that is applicable in the current state. The second type of impasses happen when an operator is applicable in the current state, but its application does not change the current state. Here, the new goal that is created aims at modifying the operator so that its application changes the current state. Alternatively, the operator could be modified so that it is no longer deemed applicable in

the current state. The third type of impasses is called a ‘tie’. A tie happens when two or more operators are applicable in the current state but neither is preferred. The new goal created by this impasse is to further evaluate the options and make one of the operators preferred to the others. The fourth type of impasse results from having more than one applicable operator, and simultaneously having knowledge in working memory favoring two or more operators in the current state. In this case, further processing is needed, and the new goal is to resolve the conflict by removing from working memory one of the contradictory preferences.

Regardless of which type of impasses is reached, resolving an impasse is an opportunity for learning in Soar (Wray & Jones, 2006). Each time a new result is produced while achieving a subgoal, a new rule associating the current state with the new result is added in long-term memory to ensure that the same impasse will not be reached in the future. This new rule is called a ‘chunk’[4] to distinguish it from rules that were precoded by the modeler (and learning is called ‘chunking’). This learning method is deductive (Lehman et al., 2006).

4.2.4 Extensions

Unlike ACT-R (and Clarion, as described next), Soar was originally designed as an artificial intelligence model (Lehman et al., 2006). Hence, initially, more attention has been paid to functionality and performance than to psychological realism. However, Soar has been used in psychology and Soar 9 has been extended to increase its psychological realism (Laird, 2008). This version of the architecture is illustrated in Figure 2. First, the

[4] Not to be confounded with the use of ‘chunk’ in ACT-R. In ACT-R, a chunk is a passive knowledge representation residing in declarative memory. In Soar, a chunk is a new associative rule learned by the model.

long-term memory has been further subdivided in correspondence with psychology theory (Schacter et al., 2000). The associative rules (as described above) are now part of the procedural memory. In addition to procedural memory, long-term memory now also includes a semantic and an episodic memory. Semantic memory contains knowledge structures representing factual knowledge about the world (e.g., the earth is round), while episodic memory contains a snapshot of working memory representing an 'episode' (e.g., Fido the dog is now sitting in front of me). Unlike procedural memory, semantic and episodic memories allow for the retrieval of memory structures/episodes with partial match of conditions.

Another addition is the presence of activation in working memory to capture recency/usefulness (as in ACT-R). While the activation is not used to select the rules to be fired (because all rules fire simultaneously in Soar), it is a useful addition when the content of working memory is stored in episodic memory and can guide recall. It should be noted that this numerical activation is a new non-symbolic component in Soar. Another new non-symbolic component in Soar is the use of numerical values to model operator preferences. These are akin to utility functions, and can be learned using reinforcement learning (Laird, 2008). Finally, recent work has been initiated to include a clustering algorithm that would allow for the creation of new symbolic structures and a visual imagery module to facilitate symbolic spatial reasoning. Also, the inclusion of emotions is now being considered (via appraisal theory; Scherer, 2001).

4.3 Clarion[5]

Clarion is an integrative cognitive architecture consisting of a number of distinct subsystems with a dual representational structure in each subsystem (implicit versus explicit representations; see Figure 3). Clarion is based on the following basic principles (Sun, 2002). First, humans can learn continuously from on-going experience in the world without much a priori knowledge (as psychological research suggests that autonomous learning is an important cognitive capability). Second, there are different types of knowledge involved in human learning (e.g., procedural vs. declarative, implicit vs. explicit; Sun, 2002), and different types of learning processes are involved in acquiring different types of knowledge. Third, motivational processes as well as meta-cognitive processes are important and should be incorporated in a psychologically realistic cognitive architecture.

Insert Figure 3 about here

The Clarion subsystems include the action-centered subsystem (the ACS), the non-action-centered subsystem (the NACS), the motivational subsystem (the MS), and the meta-cognitive subsystem (the MCS). The role of the ACS is to control actions, regardless of whether the actions are for external physical movements or for internal mental operations (e.g., goal setting, working memory operations, initiating reasoning, etc.). The role of the NACS is to maintain and reason over general knowledge. The role of the MS is to provide underlying motivations for sustainability, purposefulness, focus, and adaptivity (Sun, 2009). The role of the MCS is to monitor, direct, and modify

[5] More details and a simulation environment can be found on the Clarion project webpage: http://www.clarioncognitivearchitecture.com/.

dynamically the operations of the other subsystems in terms of providing impetus and feedback (e.g., indicating whether outcomes are satisfactory or not).

Each of these interacting subsystems consists of two "levels" of representations. In each subsystem, the top level encodes explicit knowledge and the bottom level encodes implicit knowledge. The distinction of implicit and explicit knowledge has been amply argued for before (see, e.g., Helie & Sun, 2010; Reber, 1989; Seger, 1994; Stanley et al., 1989; Sun, 2002; Sun et al., 2005; Wilson et al., 2009). The two levels interact, for example, by cooperating through a combination of the action recommendations from the two levels respectively, as well as by cooperating in learning through bottom-up and top-down processes. Essentially, it is a dual-process theory of mind (Sun, 2002).

4.3.1 The Action-Centered Subsystem

The ACS includes a top and a bottom level. The bottom level of the ACS uses distributed representations (e.g., connectionist networks) that learn implicit reactive routines (incorporating the situated cognition view). Clarion was the first cognitive architecture centered on such implicit reactive routines with reinforcement learning (Sun et al., 2001). In addition, the bottom level of the ACS is modular; that is, a number of networks co-exist, each of which is adapted to a specific modality, task, or group of input stimuli. This coincides with the modularity claim that much processing is done by limited, encapsulated, and specialized processors that are highly efficient (Fodor, 1983; Karmiloff-Smith, 1986; Sun, 2004). These modules can be developed by interacting with the world (computationally, through various decomposition methods; see, e.g., Sun & Peterson, 1999). However, some of them are formed evolutionarily, reflecting hardwired instincts and propensities (Hirschfield & Gelman, 1994). Because of these networks,

Clarion is able to handle very complex situations that are not amenable to simple rules (see Section 6.3 below).

In the top level of the ACS, explicit conceptual knowledge is captured in the form of symbolic rules (for details, see Sun, 2002). There are many ways in which explicit knowledge may be learned, including independent hypothesis-testing and bottom-up learning. The basic process of bottom-up learning is as follows (Sun et al., 2001): if an action implicitly decided by the bottom level is successful, then the model extracts an explicit rule that corresponds to the action selected by the bottom level and adds the rule to the top level. Then, in subsequent interactions with the world, the model verifies the extracted rule by considering the outcome of applying the rule: if the outcome is not successful, then the rule should be made more specific; if the outcome is successful, the agent may try to generalize the rule to make it more universal (Michalski, 1983). After explicit rules have been learned, a variety of explicit reasoning methods may be used. Learning explicit conceptual representations at the top level can also be useful in enhancing learning of implicit reactive routines in the bottom level (e.g., Sun et al., 2001).

Although Clarion can learn even when no a priori or externally provided knowledge is available, it can make use of such knowledge when available (Anderson, 1983). To deal with instructed learning, externally provided knowledge can (1) be combined with existent conceptual structures at the top level (i.e., internalization), and (2) be assimilated into implicit reactive routines at the bottom level (i.e., assimilation).

4.3.2 The Non-Action-Centered Subsystem

The NACS may be used for representing general knowledge about the world constituting the semantic memory and the episodic memory, and for performing various kinds of memory retrieval and inference (Sun, 2002). The NACS is also composed of two levels (a top and a bottom level) and is under the control of the ACS (through its actions).

At the bottom level, "associative memory" networks encode non-action-centered implicit knowledge. Associations are formed by mapping an input to an output (such as mapping "2+3" to "5").[6] The backpropagation (Sun et al., 2001, 2005) or Hebbian (Helie & Sun, 2010) learning algorithms can be used to establish such associations between pairs of inputs and outputs. In bottom-level networks, chunks[7] are specified through dimensional values (features). At the top level of the NACS, a general knowledge store encodes explicit non-action-centered knowledge (Helie & Sun, 2010; Sun, 1994). A node is set up in the top level to represent a chunk. The chunk node connects to its corresponding features represented as individual nodes in the bottom level of the NACS (see, e.g., Helie & Sun, 2010; Sun, 1994). Additionally, links between chunk nodes encode explicit associations between pairs of chunks, known as 'associative rules'. Similar to the processes in the ACS, explicit associative rules may be learned in a variety of ways (Sun, 2002).

During reasoning, in addition to applying associative rules, similarity-based reasoning may be employed in the NACS. Specifically, a known (given or inferred) chunk may be automatically compared with another chunk. If the similarity between them is sufficiently high, then the latter chunk is inferred (for details, see Sun, 1994;

[6] Note that the output can be the same as the input, as is the case for auto-associative networks.

[7] In Clarion, chunks are passive knowledge structures, similar to ACT-R.

2002). As in the ACS, top-down or bottom-up learning may take place in the NACS, either to extract explicit knowledge in the top level from the implicit knowledge in the bottom level or to assimilate explicit knowledge of the top level into implicit knowledge in the bottom level.

4.3.3 The Motivational and Meta-Cognitive Subsystems

The motivational subsystem (the MS) is concerned with drives and their interactions (Toates, 1986). It is concerned with why an agent does what it does. Simply saying that an agent chooses actions to maximize gains, rewards, reinforcements, or payoffs leave open the question of what determines these things (Sun, 2009). The relevance of the MS to the ACS lies primarily in the fact that it provides the context in which the goal and the reinforcement of the ACS are set. It thereby influences the working of the ACS, and by extension, the working of the NACS.

Dual motivational representations are in place in Clarion. The explicit goals (such as “finding food”) of an agent may be generated based on internal drives (for example, “being hungry”; see Sun, 2009 for details). Beyond low-level drives (concerning physiological needs), there are also higher-level drives (which are more sociologically based). Some of them are primary, in the sense of being “hard-wired”. For example, Sun (2009) developed a set of these drives: affiliation & belongingness, dominance & power, nurturance, fairness, honor, curiosity, etc. While primary drives are built-in and relatively unalterable, there are also “derived” drives, which are secondary, changeable, and acquired mostly in the process of satisfying primary drives.

The meta-cognitive subsystem (the MCS) is closely tied to the MS. The MCS monitors, controls, and regulates cognitive processes for the sake of improving cognitive

performance (Nelson, 1993; Smith, Shield, & Washburn, 2003). Control and regulation may be in the forms of setting goals, setting parameters, interrupting and changing on-going processes, and so on. Control and regulation can also be carried out through setting reinforcement functions. All of the above can be done on the basis of drive activation in the MS as well as other factors, such as performance monitoring statistic or environmental/contextual queues. The MCS is also made up of two levels: the top level (explicit) and the bottom level (implicit), although generally only the bottom level is used.

4.4 Other cognitive architectures

Several other cognitive architectures have been proposed in recent years (for surveys, see, e.g., Chong, Tan, & Ng, 2007; Langley et al., 2009). The most popular of these in psychology is EPIC (Kieras & Meyer, 1997; Meyer & Kieras, 1997), which has been used mostly for the purpose of human-computer interaction. EPIC is briefly discussed below. Other cognitive architectures have been proposed mostly in computer science (e.g., LIDA; Franklin & Patterson, 2006), robotics (e.g., SS-RICS; Kelley & Avery, 2010), or in computational neuroscience (e.g., ART; Carpenter & Grossberg, 1987). As such, most of these have had little impact on cognitive psychology research. A more complete list is presented in Vernon, Metta, and Sandini (2007).

Similar to ACT-R, EPIC (Kieras & Meyer, 1997; Meyer & Kieras, 1997) is a production system aimed at modeling psychological phenomena. However, unlike the cognitive architectures discussed above, EPIC focuses on the interaction between a cognitive agent and its environment (*embodied cognition*). As such, special care was devoted to modeling perceptual-motor components. EPIC is composed of a long-term

memory (roughly equivalent to the declarative memory in ACT-R), a production memory (roughly equivalent to the procedural memory in ACT-R), a cognitive processor, and a working memory (similar to the module buffers in ACT-R). Hence, EPIC is very similar to ACT-R except that all rules fire simultaneously in EPIC (the processing bottleneck is peripheral) and that the perceptual-motor modules have been more developed than the central processing modules. As such, EPIC has been applied mainly to explain human-computer interaction tasks (e.g., Kieras & Meyer, 1997). The EPIC philosophy is to start small and add components when needed in simulating a task (Kieras & Meyer, 1997). So far, EPIC does not include learning mechanisms and therefore we will not discuss it further.

5 Autonomous learning in cognitive architectures

Autonomous learning should be an essential part of any psychologically-realistic cognitive architecture. From a psychological perspective, learning is essential to being humans, and facilitates adaptation to complex environments. From a cognitive scientist's perspective, autonomous learning is important for psychological theorizing and model development. From an artificial intelligence perspective, learning is essential in reducing the time and effort needed to collect and code the knowledge necessary for performing a task. The different cognitive architectures reviewed in the present survey have each emphasized learning to different degrees (Taatgen & Anderson, 2008). However, they all included some forms of learning (Chong et al., 2007). Below we address in more details the learning processes of the cognitive architectures in relation to autonomous learning.

5.1 ACT-R

The main learning methods in ACT-R include production compilation and a Bayesian learning algorithm for rule utility (Anderson et al., 2004; Taatgen & Anderson, 2002). Production compilation is a form of deductive learning that allows for the creation of more specific production rules, thus eliminating the need for sequentially firing several rules. In contrast, rule utility is learned using a statistical learning process that optimally matches the probability of using a production rule with its past frequency of success. Together, these two learning processes allow for modeling learning trajectories in psychology tasks, as well as gradual shifts in response strategy.

In ACT-R, the focus has traditionally been on top-down learning: Turning externally provided declarative knowledge into procedural knowledge. This is because ACT-R can only generate new rules using some arrangement of the existing rule structure. For instance, modeling an inductive process in ACT-R requires the addition of rules representing general cognitive strategies to allow a deductive learning process (i.e., production compilation) to emulate an inductive learning process (Taatgen et al., 2006; for an example, see Section 6.1.2 below). Also, ACT-R does not naturally exploit statistical regularities to create completely new production rules. Statistical regularities are only used to fine-tune the existing knowledge structure.

Another learning possibility is “strategy shift” (Taatgen et al., 2006). For instance, solving a problem using rules can sometimes be more demanding than retrieving solutions from memory in ACT-R (Taatgen & Wallach, 2002). This is because repeated experiences with the environment automatically create or increase the base-level activation of memory traces stored in declarative memory (see Section 6.1.1 below for an

example). Hence, memory retrieval can eventually replace rule application when the rules are costly.[8]

Overall, ACT-R has been successful at modeling human learning, but learning algorithms have to start with a large number of a priori (or externally provided) production rules, which can make learning less autonomous and the model development processes cumbersome (Karmiloff-Smith, 1986; Sun et al., 2001). While one can argue that some of the included knowledge is innate or has been learned in previous developmental stages, the justification can be tricky in some cases.

5.2 Soar

The Soar cognitive architecture has its roots in the classical (i.e., first wave; Russell & Norvig, 1995) artificial intelligence tradition (Lehman et al., 2006). As such, emphasis has initially been put on pre-inserting expert knowledge relevant for problem solving. However, not all situations can be anticipated, and learning was incorporated in the form of 'chunking' (Wray & Jones, 2006). Chunking is used for learning each time an impasse is reached. Chunking produces a new production rule that allows the avoidance of the same impasse in the same situation in the future. Like the compilation production process in ACT-R, chunking is a deductive, compositional learning mechanism (e.g., explanation-based learning; Minton, 1990). It has been used to emulate several other learning methods, such as abduction (Lehman et al., 2006), learning by instructions (for an example, see Section 6.2.1 below), and generalization (Wray & Jones, 2006).

The limited role of learning in 'traditional' Soar has been acknowledged (Wray & Jones, 2006) and the latest implementation (Soar 9) aims at closing the gap with other

[8] Consistent with Logan's (1988) instance-based theory of automaticity.

cognitive architectures (Laird, 2008). Soar 9 includes a reinforcement learning algorithm (similar to what was in Clarion) allowing numerical preference values for the operators to be learned (for an example, see the Section 6.2.2 below). Numerical operator preferences play a similar role to rule utilities in ACT-R (Laird, 2008). Work is also now being done to add a clustering algorithm to learn new symbolic structures.

To summarize, Soar initially only included symbolic, deductive learning structures that limited its ability to autonomously learn and adapt to the environment without much a priori knowledge to begin with. However, recent new developments in Soar 9 are more promising.

5.3 Clarion

Clarion has been a psychologically-driven cognitive architecture from its inception (Sun, 2002, Sun et al., 2001). The focus in Clarion is on learning of both explicit and implicit knowledge, as well as the consequence of their interaction on learning processes (Sun et al., 2005). As such, Clarion includes symbolic rule-learning in the top level (for explicit knowledge) and non-symbolic/statistical learning in the bottom level (for implicit knowledge). These learning algorithms are capable of similar functionalities as learning in ACT-R and Soar (Sun, 2002).

However, Clarion also includes learning mechanisms that are unique to this cognitive architecture and facilitate autonomous learning. First, bottom-level implicit learning is trial-and-error based and requires (almost) no a priori knowledge to begin with (except for structures). Second, bottom-up learning (i.e., the transformation of implicit knowledge into explicit knowledge) allows for the creation and the subsequent tuning of new symbolic rule structures (Sun et al., 2001). Briefly, the bottom level of Clarion learns

statistical regularities in the environment (implicit knowledge) using neural networks trained with, e.g., reinforcement learning. The information in the bottom level is used to create new symbolic structures (rules and symbols) or tune existing rules (e.g., generalization, specialization; for an example, see Section 6.3.2 below). Third, top-down learning (i.e., the transformation of explicit knowledge into implicit knowledge) allows for the assimilation of symbolic knowledge (in the top level) into a non-symbolic format in the bottom level. This is done, e.g., through training the bottom-level networks with instances encountered (or consistent with the symbolic rules). These two types of learning favor the creation of redundant representations in the top and bottom levels, which leads to Clarion's robustness and synergistic performance (Helie & Sun, 2010; Sun et al., 2005).

In addition to the multiple learning algorithms included in Clarion, the motivational and the meta-cognitive subsystems play a key role in autonomous learning and adaptation. The inclusion of the MS allows the model to autonomously select what is important (i.e., what its goals should be) and what needs to be learned. The MCS allows for the internal selection of learning mechanisms and tuning of learning processes and parameters. Both of these structures, essential for fully autonomous learning, are absent from other cognitive architectures.[9] Overall, Clarion is currently more autonomous because it can adapt to its environment with a minimum amount of a priori (pre-inserted) knowledge (compared with ACT-R and Soar). However, this does not prevent Clarion from using pre-existing knowledge if available (as is possible in ACT-R and Soar).

[9] Although the goal structures in ACT-R and Soar can be used to model some motivational and meta-cognitive phenomena.

5.4 The role of connectionist networks in autonomous learning

ACT* (ACT-R's early version; Anderson, 1983) and Soar (Laird et al., 1987) were some of the first cognitive architectures available and have been around since the early eighties while Clarion was first proposed in the mid-nineties (Sun, 1994). This chronology is crucial when exploring their autonomous learning capacity. ACT* and Soar were developed before the connectionist revolution of the PDP Research Group (Rumelhart, McClelland, & The PDP Research Group, 1986), and are therefore implemented using knowledge-rich production systems (Russell & Norvig, 1995). In contrast, Clarion was proposed after the connectionist revolution and thus makes use of connectionist networks. While some attempts have been made to implement ACT-R (Lebiere & Anderson, 1993) and Soar (Cho, Rosenbloom, & Dolan, 1991) with neural networks, these architectures remain mostly knowledge rich production systems grounded in the artificial intelligence tradition (but see SAL below). One of the most important impacts of the connectionist revolution has been data-driven learning rules (e.g., backpropagation) that allows for autonomous learning. Clarion was created within this tradition, and every component in Clarion can be implemented using connectionist networks. For instance, explicit knowledge may be implemented using linear two-layer neural networks (e.g., Helie & Sun, 2010; Sun et al., 2001, 2005; Sun & Helie, 2013) while implicit knowledge can be implemented using, e.g., nonlinear multilayer backpropagation networks in the ACS (e.g., Sun et al., 2001, 2005) and recurrent associative memory networks in the NACS (e.g., Helie & Sun, 2010; Sun & Helie, 2013). This general philosophy has also been applied to modeling the MS and MCS using linear (explicit) and nonlinear (implicit) neural networks (Wilson et al., 2009). As such, Clarion

requires less pre-coded knowledge to achieve its goals, and can be considered more autonomous. However, future work on ACT-R and Soar can conceivably revive efforts to transform these cognitive architectures using connectionist network components to allow for more autonomous learning.

One promising example connectionist cognitive architecture effort is SAL (Jilk, Lebiere, O'Reilly, & Anderson, 2008). SAL uses the Leabra framework (O'Reilly, 1996) to implement the ACT-R cognitive architecture. Jilk and colleagues argue that ACT-R and Leabra provide convergent descriptions of similar phenomena (albeit at different levels) that are highly compatible, and that therefore integrating these two frameworks can provide for a more complete description of many phenomena encompassing both the neurobiological and the psychological/cognitive levels. SAL has already been used to simulate a navigation task, and the results show a synergistic integration in which SAL can provide a more complete explanation than either ACT-R or Leabra alone (Jilk et al., 2008). These results are highly promising and SAL could be a significant game changer in the realm of autonomous learning in cognitive architectures. Hence, if successful, future work with SAL may provide for a new way to benchmark autonomous learning in psychologically-oriented cognitive architectures.

5.5 Revisiting the desiderata

This section so far presented an overview of the learning mechanisms included in some of the most successful cognitive architectures used in psychology. This subsection evaluates these architectures with regard to the learning-related desiderata from Section 3. A summary of each cognitive architecture’s desiderata-related components is shown in Table 1.

Insert Table 1 about here

ACT-R is arguably the most widely known cognitive architecture. It succeeded in capturing a variety of human data in many different task domains, ranging from skill learning to language production. It can learn sequences of events, and chain them using production compilation. Also, learning of the utility function is incremental and done on a trial-by-trial basis. However, ACT-R does not have a clear-cut (process-based or representation-based) distinction between implicit and explicit processes, and thus explicit and implicit memories are not separated into different modules. While both explicit (production compilation) and implicit (reinforcement learning of the utility functions) learning is implemented in ACT-R, the lack of modularity in terms of explicit and implicit memories prevents the exploration of the interactions between implicit and explicit processes. Relatedly, ACT-R addresses top-down learning (using production compilation) but does not directly address bottom–up learning. Both types of learning have been shown to be important to cognition (Sun, 2002). To summarize, ACT-R already possesses many learning capabilities (as described above), but more complete autonomous learning may be achieved by adding a modular separation of explicit and implicit knowledge. This could allow for the integration of explicit and implicit processing and a bottom-up learning procedure.

Soar is based on the ideas of problem spaces, states, and operators. Prominently in the model is a goal stack. When there is an outstanding goal on the goal stack, different productions propose different operators and operator preferences for accomplishing the goal. Learning consists of chunking—the creation of a new production that summarizes the process leading up to achieving a goal, so as to avoid subsequent impasses. Because it incorporates few psychological constraints (not based on empirical data), Soar is free in

implementing complex algorithms and thus is able to deal with very complex tasks (e.g., real-time large-scale multi-agent simulation, spatial problem solving, theorem proving etc). Soar can learn sequences of behaviors and automatize them using chunking. While Soar traditionally only included symbolic 'one-shot' learning, Soar 9 now includes a reinforcement learning algorithm that constitute trial-and-error adaptation. However, as in ACT-R, Soar includes no clear (representation-based) difference between implicit and explicit cognition (Sun et al., 2005). These two types of memory are not separated into distinct modules. However, symbolic learning (e.g., chunking) shares many of the explicit learning characteristics (in 'traditional' Soar), and subsymbolic learning of preferences (e.g., in Soar 9) shares many of the characteristics of implicit learning. Hence, in a sense, explicit and implicit learning are both included in Soar 9. Finally, there is no bottom–up[10] or top–down learning, and the synergistic interaction of explicit/implicit processing is not accounted for. Hence, 'traditional' Soar satisfies very few of the learning-related desiderata. However, the newest extension/implementation (Soar 9; Laird, 2008) has increased psychological realism, which suggests a renewed interest in psychological modeling and the Soar cognitive architecture should be re-evaluated after Soar 9 has become the standard implementation (or a newer implementation is made available).

Clarion employs a variety of modularity, including that between implicit and explicit processes, as well as that between various memory components. In particular, Clarion has a dual-representational structure that consists of two "levels": the top level captures explicit processes and the bottom level implicit processes. Clarion is the only

[10] Note that the new clustering algorithm included in Soar 9 may be able to explain some form of bottom-up learning. However, further research is needed to address this possibility.

(reviewed) psychologically-oriented cognitive architecture to include a structural distinction between explicit and implicit cognition. Clarion can learn sequences of behaviors and automatize them using top-down assimilation. Learning in the bottom-level is autonomous gradual trial-and-error adaptation. Clarion directly addresses the interaction between implicit and explicit processes, and in particular bottom–up learning and top-down learning. Explicit symbolic conceptual representations are derived, literally, from implicit reactive routines (the bottom level of Clarion) through bottom–up learning within the context of ongoing activities in the everyday world. Explicit processes can generate and fine-tune explicit knowledge structures (within the top level), and implicit knowledge is learned using reinforcement learning (within the bottom level). Hence, Clarion addresses all the learning related desiderata listed earlier, and the incorporation of all these components results in synergistic learning (e.g., speeding up learning) and performance (e.g., improving performance, improving transfer, and so on). On the negative side, Clarion's focus on psychologically-realistic learning and module interactions slowed down the development of symbolic reasoning capabilities, which are necessary for some very complex tasks (e.g., theorem proving, natural language understanding). While ACT-R and Soar have started to address these very complex tasks, more work is needed to use Clarion to tackle these complex human endeavors and test implications of how the currently available structures allow for such functionality.

6 Example simulations

This section presents two simulation examples for each of the psychologically-oriented cognitive architectures surveyed. While we could not find

identical tasks that were simulated with all three architectures, we selected some representative simulation examples that share common structures (e.g., navigation task, sugar factory task) but illustrate the different autonomous learning mechanisms used by each cognitive architecture. The key learning mechanisms used for each simulation are highlighted in Table 2.

Insert Table 2 about here

6.1 ACT-R

6.1.1 The sugar production factory task

In the sugar production factory task (Berry & Broadbent, 1984), subjects were to interact with a simulated sugar factory to maintain a particular sugar production level through adjusting the size of the workforce. The behavior of the simulated system was determined by a linear equation that depend on the previous state. The results show that subjects quickly learned to improve their performance without much awareness of the mathematical relationship between the system input and output (Berry & Broadbent, 1984; Stanley et al., 1989).

In ACT-R, this rapid improvement without the accompanied awareness of the system functioning is explained by instance-based learning (Taatgen et al., 2006; Taatgen & Wallach, 2002). Briefly, this means that the ACT-R model stores every experienced input-output relationship acquired through interacting with the simulated sugar factory in declarative memory. Each such experience is called an 'instance' (Logan, 1988). When a new trial is initiated, all instances stored in declarative memory are recalled with different activation levels depending on their match (similarity) with the current trial. The instance

that is most activated by (most similar to) the current situation is retrieved and the previously stored output of the retrieved instance is used as the current model response.

The instance-based memory retrieval process can be accomplished in ACT-R using two production rules (Taatgen et al., 2006):

(1) Retrieval-request-rule

IF the goal is to determine the number of workers to achieve output *G* and the output of the previous day was *O*:

THEN send a request to declarative memory for an instance with previous output *O* and current output *G*.

(2) Retrieval-harvest-rule

IF the goal is to determine the number of workers;

AND an instance has been retrieved with *W* workers:

THEN set the number of workers to *W*.

The human and ACT-R simulation results are shown in Figure 4. As can be seen, the ACT-R simulation captured the human data in the sugar production factory task. Both the human and model improved their accuracy with practice. Also, because ACT-R learned to respond using stored instances (instead of a production rule representing the simulated sugar factory system equation), the ACT-R simulation has no direct or detailed explicit knowledge of the system functioning (Taatgen et al., 2006; Taatgen & Wallach, 2002). This corresponds with the lack of explicit task knowledge of human subjects (Berry & Broadbent, 1984).

Insert Figure 4 about here

6.1.2 Learning verb past tenses

The simulation of the sugar production factory task showed that ACT-R can carry out instance-based learning and reasoning (Taatgen et al., 2006). However, true generalization requires the abstraction of regularities to form new rules (e.g., rule induction). A well-known example in psychology is children learning of the past tense of English verbs (Taatgen & Anderson, 2002). This classical result shows that children's accuracy in producing the past tense of irregular verbs follows a U-shaped curve (Marcus, Pinker, Ullman, Hollander, Rosen, & Xu, 1992). Early in learning, children have a separate memory representation for the past tense of each verb (and no conjugation rule). Hence, the past tenses of irregular verbs are used mostly correctly (Phase 1). After moderate training, children notice that most verbs can be converted to their past tense by adding the suffix '-ed'. This leads to the formulation of a default rule (e.g., to find the past tense of a verb, add the suffix '-ed' to the verb stem). This rule is a useful heuristic and works for all regular verbs in English. However, children tend to over-generalize and (incorrectly) apply the rule to irregular verbs. This leads to errors and the low point of the U-shaped curve (Phase 2). Finally, children learn that there are exceptions to the default rule, and memorize the past tense of irregular verbs. Performance improves again (Phase 3).

In ACT-R, the early phase of training uses instance-based retrieval (as in the simulation of the sugar production factory task described above; for details, see Taatgen et al., 2006). Hence, the focus of the presentation is on the induction of the default rule (which is over-generalized in Phase 2 and correctly applied in Phase 3). This is

accomplished by joining two production rules. First, consider the following memory retrieval rule used in Phase 1 (Taatgen et al., 2006):

(3) Retrieve-past-tense

IF the goal is to find the past tense of a word *w*:

THEN issue a request to declarative memory for the past tense of *w*.

If a perfect match is retrieved from declarative memory, a second rule is used to produce the (probably correct) response. However, if Rule 3 fails to retrieve a perfect match, the verb past tense is unknown and an analogy rule (an example general cognitive strategy) is used instead (Taatgen et al., 2006):

(4) Analogy-find-pattern

IF the goal is to find the past tense of word *w*1;

AND the retrieval buffer contains past tense *w*2-suffix of *w*2:

THEN set the answer to *w*1-(*w*2-suffix).

This rule produces a form of generalization using an analogy. Because Rule 4 always follows Rule 3, they can be combined using production compilation (Taatgen et al., 2006). Also, *w*2 is likely to be a regular verb, so *w*2-suffix is likely to be '-ed'. Hence, combining Rules 3 and 4 yields (Taatgen et al., 2006):

(5) Learned-rule

IF the goal is to find the past tense of word *w*:

THEN set the answer to *w*-ed.

which is the default rule that can be used to accurately find the past tense of regular verbs.

The U-shaped curve representing the performance of children learning irregular verbs can thus be explained with ACT-R as follows (Taatgen & Anderson, 2002): In Phase 1, Rule 5 does not exist and Rule 3 is applied to correctly conjugate irregular verbs. In Phase 2, Rule 5 is learned and has proven useful (with regular verbs). Hence, it is often selected to incorrectly conjugate irregular verbs. In phase 3, the irregular verbs become more familiar as more instances have been encountered. This increases their base-level activation in declarative memory, which facilitates retrieval and increases the likelihood that Rule 3 is selected to correctly conjugate the irregular verbs (because Rule 3 is less costly than Rule 5). Figure 5 shows ACT-R simulation results.

Insert Figure 5 about here

6.1.3 Discussion

This subsection presented two example ACT-R simulations: the sugar production factory task and the learning of the past tense of English verbs. In the former, ACT-R adapted by using instances stored in declarative memory. This learning process corresponds with some data of human performance (Berry & Broadbent, 1984). In the latter simulation, ACT-R was able to overcome the reliance on memory retrieval and use a different general cognitive strategy (i.e., analogy) to extract a production rule useful in conjugating regular verbs (Taatgen & Anderson, 2002). A mixture of memory retrieval and default rule application produced the well-known U-shaped curve of children learning the English past tense of irregular verbs (Marcus et al., 1992). Overall, ACT-R was successful at producing two different patterns of autonomous learning observed in psychology tasks. However, note that the learning of English verb past tense was not

fully autonomous; a certain amount of prior knowledge was pre-inserted in the model (e.g., the analogy rule).

6.2 Soar

6.2.1 Room navigation

Konik and Laird (2006) have designed a Soar agent capable of learning how to navigate in an artificial environment from “Haunt 2 game” (a 3D first-person perspective game built using the Unreal game engine; see Magerko, Laird, Assanie, Kerfoot, & Stokes, 2004). This environment is structured and large, with a state space consisting of objects that are interrelated dynamically, have unobservable features, require real-time decisions, exist in continuous space, and include external agents and events. Learning was implemented by using inductive logic programming (Muggleton, 1992). One of the objectives of inductive logic programming is to learn an input space adequate (or inadequate) for representing a concept using positive and negative example situations. Here, the goal of the agent was to reach objects by learning how to navigate in the environment by observing (i.e., copying) an expert annotated behavior trace used for feedback.

In this simulation, it was assumed that the agent had access to a map composed of 13 rooms (Konik & Laird, 2006). The agent needed to learn a series of subgoals and operations to move in the state space. States, goals, and operations were coded using first-order logical rules and implicitly represented a decision tree. In this particular simulation, goals and actions (operators) were similar, except that a goal was more abstract than an action. For instance, “Pick-up item *i*” would be an action if item *i* was

directly accessible, but a goal if item *i* was in another room (because a subgoal "go to the other room" would have to be created).

The objective in the simulation is to find and pick-up an item of interest. However, the rules of navigation needed to be learned. More precisely, the Soar agent needed to select the operator "go-to-room(*r*)", where *r* represents the room where the item of interest is located. If room *r* is not directly accessible to the agent, a subgoal "go-to-next-room" is selected, which allows the agent to move to a different room. These two steps are iterated until the correct room (i.e., the room containing the item of interest) is reached.[11] The agent has to learn when to select and terminate the application of these operators. A typical simulation contained roughly 2,000 predicates, each having up to 1,000 literals. The expert used for providing feedback was a hand coded Soar agent that had previously experienced about 30,000 situations in the artificial environment.

Twelve different learning conditions were designed by varying the number of positive and negative training exemplars. Ten simulations were run in each learning condition and the distribution of the results is shown in Figure 6. As can be seen, the number of successful simulations grew with the number of negative exemplars, and a similar trend was present for the number of positive exemplars (although not as systematic). Overall, the navigation problem was consistently learned when the number of examples (both positive and negative) was sufficiently large (Konik & Laird, 2006).

Insert Figure 6 about here

[11] Note that other (intermediate) operators were also present but they were not learned. Hence, they will not be discussed here (for details, see Konik & Laird, 2006).

6.2.2 Learning problem solving

Nason and Laird (2005) proposed a variation of Soar that includes a reinforcement learning algorithm (following the precedents of Clarion and ACT-R) to learn numerical preference values for the operators (instead of relying only on symbolic preferences). In this implementation (called Soar-RL), the preferences are replaced by *Q*-values (Watkins, 1989) that are learned using environmental feedback. After the *Q*-value of each relevant operator has been calculated (i.e., all the operators available in working memory), an operator is stochastically selected.

Soar-RL has been used to simulate the Missionaries & Cannibals problem. The goal in this problem is to transport three missionaries and three cannibals across a river using a boat that can carry at most two persons at a time. Several trips are required, but the cannibals must never outnumber the missionaries on either riverbank. This problem has been used as a benchmark in problem solving research because, if the desirability of a move is evaluated in term of the number of peoples that have crossed the river (which is a common assumption), a step backward must be taken midway in solving the problem (i.e., a move that reduces the number of peoples that crossed the river must be selected).

In the Soar-RL simulation, the states were defined by the number of missionaries and cannibals on each side of the river and the location of the boat. The operators were boat trips transporting people, and the *Q*-values of the operators were randomly initialized. Also, to emphasize the role of reinforcement learning in solving this problem, chunking had been disengaged. Hence, the only form of adaptation was the adjustment of the operator *Q*-values. Success states (i.e., all peoples crossed the river) were rewarded,

failure states (i.e., cannibals outnumbering missionaries on a riverbank) were punished, and all other states received neutral reinforcement. The problem was simulated 500 times.

The simulation results are shown in Figure 7. As can be seen, Soar-RL generally learned to solve the Missionaries & Cannibals problem. Most errors (gray squares) resulted from the stochastic decision process (Nason & Laird, 2005). Nason and Laird also showed that the model performance can be improved five-fold by adding a symbolic preference preventing an operator at time *t* from undoing the result of the application of the operator at time *t*-1.

Insert Figure 7 about here

6.2.3 Discussion

This subsection presented example simulations using the Soar cognitive architecture. In the first simulation, Soar was shown to be able to learn how to navigate in an artificial environment by using an expert tutor along with a set of productions (a priori knowledge) using inductive logic programming (Konik & Laird, 2006). This simulation is interesting because it shows that a knowledge-rich Soar agent can be used to train other Soar agents, which can substantially reduce development/programming time. In the second simulation, it was shown how the addition of a reinforcement learning algorithm in Soar could be used to learn numerical preferences for operator selection (Nason & Laird, 2005).

While Soar has simulated tasks that are commonly done by subjects in psychology experiments, no attempt was made at comparing in detail the simulation results with human data. This reflects the emphasis of Soar on reproducing intelligent behavior (i.e., artificial intelligence); not the detailed modeling of psychological

processes. Learning in Soar is not fully autonomous because of the amount of a priori knowledge needed to get learning started. However, some of the a priori knowledge in the Soar simulations can be argued to represent innate/developmental knowledge.

6.3 Clarion

6.3.1 The sugar production factory task

Sun, Slusarz, and Terry (2005) simulated an extension of the sugar production factory task (Stanley et al., 1989). The aim of the Clarion simulations was to capture the effects of verbalization, explicit (how-to) instructions, and more autonomous learning in this task.

Stanley and his colleagues (1989) tested four groups of subjects. The control group was not given any explicit (how-to) instruction and was not asked to verbalize (thus reproducing Berry & Broadbent, 1984). The other three groups were subjected to different experimental manipulations. The verbalization group was required to provide a verbal account of the response strategy used after each block of trials. Other groups of subjects were given explicit instructions in various forms. For example, subjects in the "memory training" group studied a series of correct input–output pairs, and subjects in the "simple rules" group were given a simple heuristic rule ("always select the response level halfway between the current production level and the target level"). The results are shown in Table 3. Statistical analysis showed that all three experimental groups produced higher scores than the control group.

Insert Table 3 about here

To simulate this task with Clarion, the input in the bottom level consisted of a moving time window. At the top level, Clarion used a generate-and-test learning algorithm to learn the rules. Rules for this task were mostly numerical relations. In hypothesizing rules, Clarion progresses from the simplest rule form to the most complex. Top-level rules are refined using bottom-up information (Sun et al., 2001). When a rule cannot be further specialized, it is deleted. Whenever all the rules of a certain form were deleted, a new set of rules of a different form (more complex) were hypothesized, and the cycle repeated itself. The different experimental conditions were modeled using a parameter/threshold on rule deletion (specialization) and/or pre-coding the instruction in the top level. As shown in Table 3, the simulation captured the human data well. The mean square error between the human and the model data was only 0.19. In addition, the simulation captured the verbalization and instruction effects in the human data. Detailed statistical analysis on the simulation results confirmed that Clarion reproduced the same qualitative and quantitative results as the human subjects.

6.3.2 Minefield navigation

Sun, Merrill, and Peterson (2001) empirically tested and simulated a complex minefield navigation task. In the empirical task, the subjects were seated in front of a computer monitor that displayed an instrument panel containing several gauges that provided current information on the status/location of a vehicle. The subjects used a joystick to control the direction and speed of the vehicle. In each trial, a random mine layout was generated and the subjects had limited time to reach a target location without hitting a mine. Control subjects were trained for several consecutive days in this task. Sun

and colleagues also tested three experimental conditions with the same amount of training but emphasizing verbalization, over-verbalization, and dual-tasking (respectively).

The human results are shown in Figure 8 (left panels). First, as shown in Figure 8a, learning was slower in the dual-task condition than in the single-task condition. Second, Figure 8b shows that verbalization speeds up learning. However, the effect of verbalization is reversed in the over-verbalization condition; over-verbalization interfered with (slowed down) learning (not shown in the Figure).

Insert Figure 8 about here

In the Clarion simulation, simplified (explicit) rules were represented in the form "State --> Action" in the top level. In the bottom level, a backpropagation network was used to (implicitly) learn the input-output function using reinforcement learning. Reinforcement was received at the end of every trial. The bottom-level information was used to create and refine top-level rules (with bottom-up learning). The model started out with no specific a priori knowledge about the task (the same as a typical subject) so that autonomous learning could be captured. The bottom level contained randomly initialized weights. The top level started empty and contained no a priori knowledge about the task (either in the form of instructions or rules). The interaction of the two levels was not determined a priori either. There was no fixed weight in combining outcomes from the two levels; the weights were automatically set based on relative performance of the two levels on a periodic basis. The effects of the dual task and the various verbalization conditions were modeled using rule-learning thresholds so that more/less activities could occur at the top level.

As can be seen in Figure 8 (right panels), all the human results were accurately reproduced by the Clarion simulations. The negative effect of over-verbalization (not shown) was also captured by the Clarion simulation. In addition, the human and simulated data were input into a common ANOVA and no statistically significant difference between human and simulated data was found in any of the conditions. Hence, Clarion did a good job of simulating detailed human data in the minefield navigation task.

6.3.3 Discussion

This subsection presented simulation results with the Clarion cognitive architecture. Because Clarion was developed as a psychological model, emphasis was placed on the careful modeling of human experiments and detailed matching of human data. The first simulation presented the results of a simulation of the sugar production factory task (Stanley et al., 1989). This simulation illustrates how Clarion can differently adapt to its environment by using the information/instruction that is provided. Here, various levels of information were provided, and Clarion was able to exploit the available instructions to improve its performance (in correspondence with the human data). The second simulation presented a complex minefield navigation task in which no initial a priori knowledge was available (Sun et al., 2001). The Clarion simulation was able to learn autonomously to achieve the task and build general rules that could potentially be used in similar situations (through its bottom-up learning process). The simulation data matched well the human data, and detailed comparison was performed, which was facilitated by the fact that both the human and simulation data were collected in the same task environment.

7 Concluding remarks

This survey started with some learning-related desiderata in the development of psychologically-oriented cognitive architectures and reviewed three of the most popular architectures used in psychology. The emphasis was on autonomous learning, which is an important part of development for both natural and artificial agents. It was shown that ACT-R and Soar can use existing symbolic rule-based knowledge to create complex new symbolic rule-based knowledge, and non-symbolic statistical regularities to adjust the relevance of the symbolic rule structures. Specifically, ACT-R was able to use production rules implementing learning strategies to acquire the past-tense of English verb in a psychologically-realistic way and to implicitly learn the sugar factory task using a process consistent with psychology theory (Logan, 1988). Likewise, Soar was able to learn to navigate a complex environment using feedback from another Soar agent and to solve a typical problem solving task used in psychology. However, both ACT-R and Soar required the modeler to pre-code some knowledge using rules to kick start the autonomous learning capabilities of the architectures.

The third architecture, Clarion, also includes learning processes similar to ACT-R and Soar, but can also use statistical regularities to create brand new symbolic rule-based knowledge, and tune existing rule-based symbolic structures. Clarion can learn without much prior knowledge, such as initial productions necessary in ACT-R or Soar. This was shown by allowing Clarion to build its own rule in the sugar factory task and the navigation task, without the pre-insertion of production rules to guide learning. Hence, Clarion appears to be more autonomous (due in large part to its connectionist philosophy). In addition, different learning options in Clarion are managed by the

motivational and the meta-cognitive subsystem (structures that are not included in ACT-R and Soar), so the model can adapt autonomously and select the appropriate strategy (for a discussion and example use of the MS and MCS, see Sun, 2002; Wilson et al., 2009). However, Clarion has not yet been applied to some of the most complex tasks where ACT-R and Soar have been successful (e.g., theorem proving, tutoring systems, combat simulation, etc.), so more research is required to test whether efficient strategies would automatically emerge from Clarion, and how they would match human results.

To summarize, a series of simulation examples showed that ACT-R, Soar, and Clarion have all had some success at addressing learning to accomplish simple and complex tasks. However, autonomous learning in cognitive architectures is clearly a work in progress. None of the aforementioned models has achieved human-level performance in relation to learning (Shi, 2011). To achieve human-level intelligence, one would need to design a model that can account for a variety of learning capabilities spanning from low-level perceptual learning to high-level complex reasoning using a few unifying frameworks (Ormrod, 2012). Hence, future work should be devoted to further development of autonomous learning in cognitive models by attempting to use the smallest possible number of common unifying frameworks.

8 Acknowledgments

This research was supported by the ONR grant N00014-08-1-0068 to the second author. Requests for reprints should be addressed to Sébastien Hélie, Department of Psychological Sciences, Purdue University, West Lafayette, IN 47907-2081, e-mail: shelie@purdue.edu; or to Ron Sun, Department of Cognitive Science, Rensselaer Polytechnic Institute, email: rsun@rpi.edu.

9 References

Anderson, J.R. (1983). *The Architecture of Cognition*. Cambridge, MA: Harvard University Press.

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

Anderson, J. R. (1993). *Rules of the Mind*. Hillsdale, Erlbaum.

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

Anderson, J. R., & Lebiere, C. (1998). *The Atomic Components of Thought*. Mahwah, NJ: Lawrence Erlbaum Associates.

Anderson, J.R., & Lebiere, C. (2003). The Newell test for a theory of cognition. *Behavioral and Brain Sciences*, *26*, 587-640.

Ashby, F. G., & O'Brien, J. B. (2005). Category learning and multiple memory systems, *Trends in Cognitive Sciences*, *2*, 83-89.

Beilock, S. L., Carr, T. H., MacMahon, C., and Starkes, J. L. (2002). When paying attention becomes counterproductive: impact of divided versus skill-focused attention on novice and experienced performance of sensorimotor skills. Journal of Experimental Psychology: Applied, 8, 6–16.

Berry, D. C., & Broadbent, D. E. (1984). On the relationship between task performance and associated verbalizable knowledge. *The Quarterly Journal of Experimental Psychology*, *36A*, 209-231.

Bickard, M. (1993). Representational content in humans and machines. *Journal of Experimental and Theoretical Intelligence*, *5*, 285-333.

Bornstein, R. F., & Pittman, T. S. (1992). *Perception Without Awareness*. New York: Guilford.

Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*, 123-140.

Cangelosi, A., Greco, A., & Harnad, S. (2002). Symbol grounding and the symbolic theft hypothesis. In A. Cangelosi & D. Parisi (Eds.) *Simulating the Evolution of Language* (pp. 191-210). London: Springer.

Carpenter, G. A., & Grossberg, S. (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics, and Image Processing*, *37*, 54-115.

Cho, B., Rosenbloom, P. S., & Dolan, C. P. (1991) Neuro-Soar: A neural-network architecture for goal-oriented behavior. In K. J. Hammond & D. Gentner (Eds.) *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society* (pp. 673-677). Hillsdale, NJ: Lawrence Erlbaum Associates.

Chong, H. –Q., Tan, A. –H., & Ng, G. –W. (2007). Integrated cognitive architectures: A survey. *Artificial Intelligence Review*, *28*, 103-130.

Craik, F. I. M., & Tulving, E. (1975). Depth of processing and the retention of words in episodic memory. *Journal of Experimental Psychology: General*, *104*, 268-294.

Dreyfus, H., & Dreyfus, S. (1987). *Mind Over Machine: The Power of Human Intuition*. New York: The Free Press.

Eichenbaum, H. (1997). Declarative memory: Insights from cognitive neurobiology. *Annual Review of Psychology*, *48*, 547-572.

Evans, J. B. T. (2006). The heuristic-analytic theory of reasoning: Extension and evaluation. *Psychonomic Bulletin & Review*, *13*, 378-395.

Fodor, J. (1983). *The Modularity of the Minds*. Cambridge: MIT Press.

Franklin, S., & F. G. Patterson, Jr. (2006). The Lida Architecture: Adding New Modes of Learning to an Intelligent, Autonomous, Software Agent. *Integrated Design and Process Technology, IDPT-2006*. San Diego, CA: Society for Design and Process Science.

Helie, S., & Ashby, F. G. (2012). Learning and transfer of category knowledge in an indirect categorization task. *Psychological Research*, *76*, 292-303

Helie, S., & Cousineau, D. (2011). The cognitive neuroscience of automaticity: Behavioral and brain signatures. *Cognitive Sciences, 6*, 25-43.

Helie, S., Proulx, R., & Lefebvre, B. (2011). Bottom-up learning of explicit knowledge using a Bayesian algorithm and a new Hebbian learning rule. *Neural Networks*, *24*, 219-232.

Hélie, S., & Sun, R. (2010). Incubation, insight, and creative problem solving: A unified theory and a connectionist model. *Psychological Review*, *117*, 994-1024.

Helie, S., Waldschmidt, J. G., & Ashby, F. G. (2010). Automaticity in rule-based and information-integration categorization. *Attention, Perception, & Psychophysics, 72*, 1013-1031.

Hirschfield, L., & Gelman, S. (Eds.). (1994). *Mapping the Mind: Domain Specificity in Cognition and Culture*. Cambridge: Cambridge University Press.

Hutchins, E. (1995). How a cockpit remembers its speeds. *Cognitive Science*, *19*, 265-288.

Jilk, D. J., Lebiere, C., O'Reilly, R. C., & Anderson, J. R. (2008). SAL: An explicitly pluralistic cognitive architecture. *Journal of Experimental & Theoretical Artificial Intelligence*, *20*, 197-218.

Karmiloff-Smith, A. (1986). From meta-processes to conscious access: Evidence from children's metalinguistic and repair data. *Cognition, 23*, 95-147.

Kelley, T. D., & Avery, E. (2010). A Cognitive Robotics System: The Symbolic and Subsymbolic Robotics Intelligence Control System (SS-RICS). *Proceedings of SPIE*.

Kieras, D. E., & Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, *12*, 391-438.

Konik, T., & Laird, J. E. (2006). Learning goal hierarchies from structured observations and expert annotations. *Machine Learning*, *64*, 263-287.

Laird, J. E., (2008). Extending the Soar cognitive architecture. *Proceedings of the First Conference on Artificial Intelligence*. Memphis, TN.

Laird, J. E., Newell, A., Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence*, *33*, 1-64

Langley, P., Laird, J. E., & Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, *10*, 141-160.

Lebiere, C., & Anderson, J. R. (1993). A connectionist implementation of the ACT-R production system. In W. Kintsch (Ed.) *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society* (pp. 635-640). Hillsdale, NJ: Lawrence Erlbaum Associates.

Lehman, J. F., Laird, J., & Rosenbloom, P. (2006). *A Gentle Introduction to Soar, an Architecture for Human Cognition*. University of Michigan.

Logan, G. D. (1988). Toward an instance theory of automatization. *Psychological Review*, *95*, 492-527.

Maddox, T. W., Ashby, F. G., & Bohil, C. J. (2003). Delayed feedback effects on rule-based and information-integration category learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *29*, 650-662.

Magerko, B., Laird, J. E., Assanie, M., Kerfoot, A., & Stokes, D. (2004). AI characters and diectors for interactive computer games. *Proceedings of the Sixteenth Innovative Applications of Artificial Intelligence Conference*. Menlo Park, CA: AAAI.

Marcus, G. F., Pinker, S., Ullman, M., Hollander, M., Rosen, T. J., & Xu, F. (1992). Overregularization in language acquisition. *Monographs of the Society for Research in Child Development*, *57*, 1–182.

Mathews, R.C., Buss, R.R., Stanley, W.B., Blanchard-Fields, F., Cho, J.R., & Druhan, B. (1989). Role of implicit and explicit processes in learning from examples: A synergistic effect. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *15*, 1083-1100.

Merleau-Ponty, M. (1963). *The Structure of Behavior*. Boston, MA: Beacon Press.

Meyer, D. & Kieras, D. (1997). A computational theory of executive cognitive processes and human multiple-task performance: Part 1, basic mechanisms. *Psychological Review*, *104*, 3-65.

Michalski, R. (1983). A theory and methodology of inductive learning. *Artificial Intelligence*, *20*, 111-161.

Minton, S. N. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, *42*, 363-391.

Muggleton, S. (1992). *Inductive Logic Programming*. London: Academic Press.

Nason, S., & Laird, J. E. (2005). Soar-RL: Integrating reinforcement learning with Soar. *Cognitive Systems Research*, *6*, 51-59.

Nelson, T. (Ed.). (1993). *Metacognition: Core Readings*. Boston, MA: Allyn and Bacon.

Newell, A. (1990). *Unified Theories of Cognition*. Cambridge: Harvard University Press.

O'Reilly, R.C. (1996). Biologically Plausible Error-Driven Learning Using Local Activation Differences: The Generalized Recirculation Algorithm. *Neural Computation*, *8*, 895–938.

Ormrod, J. E. (2012). *Human Learning. Sixth Edition*. Pearson.

Reber, A.S. (1989). Implicit learning and tacit knowledge. *Journal of Experimental Psychology: General*, *118*, 219-235.

Rescorla, R., & Wagner, A. (1972). A theory of Pavlovian conditioning. In A. Black & W. Prokasy (Eds.) *Classical Conditioning II: Current Research and Theory* (pp. 64-99). New York: Apple – Century – Crofts.

Rumelhart, D., McClelland, J., & The PDP Research Group. (Eds.). (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations.* Cambridge, MA: MIT Press.

Russell, S., & Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Upper Saddle River, NJ: Prentice Hall.

Schacter, D. L., Chiu, C. -Y. P., & Ochsner, K. N. (1993). Implicit memory: A selective review. *Annual Review of Neuroscience*, *16*, 159-182.

Schacter, D. L., Wagner, A. D., & Buckner, R. L. (2000). Memory systems of 1999. In E. Tulving & F.I.M. Craik, (Eds.), *The Oxford Handbook of Memory* (pp. 627-643), New York: Oxford University Press.

Scherer, K. R. (2001). Appraisal considered as a process of multi-level sequential checking. In K. R. Scherer, A. Schor, & T. Johnstone (Eds.) *Appraisal Processes in Emotion: Theory, Methods, Research* (pp. 92-120). New York: Oxford University Press.

Seger, C. (1994). Implicit learning. *Psychological Bulletin*, *115*, 163–196.

Shi, Z. (2011). *Advanced Artificial Intelligence*. Singapore: World Scientific Publishing.

Smith, J. D., Shields, W. E., & Washburn, D. A. (2003). The comparative psychology of uncertainty monitoring and metacognition. *Behavioral and Brain Sciences*, *26*, 317-373.

Smolensky, P. (1988). On the proper treatment of connectionism. *Behavioral and Brain Sciences*, *11*, 1-74.

Stanley, W., Mathews, R., Buss, R., & Kotler-Cope, S. (1989). Insight without awareness: On the interaction of verbalization, instruction and practice in a simulated process control task. *Quarterly Journal of Experimental Psychology: Human Experimental Psychology*, *41A*, 553–577.

Sun, R. (1994). *Integrating Rules and Connectionism for Robust Commonsense Reasoning*. New York: John Wiley and Sons.

Sun, R. (2000). Symbol grounding: A new look at an old issue. *Philosophical Psychology*, *13*, 403-418.

Sun, R. (2002). *Duality of the Mind: A Bottom-up Approach Toward Cognition*. Mahwah, NJ: Lawrence Erlbaum Associates.

Sun, R. (2004). Desiderata for cognitive architectures. *Philosophical Psychology*, *17*, 341-373.

Sun, R. (2009). Motivational representations within a computational cognitive architecture. *Cognitive Computation*, *1*, 91-103.

Sun, R., & Helie, S. (2013). Psychologically realistic cognitive agents: Taking human cognition seriously. *Journal of Experimental & Theoretical Artificial Intelligence*, *25*, 65-92.

Sun, R., Merrill, E., & Peterson, T. (2001). From implicit skills to explicit knowledge: A bottom-up model of skill learning. *Cognitive Science, 25*, 203-244.

Sun, R., & Peterson, T. (1999). Multi-agent reinforcement learning: Weighting and partitioning. *Neural Networks*, *12*, 127-153.

Sun, R., Slusarz, P., & Terry, C. (2005). The interaction of the explicit and the implicit in skill learning: A dual-process approach. *Psychological Review, 112*, 159-192.

Sutton, R., & Barto, A. (1981). Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review*, *88*, 135-170.

Taatgen, N. A., & Anderson, J. R. (2002). Why do children learn to say “broke”? A model of learning the past tense without feedback. *Cognition*, *86*, 123-155.

Taatgen, N. A., & Anderson, J. R. (2008). Constraints in cognitive architectures. In R. Sun (Ed.) *The Cambridge Handbook of Computational Psychology* (pp. 170-185). New York: Cambridge University Press.

Taatgen, N. A., Lebiere, C., & Anderson, J. R. (2006). Modeling paradigms in ACT-R. In R. Sun (Ed.) *Cognition and Multi-Agen Interaction: From Cognitive Modeling to Social Simulation* (pp. 29-52). New York: Cambridge University Press.

Taatgen, N. A., & Wallach, D. (2002). Whether skill acquisition is rule or instance based is determined by the structure of the task. *Cognitive Science Quarterly*, *2*, 163-204.

Thorndike, E. (1911). *The Study of Instinct*. London: Oxford University Press.

Toates, F. (1986). *Motivational Systems*. Cambridge: Cambridge University Press.

Ungerleider, L. G., & Haxby, J. V. (1994). 'What' and 'where' in the human brain. *Current Opinion in Neurobiology*, *4*, 157-165.

Vernon, D., Metta, G., & Sandini, G. (2007). A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *IEEE Transactions on Evolutionary Computation*, *11*, 151-180.

Wasserman, E., Elek, S., Charlosh, D., & Baker, A. (1993). Rating causal relations. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, *19*, 174-188.

Watkins, C. (1989). *Learning From Delayed Rewards*. Doctoral Dissertation, Cambridge University, Cambridge, UK.

Willingham, D. B. (1998). A neuropsychological theory of motor skill learning. *Psychological Review*, *105*, 558-584.

Willingham, D., Nissen, M., & Bullemer, P. (1989). On the development of procedural knowledge. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *15*, 1047-1060.

Wilson, N., Sun, R., & Mathews, R. (2009). A motivationally-based simulation of performance degradation under pressure. *Neural Networks*, *22*, 502-508.

Wray, R. E., & Jones, R. M. (2006). Considering Soar as an agent architecture. In R. Sun (Ed.) *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation* (pp. 53-78). New York: Cambridge University Press.

Table 1. Learning-related desiderata and components for each psychologically-oriented cognitive architecture

	ACT-R	Soar	Clarion
Dichotomy of implicit / explicit processing	Utility/ Structure	Preference/ Structure	Modularity of levels
Explicit learning / memory	Production compilation	Chunking	'One-shot' symbolic
Implicit learning / memory	Utility learning	Preference learning (Soar 9)	Reinforcement learning
Synergistic interaction	-	-	Knowledge integration
Bottom-up / top-down learning	Only top-down (production compilation)	-	Bottom-up rule extraction / top-down assimilation

Table 2. Autonomous learning mechanisms used in the simulations

Simulations	Mechanisms
ACT-R	
Sugar production	Instance-based memory retrieval implemented as production rules.
Verb past-tense	Analogical processing learned using production compilation.
Soar	
Navigation	Inductive logic programming used to learn by observing an expert annotated behavior trace.
Problem solving	Reinforcement learning of *Q*-values used as operator preferences.
Clarion	
Sugar production	Generate-and-test explicit rule learning aided by rule refinement with bottom-up learning.
Navigation	Implicit reinforcement learning allowing for bottom-up learning of explicit rules.

Table 3. Human and model data for the Clarion simulation of the sugar production factory task

Human data (Stanley et al., 1989)

Group	Sugar task	Person task
Control	1.97	2.85
Verbalization	2.57	3.75
Memory training	4.63	5.33
Simple rule	4.00	5.91

Model data (Sun et al., 2005)

Group	Sugar task	Person task
Control	2.276	2.610
Verbalization	2.952	4.187
Memory training	4.089	5.425
Simple rule	4.073	5.073

Note. Two different cover stories were used in Stanley et al. (1989) with the same underlying equation. Pretraining was used in simulating the person task because of the subjects' experience interacting with persons (Sun et al., 2005).

10 Figure captions

Figure 1. General architecture of ACT-R. DLPFC = Dorsolateral prefrontal cortex; VLPFC = Ventrolateral prefrontal cortex. (This Figure is from Anderson et al., 2004).

Figure 2. The general architecture of Soar. The main components of 'traditional' Soar are the Long-Term Memory and the Working Memory. Additional details are from Soar 9.

Figure 3. The Clarion architecture. ACS stands for the action-centered subsystem, NACS the non-action-centered subsystem, MS the motivational subsystem, and MCS the meta-cognitive subsystem.

Figure 4. Human and ACT-R simulation results in the sugar production factory task.

Figure 5. Proportion of correct past tense verbs produced by ACT-R. The dotted-line shows the performance with regular verbs while the full line shows performance with irregular verbs. (This Figure is from Taatgen & Anderson, 2002).

Figure 6. Distribution of the simulation results for each number of positive and negative training exemplars. (This Figure is from Konik & Laird, 2006).

Figure 7. Success and failure of Soar-RL in the Missionaries & Cannibals problem. The *x*-axis represents training trials while the *y*-axis represents the number of moves. (This Figure is from Nason & Laird, 2005).

Figure 8. (a) Single vs. dual task training. (b) Verbalization vs. no verbalization. The left panel in each row shows human data while the right panel in each row shows Clarion simulation data. (This Figure is from Sun et al., 2001).

Figure 1

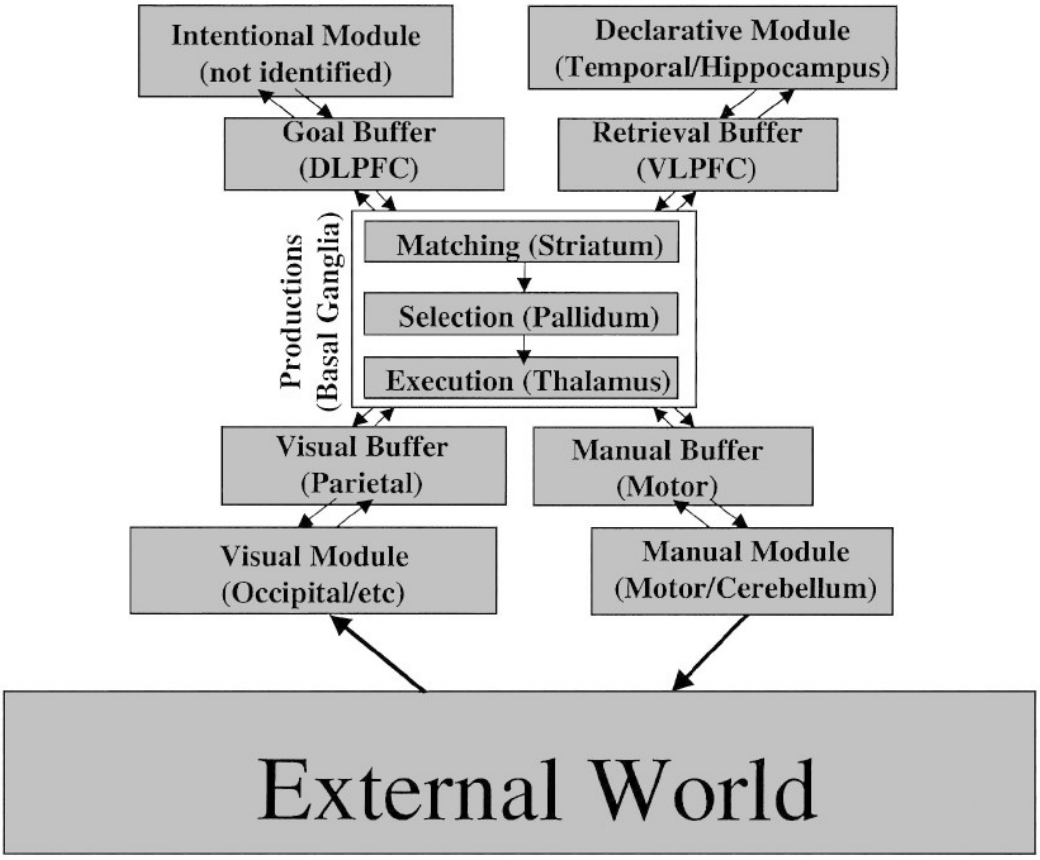

Intentional Module
(not identified)
Declarative Module
(Temporal/Hippocampus)
Goal Buffer
(DLPFC)
Retrieval Buffer
(VLPFC)
Productions
(Basal Ganglia)
Matching (Striatum)
Selection (Pallidum)
Execution (Thalamus)
Visual Buffer
(Parietal)
Manual Buffer
(Motor)
Visual Module
(Occipital/etc)
Manual Module
(Motor/Cerebellum)
External World

Figure 2

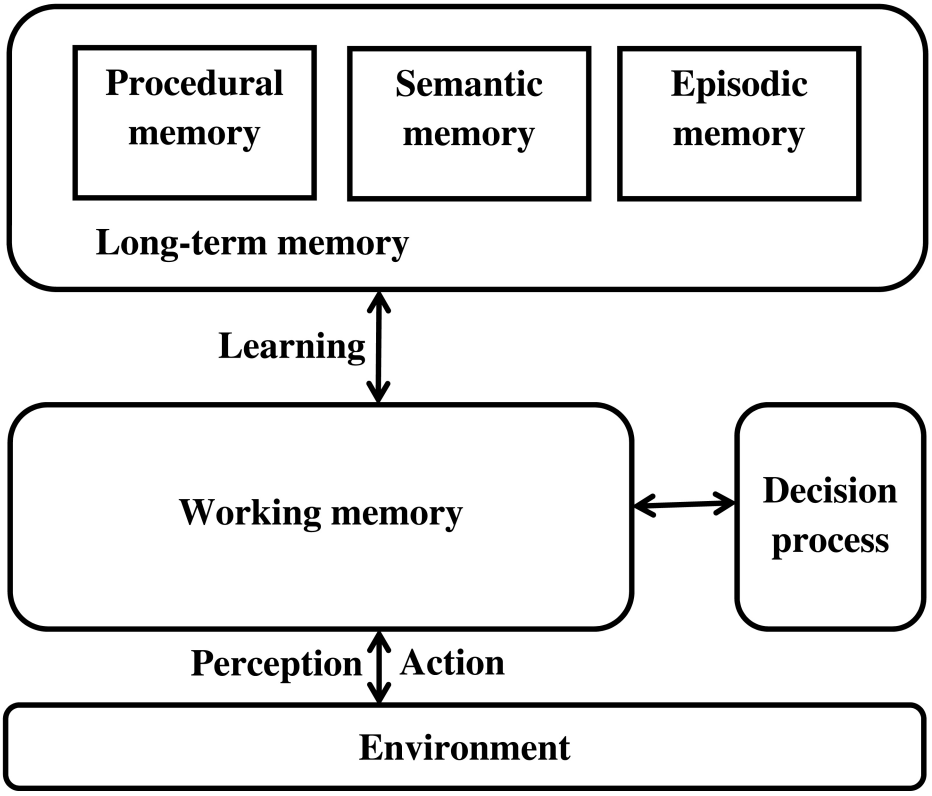

Procedural memory
Semantic memory
Episodic memory
Long-term memory
Learning
Working memory
Decision process
Perception
Action
Environment

Figure 3

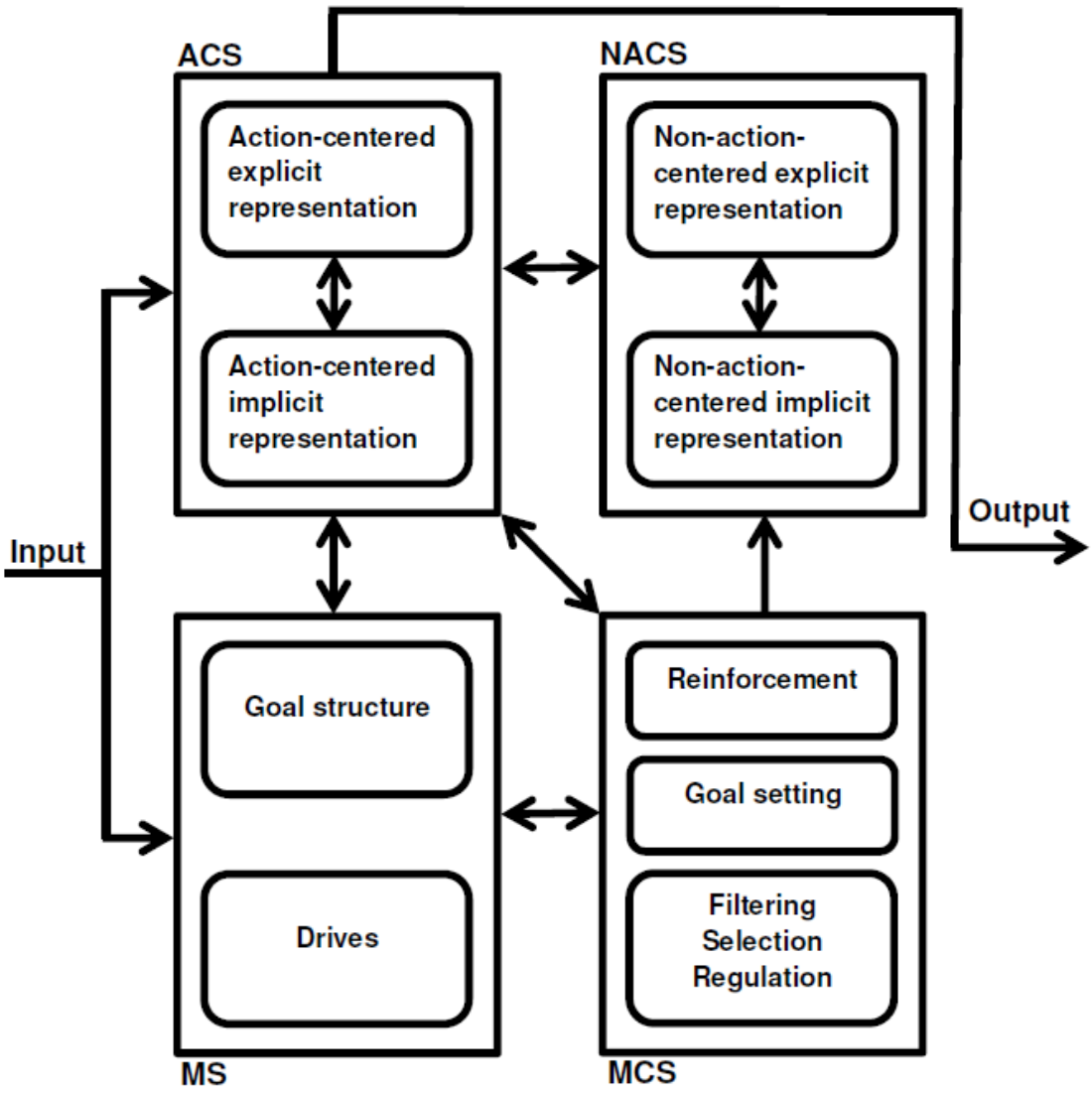

ACS
NACS
Action-centered explicit representation
Action-centered implicit representation
Non-action-centered explicit representation
Non-action-centered implicit representation
Input
Output
Goal structure
Drives
Reinforcement
Goal setting
Filtering
Selection
Regulation
MS
MCS

Figure 4

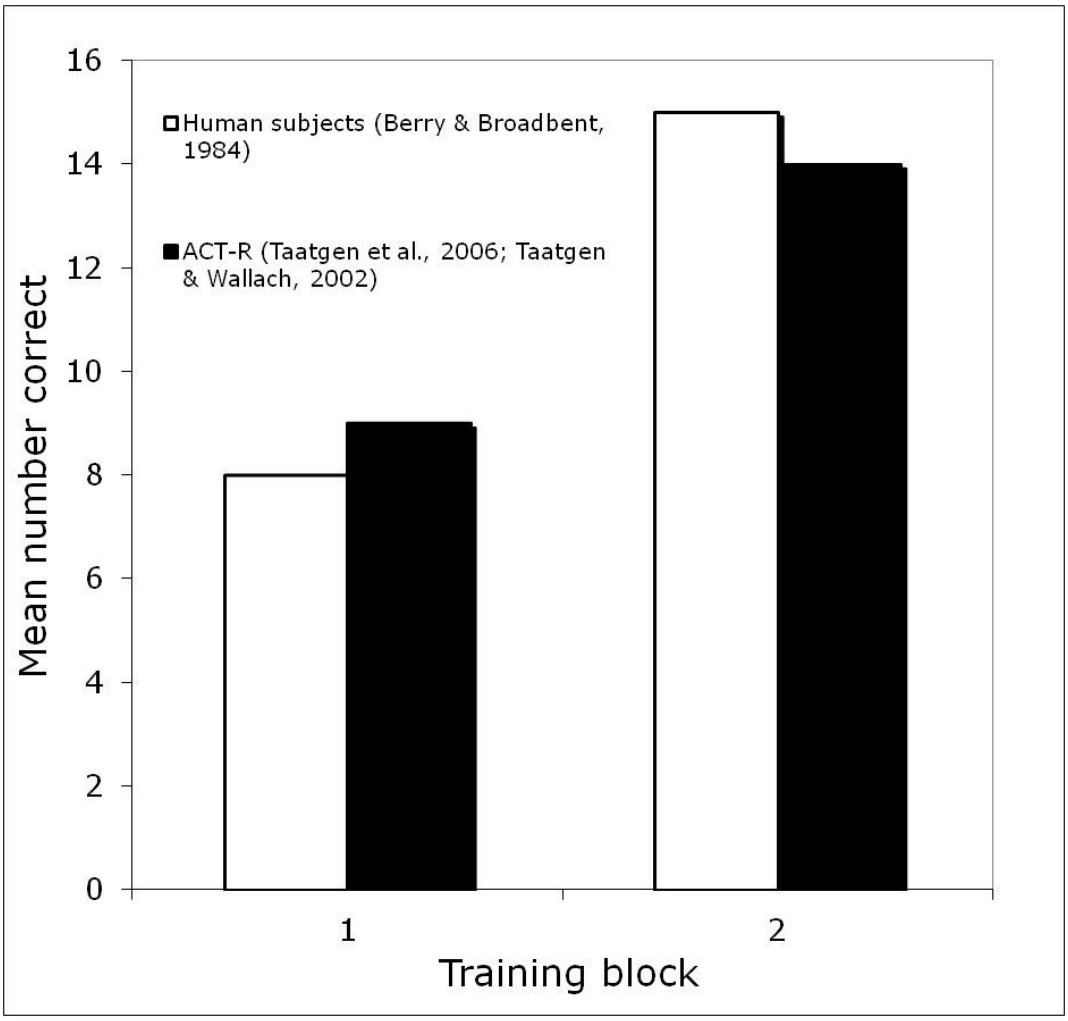

16
14
12
10
8
6
4
2
0
Mean number correct
Human subjects (Berry & Broadbent, 1984)
ACT-R (Taatgen et al., 2006; Taatgen & Wallach, 2002)
1
2
Training block

Figure 5

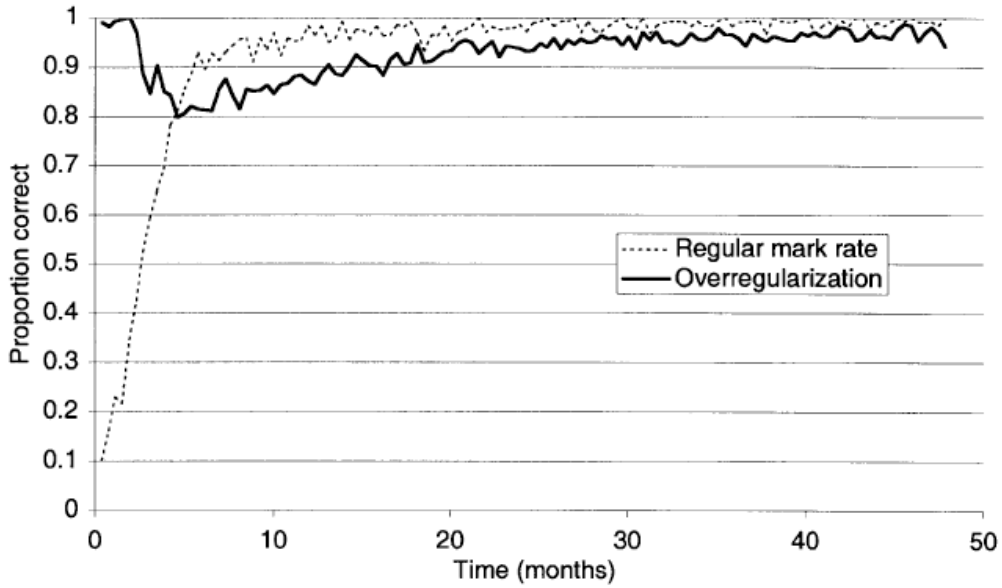

Proportion correct
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
Time (months)
Regular mark rate
Overregularization

Figure 6

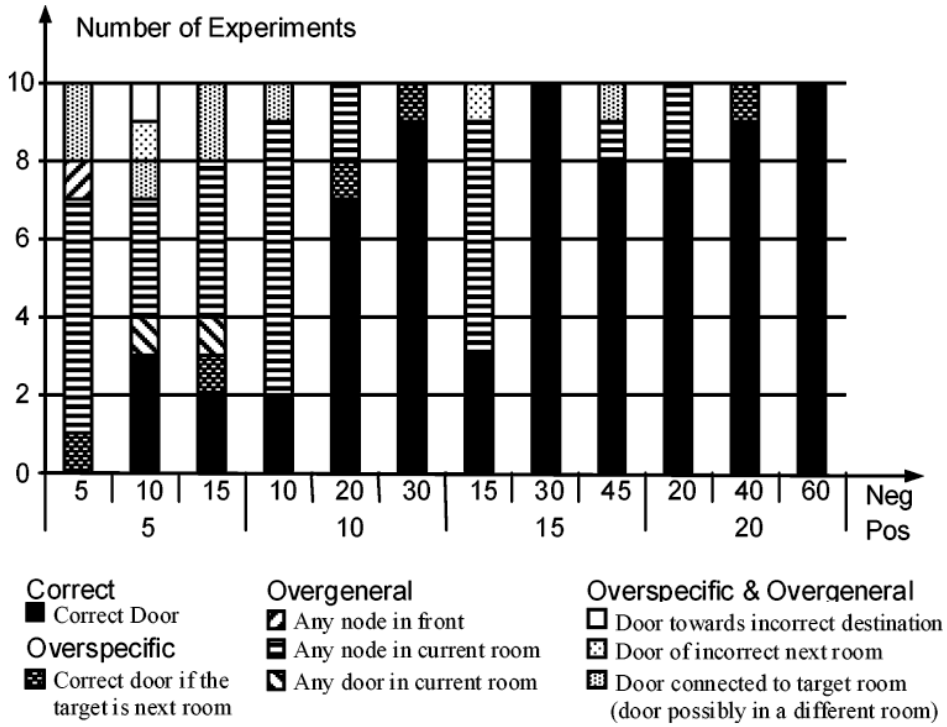

Number of Experiments
10
8
6
4
2
0
5
10
15
10
20
30
15
30
45
20
40
60
Neg
5
10
15
20
Pos
Correct
Correct Door
Overspecific
Correct door if the target is next room
Overgeneral
Any node in front
Any node in current room
Any door in current room
Overspecific & Overgeneral
Door towards incorrect destination
Door of incorrect next room
Door connected to target room (door possibly in a different room)

Figure 7

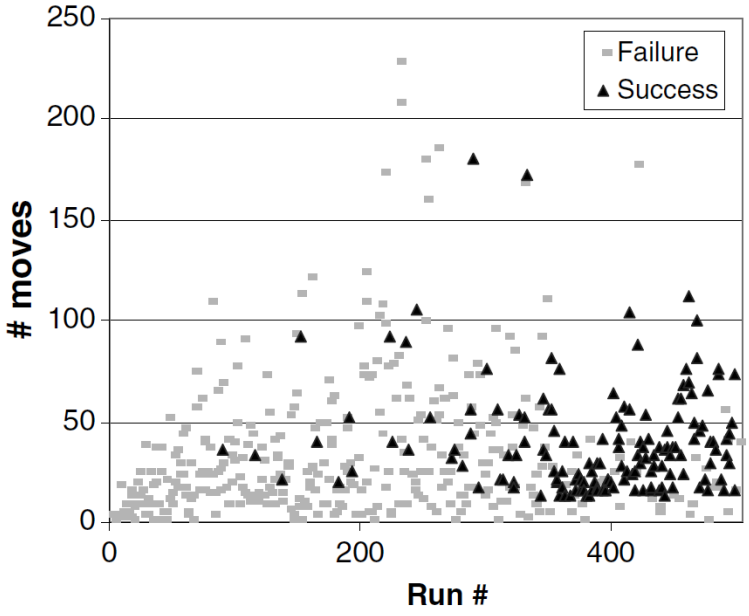
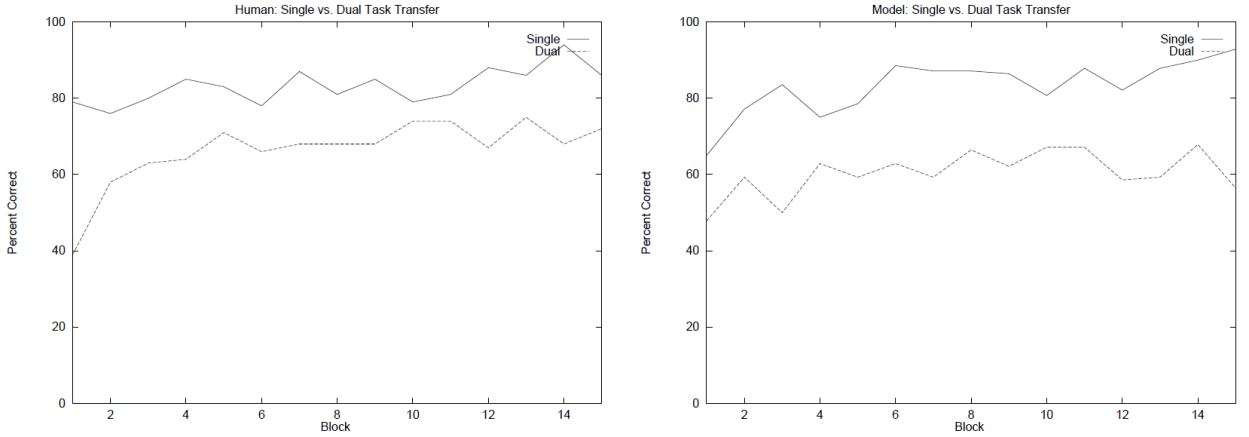
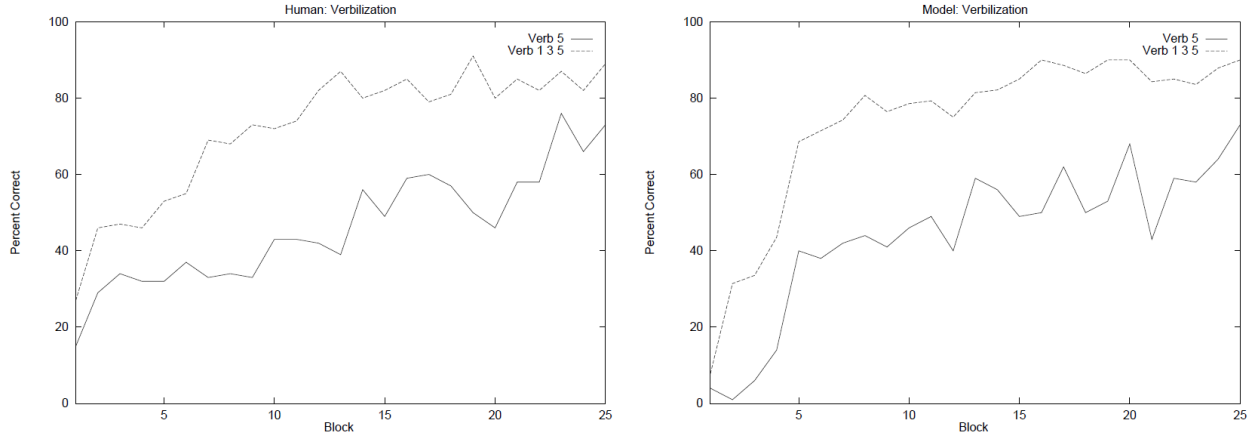


Figure 8


Human: Single vs. Dual Task Transfer
Single
Dual
Percent Correct
Block
Model: Single vs. Dual Task Transfer
Single
Dual
Percent Correct
Block

(a)


Human: Verbilization
Verb 5
Verb 1 3 5
Percent Correct
Block
Model: Verbilization
Verb 5
Verb 1 3 5
Percent Correct
Block

(b)