

Integrated Cognitive Architectures: A Survey

Hui-Qing Chong

School of Chemical and Biomedical Engineering, Nanyang Technological University
Nanyang Avenue, Singapore 639798
E-mail: ch0015ng@ntu.edu.sg

Ah-Hwee Tan*

School of Computer Engineering, Nanyang Technological University
Nanyang Avenue, Singapore 639798
E-mail: asahtan@ntu.edu.sg

Gee-Wah Ng

DSO National Laboratories
20 Science Park Drive, Singapore 118230
E-mail: ngeewah@dso.org.sg

Artificial Intelligence Review
To appear

Abstract

This paper aims to present an account of the state of the art research in the field of integrated cognitive architectures by providing a review of six cognitive architectures, namely Soar, ACT-R, ICARUS, BDI, the subsumption architecture and CLARION. We conduct a detailed functional comparison by looking at a wide range of cognitive components, including perception, memory, goal representation, planning, problem solving, reasoning, learning, and relevance to neurobiology. In addition, we study the range of benchmarks and applications that these architectures have been applied to. Although no single cognitive architecture has provided a full solution with the level of human intelligence, important design principles have emerged, pointing to promising directions towards generic and scalable architectures with close analogy to human brains.

Keywords: Integrated cognitive architectures; Soar; ACT-R; ICARUS; BDI; subsumption architecture; CLARION

*Corresponding author

1 Introduction

An integrated cognitive architecture can be defined as a single system that is capable of producing all aspects of behaviour, while remaining constant across various domains and knowledge bases (Newell, 1990; Anderson, Bothell, Byrne, Douglass, Lebiere, & Qin, 2004). This system would consist of many modules (or components) working together to produce a behaviour. These modules contain representations of knowledge, memories for storage of content and processes utilizing and acquiring knowledge. Integrated cognitive architectures are often used to explain a wide range of human behaviour, and to mimic the broad capabilities of human intelligence (Anderson et al., 2004; Langley & Choi, 2006).

Research on integrated cognitive architectures is interdisciplinary by nature, spanning the fields of artificial intelligence, cognitive psychology, and neurobiology. Over the past decades, many cognitive architectures have been proposed and steadily developed, based on different approaches and methodologies. However, despite that integrated cognitive architectures has been an important research area, there is a lack of formal reviews on the subject domain.

An early survey by Wray et al. (1994) provided an introduction to twelve cognitive architectures, covering the subsumption architecture, ATLANTIS, Theo, Prodigy, ICARUS, Adaptive Intelligent Systems, A Meta-reasoning Architecture for 'X', Homer, Soar, Teton, RALPH-MEA, and Entropy Reduction Engine. The survey made comparison based on the properties of the architectures, capabilities of the agents, environmental consideration, generality, psychological validity, and efficiency. However, the references used by the authors of this website were dated before 1992. In addition, the authors simply made comparison by describing the criteria, followed by a list of architectures displaying each property.

Köse (2000) perhaps presented the most comprehensive survey of all in terms of a wide coverage of architectures and the capabilities compared based on autonomous mobile robots. However, this paper was written in 2000 and therefore missed the many new development in the recent years. For instance, ICARUS describes a problem solving module explicitly (Langley & Choi, 2006), yet Köse stated that problem solving capability was absent in ICARUS.

Stollberg and Rhomberg (2006) gave a detailed overview of Soar, BDI and others, as well as comparison of these architectures based on goal types and resolution techniques. Though the number of aspects compared was limited, the authors made functional comparisons between the architectures, rather than just stating whether each feature compared had been well addressed in the architecture.

The latest review by Vernon et al. (2007) included an overview of various cognitive architectures, such as Soar, ICARUS, ACT-R and others, as well as a comparison of the architectures. However, they only compared if the different aspects had been strongly addressed in the design of the architectures.

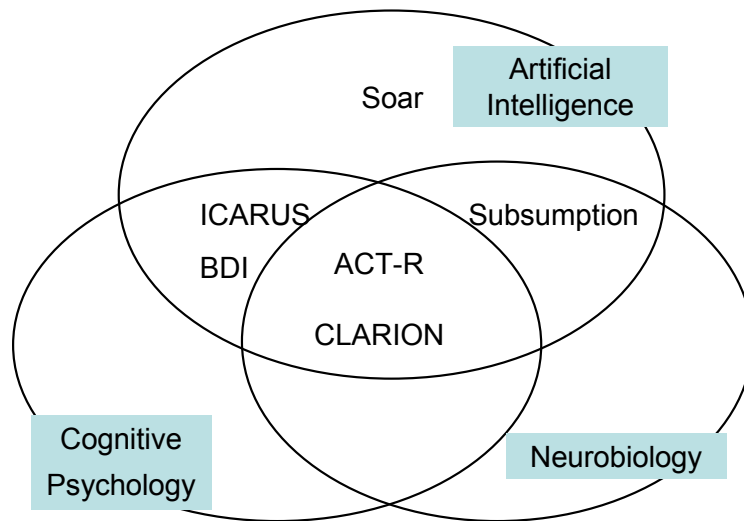


Figure 1: An overview of the six cognitive architectures.

Among all prior surveys, only Stollberg and Rhomberg (2006) had made functional comparisons of the various architectures. However, they had merely focused on the aspect of goal. In this paper, we aim to fill the gap of the prior surveys by providing a detailed comparison of a selected group of cognitive architectures. Instead of simply stating if specific functions are strongly addressed in the design, our comparison shall be at the mechanism level, describing how the architectures differ from one another in realizing the functions.

While it is clearly impossible for the paper to include all existing research and individuals, who have contributed significantly to the topic, we aim to cover a good mix of the systems that are representative of a diverse range of approaches and disciplines. Figure 1 shows the six cognitive architectures that we study, namely Soar, ACT-R, ICARUS, BDI, the subsumption architecture, and CLARION, roughly classified according to their roots and emphases.

Soar (Laird, Rosenbloom, & Newell, 1986a, 1986b; Laird, Newell, & Rosenbloom, 1987; Lehman, Laird, & Rosenbloom, 2006), based on the physical symbolic hypothesis (Newell, 1990), is one of the earliest and most extensively developed AI architectures in the history. ACT-R (Anderson et al., 2004) (also with a long history) and ICARUS (Langley & Choi, 2006) (a relatively recent model) are cognitive systems developed with the primary aim of producing artificial intelligence mimicking human cognition. While the three architectures share many features of classical artificial intelligence, including symbolic representation, production rule based inference, and means-end analysis for problem solving, ACT-R and ICARUS are notably different from Soar by their strong emphasis of producing a psychologically motivated cognitive model.

Belief-Desire-Intention (BDI) architecture (Bratman, Israel, & Pollack, 1988; Rao & Georgeff, 1991) is a popularly used framework, incorporating beliefs, desires and inten-

tions, for designing intelligent autonomous agents. Based on the studies of folk psychology and intentional systems, BDI has a special focus on intentions, representing an agent's commitments to carry out certain plans of actions (Georgeff & Ingrand, 1989).

Coined as the new artificial intelligence, the subsumption architecture (Brooks, 1999) is notably different from the other cognitive architectures in its approach and design. The subsumption architecture is behaviour based and thus does not contain any problem solving or learning module. The idea of higher layers subsuming lower layers in the subsumption architecture has its root from neurobiology.

CLARION (Sun & Peterson, 1996, 1998; Sun, Merrill, & Peterson, 2001; Sun & Zhang, 2006) is a hybrid model integrating both symbolic and connectionist information processing. The design of CLARION is based on neural networks as well as cognitive psychology. As a result, it is similar to ACT-R as both models are based on a combination of artificial intelligence, cognitive psychology and some favour of neurobiology.

The rest of this paper is organized as follows. Section 2 first presents a brief review of the individual cognitive architectures, in terms of the underlying philosophy, architecture, functions, and processes. Section 3 compares the six architectures in terms of eight cognitive functions, covering perception, memory, goals, planning, problem solving, reasoning/inference, learning, and relevance to cognitive neuroscience. Section 4 presents the benchmarks and applications of the various architectures. The final section rounds up the review, discusses some interesting convergence in design principles, and suggests promising directions for future research.

2 A Review of Six Cognitive Architectures

2.1 State, Operator, And Result (Soar)

Soar (Laird et al., 1986a, 1986b, 1987; Lehman et al., 2006) has its root in the classical artificial intelligence (Newell, 1990) and is one of the first cognitive architectures proposed. The main objective of Soar is to handle the full range of capabilities of an intelligent agent through a general mechanism of learning from experience. Therefore, Soar incorporates a wide range of problem solving methods and learns all aspects of the tasks to perform them (Laird et al., 1987). It is used for the understanding of the mechanisms required for intelligent behaviour and the incorporation of the mechanisms to form a general cognitive architecture in classical artificial intelligence (Laird et al., 1986a, 1986b).

2.1.1 Architecture

Figure 2 illustrates the Soar architecture, consisting of the various memory structures and a decision making mechanism linking perception to action. The memory structures present

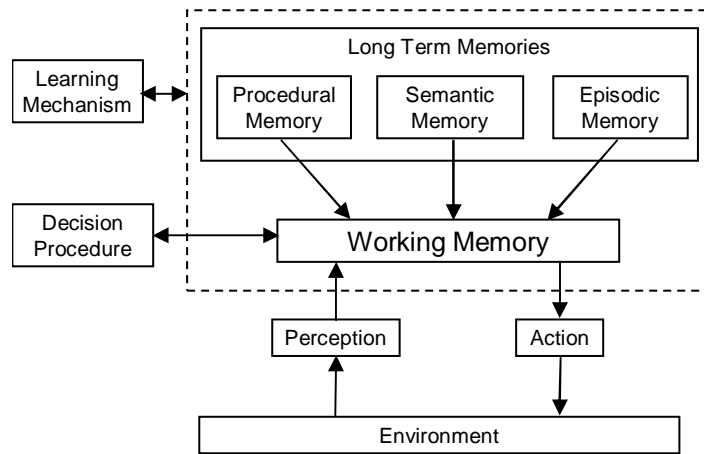


Figure 2: The Soar cognitive architecture (Adapted from Lehman et al., 2006).

in the Soar architecture include long term memory and working memory, which can be considered as short term memory. Information from the environment is made available in the working memory via perception, allowing appropriate actions to be chosen during domain-independent problem solving. The external environment can also be influenced by the architecture through the implementation of selected actions.

Knowledge is stored in the long term memory, which can be classified into procedural, semantic and episodic memories. Procedural memory provides the knowledge of performing tasks. Semantic memory stores general facts about the world and is considered as declarative knowledge. On the other hand, episodic memory contains specific memory of an event experienced. Hence, both procedural and semantic memories are universally applicable, whereas episodic memory is contextual specific. In instances whereby procedural knowledge is insufficient, semantic and episodic memories are employed as cues to aid in problem solving.

The working memory in Soar cognitive architecture houses all the knowledge that are relevant to the current situation. It contains the goals, perceptions, hierarchy of states, and operators. The states (and sub-states) give the information on the current situation. The operator provides the steps to apply during problem solving, while the goal directs the architecture into the desired state. The content of the working memory, also known as working memory elements, can trigger both the retrieval of relevant knowledge from the long term memory into the working memory, and motor actions.

2.1.2 Functions and Processes

Soar supports a number of problem solving methods. Through the means-ends analysis, the system selects and applies operators to result in a new state that is closer to the desired state. In order to bring the system closer to its goal, Soar implements a five-phase

decision cycle, constituting of input, elaboration, decision, application, and output. The main function of the decision cycle is to choose the next operator to apply (Laird et al., 1986b, 1987; Lehman et al., 2006).

Percepts are added into the working memory in the input phase for use during the elaboration phase. Production rules are then matched with the working memory elements in order to bring knowledge relevant to the current problem into the working memory. Meanwhile, preferences are created to act as recommendations for selection of appropriate operators. The elaboration phase continues till the firing of rules in the long term memory ceases, ensuring that all knowledge relevant to the current situation are considered before a decision is made (Laird et al., 1986a, 1986b; Lehman et al., 2006). When the elaboration phase reaches a quiescence, the decision cycle proceeds to the decision phase, wherein the preferences are evaluated. The better of the suggested operators are chosen and applied during the application phase. A motor action is then performed during the output phase as a result of applying the selected operator.

An impasse is encountered whenever the procedural knowledge is inadequate for problem solving (Laird et al., 1986a; Chong, 2004). In Soar, there are four types of impasse: no-change, tie, conflict, and rejection. The no-change impasse occurs when the elaboration phase enters a quiescence without any suggestion, while the tie impasse refers to the situation whereby no object is superior over the others. Cases in which two or more candidate objects are better than each other result in the conflict impasse. The rejection impasse is encountered when all operators are rejected.

Any impasse encountered provides an opportunity for Soar to learn from its experience. The learning mechanisms proposed in the Soar architecture include chunking, reinforcement learning, episodic memory, and semantic memory. A successful resolution of an impasse results in the termination of a goal or subgoal, which in turn leads to formation of chunks. The chunking mechanism enables new production rules to be added into the long term memory. These chunks are used whenever a similar situation is encountered, thereby avoiding the same impasse and improving the performance of the agent in the future. Soar also receives rewards from successes and punishments from failures, allowing the agent to undergo reinforcement learning. Any operator resulting in a reward upon an execution is given a positive reinforcement, and such operators are more likely to be selected in the future. The episodic and semantic memories store information on the agent's past experiences, and therefore both are used as additional cues to select applicable operators (Laird et al., 1986a; Lehman et al., 2006; Chong, 2004).

2.2 Adaptive Control of Thought-Rational (ACT-R)

The key motivation underlying ACT-R (Anderson et al., 2004) is to develop a model of human cognition, using empirical data derived from experiments in cognitive psychology

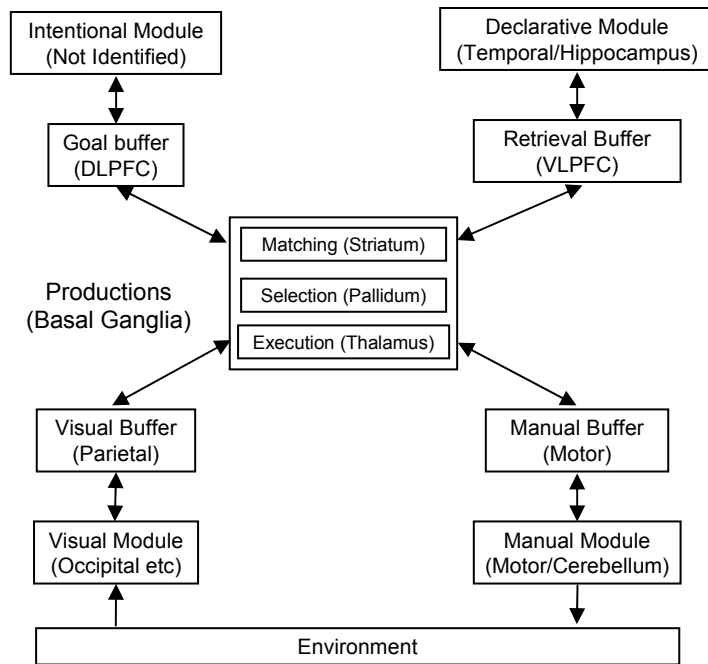


Figure 3: The ACT-R 5.0 model (Adapted from Anderson et al., 2004).

and brain imaging. It provides a step by step simulation of human behaviour for detailed understanding of human cognition. In recent projects, ACT-R is also used in the prediction of activation patterns in brain with the aid of functional Magnetic Resonance Imaging (fMRI).

2.2.1 Architecture

In this cognitive architecture, information from the external environment and knowledge stored in the memories work conjunctively to select actions for execution to satisfy the goal(s) of the agent. The four basic modules in the ACT-R architecture are the visual, manual, declarative memory, and goal modules as illustrated in Figure 3. The visual module is needed for identifying objects in the visual field and can be further classified into the visual-location and visual-object modules. The visual-location module is required for determining the locations of objects, while the visual-object module aids in the recognition of the objects. The manual module is used for the control of the hand actuators. Information from memories can be retrieved via the declarative memory module, while the goal module keeps track of the agent’s current goals, and enables the maintenance of the agent’s thought in the absence of supporting external stimuli (Anderson et al., 2004).

All the above modules are coordinated through the central production system, which is identified to be the basal ganglia in human brain. Production rules are implemented in the central production system, corresponding to striatum, pallidum and thalamus, as well as

the associated connections with the various buffers. The response of the central production system is limited by the amount of information available in the buffers of various modules. Similarly, the modules can only communicate with the central production system through the information present in the buffers. As a result, the system is not aware of all information stored in the long term memory but only those that have been retrieved.

The goal buffer in ACT-R helps to keep track of one's internal state during problem solving and it has been identified that the dorsolateral prefrontal cortex (DLPFC) holds the role of a goal buffer. On the other hand, the declarative memory retrieved from the long term memory store is passed to the retrieval buffer, which is associated with the ventrolateral prefrontal cortex (VLPFC). The manual buffer for controlling the agent's hand is associated with the motor and somatosensory cortical areas, which are the two areas in the brain that control and monitor hand movement. The visual buffers in this model include both the dorsal 'where' path of visual system and the ventral 'what' system. The dorsal 'where' system is important for locating the object, while the ventral 'what' system tracks visual objects and their identities. The various buffers interact with one another and the central production system to achieve cognition (Anderson et al., 2004).

2.2.2 Functions and Processes

This model assumes a mixture of parallel and serial processing. Parallelism occurs within each module as well as between different modules. For example, the declarative module can execute a parallel search through many memories for retrieval. However, there are two levels of serial bottleneck in this model. The first bottleneck is that the content of any buffer is only limited to a single declarative unit of knowledge, known as chunks in this model. Hence, either the retrieval of a memory or the encoding of an object can occur at a time. The second bottleneck is that only a single production is fired in every cycle.

The knowledge in ACT-R can be stored in either procedural memory or declarative memory. Knowledge in declarative memory is represented as chunks, while knowledge in procedural memory is available in terms of production rules. In order to retrieve the necessary knowledge for problem solving, chunks stored in the declarative modules are first activated. A chunk is only retrieved when its activation level raises above a threshold level. Whenever a production rule matches the chunk retrieved, the production rule is fired.

A learning mechanism being developed in ACT-R currently is production compilation. Production compilation enables separate production rules to combine into one and allows performing of action without the retrieval of declarative instructions (Anderson et al., 2004).

2.3 ICARUS

ICARUS (Langley, 2004; Langley & Choi, 2006) has its root in designing physical and embodied agents, through integrating perception and action with cognition. This cognitive architecture also aims to unify reactive execution with problem solving, combine symbolic structures with numeric utilities, and learn structures and utilities in a cumulative manner.

The design for ICARUS has been guided by the following principles, which provide the differentiation of ICARUS from other cognitive architectures: (1) Cognitive reality of physical objects; (2) Cognitive separation of categories and skills; (3) Primacy of categorization and skill execution; (4) Hierarchical organization of long term memory; (5) Correspondence of long term or short term structure; and (6) Modulation of symbolic structures with utility functions.

2.3.1 Architecture

Figure 4 presents an overview of the ICARUS architecture, consisting of four main components, namely the perceptual buffer, the conceptual memory, the skill memory, and the motor buffer. The perceptual buffer is involved in the temporary storage of percepts. Conceptual memory can be further classified into short term conceptual memory and long term conceptual memory. The short term conceptual memory, otherwise known as the belief memory, contains the set of active inferences describing the relations among the objects perceived. On the other hand, the long term conceptual memory consists of the known conceptual structures describing objects or classes of the environmental situations. Similarly, the skill memory is subdivided into short term skill memory and long term skill memory. All the skills that can be executed by the ICARUS agent are stored in the long term skill memory, while the chosen skill to be implemented is brought into the short term skill memory. The skill signals in the motor buffer, read out from the short term skill memory, are then executed by the ICARUS agent to bring about changes to its environment.

2.3.2 Functions and Processes

The key processes in ICARUS include conceptual inference, goal selection, skill execution, problem solving, and learning (Langley & Choi, 2006). Conceptual inference is the mechanism responsible for matching the conceptual structures against the percepts and beliefs. It is dependent on the content and representation of elements that have been stored in both the short term and long term memories. The environment, in which the ICARUS agent is situated, affects the perceptual buffer and thus the conceptual memory, leading to a repeat of the conceptual inference that updates the description of the environment. In each cycle, the agent retrieves the attributes of the perceived objects into the perceptual buffer and matches them against the conceptual definitions in the long term memory. As a result, all

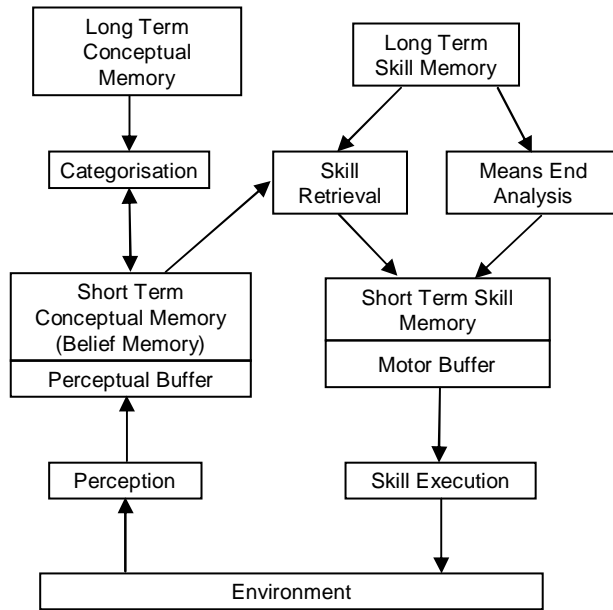


Figure 4: The schematic diagram of ICARUS (Adapted from Langley, 2004).

elements that are implied deductively by the percepts and the conceptual definitions are added to the belief memory. Conceptual inference occurs in a bottom-up (data driven) manner, and is similar to the elaboration phase in Soar (Langley, 2004; Langley, Arai, & Shapiro, 2004; Langley & Choi, 2006).

Goals in ICARUS represent the objectives that an agent aims to satisfy. In order to achieve the goals, the chosen skills within the long term skill memory are executed, thereby altering the environment to bring the agent closer to its goals. ICARUS focuses on only one goal at a time. Therefore in each cycle, only the highest priority goal not yet satisfied is attended to by the ICARUS agent. The agent then selects the skills that are applicable to the agent’s current belief and executes the skills chosen via the motor buffer.

Whenever the agent is unable to find an applicable skill to achieve its current goal upon reaching an impasse, it employs means-ends analysis as its problem solving strategy. Means-ends analysis involves the decomposition of a problem into subgoals. An applicable skill path is selected for each subgoal. After accomplishing a subgoal, the agent returns to the parent goal, before selecting another skill path for the next subgoal.

ICARUS is similar to Soar in terms of learning, since both architectures learn from impasse. The difference is that ICARUS retrieves information from the goal stack to select applicable skill, while Soar uses task-dependent knowledge to solve an impasse. Skill learning in ICARUS occurs as a result of problem solving whereby a skill is learnt when the agent is able to execute the action successfully.

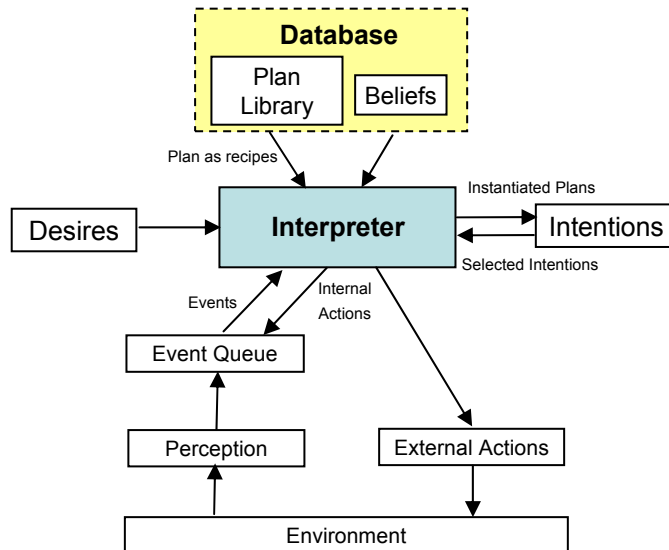


Figure 5: The BDI architecture in dMARS specification (Adapted from Guerra-Hernández et al., 2004).

2.4 Belief-Desire-Intention (BDI)

BDI is based on the Dennett’s theory of intentional systems (Dennett, 1987) and the theory of human practical reasoning (Bratman et al., 1988). Originally developed as a system that can reason and plan in a dynamic environment, BDI meets real-time constraints by reducing the time used in planning and reasoning. BDI is designed to be situated, goal directed, reactive, and social. This means a BDI agent is able to react to changes and communicate in their embedded environment, as it attempts to achieve its goals. Mechanisms for responding to new situations or goals during plan formation for general problem solving and reasoning in real time processes are also included in BDI systems (Georgeff & Ingrand, 1989; Sardina, de Silva, & Padgham, 2006). BDI agents are typically implemented as Procedural Reasoning System (PRS). Later implementations of BDI architectures are also largely based on PRS (Guerra-Hernandez, Fallah-Seghrouchni, & Soldano, 2004; Sardina et al., 2006).

2.4.1 Architecture

Figure 5 shows the schematic of the BDI architecture, consisting of the main components of beliefs, desires, and intention, together with an interpreter. Having its origin in Computer Science, BDI adopts a database for storing beliefs and plans. Beliefs are facts about the world as well as inference rules that may lead to acquisition of new beliefs. The beliefs are updated by the perception of the environment and the execution of the intentions (Georgeff & Ingrand, 1989; Guerra-Hernandez et al., 2004).

Plans refer to sequences of actions that a BDI agent can perform to achieve one or

more of its intentions. It is a type of declarative knowledge containing ideas on how to accomplish a set of given goals or react to certain situations. A plan consists of a body and an invocation condition. The body of a plan comprises of possible courses of actions and procedures to achieve a goal, while the invocation condition specifies the prerequisites to be met for the plan to be executed, or to continue executing. A plan can also consist of subgoals to achieve, and an execution of the plan may create new subgoals, leading to the formation of goal-subgoal hierarchy (Sardina et al., 2006).

Desires, also known as goals, are the objectives that a BDI agent aims to accomplish. The usage of the term *goal* in this architecture is only restricted to non-conflicting desires. Conflicts among desires are also resolved using rewards and penalties (Dastani, Hulstijn, & van der Torre, 2001). A goal is said to be successfully achieved when a behaviour satisfying the goal description is executed.

Intentions are the actions that the agent is committed to perform in order to achieve the desires. Thus, it contains all the tasks chosen by the system for execution. Each intention is implemented as a stack of plan instances, and is only considered as executable if the context of the plan matches with the consequence of the beliefs.

2.4.2 Functions and Processes

A system interpreter manipulates the components of the BDI architecture described above. In each cycle, it updates the event queue by the perceptual input and the internal actions to reflect the events observed, followed by the selection of an event. New possible desires are then generated by finding relevant plans in the plan library for the selected event. From the set of relevant plans, an executable plan is then selected and an instance plan is thus created. The instance plan is pushed onto the existing intention stack (when the plan is triggered by an internal event such as another plan) or a new intention stack (when the plan is triggered by an external event). The BDI agent interacts with its environment either through its database when new beliefs are acquired or through actions performed during the execution of the intention. The interpreter cycle repeats after the intention is executed. However, the acquisition of a new belief or goal may lead to an alteration of plans and cause the agent to work on another intention (Dastani & van der Torre, 2002).

The BDI architecture integrates means-ends reasoning with the use of decision knowledge in its reasoning mechanism. It is however notable that the system performs means-ends reasoning and planning in the context of existing intentions. A special feature of the BDI architecture is that once it is committed to a process of achieving goals, it does not consider other pathways although they may be better than the one chosen. In this way, the architecture is able to cut down its decision time.

An active goal in the BDI architecture is dropped once the system recognizes the goal as accomplished or cannot be readily accomplished. Only when all attempts at achieving

a goal by trying all applicable plans have failed before the goal can be labelled as “cannot be readily accomplished” (Georgeff & Ingrand, 1989).

The original BDI architecture does not have the mechanism of learning. However, there have been recent attempts to include learning mechanisms into BDI systems. For example, Norling (2004) extended BDI for learning recognition primed decision making. A table lookup version of Q-learning was adopted to learn reactive rules for path finding in a grid world. More recently, Subagdja and Sonenberg (2005) further extended the BDI architecture to incorporate learning by generating and testing hypothesis for the purpose of formulating plans. At the multi-agent level, Guerra-Hernandez et al. (2004) expanded the BDI architecture to incorporate learning in multi-agent systems using a first order method called induction of decision tree.

2.5 The Subsumption Architecture

The subsumption architecture represents a “new” approach to artificial intelligence derived from behaviour-based robotics. In view that classical artificial intelligence often has the problems of extensibility, robustness, and achieving multiple goals, The subsumption architecture was proposed as an incremental and bottom-up approach to deal with these problems (Brooks, 1999). The subsumption architecture decomposes a problem in terms of the behaviours exhibited by the robots instead of the stages of information flowing within the controller as in a traditional AI design. Conversely, some researchers believed that traditional cognitive architectures would be useful for higher cognitive functions, while the subsumption architecture was only meant for reflexive tasks (Hartley & Pipitone, 1991).

2.5.1 Architecture

Figure 6 depicts the subsumption architecture, comprising of a hierarchy of “simple” behaviour-based modules organized into layers (levels of competence). Subsumption allows all layers to access the sensor’s data and multiple (simple) behaviours to operate in parallel. Each layer is capable of controlling the system by itself, unlike classical artificial intelligent agents. Each level of competence displays a behaviour to pursue a particular goal and a higher-level layer tends to subsume the underlying layers. The lower layers work like fast-adapting mechanisms, allowing the agent to react quickly to changes in its environment. In contrast, the higher layers control the system towards the overall goals. Consequently, the lower level behaviours are the results of the reactions towards the environment, while the higher level behaviours are driven by the aim to pursue the primary goals. In order to achieve a higher level of competence, a new layer of control can be simply added without altering existing layers (Brooks, 1999; Nakashima & Noda, 1998; Toal, Flanagan, Jones, & Strunz, 1996). The layered design of the architecture is thus analogous to a biological nervous system, wherein new sections of brain are developed for new functions, but old

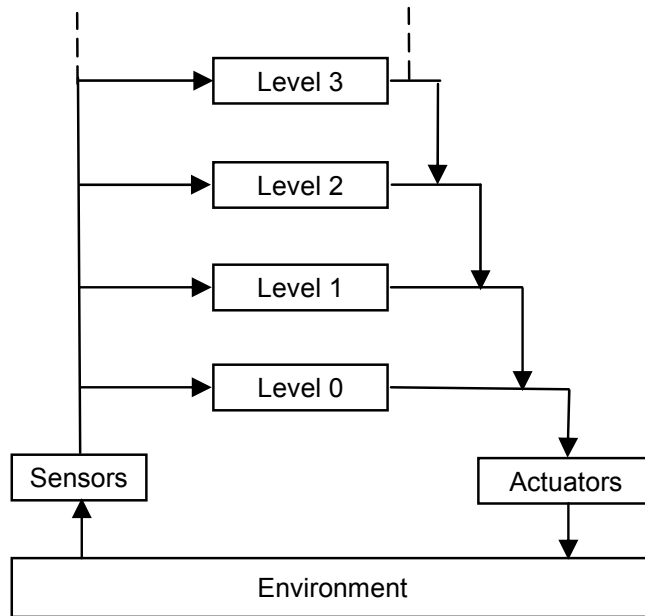


Figure 6: The subsumption architecture (Adapted from Brooks, 1999).

sections are still preserved to perform their original tasks (Brooks, 1991; Bryson, Smaill, & Wiggins, 2005).

The layered-based design of the subsumption architecture also allows for easier implementation. Specifically, each module of the system can be tested and debugged until flawless before proceeding to the next higher level. This type of incremental, bottom-up approach of the model ensures the workability of the system and also simplifies the debugging process. The architecture is also able to cope with noise due to imperfect sensory information or unpredictability in the environment. By designing multiple distributed layers of behaviour into the system, the possibility of system collapse due to a drastic change in environment is reduced. This allows the system's performance to degrade gradually, rather than failing completely when facing non-ideal inputs.

2.5.2 Functions and Processes

It is notable that symbolic processing is absent in the subsumption architecture. There is no explicit representation of knowledge and therefore there is no matching of rules. The subsumption architecture is also free from a central control (Brooks, 1999). The layers in the architecture are driven by the data collected, with no global data or dynamic communication. This implies that the system is being reactive via implementing an activity as a consequence of events. As such, the perception of the system is said to be tightly coupled to the action within each layer.

The layers are also expected to react quickly in order to sense the rapid changes in the

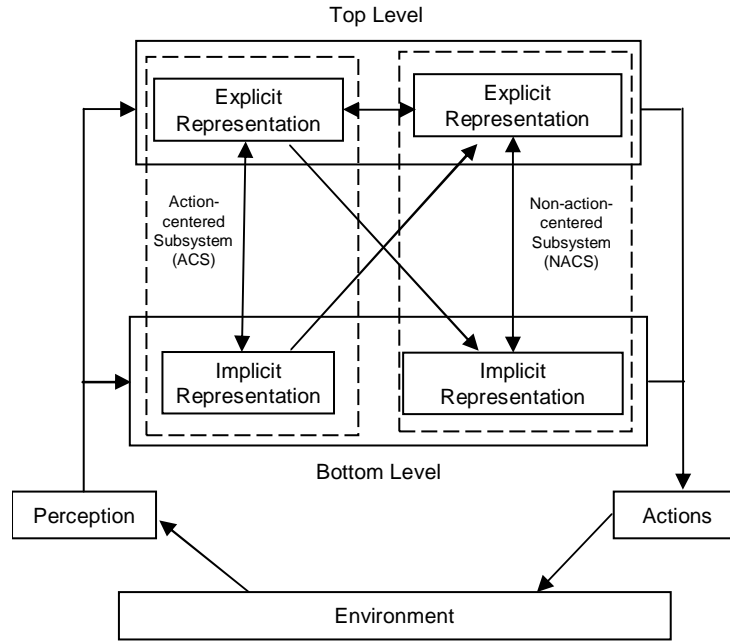


Figure 7: The CLARION architecture (Adapted from Sun et al., 2005).

environment. Therefore, the communication between the layers occur in one direction only, resulting in minimal interactions between the layers. However, as mentioned earlier, the higher layers can suppress the inputs and inhibit the outputs of the lower layers, leading to an adjustment in the behaviour for the purpose of fulfilling the overall goal (Butler, Gantchev, & Grogona, 2001; Brooks, 1999; Amir & Maynard-Zhang, 2004).

2.6 Connectionist Learning with Adaptive Rule Induction ON-line (CLARION)

With its root in connectionist systems (i.e. neural networks), CLARION is a hybrid architecture that incorporates both implicit and explicit memories for reasoning and learning (Sun et al., 2001). Procedural knowledge (implicit memory) can be gradually accumulated with repeated practice, and subsequently applied to practiced situations of minor variations. To deal with novel situations, declarative knowledge is required to guide in the exploration of new situations, thereby reducing time for developing specific skills. It also unifies neural, reinforcement and symbolic methods to perform on-line, bottom-up learning. Hence, CLARION is able to react in a dynamically changing environment without any preexisting knowledge installed into the architecture (Sun & Peterson, 1996, 1998).

2.6.1 Architecture

As shown in Figure 7, the CLARION architecture consists of two levels: a top level containing propositional rules and a bottom level (reactive level) containing procedural knowledge. The top level comprises of explicit symbolic mechanisms and the bottom level uses subsymbolic neural mechanisms. Procedural knowledge can be acquired through reinforcement learning in a gradual and cumulative fashion, while declarative knowledge is acquired through rule extraction by trials and errors. The architecture consults both the rules and the Q-values computed, combining their recommendations in a weighted sum, in order to select the most appropriate action to perform at each step.

It is generally accepted that declarative knowledge is explicit, while procedural knowledge is implicit. As a result, declarative knowledge is usually accessible but procedural knowledge is not. However, there may be exceptions. Thus, the CLARION architecture includes non-action-centered subsystem (NACS) and action-centered subsystem (ACS). NACS contains mostly declarative knowledge, whereas ACS contains mainly procedural knowledge. The top level of NACS is a general knowledge store (GKS) and contains explicit representation. On the other hand, an associative memory networks (AMN) forms the bottom level of NACS, which consists of implicit representation. In ACS, explicit action rules are stored in the top level. These rules are either provided by the agent’s external environment or extracted from information in the bottom level. The bottom level of ACS comprises of implicit decision networks, which can be trained by reinforcement learning (Sun, Slusarz, & Terry, 2005; Sun & Zhang, 2006).

Besides ACS and NACS, CLARION further consists of two other subsystems, namely the motivational subsystem for guiding the agents overall behavior by forming goals; and the meta-cognitive subsystem as the main controller managing the processes of the other subsystems so as to provide structured behavior.

2.6.2 Functions and Processes

In view that people often use similarities between objects in determining the outcomes of inductive reasoning, CLARION includes both rule-based and similarity-based reasoning to mimic human reasoning. Reasoning takes place in CLARION through comparing a known chunk with another chunk. When the similarity between two chunks is sufficiently high, an inference regarding the relations between them can be made. While the comparison of chunks gives rise to similarity-based reasoning, the usage of chunks during reasoning constitutes to rule-based reasoning. The process of reasoning in CLARION can also occur iteratively to allow all possible conclusions to be found. In iterative reasoning, the conclusion reached at each step can be used as a starting point for the next step. The overall outcome is to select the appropriate course of action for the agent to react to its environment (Sun & Zhang, 2006).

Learning in CLARION can be differentiated between implicit learning or explicit learning. Implicit learning can be considered as learning procedural skills, and explicit learning takes place by learning those rules as declarative knowledge. Learning of procedural knowledge at the bottom level occurs through the reinforcement learning paradigm, which works by making adjustments to the probability of selecting a particular action. When an action receives a positive reinforcement, the chance of selecting the action increases; otherwise, the chance of selection decreases. The learning processes employed in CLARION are through neural mechanisms, such as multi-layer neural networks and backpropagation algorithm, which are used to compute Q-values. When the situation becomes better due to an action performed, the Q-value of the action raises, and thus increases the tendency of performing that action.

Learning of rules in the top level takes place by extracting knowledge from the bottom level. Upon the successful execution of an action, the agent extracts a rule corresponding to the action selected by the bottom level and adds it to the rule network. The agent in turn removes more specialized rules in the rule network and keeps the general rules. The agent then tries to verify the rules learnt via applying them in the subsequent interactions with its environment. If the performed action results in an unsuccessful attempt, the rule is modified to be more specific and exclusive of the current situation. In contrast, a successful attempt enables a rule to be generalized, making it more universal in application (Sun & Peterson, 1996, 1998).

3 Functional Comparison

In this section, we compare the six architectures based on the cognitive functions that they support. This is notably not an easy task, especially when each architecture places emphases on different aspects of cognition and may not use a consistent set of terminologies.

3.1 Perception

As shown in Table 1, all six architectures identify some forms of perception as the input module to the system. For example, a Soar agent stores the information from the external environment directly into its working memory. The perceptual input can be from a simulation programme, video camera or feedback from a robot (Lehman et al., 2006). In ACT-R, the perceptual information from the environment enters the visual module and is made available to the central production system via the visual buffer (Anderson et al., 2004). ICARUS perceives objects in its environment through its perception buffer (Langley & Choi, 2006).

In the BDI architecture, perception is available in the form of events stored in the event queue. An event can be an acquired/removed belief or any sensory input received from the

Table 1: Realization of perception in the six cognitive architectures.

Architecture	Means of Perception
Soar	Perceptual input stored directly as part of the working memory
ACT-R	Perception stored in visual module and made available through visual buffer
ICARUS	Perceptual stored in perceptual buffer as part of conceptual memory
BDI	Perception mapped to events stored in event queue
Subsumption	Perception is available through sensors
CLARION	Perceptual input represented as dimension/value pairs

environment. Multiple items can be present in an event queue, and each item consists of an event paired with either a failed plan or a currently executed plan (Guerra-Hernandez et al., 2004; Sardina et al., 2006). An event queue supports a time-ordered sequence of events to be processed by a system. It thus involves a temporal dimension comparing to a buffer that contains a snapshot of the current situation or a single chunk of knowledge (Anderson et al., 2004).

The subsumption architecture has been widely used for building robots. The system perceives the environment through its sensors, which may be sonar or visual sensors, such as CCD cameras (Brooks, 1991, 1999). The input into CLARION is a series of dimension/value pairs describing the state of the world. It consists of the objects in the environment, as well as the items in the working memory and the current goal(s).

3.2 Memory

As presented in Table 2, all systems except the subsumption architecture make use of both short term memory and long term memory explicitly. In Soar, short term memory, also known as working memory, contains information received from the environment and all rules relevant to the current situation. Long term memory in Soar can be procedural, declarative (semantic) or episodic and is realized through the production rule system (Nuxoll & Laird, 2004; Lehman et al., 2006). Short term memory in ACT-R is available to the architecture through the buffers. Whereas Soar suggests a centralized working memory containing preferences, perception, states, and sub-states etc., ACT-R uses a distributed memory system, wherein the goals, beliefs, sensory, and motor signals are situated in distinct buffers (Anderson et al., 2004). Similar to that of Soar, the production rules in ACT-R are stored in the procedural memory. However, whereas the chunks in Soar refer to acquired knowledge from the chunking mechanism, the chunks in ACT-R are knowledge stored in the declarative memory. The CLARION architecture also contains working memory as a temporary information storage to facilitate decision making (Sun et al., 2001). However, the propositional rules are stored in the declarative knowledge (top level), while the bottom

Table 2: Implementation of memory functions in the six cognitive architectures.

Architecture	Representation	Working Memory	Long Term Memory
Soar	Symbolic	Contains perceptual input, states, and production rules relevant to current situation	Contains procedural, declarative (semantics), and episodic memory
ACT-R	Symbolic	Contains goal, perception, relevant knowledge, and motor action in the various buffers	Contains declarative knowledge in declarative module and procedural knowledge in production system
ICARUS	Symbolic	Consists of perceptual buffer, belief memory, and short term skill memory	Procedural knowledge stored in long term skill memory, declarative knowledge in long term conceptual memory
BDI	Symbolic	Belief as working memory	Plans as long term memory
Subsumption	No explicit representation	No explicit working memory	Heuristics within each layer
CLARION	Symbolic + Subsymbolic	As temporary information storage	Procedural at bottom level Declarative at top level

level contains the procedural knowledge (Sun & Peterson, 1996, 1998). Thus, the long term memory representation in CLARION differs from those of Soar and ACT-R, wherein the rules are stored as procedural knowledge.

The short term memory in ICARUS takes the form of *belief memory* updated through a conceptual inference process as well as *short term skill memory* (Langley & Choi, 2006). Although ICARUS does not refer to declarative and procedural memories explicitly, its conceptual memory can be considered as declarative memory and the skill memory as procedural memory.

In BDI, the beliefs and facts about the world are stored in the database as symbolic representation of the world perceived. A database contains both the plan library and the agent’s beliefs. As a result, the database represents a more permanent form of storage of knowledge than buffers (Georgeff & Ingrand, 1989). It is believed that different types of databases can be used for various agent systems, depending on the design of the agent. A plan comprises of both declarative and procedural knowledge. The declarative knowledge is made available in the plan library as plan rules, which specify the conditions a given plan can be applied to achieve the agent’s intentions. The procedural knowledge defines the actions to be undertaken when the plan is carried out (Wooldridge, 1996).

In contrast to the others, the subsumption architecture does not model any knowledge of the world explicitly, since it does not engage in any higher cognitive function. As a result, it does not refer to explicit memory structure (Brooks, 1999).

3.3 Goals

Goals are critical in all architectures, as the objectives to be fulfilled by an agent. Table 3 summarizes the goal mechanisms of the six cognitive architectures. In most architectures, a goal is represented by a target state that an agent wants to achieve. Soar further treats goals as a type of objects in the centralized working memory, which can be altered in each decision cycle. The goals can be divided into subgoals whenever an impasse is encountered, leading to the creation of the goal-subgoal hierarchy (Laird et al., 1986a).

In ACT-R, goals are stored in the intentional module and are available to the central production system via the goal buffer (Anderson et al., 2004). A goal can be decomposed into subgoals and the new goals are added into the goal stack in ACT-R. A goal is subsequently removed from the goal stack once it is accomplished (Anderson & Schunn, 2000).

Whereas ICARUS takes a simple approach of maintaining goals in the goal memory (Langley & Choi, 2006), BDI has a much more elaborated treatment of goals as *desires*. BDI also supports a goal life cycle, maintaining the distinction among active goals, inactive goals, accomplished goals, abandoned goals, and unattainable goals (Georgeff & Ingrand, 1989). In addition to defining goals as target states, some BDI architectures also incorporate *perform goals*, which only require certain activities to be performed regardless of the outcome. In other words, as long as the specified activity is executed, a *perform goal* is considered as accomplished (Braubach, Pokahr, Moldt, & Lamersdorf, 2005).

The subsumption architecture is different from the other architectures as it allows multiple goals to be processed in parallel. This is made possible as each layer works to achieve its goal(s) independently, although the higher layers can subsume the goals of lower layers. For example, consider a robot system, wherein the goal of the first (lower) layer is to avoid object and the goal of the second (higher) layer is to wander around. Whenever the robot remains stationary to avoid colliding with any obstacle, the second layer subsumes the output behaviour and requests the robot to wander around. As each layer operates independently, multiple goals can be active and pursued at any one time (Brooks, 1991, 1999).

In CLARION, a motivational subsystem creates and stores goals using a goal structure, which can be a goal stack or a goal list (Sun, 2003). Whereas the goal stack works in a similar way as other architectures, the goals in a goal list can be accessed randomly and they compete with each other to be the current goal.

3.4 Problem Solving

Problem solving refers to the capability of searching for a solution, when an agent does not know how to move from a given state to a goal state. Table 4 compares the problem solving mechanisms of the six cognitive architectures.

Table 3: Goal representation in the six cognitive architectures.

Architecture	Goal Representation
Soar	Goals represented as states in working memory Support goal-subgoal hierarchy
ACT-R	Goals stored in the intentional module and made available through the goal buffer
ICARUS	Goals placed in the goal stack Support goal-subgoal hierarchy
BDI	Non-conflicting goals as desires Support goal-subgoal hierarchy
Subsumption	Multiple goals across modules in different layers Allow multiple and parallel goal processing
CLARION	Goals stored in a goal structure, such as goal stack or goal list

Problem solving in ACT-R occurs via the activation of chunks in the declarative memory and the retrieval of knowledge from the procedural memory. Soar implements the decision procedure to select the appropriate operator using preferences created during the elaboration phase. The selected operator is then applied to perform a motor action to advance in a problem space (Lehman et al., 2006). In addition, Soar makes use of means-ends analysis to aid in problem solving (Laird et al., 1986a), by reducing the differences between the current state and the goal state. ICARUS uses a similar version of means-ends analysis, which places goals in the goal stack and searches for applicable skills using a backtracking strategy (Langley & Choi, 2006).

The BDI architecture combines means-ends analysis with decision making to select the best plan to implement (Guerra-Hernandez et al., 2004; Georgeff & Ingrand, 1989). The subsumption architecture does not have any explicit problem solving mechanism. Specifically, the actions executed by the agent are reflexive in nature, rather than being chosen via a problem solving mechanism (Brooks, 1999).

The problem solving mechanism for CLARION has not been explicitly outlined in the literatures reviewed. However, CLARION combines recommendations from the top and bottom levels to decide on appropriate reactions (Madden & Howley, 2003). The process of searching for appropriate actions may be considered as a form of problem solving.

3.5 Planning

Table 5 compares the six architectures in terms of their planning capabilities. Planning can be viewed as the generation of action sequences to achieve a given goal. It can also be considered as making choice of the course of actions to pursue (Georgeff & Ingrand, 1989). Soar does not have a strong emphasis on planning. However, by following the

Table 4: Problem solving in the six cognitive architectures.

Architecture	Mechanism for Problem Solving
Soar	Decision procedure for selecting appropriate operators; Means-ends analysis
ACT-R	By activation of chunks in Bayesian framework and production rule firing when chunks match with rules
ICARUS	Means-ends analysis by searching and backtracking
BDI	Means-ends analysis
Subsumption	No explicit problem solving mechanism
CLARION	Combination of Q-values calculated in the bottom level and rules in the top level to choose the course of actions

second definition, the decision cycle in Soar can be seen as a planning mechanism, whereby the actions are selected based on preferences to reach a goal (Laird et al., 1986a). Similarly, there is no detailed description of planning in the ACT-R model, although it is mentioned that subgoals are created leading to a sequence of actions to perform in a plan during the problem solving process (Anderson et al., 2004).

Different from Soar and ART-R, planning is explicitly employed in ICARUS through the instantiation of skills, which is the top-down selection of skills applicable to the current beliefs for execution (Langley & Choi, 2006).

In BDI, plans are stored in the plan library and intentions are plans committed by the agent for execution¹. Each plan comprises of possible courses of action and information on situations for initiating and/or continuing its execution (Guerra-Hernandez et al., 2004; Georgeff & Ingrand, 1989).

An explicit planning mechanism is absent in the subsumption architecture. Instead, planning within the architecture is implicit by decomposing a given task into multiple processes across layers, each can be activated with the appropriate sensory inputs. The activated processes then combine to produce the required behaviours for achieving the overall goal. Nevertheless, the behaviours of the subsumption agents are still mainly reactive in nature (Brooks, 1991, 1999).

In CLARION, Q-values are used in a method known as beam search for selecting a sequence of actions during the formulation of a plan. The actions are selected such that the probability of achieving a given goal is the highest (Sun & Sessions, 1998; Sun & Zhang, 2006).

¹It is interesting to note that ACT-R contains an intentional module, which actually stores the goals of the architecture.

Table 5: Planning in the six cognitive architectures.

Architecture	Mechanism for Planning
Soar	Decision cycle selects appropriate actions, bringing system closer to goal
ACT-R	Planing by creating subgoals
ICARUS	Planning through instantiation of skills
BDI	Predefined plans stored in the plan library
Subsumption	Implicit planning by task decomposition
CLARION	Planning by beam search in the Q-value space

3.6 Reasoning/Inference

Table 6 presents the reasoning or inference mechanisms in the six cognitive architectures. Both Soar and ACT-R architectures are built on production systems and thus they use production rules for reasoning. When a production rule matches with the current state, the rule execution leads to an action (Laird et al., 1986a; Lehman et al., 2006; Anderson et al., 2004). Soar has an additional reasoning mechanism, known as elaboration. During the elaboration phase, the relevant knowledge is brought into the working memory and preferences are created, giving suggestions to which operators to choose (Laird et al., 1986a). In contrast, ACT-R uses probabilistic reasoning in the matching of production rules, after activating the relevant chunks in the declarative knowledge (Anderson et al., 2004; Sun & Zhang, 2006). ICARUS, on the other hand, uses conceptual inference, which occurs in a bottom up, data driven manner. Concepts in ICARUS are represented as Boolean structure, which are matched in an all or none manner (Langley & Choi, 2006).

The BDI architecture makes use of the procedural reasoning system (PRS), which is used to select the appropriate plans for goal satisfaction (Georgeff & Ingrand, 1989). Again, a reasoning system is absent in the subsumption architecture (Bryson et al., 2005; Nakashima & Noda, 1998). CLARION combines both similarity-based and rule-based mechanisms to mimic the process of human reasoning. This works by comparing the similarities between chunks as part of the inference process (Sun & Zhang, 2006).

3.7 Learning

The learning mechanisms employed in the six architectures are summarized in Table 7. All architectures discussed contain some form of learning, except the behaviour-based subsumption architecture, wherein all the behaviours produced in reaction to the environmental cues are pre-wired into the design of the architecture (Brooks, 1999; Hartley & Pipitone, 1991).

Soar employs chunking as a general learning mechanism. When an impasse is encountered, a subgoal is created. Upon the successful resolution of the impasse, a chunk is created

Table 6: Reasoning and inference in the six cognitive architectures.

Architecture	Mechanism for Reasoning/Inference
Soar	Rule matching to bring relevant knowledge into working memory; Elaboration phase to create preferences
ACT-R	Probabilistic reasoning
ICARUS	Boolean match of conceptual clauses, bottom-up and data driven
BDI	Procedural reasoning system
Subsumption	Absence of reasoning mechanism
CLARION	Integrate rule-based reasoning and similarity-based reasoning

and added into the production memory as new knowledge (Laird et al., 1986a). Another learning mechanism incorporated into Soar is Q-learning, which is a classical reinforcement learning method (Watkins & Dayan, 1992). The architecture experientially learns an estimation of the long term rewards of performing an action in a given state and applies this estimation in the selection of actions (West, Stewart, Lebiere, & Chandrasekharan, 2005). Another two learning mechanisms used in Soar are episodic memory and semantic memory, which are recordings of specific events in episodes and non-contextual knowledge respectively. These two forms of memories are important in the resolution of impasse, enabling rule acquisition through chunking and reinforcement learning (Lehman et al., 2006).

Likewise in ICARUS, learning occurs when an impasse is reached. This allows the formation of new skills/knowledge that can be used for similar situations in the future (Langley & Choi, 2006). Learning takes place in the form of production compilation in ACT-R. This mechanism is somewhat similar to chunking in Soar. It involves a combination of various production rules into one, eventually leading to performance without the retrieval of instructions from the declarative memory (Anderson et al., 2004).

Although traditional BDI does not consider learning as part of the architecture, many learning methods, such as Q-learning, decision trees, and learning from interpretations, have been shown to work with BDI systems (Guerra-Hernandez et al., 2004).

As a hybrid system, learning in CLARION takes place at both the top and bottom layers. Learning of procedural knowledge in the bottom level of CLARION is by multi-layer neural networks and Q-learning, whereas learning of rules in the top level occurs through extracting rules from the bottom level (Sun & Peterson, 1996, 1998). This implies that CLARION also differentiates between implicit learning (in the bottom level) and explicit learning (in the top level). This dual-process approach is notably absent in other cognitive architectures.

Table 7: Learning in the six cognitive architectures.

Architecture	Mechanisms for Learning
Soar	Chunking for creation of new production rules Reinforcement learning for updating reward of each rule Episodic and semantic memory to aid in decision making
ACT-R	Production compilation for combining multiple production rules into one
ICARUS	Skill learning
BDI	Q-learning, top-down induction of decision tree Learning from interpretations
Subsumption	Absence of learning mechanism Reactions to environment pre-wired into each module
CLARION	Q-learning at bottom level (procedural knowledge) and rule extraction at top level (declarative knowledge)

3.8 Relevance to Neurobiology

Having their roots in artificial intelligence and cognitive psychology, Soar, ICARUS, and BDI do not make explicit references to neural anatomy (Table 8). In contrast, the ACT-R architecture makes extensive references of its components to specific regions in human brains. Most notably, the central production system and the memory modules in ACT-R are mapped to the basal ganglia and the various cortical areas respectively (Anderson et al., 2004).

Although CLARION is based on a hybrid neural architecture, there is no reference to specific neural substrates for its modules. Similarly, the subsumption architecture does not claim any significant biological connection, although its layered design is said to be consistent with that of a human nervous system. Most notably, the addition of new layers is similar to the development of new brain sections for new functions, leaving existing parts unaltered (Brooks, 1991).

4 Benchmarks and Applications

As shown in Table 9, all the six cognitive architectures reviewed have been applied to a wide variety of cognitive tasks and real-life applications. As one of the oldest models, Soar has been used in many problem solving tasks, including Eight Puzzle, the Tower of Hanoi, Fifteen Puzzle, Think-a-dot and Rubik Cube. Recently, the Soar team has also developed the Eater’s World, a task that requires the agent to make decisions of which directions to move (Nuxoll, Laird, & James, 2004; Nuxoll & Laird, 2004). The Eater’s World tests the agent’s navigation skills in response to its environment, with rewards in the form of food and bonus food.

Table 8: The relevance of the six cognitive architectures to neurobiology.

Architecture	Relevance to Neurobiology
Soar	No reference to brain anatomy
ACT-R	Used for prediction of brain activation pattern; Modules and production system are mapped to various brain regions
ICARUS	No reference to brain anatomy
BDI	No reference to brain anatomy
Subsumption	No reference to brain anatomy; Layered design is analogous to biological nervous systems
CLARION	Based on neural networks but no reference to brain anatomy

More importantly, Soar has been used by US Army, Navy, and Air Force to develop robots and software for the purposes of modelling, simulations, and control. For instance, TacAir-Soar and RWA-Soar are developed as training models for human pilots. Soar MOUTBOT is used for military operations on land (Lehman et al., 2006). Soar has also been applied in the form of embodied conversational agents (ECA). Some specific systems developed include Soar Training Expert for Virtual Environment (STEVE) and Mission Rehearsal Exercise (MRE). The former is a virtual agent, designed for teaching the operation and maintenance of the gas turbine engines on naval ships (Rickel & Johnson, 2000). On the other hand, MRE is developed to teach leadership skills for dealing with high stake social dilemmas through the use of virtual human technology (Swartout, Gratch, Hill, E., Marsella, Rickel, & Traum, 2006).

Ever since its development, ACT-R has been used as a framework for various cognitive tasks, such as the Tower of Hanoi, memory for text or lists of words, language comprehension, and communication. Other applications of ACT-R architecture include aircraft control as in the Anti-Air Warfare Coordinator and predictions of activation patterns in brain imaging studies (Anderson et al., 2004). In addition, ACT-R has been applied to develop education software, for predicting the students' areas of difficulties so that appropriate help can be provided. ACT-R has also been used in the investigation of human-computer interactions, such as ACT-R versus human player in scissor-paper-stone. West et al. (2005) show that ACT-R is able to capture significant portion of cognitive functions involved in human game playing .

Similar to Soar and ACT-R, ICARUS has been applied to many cognitive tasks, including the Tower of Hanoi, multi-column subtraction, and peg solitaire. Other key domains, which have been studied to date, include in-city driving and pole balancing (Langley, 2004).

BDI was originally designed for the diagnosis of the reaction control system (RCS) in the NASA's space shuttles (Georgeff & Ingrand, 1989). It has also been applied to the problems of factory process control and business process management (Guerra-Hernandez et al.,

Table 9: Benchmarks and Applications.

	Soar	ACT-R	ICARUS	BDI	Subsumption	CLARION
Puzzles and Cognitive Tasks	Tower of Hanoi, Eight Puzzle, Fifteen Puzzle, Think-a-dot, Rubik Cube	Tower of Hanoi, Memory of text, Language comprehension, Communication	Tower of Hanoi, Peg Solitaire, Multi-column, subtraction	Tower of Hanoi		Tower of Hanoi, Serial reaction time task, Grammar learning, Alphabetical arithmetic
ECA, Training and Diagnosis	Operation and maintenance of engines, leadership training			Museum guide, diagnosis of space shuttles		
Command control and virtual games	TacAir-Soar, RWA-Soar, MOUTBOT for US military, Eater's World	Aircraft control	In-City driving, Pole balancing	Factory process control, Business process management	Truckin' Game, airplane control	Maze and minefield navigation, process control
Robotics	Robotics for US military				Navigation, obstacle avoidance, robot soccer	

2004). The implementations of the BDI architecture include procedural reasoning system (PRS), dMARS, JACK, 3APL, Jason, JADEX, JAM, AgentSpeak(L), and UM-PRS. Besides performing cognitive tasks, such as the Tower of Hanoi, BDI has also been applied as embodied conversational agents through the creation of *Max*. As a guide in a public computer museum, *Max* is able to make small talk conversations with the visitors, while providing information about the museum (Kopp, Gesellensetter, Krämer, & Wachsmuth, 2005).

The subsumption architecture was originally proposed as a new robot design (Brooks, 1991, 1999) and has since been extensively applied. Unlike other cognitive architectures, this type of behaviour-based robots does not require any complex mathematical computation built into the robots. Many robots with different levels of navigation abilities have been built, including *Allen* that avoids both static and dynamic obstacles while wandering randomly; *Herbert* with the additional capability of stealing empty soda cans; and *Tom* and *Jerry* that are capable of heading towards distant places. Other robots that have been designed with the subsumption architecture are *Seymour* with vision ability; *Genghis*, a six-legged walking robot, and *ToTo* that navigates as if it has a built-in map (Brooks, 1991, 1999). There are also attempts to incorporate the subsumption architecture into software design, such as “reactive accompanist”. A reactive accompanist is a computer that can accompany unfamiliar melodies with musicians in real time without any knowledge of music theory or any form of rules (Bryson et al., 2005). The subsumption architecture is also employed in multi-agent systems, such as soccer match in RoboCup (Nakashima & Noda, 1998).

The subsumption architecture has also been applied in Truckin’ simulation game. The objective of the game is to ensure a steady flow of goods from the producers to the consumers through the trading by the retailers and the truck agents played by the architecture (Butler et al., 2001). Hartley and Pipitone (1991) have also attempted to use the subsumption architecture in airplane control.

CLARION has been used in both the simulation of navigation and cognitive tasks. Sun and Peterson (1996,1998) report the use of CLARION for navigation in simple mazes, as well as complex minefield navigation, wherein the agent has to navigate an underwater vessel through a minefield to reach a given target location. The cognitive tasks using CLARION include serial reaction tasks, artificial grammar learning tasks, process control tasks, alphabetical arithmetic tasks, and the Tower of Hanoi (Sun & Zhang, 2006).

Considering all the domains studied, it appears that the Tower of Hanoi and maze navigation tasks have been used across most architectures. In particular, the Tower of Hanoi has long been regarded as the classic paradigm for behaviour studies of goal manipulations and has been used as the benchmark cognitive task for testing many cognitive architectures. Maze navigation is also popularly chosen as one of the benchmark tasks simply because mobility is a key function of autonomous agents. In addition, navigation evaluates the

architecture's reactivity to the environment, which is an essential aspect of intelligence. It is also notable that navigation can be carried out readily in the subsumption architecture, which is less likely to be used for cognitive tasks such as the Tower of Hanoi.

5 Conclusions

The ultimate goal of studying integrated cognitive architectures is to build coherent systems that display a wide range of functions robustly across different problem settings. This is the path towards understanding human cognition and developing systems with human level intelligence. In this paper, we have reviewed six representative cognitive architectures and made comparison based on their functionalities and modules.

An integrated cognitive architecture should be applicable across different problem domains. Though the existing architectures have been increasingly applicable to many tasks, there remains a challenge to build one system capable of solving all types of problems in a real environment. For example, although some existing architectures have shown promising results in their computational modelling of the brain, the parameters defined in the computational models are based on empirical evidences, and therefore the models are only valid within the specific experimental ranges. Hence, it would be better if the architectures were applicable to universal situations so that the agent is able to solve problems in all kinds of novel situations, and not just confined to test situations.

In addition, many cognitive architectures focus on achieving the cognitive functions and give little emphasis to their biological validity. Although most cognitive architectures contain modules, such as problem solving, learning, and memory for storing knowledge, which are key processes and components in the human brain, they make little reference, if at all, to brain anatomy. The only significant attempt to do this is ACT-R. In this aspect, it appears that ACT-R has presented a relatively more balanced approach, by covering most basic functions of human cognition and identifying the modules in its architecture in an extensive scale to specific regions of human brains.

5.1 Converging Design Principles

All cognitive architectures surveyed in this paper contain certain features which make them unique. They however do bear important similarities in their principles and approaches. Specifically, problem solving, reasoning/inferencing, and learning are essential components of most cognitive systems, with the exception of the subsumption architecture.

One fundamental construct found in many cognitive architectures is the use of short term working memory, production rules and a decision process. The use of short term working memory is critical to hold the relevant input from the environment and the knowledge from the long term memory for the selection of appropriate actions. Most cognitive

architectures rely on short term memory, which is known as working memory in Soar, memory buffers in ACT-R and short term conceptual memory in ICARUS. All cognitive architectures except the subsumption architecture also make use of some forms of rules to aid in selecting the appropriate courses of action. For instance, the rules in CLARION are stored in the top level of the architecture, while the production rules in ACT-R are available as procedural knowledge. The decision processes in Soar, ACT-R and ICARUS are also similar, which involve the selection of skills/actions based on the desired goals and the information available in the short term memory.

Most of the architectures, in particular, Soar, ACT-R, ICARUS, and CLARION, make a distinction between declarative and procedural knowledge. While declarative knowledge contains the facts and inference rules required for reasoning, procedural knowledge encodes the sequences of action to perform in response to specific situations. One promising addition to some of the architectures, such as CLARION, is the inclusion of other forms of memory, for instance, episodic memory. These additions would enhance the system complexity and are more likely to capture the features of human cognition. After all, we make use of various types of memories and processes to reason, function, and learn.

5.2 Promising Research Trends

With the high level of interests and extensive amount of work done in the field of integrated cognitive architecture, we expect to see more exciting results in the next one to two decades. To conclude this paper, we highlight four research trends below, which in our opinion are promising in the advancement of the state of the art. Many of these directions, especially interaction-based learning and biologically-inspired cognitive architectures, are being actively pursued as we speak.

1. **Real-time Embodiment:** A cognitive agent needs to be situated. Some of the architectures such as BDI, the subsumption architecture and CLARION are designed to remain responsive even in a dynamic environment, therefore enabling situatedness. These architectures are able to sense changes in the environment, as well as respond rapidly to the changes. Although most cognitive architectures include a vision or perceptual buffer, none has been integrated with a machine vision system. There has been much progress made in the field of computer vision. We are eager to witness the development of truly embodied intelligent systems, that are capable of sensing and interacting with the environment in real time, just like human beings.
2. **Interaction based learning:** Related to the issue of embodiment, a cognitive agent should be able to learn and function in a real-time dynamic environment. Under the notion of embodied cognition (Anderson, 2003), an agent acquires its intelligence through interaction with the environment. Reinforcement Learning (Sutton & Barto,

1998; Si, Barto, Powell, & Wunsch, 2004) is a field that has received intensive research effort, but has not been incorporated into many cognitive architectures in a major and principled manner. In recent years, a family of self-organizing neural models, known as fusion Adaptive Resonance Theory (fusion ART), has been steadily developed (Tan, 2004, 2007; Tan, Carpenter, & Grossberg, 2007). By expanding the original ART model consisting of a single pattern field into a multi-channel architecture, fusion ART unifies a number of network designs supporting a myriad of interaction based learning paradigms, including unsupervised learning, supervised learning and reinforcement learning. A specific instantiation of fusion ART, known as Temporal Difference-Fusion Architecture for Learning and Cognition (TD-FALCON), has shown to produce competitive learning capabilities, compared with gradient descent based reinforcement learning systems (Xiao & Tan, 2007; Tan, Lu, & Xiao, 2008). Such models may provide a promising approach to designing cognitive systems for functioning and learning in real-time environment.

3. **Evaluation:** Although the various architectures have been applied to a wide range of benchmarks and applications, their implementations, with the exception of Soar, are generally not available in the public domain. As such, it is difficult to compare the capabilities of the various architectures across common tasks. In addition, there is a lack of standard benchmarks and problem domains for comparing the various systems side by side at the system level. To date, Robocup and in-city driving are some of the best known domains for such purposes. However, the tasks are still specific and success could be achieved through customized designs in place of generic principles. As such, it is essential to have a collection of general domain tasks, each demanding a variety of cognitive functionalities, based on which the various architecture can be evaluated.
4. **Biologically-inspired Architectures:** One key objective of developing integrated cognitive architectures is to mimic and explain human cognition. It is thus meaningful for a good architecture to fulfil both biological and psychological validity. As of today, most cognitive architectures have already displayed psychological validity. For example, ACT-R has originally been a computational model based in psychology findings. Soar and ICARUS have also attempted to show that they contain features that coincide with psychological findings. However, despite that some architectures, such as ACT-R, have made references to specific brain regions, very few have restricted themselves to use only neurally plausible mechanisms for implementing their functions. Indeed, while powerful constructs, like problem solving and goal-subgoal hierarchy, have been available for a long time, little is known how to achieve the same level of capabilities in a biological based architecture. The answer to this question could well be the key to the next generation integrated cognitive architectures.

Acknowledgments

This survey work is supported in part by the DSO National Laboratories and Nanyang Technological University under the DSO-Undergraduate Research Experience on Campus (DSO-URECA) programme. The authors thank the anonymous reviewers for the useful comments and suggestions.

References

- Amir, E., & Maynard-Zhang, P. (2004). Logic-based subsumption architecture. *Artificial Intelligence*, *153*, 167–237.
- Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., & Qin, Y. (2004). An intergrated theory of the mind. *Psychological Review*, *111*, 1036–1060.
- Anderson, J., & Schunn, C. (2000). Implications of the ACT-R learning theory: No magic bullets. *Advances in Instructional Psychology*, *5*, 1–34.
- Anderson, M. (2003). Embodied cognition: A field guide. *Artificial Intelligence*, *149*, 91–130.
- Bratman, M., Israel, D., & Pollack, M. (1988). Plans and resource-bounded practical reasoning. *Computational Intelligence*, *4*(4), 349–355.
- Braubach, L., Pokahr, A., Moldt, D., & Lamersdorf, W. (2005). Goal representation for BDI agent systems. In *Proceedings, 2nd International Workshop on Programming Multiagent Systems: Languages and Tools*, pp. 9–20.
- Brooks, R. A. (1991). How to build complete creatures rather than isolated cognitive simulators. In *Proceedings, Architectures for Intelligence*, pp. 225–240.
- Brooks, R. (1999). *Cambrian Intelligence: The Early History of the New AI*. Boston, MA: MIT Press.
- Bryson, J., Smaill, A., & Wiggins, G. (2005). The reactive accompanist: Applying subsumption architecture to software design. Tech. rep., Department of Artificial Intelligence, University of Edinburgh.
- Butler, G., Gantchev, A., & Grogona, P. (2001). Object-oriented design of the subsumption architecture. *Software - Practice and Experience*, *31*, 911–923.
- Chong, R. S. (2004). Architectural explorations for modeling procedural skill decay. In *Proceedings, Sixth International Conference on Cognitive Modeling*.

- Dastani, M., Hulstijn, J., & van der Torre, L. (2001). BDI and QDT: A comparison based on classical decision theory. In *Proceedings, AAAI Symposium*, pp. 16–26.
- Dastani, M., & van der Torre, L. (2002). An extension of BDI_{CTL} with functional dependencies and components. In *Proceedings, 9th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, pp. 115–129.
- Dennett, D. (1987). *The Intentional Stance*. MIT Press, Cambridge, MA.
- Georgeff, M., & Ingrand, F. (1989). Decision-making in an embedded reasoning system. In *Proceedings, International Joint Conference on Artificial Intelligence*, pp. 972–978.
- Guerra-Hernandez, A., Fallah-Seghrouchni, A. E., & Soldano, H. (2004). Learning in BDI multi-agent systems. In *Proceedings, Fourth International Workshop on Computational Logic in Multi-Agent Systems, Fort Lauderdale, FL*.
- Hartley, R., & Pipitone, F. (1991). Experiments with the subsumption architecture. In *Proceedings, IEEE International Conference on Robotics and Automation*, pp. 1652–1658.
- Köse, H. (2000). Towards a robust cognitive architecture for autonomous mobile robots. Master’s thesis, Boğaziçi University.
- Kopp, S., Gesellensetter, L., Krämer, N., & Wachsmuth, I. (2005). A conversational agent as museum guide - design and evaluation of real-world application. *Intelligent Virtual Agents*, 5, 329–343.
- Laird, J., Newell, A., & Rosenbloom, P. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33(1), 1–64.
- Laird, J., Rosenbloom, P., & Newell, A. (1986a). Chunking in Soar: The anatomy of a general learning mechanism. *Machine Learning*, 1, 11–46.
- Laird, J., Rosenbloom, P., & Newell, A. (1986b). *Universal Subgoaling and Chunking: The Automatic Generation and Learning of Goal Hierarchies*. Boston, MA: Kluwer.
- Langley, P. (2004). A cognitive architecture for physical agents.. Retrieved October 28, 2006 from <http://www.isle.org/langley/talks/icarus.6.04.ppt>.
- Langley, P., Arai, S., & Shapiro, D. (2004). Model-based learning with hierarchical relational skills. In *Proceedings, ICML-2004 Workshop on Relational Reinforcement Learning*.
- Langley, P., & Choi, D. (2006). A unified cognitive architecture for physical agents. In *Proceedings, Twenty-First National Conference on Artificial Intelligence*, pp. 1469–1474.

- Lehman, J., Laird, J., & Rosenbloom, P. (2006). A gentle introduction to soar, an architecture for human cognition: 2006 update.. Retrieved May 17, 2007 from <http://ai.eecs.umich.edu/soar/sitemaker/docs/misc/GentleIntroduction-2006.pdf>.
- Madden, M., & Howley, T. (2003). Transfer of experience between reinforcement learning environments with progressive difficulty. *Artificial Intelligence Review*, 21(3-4), 375–398.
- Nakashima, H., & Noda, I. (1998). Dynamic subsumption architecture for programming intelligent agents. In *Proceedings, IEEE International Conference on Multi-Agent systems*, pp. 190–197.
- Newell, A. (1990). *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.
- Norling, E. (2004). Folk psychology for human modelling: Extending the BDI paradigm. In *Proceedings, International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'04)*, pp. 202–209.
- Nuxoll, A., & Laird, J. (2004). A cognitive model of episodic memory integrated with a general cognitive architecture. In *Proceedings, Sixth International Conference on Cognitive Modeling*, pp. 220–225.
- Nuxoll, A., Laird, J., & James, M. (2004). Comprehensive working memory activation in soar. In *Proceedings, Sixth International Conference on Cognitive Modeling*, pp. 226–230.
- Rao, A. S., & Georgeff, M. P. (1991). Modeling rational agents within a bdi-architecture. In *Proceedings, Second International Conference on Principles of Knowledge Representation and Reasoning*, pp. 473–484. Morgan Kaufmann, San Mateo, CA.
- Rickel, J., & Johnson, W. (2000). Task-oriented collaboration with embodied agents in virtual worlds. In Cassell, J., Sullivan, J., & Prevost, S. (Eds.), *Embodied Conversational Agents*, pp. 95–122. Boston, USA: MIT Press.
- Sardina, S., de Silva, L., & Padgham, L. (2006). Hierarchical planning in BDI agent programming languages: A formal approach. In *Proceedings, Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 1001–1008.
- Si, J., Barto, A. G., Powell, W. B., & Wunsch, D. (Eds.). (2004). *Handbook of Learning and Approximate Dynamic Programming*. Wiley-IEEE Press.
- Stollberg, M., & Rhomberg, F. (2006). Survey on goal-driven architectures. Tech. rep., DERI Austria.

- Subagdja, B., & Sonenberg, L. (2005). Learning plans with patterns of actions in bounded-rational agents. In *Proceedings, 9th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES'05)*, Vol. 3, pp. 30–36.
- Sun, R. (2003). A tutorial on CLARION 5.0. Tech. rep., Cognitive Science Department, Rensselaer Polytechnic Institute.
- Sun, R., Merrill, E., & Peterson, T. (2001). From implicit skills to explicit knowledge: a bottom-up model of skill learning. *Cognitive Science*, *25*(2), 203–244.
- Sun, R., & Peterson, T. (1996). Learning in reactive sequential decision tasks: The CLARION model. In *Proceedings, IEEE International Conference on Neural Networks*, pp. 1073–1078.
- Sun, R., & Peterson, T. (1998). Hybrid learning incorporating neural and symbolic processes. In *Proceedings, IEEE International Conference on Fuzzy Systems*, pp. 727–732.
- Sun, R., & Sessions, C. (1998). Learning to plan probabilistically from neural networks. In *Proceedings of IEEE International Conference on Neural Networks*, pp. 1–6.
- 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*(1), 159–192.
- Sun, R., & Zhang, X. (2006). Accounting for a variety of reasoning data within a cognitive architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, *18*(2), 169–191.
- Sutton, R., & Barto, A. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.
- Swartout, W., Gratch, J., Hill, R., E., H., Marsella, S., Rickel, J., & Traum, D. (2006). Toward virtual humans. *Artificial Intelligence Magazine*, *27*(2), 96–108.
- Tan, A.-H. (2004). FALCON: A fusion architecture for learning, cognition, and navigation. In *Proceedings, International Joint Conference on Neural Networks*, pp. 3297–3302.
- Tan, A.-H. (2007). Direct code access in self-organizing neural networks for reinforcement learning. In *Proceedings, International Joint Conference on Artificial Intelligence (IJCAI'07)*, pp. 1071–1076.
- Tan, A.-H., Carpenter, G. A., & Grossberg, S. (2007). Intelligence through interaction: Towards a unified theory for learning. In *Proceedings, International Symposium on Neural Networks (ISNN'07), LNCS4491*, pp. 1098–1107.
- Tan, A.-H., Lu, N., & Xiao, D. (2008). Integrating temporal difference methods and self-organizing neural networks for reinforcement learning with delayed evaluative feedback. *IEEE Transactions on Neural Networks*, *9*(2), 230–244.

- Toal, D., Flanagan, C., Jones, C., & Strunz, B. (1996). Subsumption architecture for the control of robots. In *IMC-13*.
- 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(2), 151–180.
- Watkins, C., & Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3/4), 279–292.
- West, R., Stewart, T., Lebiere, C., & Chandrasekharan, S. (2005). Stochastic Resonance in Human Cognition: ACT-R versus Game Theory, Associative Neural Networks, Recursive Neural Networks, Q-learning, and Humans. In *27th Annual Meeting of the Cognitive Science Society*.
- Wooldridge, M. (1996). A logic for BDI planning agents. In *Working Notes, 3rd Model Age Workshop: Formal Models of Agents*.
- Wray, R., Chong, R., Phillips, J., Rogers, S., & Walsh, B. (1994). A survey of cognitive and agent architectures.. Retrieved January 28, 2007 from <http://ai.eecs.umich.edu/cogarch0/>.
- Xiao, D., & Tan, A.-H. (2007). Self-organizing neural architectures and cooperative learning in multi-agent environment. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 37(6), 1567–1580.