Advances in Cognitive Systems 2 (2013) 277-296 Submitted 12/2012; published 12/2013

Systematic Integration of Cognitive and Robotic Architectures

Matthias Scheutz MATTHIAS.SCHEUTZ@TUFTS.EDU
Human-Robot Interaction Laboratory, Department of Computer Science, Tufts University, Medford, MA 02144 USA

Jack Harris JACKHARR@INDIANA.EDU
Air Force Research Laboratory, Wright-Patterson Air Force Base, OH 45433 USA

Paul Schermerhorn PSCHERME@INDIANA.EDU
Human-Robot Interaction Laboratory, Department of Computer Science, Tufts University, Medford, MA 02144 USA

Abstract

Originally, progress towards the AI goal of building artificial agents with human-like intelligence was best seen in cognitive architecture research that focused on developing complete agents in a systematic, theory-driven way. Later, research in embodied AI and robotics turned away from this focus on higher-level cognition in favor of making robots robustly achieve simple tasks in the real world. The ensuing hiatus between *cognition-focused* and *action-focused* research perspectives is still reflected in cognitive and robotic architectures today. In this paper, we attempt to reunite the two views by introducing a theoretically motivated, generic interface between cognitive and robotic architectures. From this integration the advances in both cognitive and robotic architectures can be leveraged to produce more complex adaptive robotic behavior. We start by reviewing the differences between cognitive and robotic architecture, followed by a comparison of two alternative methods for integrating such architectures. As a result of this comparison, we propose a three-part interface framework for architecture integration. We then report two specific instances of the interface for integrating the ICARUS and ACT-R cognitive architectures into the robotic DIARC architecture, along with proof-of-concept implementations with two sets of knowledge structures for executing a simple office environment exploration task using a Pioneer P3-AT robot. We describe qualitative evaluations of both integrated architectures and discuss directions for future research with the proposed framework.

1. Introduction

Architectures for intelligent robots have improved steadily over the years. A key contribution to this advancement is the inclusion of more diverse components that let robots negotiate increasingly complex indoor and outdoor environments. As a result, current robot architectures integrate multiple sophisticated algorithms for real-time perceptual, planning and action processing, from 3D object recognition, to simultaneous localization and mapping, to navigation and task planning, to action sequencing. But typically, robotic architectures lack components for high-level cognition such as

general-purpose reasoning and problem solving. Similarly, although some cognitive architectures have been applied to robotic domains (e.g., Hanford et al., 2009; Trafton et al., 2005), most focus on understanding higher-level cognitive processes in non-robotic domains.

Yet cognitive architectures have much to offer for many robotic applications, in particular, mixed-initiative scenarios with human teammates or domains where online problem solving is required as part of the task. For example, being able to solve problems during task execution that have not been anticipated by the agent designer can make the difference between mission success and mission failure, in addition to giving robots much more flexibility in coping with unforeseen events. Furthermore, being able to model human performance, a focus of much cognitive architecture research, and to use those models in decision-making and behavior coordination can lead to better mixed-initiative teams and improve general human-robot interactions. Similarly, robotic architectures can add important functionality and insights regarding real-world perception and action to cognitive architectures such as learning about new objects and what actions can be performed on them.

While the above examples are by no means intended to be exhaustive, they should provide a flavor of the potential gains in functionality produced from a cognitive-robotic architecture integration. We will thus take the utility of such integration for granted and focus on the methodology and technical requirements needed for realizing it. For building integrated agents is a non-trivial effort given the requirement for harmonizing disparate data representations and balancing control flow issues between different subcomponents of the integrated system.

In this paper, we address the challenge of integrating cognitive and robotic architectures by providing a theoretically motivated and practically guided *generic interface* that will let both cognitive and robotic architecture developers utilize the strengths of each paradigm while retaining the overall individual architectural structures. In Section 2, we briefly review the design assumptions of cognitive architectures and compare them to those made by robotic architectures. Section 3 examines two possible integration methodologies, from which we derive a generic interface, described in Section 4, that bridges the gap between the two architecture types and allows for various levels of integration. In Section 5, we provide the details of how the interface can be applied to integrate two cognitive architectures, ICARUS (Langley et al., 2009a) and ACT-R (Anderson et al., 2004), with the DIARC robotic architecture (Scheutz et al., 2007). We also give examples from proof-of-concept implementations of both integrated architectures on an autonomous robot that must perform an exploration task for which we provide relevant knowledge. Section 6 discusses directions for future work, and Section 7 summarizes our accomplishments.

2. Background

Research in cognitive architectures has from the beginning attempted to provide a *unified theory of intelligence* (e.g., Laird et al., 1987; Anderson et al., 2004). Because cognitive architectures aim at generality, they provide powerful representational formalisms to encode general problems and reasoning mechanisms that use these data structures. They have also been used to model human performance on tasks of varying complexity, from tasks using simple reaction time paradigms to more complex tasks requiring social interactions.

Cognitive architectures typically differ with respect to the layout (the processing components and "control flow" among them), representational schemes (to encode and represent knowledge), and architectural processes for inference and learning (e.g., see Langley et al., 2009b or Chong et al., 2007 for details). However, there are at least four important commonalities or "core commitments" among (major) cognitive architectures: (1) percepts are given as discrete complex data types, often corresponding to or "representing" objects in the environment together with their properties; (2) goals are explicitly represented; (3) behaviors are represented by discrete actions with parameters; and (4) processing occurs in "cognitive cycles" in which percepts are processed first, followed by internal processing, and subsequent (external or internal) action selection. Alas, even though some architectures allow for parallel processing at different stages or in different components (e.g., EPIC Meyer & Kieras, 1997 or PRODIGY Carbonell et al., 1991), the "cognitive cycle" is generally sequential, even though it may or may not take a fixed amount of time (e.g., 50 ms default in ACT-R).

The core commitments of cognitive architectures, regardless of their theoretical motivation, present significant problems for integration with robotic architectures, which usually do not share them and in some cases were specifically designed to avoid them (Brooks, 1986). For example, a robotic architecture might not employ perceptual systems that can generate discrete abstractions from continuous sensory data streams. Nor might the robotic architecture have the symbolic repertoire for explicit goal representations, let alone mechanisms for systematically reasoning about ways to best achieve a set of goals. Finally, the robotic architecture might not support discrete actions that can be scheduled and executed, but might rather generate behaviors through continuous activation and interaction of low-level control components, such as continuous time feed-back controllers.

Additional integration complications arise from the lack of time sensitivity in many cognitive architectures, where either the duration of a cognitive cycle is undetermined or the cognitive timing is not tied to real time in the implementation. In robotic applications, the lack of adherence to a carefully timed update regimen can cause unwanted side effects, such as the architecture reasoning about how to best avoid a detected obstacle without stopping the robot's motors in time, causing the robot to crash into the obstacle. And even when the timing of a cognitive cycle is theoretically specified (as in ACT-R), it is not always clear whether simulation time will map to real-time performance. For example, the LISP implementation of ACT-R uses a variable amount of processing time depending on the size of its declarative and procedural knowledge.

3. Two Routes for Integration

Any integration method for cognitive and robotic architectures must address at the least the four core commitments of cognitive architectures and the associated integration challenges: discrete perceptual representations, explicit goal representations, discrete actions, and cycle-based as opposed to real-time processing. While the first three can, to some extent, be handled by "transducer components" that effect a mapping between low-level robotic representations and high-level cognitive representations, it would be very difficult, if not impossible, to instill rigorous temporal awareness into cognitive architectures. In fact, a tight timing regimen might require a complete reconceptualization of the architecture and its core cognitive cycle, which would alter and likely reduce

functionality. Such a reorganization, however, would fail to recognize the strengths of cognitive architectures, namely their ability to provide processes like general goal management, general factual and counter-factual inference, and general problem solving that do not require exact timing or close coupling to real-world events. Thus, rather than imposing an overarching timing scheme to synchronize activities between cognitive and robotic architectures, a successful integration should find ways for both architectures to exchange information subject to their respective strengths and constraints.

There are two obvious routes to integrating these types of frameworks: (1) to make the robotic architecture an extension of the cognitive architecture; or (2) to make the cognitive architecture an extension of the robotic architecture. While both approaches can generate operational architectures for agents operating in real-world contexts, they are not symmetrical due to the different processing commitments: cognitive architectures are intrinsically sequential at least with respect to the cognitive cycle, while robotic architectures are typically highly parallel, even though they may be agnostic about the temporal sequence of updates. This difference in processing style has important consequences for choosing an appropriate integration, as we discuss next, because only the latter integration approach can realize the utility of both architecture types fully.

3.1 Robotic Architecture as an Extension of Cognitive Architecture

The first type of integration is based on the idea that the robotic architecture (RA) can be viewed as a "transducer" connecting the perceptual and motor systems of the cognitive architecture (CA) to the real-world sensors and effectors of an embodied agent. The RA's job is to provide percepts to the CA in a way the CA can handle and to accept primitive directives issued by the CA to control real-world effectors. The CA can operate as it normally would and development of factual knowledge and rules is confined to the CA without the need to make modifications in the RA. The cognitive modeling community has often resorted to this approach when connecting a CA to a synthetic task environment. For example, a primitive action in the CA like "jump" will issue a command in the simulated world to make the agent jump.

While this method of integration is very intuitive from a computational modeling perspective and requires minimal changes in the typical model development process for cognitive architectures, it imposes important design commitments and functional limitations. For example, all actions carried out by the (robotic) agent must be initiated within the CA during a cognitive cycle. Even though this might be acceptable for actions in which the exact time course does not matter much (e.g., when the agent starts to pick up an object), it will be insufficient for cases that require fast reactions (e.g., when the object slips and the robot must catch it). In general, adherence to a strict sense-think-act paradigm inevitably prevents RA subsystems from operating in parallel and at the speed required to react properly to rapidly changing environments. Since the CA does not make commitments to real-time processing, it can consider outdated percepts that no longer exist in the RA and use them for decision-making. The fact that information flow and exchanges between CA and RA are dominated by the CA's modus operandi can cause an otherwise functional RA to become dysfunctional.

3.2 Cognitive Architecture as a Subsystem of Robotic Architecture

The main alternative integration approach is to use the CA as a subsystem of the RA in which all subsystems are free to operate concurrently at their respective paces. As opposed to integrations in which the CA dictates the timing, makes decisions, and initiates actions, here action decisions can be made by both architectures. And whether CA decisions have precedence over RA decisions depends on the particular RA and how it handles behavior arbitration and action selection (Scheutz & Andronache, 2004). For example, in a subsumption-style RA where behavior arbitration is hierarchical, the CA could be placed anywhere in the hierarchy, and accordingly might trump some other components, but not all of them.

While this method of integration can preserve important functional properties of the RA such as action selection and reactivity, it has problems of its own that result from a loss of cognitive control. For example, the CA can no longer always control actions, nor is it necessarily even informed about all actions that are taking place. A typical case in point are low-level reactive (involuntary) tasks (e.g., obstacle avoidance), since the RA operates routinely unbeknownst to the CA (at a "subconscious" level). As a result, the CA might issue commands that conflict with actions being executed by the RA and, if not properly handed, this could result in dysfunctional behavior.

Moreover, the CA may not receive percepts that lead to actions by the RA, say because the sensory information is never elevated to a higher-level representation, or the information is never passed on. The fact that the RA can operate in parallel and somewhat independently from the CA makes the development of knowledge structures for the cognitive parts of the integrated architecture more challenging, as they must take into account interdependencies with the RA in perceptual processing, decision making, and action execution.

4. A Generic Robotic-Cognitive Interface

The above discussion of the advantages and disadvantages of alternative integration methods showed that there is no clear-cut answer on how to proceed. For one, which integration is preferable will depend on the domains and tasks that confront the integrated architecture. For example, a robotic-inside-cognitive integration might be the best option if the main aim is to develop computational models that replicate human performance in a cognitive task, especially when the details of perceptual and action processing, including the exact timing of those processes, are not critical. Such an integration will also reduce the effort involved in developing knowledge structures (facts, rules, etc.) for tasks and domains. Under this structure, many involuntary, fast reaction-based subsystems provided by a robotic architecture would not be available to the integrated architecture.

On the other hand, if the idea is to build real-world operational, intelligent robots, then a robotic-inside-cognitive integration might not work due to timing problems imposed by the sequential processing bottleneck, as well as the limitations imposed by prescribed knowledge structure formats in CAs. In the worst case, an integration might require a complete redesign of the CA, which would essentially force it to either adapt or abandon fundamental theoretical design commitments altogether (e.g., changing the commitment of a 50ms cognitive cycle execution in order to achieve fast reflexive behavior). Rather, by integrating the CAs into the RAs as subsystems, we can retain the advantages of their cognitive functionality while letting the RA handle time-critical and otherwise

sub-cognitive functions. As a result, the CA can serve multiple functional roles in the integrated architecture, from general problem solving, decision making, and goal management, to providing mental models of human teammates and simulations of their possible behaviors that can improve human-robot interactions.

4.1 Three Interfaces for Cognitive-Inside-Robotic Integration

Although the details of a cognitive-inside-robotic integration will necessarily depend on the specifics of the two architectures, we can specify generic interfaces, processes, and representations that aid this integration and address many of the challenges. Specifically, based on the four core commitments in cognitive architectures (introduced in Section 2), we can define three generic high-level interfaces between robot and cognitive architectures.

4.1.1 The Perceptual Interface

Robotic architectures deal with a variety of low-level streaming sensory data, including data from range finders, RGB and RGB-D readings from 2D and 3D vision sensors, sound information from microphones or microphone arrays, GPS or other localization information, data from gyroscopes, force-based sensors, and others. Passing on these data without pre-processing would quickly overwhelm a functional cognitive architecture, aside from neglecting to achieve the level of abstraction for percepts that cognitive architectures assume. Three mechanisms are needed to make this interface work, which we discuss in turn. First, we must develop a set of common perceptual primitives that robotic architectures can deliver and cognitive architectures can process. Most importantly, we need a representation of a generic "object" with its associated perceivable properties, such as type, size, color, texture, location; the details of the attributes in the object representation will depend on the robotic architecture's perceptual processing capabilities. For example, the following list structure could represent a percept for a doorway two meters straight ahead of the robot that was generated by a component in the robotic architecture from laser readings:

```
(object o7 :type doorway :distance 2 :angle 0)
```

Note that "o7" is a unique identifier associated with a particular doorway. Cognitive architectures typically use such identifiers for perceptual instances to track particular objects over time.[1] Also note that the particular syntactic incarnation of generic objects will depend on the representational format employed by the cognitive architecture such as chunks in ACT-R).

Second, the cognitive architecture must select the percepts that interest it such as those relevant to a particular task or goal. For this purpose, we can employ a top-down *attention mechanism* via a specialized "attend request" that passes object types from the cognitive to the robotic architecture (possibly including additional constraints such as attending to objects of a particular size only).

1. The tracking of and thus identification via percepts of an object over time is a very complex problem that requires contributions from both robotic and cognitive architectures. The robotic architecture can provide low-level perceptual tracking that can track objects even through short-term occlusions and other interruptions, while the cognitive architecture can use context and knowledge-based constraints to *infer* that two percepts with different identifiers forwarded by the robotic architecture denote the *same* object. We will briefly return to this issue in Section 6.

Finally, the robotic architecture needs an interrupt mechanism that can override the cognitive top-down attention mechanism and provide percepts that it deems critical, thus implementing a bottom-up form of attention. For example, it may need to provide percepts that are critical for the system's safety or other "global alarm mechanisms" (Sloman, 1998).

The generic object representation together with the two attentional mechanisms constitutes the core perceptual interface. Additional information about the nature of the stimulus and the processing component could be added, such as time stamps for each percept or information of the sensor sources. The former allows for temporal sequencing of information as needed for episodic storage in the cognitive architecture or high-level multi-modal fusion, while the latter can let the cognitive architecture reason about expectations regarding sensory inputs, such as detecting that sensors are not working properly.

4.1.2 The Goal Interface

Typically, cognitive architectures need explicit maintenance or achievement goals to perform actions, as action-oriented structures are selected based on the goal context. Usually, these goals are set during the initialization of the architecture. However, this is not ideal in the robotic context because goals might come through sensory channels such as spoken natural language (e.g., Scheutz et al., 2011) or be determined by other parts of the robotic architecture (e.g., the battery charge level monitor). For example, the vision system might request a change of camera angle or perspective to get better view of a scene, which, in turn, might require the robot to stop moving (e.g., Krause et al., 2012) or the robot might have to change its location to allow for better voice processing. Hence, ways to exchange goal information between the architectures are needed. This includes sending new goals from the robotic to the cognitive architecture and allowing the cognitive architecture to reject them. It also includes monitoring the goal status in the cognitive architecture by the robotic architecture and permitting the possible modification of goal parameters (e.g., goal priorities or timeouts). If the cognitive architecture provides a special mechanism for submitting goals (e.g., a goal buffer), similar mechanisms as in the perceptual case (for percept and attention) are possible. Otherwise, extensions to the cognitive architecture are needed to enable goal communication.

4.1.3 The Action Interface

For the cognitive architecture to effect robot behavior, primitive actions defined in the former must be passed to the robot architecture. These include voice outputs, changes to camera orientation and motor actions for locomotion and manipulation. Furthermore, some actions might map onto multiple components in the robotic architecture, requiring complex parallel and sequential control, such as picking up and transferring a filled mug from one place to another, which for the cognitive architecture might be a "primitive". The overall information exchange can, as with perceptions and goals, either be accomplished via an existing interface (e.g., motor buffer), or by implementing special communication functions for each primitive action in the cognitive architecture (e.g., issuing a primitive action would then amount to sending information from the cognitive to the robotic architecture). As with goals, the action status must be accessible to the cognitive architecture, since cognitive architectures are typically not sensitive to the time duration of action execution.

This can be accomplished in the robotic architecture by responding to action commands with the corresponding action status. For example, if an action command is issued for the first time, the robotic architecture would report the action status "starting", while if the action is issued repeatedly, the action status would be "ongoing" if it has not completed yet. The cognitive architecture can then use this information to determine whether the robotic architecture is still working on the original action or whether a new action has been initiated. In some cases, the original action must be canceled, hence mechanisms for canceling actions and modifying action parameters are needed as well. One special status is that of action failure. Knowledge within a cognitive architecture could find action failures (and their causes if reported) useful for selecting what course of action to take next. Therefore, this action status message must accommodate failure types, such as when mutually exclusive actions were issued or when the action selection mechanism in the robotic architecture arbitrated an action issued by the cognitive architecture. By granting the cognitive architecture full introspective access to action execution, one can implement extended mechanisms for action selection on top of existing ones in the robotic architecture. At the same time, this introspective access provides a simple implicit timing mechanism for handling real-world timing constraints otherwise unknown to the cognitive architecture.

4.2 Levels of Integration

There are at least three ways of implementing the above interfaces to effect a cognitive-inside-robotic integration, each yielding an increasingly tight level of integration at the expense of increased integration effort. The first and loosest integration (L1) utilizes a special purpose "CA wrapper component" that can be defined in the robotic architecture. This new RA subcomponent "wraps" the cognitive architecture and directly implements the three interfaces for perceptions, goals, and actions. This component gathers perceptual information from different parts in the robotic architecture (e.g., the laser-based and vision-based subsystems) and passes them to the cognitive architecture. It similarly distributes action-based directives from the cognitive architecture to the appropriate parts in the robotic architecture (e.g., the navigation-based and manipulation-based action primitives).

One advantage of this integration is that the wrapper component can be developed in the programming paradigm used for the robotic architecture and seamlessly integrated into it, using the same robotic middleware and possibly without requiring any changes to existing components. Another advantage is that information from multiple perceptual components can be combined in the wrapper component, allowing for synchronized multi-modal representations. The main disadvantages are that the robotic architecture will lose some of the robustness it achieves via distributed components and that the wrapper component may create a serial processing bottleneck. However, the latter is probably not critical given that cognitive architectures operate at relatively slow cycles compared to the sensory data streams processed by robotic architectures. Thus, the high-level percepts gathered in the wrapper component will be orders of magnitude smaller than the raw sensory data from which they are abstracted.

The second integration method (L2) provides a deeper, more distributed integration by implementing direct connections between perceptual and action processing components in the robotic architecture and the perceptual and action interfaces in the cognitive architecture (e.g., perceptual and motor buffers). Here, each relevant component in the robotic architecture must be augmented

by communication channels that can exchange information with the cognitive architecture directly. The main advantages of this integration method are that individual components in the former can directly connect to corresponding perceptual buffers in the latter and that the former's distributed nature is retained without a centralized bottleneck; overall communication within it is not increased. A disadvantage of this integration, in addition to larger changes required at the code level, is that the explicit timing and proper integration of information from multiple sources is no longer guaranteed as information will be delivered asynchronously. Another is that explicit synchronization mechanisms in the cognitive architecture might have to be introduced.

The third method (L3) provides an even deeper level of integration that ultimately results in a complete absorption of the cognitive architecture inside the robotic one. Functional modules in the former are systematically extracted and turned into separate components that can be run in the robotic architecture's middleware, thus becoming new components. For example, the production system of a cognitive architecture could become a separate component of the robotic architecture. The procedural and declarative memory systems could even be further separated and the rule-based system might be parallelized and distributed (e.g., Andronache & Scheutz, 2004). The main advantage of this integration is that core components of the cognitive architecture become full elements of the robotic architecture, which should lead to improved reactivity and temporal awareness due to increased parallelism. The main disadvantages, aside from the effort of parallelizing and distributing functional components, are that important assumptions about the operation of the cognitive architecture might be lost and that interfacing these new distributed modules would no longer resemble the typical programming style of cognitive architectures.

4.3 Implementation Considerations

The level of integration and the details of implementing the three generic interfaces will to some extent depend on the architectures to be integrated. For example, instead of using one common perceptual buffer mandated by cognitive architectures like ICARUS (see Section 5.2.1), percepts could be split into different buffers as in ACT-R (see Section 5.2.2). Percepts could also be sent at particular times or frequencies specified by the cognitive architecture. In some cases, it might be possible to achieve better integration by combining entire modules of a cognitive architecture with modules of the robotic architecture. For instance, the vision system in ACT-R could be replaced by or merged with the vision system in the robotic architecture. This would constitute a mixture between L1 and L2 integration.

While some robotic and cognitive architectures provide straightforward access to their perceptual and action buffers (sometimes even with a well-specified interface protocol such as SOAR), others do not have interface mechanisms in place, as they were not designed with them in mind. Fortunately, it is usually possible to extend existing robotic architectures, including low-level "reactive" architectures, by adding mechanisms that bridge between perceptual, goal, and action representations at different levels of abstraction. One example of such a bridge mechanism is the "trick" used in AuRA (Arkin & Balch, 1997) to integrate a deliberative layer, including discrete goal and behavior representations, with a low-level reactive schema-based layer that uses only continuous sensory and effector information. Another is the generic mechanism used by Kramer and Scheutz (2007) to "intercept and lift" low-level representations for use in deliberative layers and, conversely,

ways of using deliberative representations to influence low-level behavior control. A third example of multi-level integration is Kuiper's (2000) Spatial Semantic Hierarchy, which achieves many degrees of abstraction and representation.

5. Case Studies: ICARUS and ACT-R in DIARC

To demonstrate the promise of the above integration strategy, we provide two separate examples of a cognitive-inside-robotic integration using two cognitive architectures: ICARUS, which allows for hierarchical skill learning and execution and is thus a natural fit for robotic applications, and ACT-R, which, as a model of human cognition, would profit from being able to control robotic bodies directly. We have integrated both cognitive architectures into the robotic DIARC architecture (described below) and we have demonstrated and evaluated the integrated systems on a simple robot exploration task via their respective proof-of-concept implementations. For both architectures, we put forth three claims about our proposed integration scheme:

(C1) integration via the three interfaces is sufficient for achieving an operational integration between cognitive and robotic architectures that can run successfully on a real robot;

(C2) no additional timing mechanisms are needed for the cognitive architecture to operate in real-time with the robotic architecture; and

(C3) we can program the integrated architecture by fixing the configurations of the robotic components (based on the given task and the robot hardware employed) and altering content models for the cognitive architecture without processing details about robot percepts and actions.

We use our runs with the two integrated systems to support these two claims.

5.1 The DIARC Robotic Architecture

We selected the DIARC robotic architecture for implementing the generic interface since it facilitates deep integration easily and it illustrates the key integration issues. DIARC has several advantages as compared with other robotic architectures in that: (1) it is implemented in an agent-based middle-ware, ADE, that allows for straightforward extension and integration of new software components by "wrapping" them as "ADE agents" and adding them to an ADE system (Scheutz, 2006); (2) it includes *goal manager* and *action manager* components that provide both explicit goal representations as well as a natural level of abstraction between continuous low-level control processes and discrete action representations; and (3) it supports different levels of integration, from the standard connection of two "black boxes", to a tight integration between components, such as action and natural language processing (see Brick et al., 2007).

DIARC's *action manager* is a priority-based scheduler for action scripts that can run in parallel, each in its own thread. Whenever a script is ready for execution, an *action interpreter* is instantiated to execute all atomic and complex actions it contains. Atomic actions are typically either requests for sensory information from the robot's sensors (e.g., distance data from the laser range finder) or motor commands that are sent to the various effectors (e.g., rotational and translational velocities

sent to the wheels of a mobile robot base). Complex actions are (possibly conditional) sequences of atomic actions. All instantiated action scripts have priorities associated with them that are used for behavior arbitration.

DIARC supports all three levels of integration (as described in Section 4.2). For L1 integrations, a special purpose action manager component can be defined that is derived from the DIARC action manager that "wraps" the cognitive architecture and adds the three interface features to it.[2] For L2 integrations, perceptual, goal, and action components of DIARC can implement direct connections to the cognitive architecture either via Java RMI calls or via specialized sockets, depending on the programming language in which the cognitive architecture is implemented, thus retaining distributed processing. For L3 integration, DIARC already provides various components that can replace functionally similar components in the cognitive architecture, such as various planners for reasoning and problem solving, as well as a scripting language used by the action manager component that can express production rules.

5.2 Cognitive Architecture Integration in DIARC

Both integration examples within DIARC are L1 integrations, thus requiring the development of a special-purpose DIARC action manager component for each cognitive architecture. These implement the communication mechanisms and protocols specific to the CA, in addition to implementing the three interfaces for exchanging percept, goal and action messages described in Section 4.1. In both cases, the CA executes in its own LISP process, letting it run concurrently with the action manager and other DIARC components. Communication is established via a socket; ADE handles the Java-LISP serialization and deserialization, making the source DIARC percepts and goals transparent to the CA. Action requests from the CA are construed as *primitive intentions*, because there is no guarantee that they will be executed (e.g., because other, higher priority actions of the RA may need to run). The underlying RA thus remains operational, able to handle low-level reactive tasks such as obstacle avoidance at a level "subconscious" to the CA. Hence, a CA intention may be overridden by an RA process. For percepts, DIARC implements the *attentional mechanism* discussed in Section 4.1 to let the CA initiate monitoring processes in the RA for percept types salient to the current goals (e.g., doorway percepts in indoor navigation and exploration). The action manager supervises these monitors in the background (from the perspective of the CA), inserting percepts when detected into the CA's perceptual buffers. The attentional mechanism also lets DIARC focus the CA's attention on percepts of predetermined unrequested types, such as hazardous obstacles, by inserting them in a perceptual buffer of the CA.

5.2.1 ICARUS*-DIARC Integration*

DIARC interacts with ICARUS in three ways: (1) through the latter's perceptual buffer by placing percepts into its buffer; (2) via action requests from primitive intentions generated by ICARUS that cause the generation of goals in DIARC, which, in turn, lead to the execution of actions in their service that run concurrently with all other actions in DIARC; and (3) through an introspective goal

2. Perceptual information is retrieved from any relevant perceptual component in DIARC while goal and action information is obtained from the action interpreter instances within the action manager.

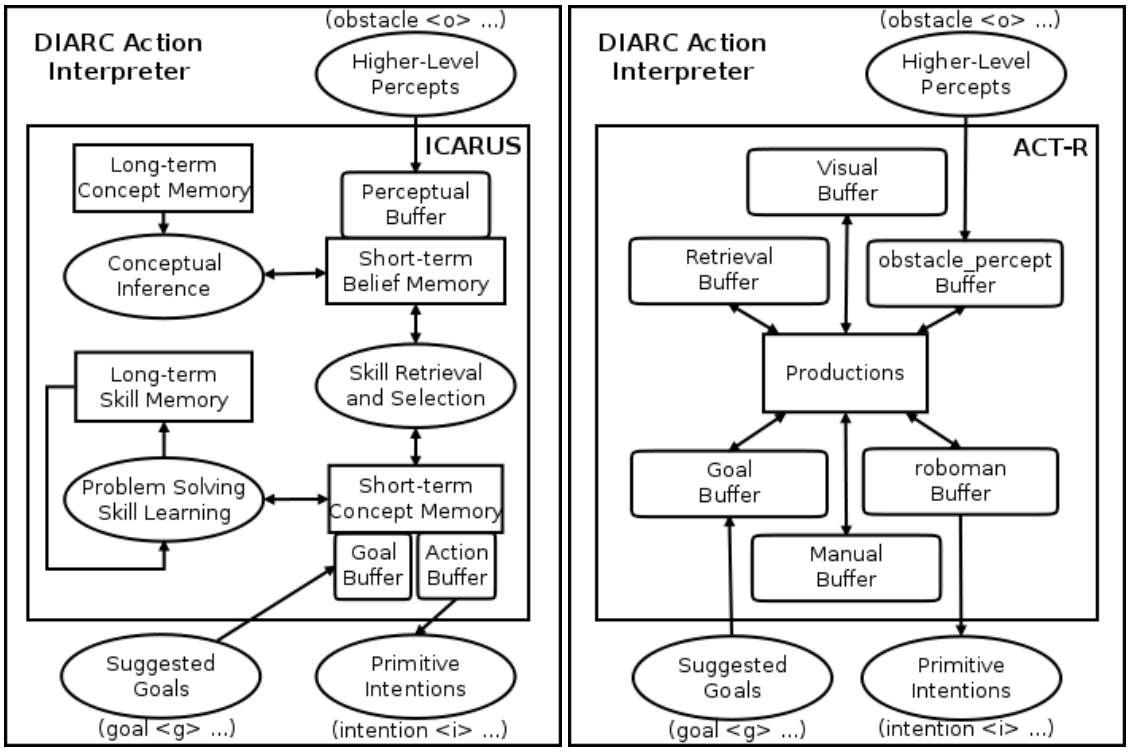

Figure 1. Schematic of the ICARUS-DIARC integration (left) and the ACT-R-DIARC integration (right) via DIARC action manager components. Both figures indicate only the higher-level percepts provided to each architecture and the action primitives sent by both architectures for the proof-of-concept evaluation knowledge structure (see text for details).

reflection mechanism (via special beliefs) that allows ICARUS to check the status of its primitive intentions.

At the start of execution (and possibly at other times, as determined by the cognitive architecture), ICARUS sends a message to specify the percepts DIARC should monitor (e.g., `(intention (attend obstacle))`). At the beginning of every cycle, ICARUS requests the current set of percepts (e.g., `(obstacle f17 location front)`) from the robot architecture.

ICARUS action requests map its action primitives directly onto DIARC goals, allowing for cycle-based issuing of the DIARC goals that it wants to pursue. Specifically, actions to be performed by DIARC (i.e., outside the ICARUS architecture) are requested via the `intention` function:

```
(intention (action1 arg1 arg2 ...)
           (action2 arg1 arg2 ...)
           ...)
```

The associated reflection mechanism lets DIARC provide ICARUS with a status summary of all current and recently completed intentions. An action in DIARC can be in one of three states at any given time: `success` (i.e., successful completion of the action), `failure` (i.e., the action

could not complete for whatever reason),[3] or `ongoing` (i.e., the action has not completed yet but is making progress). Whenever a primitive intention is scheduled, DIARC creates a new intention name used to identify it in subsequent status updates. Intention status updates are placed directly into ICARUS' belief memory on each cycle in the form:

```
(intention (i-name1 <status>)
           (i-name2 <status>)
           ...)
```

Finally, DIARC can insert explicit high-level goals directly into ICARUS' goal buffer, letting DIARC components (e.g., the DIARC NLP component; see Dzifcak et al., 2009) nominate goals that ICARUS itself might not generate. In particular, it can send goals expressed in logical form (e.g., linear temporal logic) directly to ICARUS. For instance, a supervisor's command to go to a particular office takes the form `(EVENTUALLY (location me EG826))`. The ability to add goals to the CA constitutes, in effect, another higher-level attention mechanism.

5.2.2 ACT-R-DIARC Integration

The ACT-R integration is overall very similar to that for ICARUS, the main difference being that a special action manager component for ACT-R is employed, analogous to the one developed for ICARUS. Moreover, given that both CAs are implemented in LISP, the ACT-R action manager uses the same LISP representations described above for the ICARUS version. We considered two possible approaches to the integration: (1) an L2-type *module-level* integration, which would require the replacement or extension of one or more of the *modules* that serve as the interface between the ACT-R buffers and the outside world, and (2) an L1-type *buffer-level* integration, which would require the introduction of new buffer types and corresponding control routines for operating on the buffers during each cognitive cycle.

We chose the buffer-level approach, since the standard ACT-R buffers are not well-tailored to robotic control.[4] For example, the *manual buffer* is normally used to specify hand movements, which would be a good fit if we were modeling a *human controlling a robot* (e.g., via a keyboard or a joystick interface). But in this case the goal is to construct a *cognitive architecture in a robot*, and expressing motor commands as hand motions is awkward, at best. Hence, we implemented new ACT-R buffers for the robot-specific interfaces: a *roboman* buffer for motor commands, and an *obstacle_percept* buffer (analogous to the *visual* or *aural* buffers). An ACT-R equivalent of the ICARUS command is issued before each ACT-R cycle. The LISP expressions from the DIARC action manager are converted into chunks to be deposited in the appropriate buffer (e.g., the *obstacle_percept* buffer), and chunks representing robot navigation commands are taken from the *roboman* buffer, converted into `intention` expressions, and sent back to DIARC.

3. Depending on the action there might be failure conditions that can be reflected on.
4. While the standard implementation of ACT-R is in Common Lisp, there is also a newer JAVA-based implementation (jACT-R). With the help of Anthony Harrison from the US Naval Research Laboratory, we have been able to integrate jACT-R into DIARC. The jACT-R integration is also at the buffer level, with the same new buffers as in the LISP implementation. However, one advantage of this Java implementation of ACT-R is that the action manager component can make direct method calls to ACT-R instead of communicating via sockets.

5.3 Proof-of-Concept Implementations

For the evaluation of our integration approach we chose a simple environment exploration task in which the agent must explore its environment and traverse it while avoiding obstacles. Note that this task was not designed to explicitly feature the capabilities of any particular cognitive or robotic architecture, nor was it intended to highlight any particular modeling application. Instead, the two resultant exploration systems were intended as *proof-of-concept implementations* to substantiate the three claims we formulated about the integration approach at the beginning of this section. In particular, we intended them to demonstrate that each cognitive-inside-robotic integration can successfully operate a real robot in real time in a real-world task. For a more extensive example involving human-robot speech interactions involving the ICARUS-DIARC integration, see Trivedi et al. (2011).

In the following subsections, we briefly describe the knowledge-based structures we had to develop for the ICARUS and ACT-R exploration agents. Prior to integration, the standard DIARC-level obstacle avoidance mechanisms were disabled in order to provide full control of these behaviors to the cognitive level. For space reasons we cannot provide all ICARUS concepts and skills or all ACT-R productions and chunks. We show instead representative examples for each architecture that give a flavor of the kinds of knowledge structures that are required.

5.3.1 ICARUS *Implementation*

The ICARUS version of the exploration agent is instantiated with two initial goals, namely to be moving and to keep the region in front of the agent clear of obstacles:

```
(create-goals (moving ?me)
              (can-move-forward ?me))
```

The goals are defined as ICARUS *concepts*, which are expressed in terms of percepts, relations that must hold for the concept to be satisfied, and tests that must be met. For example, the "open space ahead" goal is defined by two conceptual rules:

```
((can-move-forward ?me)
 :percepts ((robot ?me))
 :relations ((not (obst-in-front ?me ?o))))
((obst-in-front ?me ?o)
 :percepts ((robot ?me)
            (obstacle ?o location ?loc))
 :tests ((string= ?loc 'front)))
```

That is, the agent can move forward as long as there is no obstacle in front of it. ICARUS concepts are used by *skills* to effect changes in the robotic architecture. The `can-move-forward` concept is used in the skill:

```
((can-move-forward ?me)
 :percepts ((robot ?me))
 :start ((obst-in-front ?me ?o) (can-move-left ?me))
 :actions ((intention ((stop) (turn-left)))))
```

When ICARUS is notified of an obstacle in front of the agent, the `can-move-forward` concept is no longer satisfied. If there is no obstacle to the left (i.e., if `can-move-left` is satisfied) then ICARUS sends DIARC an intention message with commands causing the robot to stop and turn left. Similar concepts and skills are defined to handle other contingencies.

5.3.2 ACT-R Model Implementation

ACT-R's knowledge for this task resides in production rules that use the two new buffers discussed in Section 5.2.2. One such production causes the agent to start moving forward if the way is not blocked (i.e., if DIARC has not sent a percept indicating that there is an obstacle in front):

```
(P jbot-move
  =goal>
    ISA               goal-type
    name              "explore"
  =imaginal>
    ISA               last-actions
    - action1         "startmove"
  =obstacle_percept>
    ISA               obstacle_percept-data
    forward-blocked   false
  =roboman>
    ISA               roboman-data
==>
  =imaginal>
    action1           "startmove"
  =roboman>
    control_command   "ACOM"
    control_data      "startmove"
)
```

Similar productions are used for navigation control, such as turning and stopping, based on the contents of the *obstacle_percept* buffer, the *imaginal* buffer, which holds a representation of the current action intention, the *goal* buffer, which holds the current high-level goal of the model, and the *roboman* buffer.

5.4 Qualitative Evaluation of the Proof-of-Concept Implementations

We evaluated the two integrated cognitive-robotic architectures using a Pioneer P3-AT mobile robot platform equipped with a SICK laser range finder for detecting obstacles and doorways. All components of the system executed on an on-board laptop running Linux. We chose a qualitative evaluation as opposed to a quantitative evaluation because the proof-of-concept implementations *per se* do not provide any interesting performance measures that could be quantitatively analyzed. Rather, the idea for the evaluation is to show that the proof-of-concept implementations result in viable systems that support our three claims about the integration approach. We thus started both integrated architectures on the robot in various locations in an indoor environment. The robot was allowed to freely navigate the office space and sucessfully avoided tables, chairs, people and other obstacles as

it traversed the environment. In no run did the robot ever collide with any obstacle, get stuck in a particular place, or stop moving because it was out of actions. This shows that the robot was able to use percepts properly, that the goal for exploring the environment was used correctly, and that actions were issued properly from within the cognitive architecture, causing the robot to exhibit its exploratory behavior.

Hence, the three core components of the generic interface were fully operational in all demonstration runs. This supports both claim (C1), that the integration via the proposed three interfaces was sufficient for achieving an operational integration between cognitive and robotic architectures, and claim (C2), that no additional timing mechanisms were needed for the cognitive architecture to operate in real-time with the robotic architecture. Finally, our development of the knowledge structures for the cognitive architectures supported claim (C3), that one can program the integrated architecture by fixing the configurations of robotic components (based on the given task) and develop knowledge (chunks, rules, concepts) for the cognitive architecture to perform the task without knowing implementation details of the percepts and actions employed in the robotic architecture.

6. Discussion

The proof-of-concept evaluations demonstrated that the generic interface is viable and that we can develop operational robotic agents using the integration interface. Naturally, both the interface and its evaluation are only a first step. Various interesting algorithmic and architectural questions ensue once the doors are opened for integrated agents, including ones about the systematic evaluation of the synergistic capabilities enabled by these new mechanisms, any one of which would represent an enormous undertaking worthy of its own research agenda.

For example, how can an integrated architecture address the problem of objects' "identity"? In cognitive architectures it is often assumed that different token identifiers represent different objects, while perceptual modules in robotic architectures can usually track objects only for short periods of time (e.g., through brief temporal occlusions). Yet whether the coffee mug on the office desk today is the same as the one yesterday is not so much a perceptual fact (e.g., the coffee mug looks exactly like another one in another office) as a matter of inference (e.g., that the door to the office was locked between percepts, and nobody else had a key, so it could not have been switched). Hence, establishing the identity of the coffee mug ought to occur in the cognitive architecture, which both represents the factual knowledge (in episodic memory) of where the cup was left, what it looks like, and so forth, and has suitable inference mechanisms to conclude that the mug is the same. To utilize conceptual and factual knowledge to constrain, revise, or elaborate percepts by virtue of inference and reasoning processes, a much tighter integration between the perceptual and cognitive system is required than what we have presented in the case studies above.

One area of research that could benefit tremendously from this type of tight integration is *incremental natural language processing* (Brick & Scheutz, 2007). In other work, we have constructed a DIARC-Soar interface that lets us investigate the kinds of information exchanges that are necessary to combine perceptual, linguistic, and cognitive information to improve natural language understanding. Like the L1 integrations with ACT-R and ICARUS, DIARC components preprocess perceptual information to create complex data structures that are handed off to Soar. For example,

natural language utterances are transformed from raw audio signals into symbolic representations that DIARC then inserts into the Soar agent's working memory (analogous to the way natural language commands are inserted into the ICARUS agent's goal buffer). In this way, the Soar agent's production system is influenced by real-world perceptual information, and we can investigate how a CA's reasoning and goal structures can guide an RA's natural language processing components, say by adjusting probabilities or weights in parsing components or initiating actions that can aid in disambiguation.

Other interesting questions to pursue in future work concern the extent to which the robotic architecture should be represented in the cognitive architecture and the extent to which the cognitive architecture should be fully in control of the agent's actions. In the former case, it might be useful for the cognitive architecture to maintain representations of robotic architecture components and their processing status, which would let the cognitive architecture reason about internal resources and task progress, especially in light of perceptual errors; for example, it could detect that speech production is not working properly and devise other means of communicating information to a human team member. In the latter case, something like a "subconscious" mind that can on its own make decisions and initiate actions independent of the cognitive architecture might retain the robotic architecture's ability to respond quickly to potentially dangerous changes in its environment; this was one of the main reasons that many roboticists abandoned the "sense-think-act" cycle in the 1980s.

It would also be interesting to investigate the potential of integrating multiple cognitive architectures into one robotic architecture, given that different cognitive architectures model cognitive functions that occur at different time scales. For instance, ACT-R models typically range from tasks requiring a few cognitive cycles to multiple seconds, while Soar models capture tasks of multiple seconds and up.[5] They also have different strengths, from the biological plausibility of ACT-R to the reactive hierarchical action sequencing in ICARUS to the problem-solving mechanisms in Soar.

Finally, another intriguing use for the integration framework is mental modeling of human teammates in HRI tasks (e.g., see Scheutz, 2013), which would let a robotic system predict a teammate's response to a set of perceptions. One approach would be to instantiate a separate CA that models the teammate and feed it perceptual data, possibly in a "perspective-adjusted" manner. In effect, the CA would store knowledge about the teammate's knowledge based on specialized rules (e.g., if the robot hears a sound, assume the teammate also hears it) and subsequently reason based on that. The intentions produced by the teammate CA would then be the robot's prediction of the teammate's response. The mechanisms introduced here should facilitate the inclusion of such "supplemental" cognitive architectures.

7. Conclusions

We started our investigation by discussing the benefits of integrating cognitive and robotic architectures, and we argued that there are two main benefits of running a cognitive architecture within a robotic one: (1) the cognitive architecture can be used as a general problem solver that lets robots

5. Wray et al. (2007) have explained this idea in their C3I1 integrated architecture, which integrates both into a single "three cognitions, one intelligence" architecture.

deal with novel unfamiliar situations in more human-like ways; and (2) the cognitive architecture can model human behavior, knowledge, and other mental states that might be important for the task at hand in contexts where robots must interact with humans.

We presented a systematic and generic interface between cognitive and robotic architectures that allows for varying levels of integration between the two architecture types. In addition, we presented proof-of-concept integrations between the robotic DIARC architecture and two cognitive architectures, ICARUS and ACT-R, to validate the interface. Since the interface was based on theoretical principles that researchers in both fields should find intuitive, we believe that it will be straightforward to implement and use the interface for both reactive and hybrid robot architectures.

In addition to improving the performance of robotic systems on tasks that require online adjustments and problem solving, the interface should also let robots utilize mental models of teammates, very much in the way humans use models of other humans. Finally, we believe that the cognitive architecture community will benefit from having robotic architectures available to let them develop and evaluate models of embodied cognition in real-world settings.

Acknowledgments

The authors would like to thank Pat Langley and the anonymous action editor for their detailed feedback on the paper, and Pat Langley's group at Arizona State University (in particular, Nan Li and Nishant Trivedi) for helping with the debugging of the proof-of-concept ICARUS agent. This work was funded in part by ONR MURI grant No. N00014-07-1-1049 to the first author.

References

Anderson, J., Bothell, D., Byrne, M., & Lebiere, C. (2004). An integrated theory of the mind. *Psychological Review*, *11*, 1036–1060.

Andronache, V., & Scheutz, M. (2004). Integrating theory and practice: The agent architecture framework APOC and its development environment ADE. *Proceedings of Autonomous Agents and Multi-Agent Systems* (pp. 1014–1021).

Arkin, R. C., & Balch, T. R. (1997). AuRA: principles and practice in review. *Journal of Experimental and Theoretical Artificial Intelligence*, *9*, 175–189.

Bonasso, R., Firby, J., Gat, E., Kortenkamp, D., Miller, D., & Slack, M. (1997). Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, *9*, 237–256.

Brick, T., Schermerhorn, P., & Scheutz, M. (2007). Speech and action: Integration of action and language for mobile robots. *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 1423–1428). San Diego, CA.

Brick, T., & Scheutz, M. (2007). Incremental natural language processing for HRI. *Proceedings of the Second ACM IEEE International Conference on Human-Robot Interaction* (pp. 263–270). Washington D.C.

Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, *2*, 14–23.

Carbonell, J., Etzioni, O., Gil, Y., Joseph, R., Knoblock, C., Minton, S., & Veloso, M. (1991). Prodigy: an integrated architecture for planning and learning. *SIGART Bulletin*, *2*, 51–55.

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

Dzifcak, J., Scheutz, M., Baral, C., & Schermerhorn, P. (2009). What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. *Proceedings of the 2009 International Conference on Robotics and Automation*. Kobe, Japan.

Hanford, S. D., Janrathitikarn, O., & Long, L. N. (2009). Control of mobile robots using the soar cognitive architecture. *Journal of Aerospace Computing, Information, and Communication*, *6*, 69–91.

Kramer, J., & Scheutz, M. (2007). Radic – towards a general method for integrating reactive and deliberative layers. *International Transactions on Systems Science and Applications*, *3*, 183–192.

Krause, E., Schermerhorn, P., & Scheutz, M. (2012). Crossing boundaries: Multi-level introspection in a complex robotic architecture for automatic performance improvements. *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

Kuipers, B. (2000). The spatial semantic hierarchy. *Artificial Intelligence*, *119*, 191–233.

Laird, J., Newell, A., & Rosenbloom, P. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence*, *33*, 1–64.

Langley, P., Choi, D., & Rogers, S. (2009a). Acquisition of hierarchical reactive skills in a unified cognitive architecture. *Cognitive Systems Research*, *10*, 316–332.

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

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

Scheutz, M. (2006). ADE: Steps towards a distributed development and runtime environment for complex robotic agent architectures. *Applied Artificial Intelligence*, *20*, 275–304.

Scheutz, M. (2013). Computational mechanisms for mental models in human-robot interaction. *Proceedings of the HCI International 2013 Conference on Human-Computer Interaction*.

Scheutz, M., & Andronache, V. (2004). Architectural mechanisms for dynamic changes of behavior selection strategies in behavior-based systems. *IEEE Transactions of System, Man, and Cybernetics Part B: Cybernetics*, *34*, 2377–2395.

Scheutz, M., Cantrell, R., & Schermerhorn, P. (2011). Toward humanlike task-based dialogue processing for human robot interaction. *AI Magazine*, *32*, 77–84.

Scheutz, M., Schermerhorn, P., Kramer, J., & Anderson, D. (2007). First steps toward natural human-like HRI. *Autonomous Robots*, *22*, 411–423.

Sloman, A. (1998). Damasio, Descartes, alarms and meta-management. *Proceedings of 1998 IEEE International Conference on Systems, Man, and Cybernetics, San Diego* (pp. 2652–2657). IEEE.

Trafton, J., Cassimatis, N., Bugajska, M., Brock, D., Mintz, F., & Schultz, A. (2005). Enabling effective human-robot interaction using perspective-taking in robots. *IEEE Transactions on Systems, Man and Cybernetics*, *25*, 460–470.

Trivedi, N., Langley, P., Schermerhorn, P., & Scheutz, M. (2011). Communicating, interpreting, and executing high-level instructions for human-robot interaction. *Proceedings of the 2011 AAAI Fall Symposium on Advances in Cognitive Systems*. Arlington, VA: AAAI Press.

Wray, R., Lebiere, C., Weinstein, P., Jha, K., Springer, J., Belding, T., Best, B., & Parunak, V. (2007). Towards a complete, multi-level architecture. *Proceedings of the 2007 Internation Conference on Cognitive Modeling* (pp. 325–330). Ann Arbor, MI.