An Overview of the Distributed Integrated Cognition Affect and Reflection DIARC Architecture

Chapter · January 2019

DOI: 10.1007/978-3-319-97550-4_11

6 authors, including:

Matthias Scheutz
Tufts University
486 PUBLICATIONS **9,075** CITATIONS
SEE PROFILE

Tom Williams
Colorado School of Mines
218 PUBLICATIONS **2,404** CITATIONS
SEE PROFILE

Evan Krause
Tufts University
18 PUBLICATIONS **279** CITATIONS
SEE PROFILE

Bradley Oosterveld
Tufts University
17 PUBLICATIONS **238** CITATIONS
SEE PROFILE

An Overview of the Distributed Integrated Cognition Affect and Reflection DIARC Architecture

Matthias Scheutz, Thomas Williams, Evan Krause, Bradley Oosterveld, Vasanth Sarathy, and Tyler Frasca

Abstract DIARC has been under development for over 15 years. Different from other cognitive architectures like SOAR or ACT-R, DIARC is an intrinsically component-based distributed architecture scheme that can be instantiated in many different ways. Moreover, DIARC has several distinguishing features, such as affect processing and deep natural language integration, is open-world and multi-agent enabled, and allows for "one-shot instruction-based learning" of new percepts, actions, concepts, rules, and norms.

In this chapter, we will present an overview of the DIARC architecture and compare it to classical cognitive architectures. After laying out the theoretical foundations, we specifically focus on the action, vision, and natural language subsystems. We then give two examples of DIARC configurations for "one-shot learning" and "component-sharing". We also briefly mention different use cases of DIARC, in particular, for autonomous robots in human-robot interaction experiments and for building cognitive models.

1 Introduction

Classical cognitive architectures (CCAs) have evolved significantly since their inception in the late 1970s, with more and more features added on top of their core production systems. The ACT-R architecture (currently at version 7), for example, started from a model of associative memory and has morphed into a system allowing multiple inheritance among chunks, together with any number of new buffers connected to the central production system that can be added to the architecture to hold memory chunks (e.g., to allow for interactions with sensory and effector modules, see the ACT-RE models by Trafton et al. [76]). Similarly, the SOAR architecture (currently at version 9.6) started with a production system that only featured

HRI Laboratory, Tufts University, Medford, MA 02155, e-mail: matthias.scheutz@tufts.edu

"chunking" as the single architectural learning mechanism and has morphed into a system that integrates reinforcement learning, as well as semantic and episodic memories (in addition to the original working memory). While some of the extensions were driven by the need for more complex mechanisms to be able to develop adequate cognitive models, others were driven by the need to provide more capabilities for applications (e.g., in virtual and robotic agents). In addition to classical cognitive architectures, newer cognitive architectures such as Icarus, Clarion, and others were developed to address specific research questions (e.g., the implicit-explicit dichotomy in cognitive systems or the questions of how to learn and execute hierarchical skills, respectively).

Different from CCAs, the "Distributed Integrated Affect Reflection and Cognition" DIARC architecture [66, 63] was originally neither designed as nor intended to be a model of *human cognition*. Rather, it was conceptualized from the beginning as an *architecture scheme* (similar to the *CogAff* architecture scheme [71]) that could subsume a large set of possible architectures, and thus be used to realize a diverse set of cognitive systems of varying complexity, especially situated embodied systems such as robots. Architecture schemes are templates that, when filled in with details (i.e., specific components and their connections), specify individual architectures. In DIARC, this means that once components and their interactions are fixed, a particular DIARC architecture (a DIARC instance) is obtained. Note that the distinction between an architecture scheme and an architecture instance is different from the distinction between the algorithms and knowledge in cognitive architectures [41], in which "algorithms" are said to define the architecture *per se* (components and links) and knowledge is viewed as being encoded in representations contained in those components (either preloaded or acquired during operation). However, the design-as-architecture scheme does not preclude using different DIARC instances as cognitive models. And, in fact, different instances of DIARC have been used to model different cognitive aspects (e.g., the interaction of affect and goal processing [55], or a language-guided conjunctive visual search [65]).

In the following, we will first lay out the theoretical commitments made by DIARC as an architecture scheme and then discuss in greater detail the notable features that distinguish DIARC from other cognitive architectures. We then briefly give examples of two instances of DIARC for "one-shot learning" and "component-sharing", respectively, as well as applications of DIARC in cognitive modeling, autonomous robotics, and human-robot interaction.

2 Theoretical Commitments

Every cognitive architecture is based on basic theoretical assumptions about the structure and nature of its components, the data representations used inside and across components, as well as the information and control flow, and possibly the timing of component updates and information exchanges. To show the commonalities and differences in theoretical commitments of DIARC compared to CCAs, we

start with the four-part framework for discussing CCAs proposed in [40]: (1) structure and processing, (2) memory and content, (3) learning, and (4) perception and motor. Following this comparison, we discuss additional theoretical commitments DIARC makes concerning its components, as well as the principles underwriting the overall polylithic design and implementation of DIARC in a multi-agent system middleware.

2.1 Structure and Processing

In line with typical assumptions in CCAs, DIARC is composed of a set of components that operate in parallel and can communicate with each other by exchanging messages using logical representations. Different from most CCAs, DIARC components operate asynchronously in real parallelism and do not assume any synchronization mechanism (e.g., as imposed by a "perceive-think-act" cycles). Moreover, in addition to there not being any prescription of a particular system-wide cognitive cycle across components, there is also no prescription of the update timing of a component (e.g., compared to 50 ms or 100 ms for cognitive cycles in some CCAs). Rather, each component can update at the rate appropriate for the information it processes (and may run multiple threads of control within itself).

2.2 Memory and Content

While the details of the interaction among different components depend on the particular architecture (e.g., ACT-R provides a buffer mechanism that serves as the interface between the core production system and other modules), CCAs typically impose a communication bottleneck when they require that different modules interact via a special (short-term) *working memory* component and two long-term memory components for procedural and declarative knowledge. In contrast, DIARC does not impose such structural or communication constraints based on memories and memory access, but rather allows components to locally implement their own short-term and long-term memories. Consequently, there is no mandated component-based distinction between declarative vs. procedural knowledge – both kinds could coexist in the same memory component – although typically, procedural knowledge is stored in the action execution component while declarative knowledge is stored in a special memory and inference component that can be instantiated multiple times as needed and used for short-term and long-term information storage. Different memories can be cross-indexed and accessed via consultants that establish those links (see Section 3.3.3). Moreover, there is no prescribed knowledge representation format in DIARC for knowledge stored within components (e.g., the way in which declarative knowledge has to be represented as "chunks" in ACT-R or procedural knowledge has to be represented in terms of production rules). Rather, knowledge representa-

tions can take different forms within components, depending on the nature of the process (e.g., saliency maps inside the vision processing component, dependency graphs in the parser, clauses in the reasoner, etc.). However, there is a requirement that messages exchanged conform to the same format across architecture instances (i.e., logical expressions are used as a common currency and data representation format across components).

2.3 Learning

In CCAs, all long-term memory entries are learnable online and incrementally during task performance using "architectural" (i.e., built-in) learning mechanisms, often through inverting the information flow. Different types of learning are employed, depending on the types of long-term memory (e.g., reinforcement learning to generate weights for action selection and procedural composition such as "chunking" for procedural learning, or the learning of facts together with their meta-data for declarative learning). In contrast, `DIARC` does not prescribe any particular architectural learning mechanism, but allows components to implement their own learning strategies, depending on the information they process. For example, the vision and auditory subsystem can employ unsupervised learning to improve the accuracy of their classifiers (e.g., adjust object recognizers to build better recognition templates in the vision system or adjusting word prototypes to be able to better recognize different word instances in the speech recognizer). Policy-based action execution systems might use reinforcement learning to improve their policies, or they could use action sequences from plan traces to learn the appropriate action sequences (very much like what "problem solving" allows in architectures like Soar). In addition to online learning, some `DIARC` components can also be trained offline (e.g., the vision and speech components, the parser, policy-based planers, etc.). Most importantly, however, `DIARC` directly supports instruction-based "zero-shot" and "one-shot" learning *across most knowledge representations in the architecture*, both through its integration of natural language processing and component capabilities for one-shot learning (e.g., in the vision and speech recognition components) [64, 38]. As a result, new words, percepts, actions, skills, rules, plan operators, norms, and other forms of knowledge can be learned quickly through natural language instruction during performance and used in "open-world" tasks for which not all required task-based knowledge is available ahead of time, but rather must be acquired online during task execution (e.g., [75]).

2.4 Perception and Motor

CCAs assume that perception modules generate symbolic structures representing the perceived object, relation, event, etc. while motor modules convert such sym-

bol structures into motor actions. Both perception and motor modules may allow for learning (e.g., to acquire new perceptual and action patterns), although such learning is typically outside of the architectural specification (as lower-level perceptual and motor control processes are typically not included in CCAs). In contrast, DIARC was specifically aimed at real-world, real-time interactions and thus takes both perceptual and motor processes very seriously, providing detailed models for both (e.g., an extensive vision system that can process information from various types of sensors in real-time and various robot body modules that can process motor behavior for different robot body types). Similar to CCAs, learning in these components is not prescribed, but rather different learning methods are allowed. Different from CCAs, DIARC permits zero-shot perception and motor learning from instruction (e.g., direct learning of new percepts or new primitive actions from natural language descriptions without exposure to the percepts or the actions). Moreover, perceptual processing and action sequencing are closely tied to the real world (e.g., update frequencies of the vision system are related to the frame rates of sensors, and action commands at the lowest motor levels are tied to the command speed of effectors and the durative nature of embodied activities).

2.5 Additional Component Commitments

In addition to the types of commitments found in CCAs, DIARC makes additional theoretical commitments about its components that are not found in CCAs:

- *Affect integration.* Affect is, surprisingly, not part of CCAs, even though it is a central component of human cognition and CCAs are often intended to be "models" of human cognition. All DIARC components must represent both positive and negative affect (in the simplest case, as measures of how well a component performs, in more complex cases, as richer representations of desired component states). Some components like the Goal Manager collect affective evaluations from other components to compute composite evaluations of how well the agent (controlled by the DIARC instance) is doing, which can then be used to prioritize goals and modulate expected utility [56].
- *Open-world processing.* All DIARC components must be open-world enabled, i.e., allow for partial and incomplete representations of the information they process, as happens in open-world scenarios for which not all of the information is available initially, but rather has to be acquired through discovery and learning processes (e.g., unknown words in goal instructions referring to unknown entities in unknown locations, e.g., [75]).
- *Multi-level introspection.* All DIARC components must allow for the introspection of their states through middleware-enabled introspection processes, which can be used to optimize component and architecture performance, but also to detect and recover from faults (e.g., [36]). In addition, introspection methods can be used by components to detect available functionality in DIARC instances

(e.g., the Goal Manager component can determine the kinds of perception and action primitives that are available in other components and can be used in action scripts, see Section 3.1). Explicit logical annotations of pre- operating, and post-conditions of services made available by components to other components can be used to enable introspective access and run-time reflection on system features and capabilities.

- *Component-sharing.* All DIARC components must allow for component sharing across multiple agents, i.e., two or more agents realized as DIARC instances might share a single DIARC component (e.g., a common natural language processing subsystem consisting of speech recognizer, semantic and pragmatic parser, and dialogue manager components). Component-sharing allows for efficient implicit realization of agent-to-agent communication in which instead of explicit communication, agents have direct access to required knowledge structures [45].

2.6 Polylithic Design and Implementation

A result of being an architecture scheme, and thus allowing for different configurations among possible components and links, DIARC is intrinsically *polylithic*, compared to the monolithic nature of classical cognitive architectures. The polylithic nature is guaranteed by the implementation of DIARC in the "Agent Development Environment" ADE [54, 35, 34, 36, 59, 3, 32, 33, 31, 1, 2], which was specifically developed to address the various challenges posed by sustained long-term operation of autonomous robots. Analogous to current robotic infrastructures (such as ROS [46], JAUS [30], Yarp [42], and several others), ADE provides a basic communication and computational infrastructure for parallel distributed processes that implements various functional components of an agent architecture (e.g., the interfaces to a robot's sensors and actuators, the basic navigation and manipulation behaviors, path and task planning, perceptual processing, natural language parsing, reasoning and problem solving, etc.).[1] Different from other robotic infrastructures, ADE was from the very beginning, *designed to be as secure and fault-tolerant as possible* (e.g., [54, 1, 3]). These features have been evaluated in HRI experiments [34, 32]. Moreover, due to ADE's extendability, DIARC is easily and systematically extendable by just adding more DIARC components (that may simply "wrap" existing libraries and algorithms) implemented in ADE to an architecture instance (this is different from CCAs such as ACT-R or Soar, in which extensions can only be accomplished through specialized mechanisms such as buffers or special I/O links).

[1] A detailed conceptual and empirical comparison of robotic infrastructures up to 2006 can be found in [35].

3 An Overview of Select DIARC Components and Processes

After our brief overview of DIARC and its theoretical commitments, we now present a few central DIARC components and processes in more detail. By "central", we intend that these components will typically be part of a DIARC instance, even though they do not necessarily have to be included for all applications: (1) The Goal Manager, (2) the Vision System, and (3) the Natural Language subsystems.[2]

3.1 Goals, Actions, and Action Execution

Goals represent terminal states of the internal or external environment that an agent may need to satisfy. In DIARC, the Goal Manager (GM) receives goals from other components in the architecture, including itself. It evaluates the incoming goals, determines what behavior or action the agent should perform, how the agent should proceed, and handles the priority of each action. The priority of the actions are computed based on the urgency, expected utilities, and overall affective state. When the GM receives a goal, it determines the validity of the goal, initializes an Action Interpreter to select a sequence of actions, which, when executed, will accomplish the goal state, and then manages the execution of that action.

3.1.1 Action Representation

Actions in DIARC are stored within the Action Database, a long-term procedural memory, and are represented by a name, arguments, as well as pre-, operating-, and post-conditions. An action is either a primitive action or an action script. Primitive actions describe the specific functionality of their advertising components. For example, a vision component would advertise a *findObject* action that allows the GM to direct the vision component to look for an object, while a manipulation component would advertise the *graspObject* and *moveObject* actions in order for the GM to direct a manipulation component to act on an object. Action scripts are complex tasks containing a sequences of primitive actions, action scripts, action operators (e.g., arithmetic, comparison, etc.), and control statements (e.g., conditional statements and loops).

[2] The description of additional relevant components, such as the Belief and Inference subsystem, the (Motion and Task) Planning subsystem, and the interfaces with other middleware, will have to await a different publication outlet.

3.1.2 Action Execution

When the GM receives a goal submission, it creates a new Action Interpreter. The Action Interpreter first initializes the process for selecting an action. Then, if an action is found, it manages the execution of that action. Within the action Interpreter, an observation mechanism, described below, allows the agent to make observations about the world state by checking the state of events, objects, and agents. This mechanism enables the agent to track the progress of action execution.

Once the Action Interpreter selects an action to perform, in order to follow social norms and core rules, it verifies that the action is neither forbidden nor that it executing it would make the system enter a forbidden state. Then, it confirms that all of the action's pre-conditions are satisfied. For each pre-condition, the Action Interpreter spawns an observer (if available) to check the state of the environment. However, if there is no observer available, it checks the State Machine, which holds the agent's knowledge of the current state of the world. If any of the pre-conditions are not satisfied, then the Action Interpreter will cancel the execution and will report the failure conditions. During the course of execution of the action, there are conditions that need to be satisfied throughout ("operating conditions"). Thus, the Action Interpreter starts observation monitors for each operating condition. If at any point one of the conditions is no longer met, then it will cancel the action and report the failure conditions.

After the Action Interpreter completes the initial preparations, it can continue the execution process by checking to see if the action is a primitive action or a complex action represented by an action script. If the selected action is a script, then a similar process as described below for the primitive actions occurs for each sub-action. Each sub-action specifies the assigned agent responsible for carrying it out. However, if it is a primitive action, then the Action Interpreter checks the agent specified to perform the action. Because each action has a specified agent involved, actions can contain multiple agents interacting with each other. If the agent delegated to perform the action is a `DIARC` agent, then it will proceed normally with the execution. Otherwise, the Action Interpreter observes the other agent performing the action and the post-conditions of the action. Finally, the Action Interpreter confirms that the post-conditions of the action have been satisfied. For each condition, the Action Interpreter spawns an observer, if available. Otherwise, it will check the State Machine. The observers can confirm that other agents have performed their appointed tasks. If all of the post-conditions are met, then the action returns successfully, otherwise the action fails and the failure conditions are reported.

3.1.3 Observers

An agent must be able to track the progress of an action by observing the world and checking the conditions of the action. For instance, if an agent picks up an object off a table, it must observe that the object is in its hand and that the object has been lifted off of the table. While the agent can execute this action blindly and simply assume it

to have been completed successfully (e.g., if the action can be performed with a very low error rate), it will not truly know whether the action was successfully executed unless it observes the action outcomes. This mechanism is particularly critical for multi-agent interactions in which one agent must wait for another agent to perform an action.

Observers are implemented as special primitive actions that adhere to a particular method signature and explicitly advertise the types of observations they enable (e.g., `touching(X,Y)`). To make use of the observations, the Action Interpreter looks for available observers in the Action Database when verifying conditions for an action. If an observer is found for a particular condition, then a new observer sub-goal is spawned. During verification of pre- and post-conditions, the Action Interpreter blocks execution until the observer process either returns successfully or has timed out. On the other hand, observers for operating conditions are spawned concurrently with the action to be executed and the capability to interrupt action execution in cases of failures.

3.2 Perception and Cognitive Affordances

The Vision component (VIS) is responsible for almost all of the visual perception capabilities in DIARC. This component consists of a highly parallel, modular, and flexible framework composed of various general purpose *Image Processors*, *Saliency Operators*, *Object Detectors*, *Object Validators*, and *Object Trackers* and is responsible for the detection, identification, and tracking of target objects and relations among them. VIS is capable of operating on a variety of input sensor types (e.g., RGB, RGB-D, depth-only), and automatically configures its available capabilities based on this information. Additionally, VIS supports multiple asynchronous "visual searches" that can optionally share parts of their processing pipelines so as to reduce redundant computations and save computational resources.

Image Processors are generally used to implement common low-level image processing tasks such as feature extraction (e.g., SIFT) and edge detection, and provide a mechanism for commonly consumed image processing results to be simultaneously shared across several vision processors and visual searches.

Saliency Operators are attentional mechanisms that can be used to guide a visual search to the most salient parts of a scene. These can be driven by top-down language-guided constraints (e.g., red, tall) and/or bottom-up models such as those by Itti and Koch [29].

Object Detectors are responsible for segmenting object candidates from the scene. Detectors can take the form of either generic object detectors that attempt to break the scene into constituent parts, or specialized detectors that search the entire scene for objects of a particular class or category (e.g., face, mug).

Object Validators consume segmented object candidates from Detectors and attempt to classify them as having particular properties (e.g., color, shape, cate-

gory/class). Successfully validated objects are passed through to the next stage of the vision pipeline.

Object Trackers are the last stage of a vision pipeline. Trackers consume object candidates that have been fully validated (i.e., meet all visual search criteria), and are responsible for tracking objects from frame to frame.

One critical aspect of the vision component is exposing and advertising its capabilites to the rest of the system. This is done through simple quantifier-free first-order predicate representations, in whih each vision processor described above (with the exception of Trackers and Image Processors) advertises what it is capable of processing (e.g., `red(X)`, `mug(X)`, `on(X,Y)`). In order for a component in the system to make use of VIS capabilities, it simply has to make a request to VIS in the form a quantifier-free first-order predicate representation. VIS will take this request and attempt to find a collection of vision processes capable of satisfying each part of the predicate request. If all parts are satisfied, the relevant visual processors are assembled into a vision pipeline and a visual search is started. Requesting components are then able to retrieve any available search results.

VIS also has the ability to dynamically learn new object representations. These representations can take the form of either definitions (e.g., "a medkit is a white box with a red cross on it") or instances (e.g., "this object is a medkit"). For learned definitions, VIS must be able to map all parts of a definition to existing vision capabilities. Then, when a request for a visual search for a learned definition is made, existing vision processors representing each part of the definition can be dynamically assembled into a vision pipeline capable of locating the target object(s). Learning new object instances, however, relies on at least one detector or validator capable of learning new object models on the fly. VIS does not impose restrictions on the underlying modeling approach, but methods to date have relied on global point cloud features (e.g., ViewpointFeatureHistogram as implemented in PCL [47]).

3.2.1 Cognitive Affordances

Affordance perception refers to the ability of an agent to extract meaning and usefulness from objects in its environment, often performed through perceptual (e.g., visual and haptic) analysis of object features [27, 98]. *Cognitive affordance* is a richer notion that extends traditional aspects of object functionality and action possibilities by incorporating the influence of non-perceptual aspects: changing context, social norms, historical precedence, and uncertainty. This allows for an increased flexibility with which to reason about affordances in a situated manner. For example, consider a knife, which offers grasp affordances across the entirety of its body, including the handle and blade (note: although one has to be careful when grasping a blade, it is nevertheless still possible, and therefore an affordance). However, the cognitive affordances of grasping offered by the same knife can vary depending on the context of the task (grasping by the handle when using it versus grasping by the blade when handing it over).

DIARC implements the current state-of-the-art formalism of cognitive affordances that uses a probabilistic logic-based approach [50, 49, 98], in which affordances are represented as condition-action rules (R), very much like production rules, in which the left-hand sides (LHS) represent perceptual invariants (F) in the environment, together with contextual information (C), and the right-hand sides (RHS) represent affordances (A) actualizable by the agent in the situation (e.g., the rule that one should grab a knife by the handle when using it would be translated by specifying the grasping parameters as F, the task context of "using a knife" as C and the constrained grasping location, together with other action parameters, as A). Affordance rules (R) take the overall form

$$r \stackrel{\text{def}}{=} f \wedge c \underset{[\alpha,\beta]}{\Longrightarrow} a,$$

where $f \in F$, $c \in C$, $a \in A$, $r \in R$, and $[\alpha,\beta] \subseteq [0,1]$. $[\alpha,\beta]$ is a confidence interval intended to capture the uncertainty associated with the truth of the affordance rule r such that if $\alpha = \beta = 1$, the rule is logically true, while $\alpha = 0$ and $\beta = 1$ assign maximum uncertainty to the rule. Similarly, each of the variables f and c also have confidence intervals associated with them, and are used for inferring affordances, as described in more detail below. Thus, rules can then be applied for a given feature percept f in a given context c to obtain the implied affordance a under uncertainty about f, c, and the extent to which they imply the presence of a. Currently, a Dempster-Shafer theoretic uncertainty-processing framework is used for reasoning with these probabilistic rules and inferring the confidence intervals [70].

The DIARC implementation is in the form of a separate affordance component (AFF) in combination with several other components. Given a set of affordance rules, AFF determines the subset of applicable rules by matching their left-hand sides given the current context and perceivable objects in the environment, together with their confidence intervals, and then determines the confidences on the fused right-hand sides (in case there are multiple rules with the same RHS) based on the inference and fusion algorithm in [49]. It uses the "confidence measure" λ defined in [44] to determine whether an inferred affordance should be realized and acted upon. For example, we could check the confidence of each affordance on its uncertainty interval $[\alpha_i,\beta_i]$: if $\lambda(\alpha_i,\beta_i) \leq \Lambda(c)$ (where $\Lambda(c)$ is an confidence threshold, possibly depending on context c), we do not have enough information to confidently accept the set of inferred affordances, and can thus not confidently use the affordances to guide action. However, even in this case, it might be possible to pass on the most likely candidates to other parts of the integrated system. Conversely, if $\lambda(\alpha_i,\beta_i) > \Lambda(c)$, then we take the inferred affordance to be certain enough to use it for further processing.

From a systems standpoint, in order to process cognitive affordances, two primary sub-components have been implemented [49]: (1) an Affordance Reasoning Sub-component (ARC), and (2) a Perceptual Semantics and Attention Control Sub-component (PAC). In addition, two supporting component-specific memories – Long-term Memory (LTM) and Working Memory (WM) – are needed for storing and updating logical affordance rules and related uncertainties. During inference, ARC

searches through all available affordance rules of the form specified above in the agent's LTM and populates WM with the relevant rules. Once the rules are in WM, both PAC and ARC can use these rules as the basis for perception and inference. as well as AFF works closely with sensory and perceptual systems (e.g., VIS) and other components in DIARC to coordinate perceptual and action processing. AFF is connected to the Goal Manager (GM/AM), and during the execution of actions, GM/AM sends affordance requests to AFF. These requests provide information about the current action to be performed and the context. AFF returns the specific perceptual features that need to be searched in the environment. This allows GM/AM to direct the attention of low-level perceptual systems like the vision component (VIS) and perform searches in a focused manner, only looking for perceptual features in the environment that are relevant to the applicable rules in AFF. The presence or absence of the searched perceptual features (along with perceptual uncertainty information) is passed back to AFF, which subsequently performs uncertain logical inferences (logical AND and modus ponens) on the rules. GM/AM is at the heart of DIARC and helps coordinate most goal-directed action. In dialogue-driven tasks, GM/AM is typically the recipient of processed language-based knowledge obtained via the natural language pipeline; instructions, questions, commands, and other utterances can flow through this NL pipeline to and from the GM/AM. Another recipient of language-based knowledge in DIARC is the belief component BEL. BEL maintains a history of all declarative knowledge passing through DIARC and is capable of performing various logically-driven knowledge-representation and inference tasks. Thus, it serves as a convenient holding-area for cognitive affordance information partially processed through the NL pipeline, which can then be retrieved and processed by AFF.

With the capability to perceive and learn cognitive affordances, the agent can learn normative behavior from instruction and immediately apply this newly acquired knowledge to the task at hand.

3.3 Natural Language Dialogues

Different from most CCAs, natural language understanding and generation for dialogue interactions is at the core of DIARC, and thus deeply integrated with other components not related to language. In the following sections, we briefly discuss the core language components and their interactions within and outside the language subsystem. The design of these architectural components is justified and inspired by a long tradition of empirical work at our laboratory, evaluating aspects of communication in both human-human teams (e.g., [26, 25, 23, 43]) and human-robot teams (e.g., [4, 14, 72, 81, 81, 82, 95]).

3.3.1 Speech Recognition

Speech is the most common way for natural language to be conveyed in interactions between humans and autonomous systems, especially when those systems are embodied in robots. The first step in understanding natural language in these interactions is understanding speech: what was said, and by whom. A speech recognizer that is part of a larger cognitive architecture has access to more and different types of information than a speech recognizer in isolation.

In DIARC, the ASR (Automatic Speech Recognition) component is responsible for recognizing speech input to the system. Its main role is to convert acoustic speech signals into a text representation. This text representation is the first step in understanding spoken natural language, and is the basis for the rest of the processioning done by language components. As technologies for performing automatic speech recognition improve, the techniques that the ASR component uses to perform speech recognition can be updated to reflect the state of the art. While the internal mechanisms of the ASR component may change, these changes do not affect the role that the component plays, or its interface with the rest of the architecture. Depending on the application of an instance of DIARC, the ASR component can be configured to operate in a variety of ways. This configuration is not limited to only the speech recognition process alone, but also includes the components to which ASR is connected, and the information it sends to them.

In "closed-world" task-driven dialogues, in which the lexicon of the interaction is known to all interactors before the interaction occurs, the ASR component can be configured to recognize only utterances that can be generated from this lexicon, given some grammar. Such configurations can be achieved by adding a specific user-defined grammar, e.g., a graph on top of the existing language model of a large vocabulary speech recognizer (LVCSR), to constrain its output to that grammar alone, and thus improve recognition rates.

In contrast, the ASR component can also be configured for "open-world" scenarios, in which the robot may hear new words that are not in its lexicon and must respond to their use in a timely fashion. For this purpose, an LVCSR that is able to not only recognize a large number of words, but also recognize when it has heard a word that is outside of its known vocabulary, can be employed to allow the system to identify when it has heard a word that it does not understand, and respond accordingly. For example, it may be desirable not only to identify words that the system has not heard before, but also to start learning about them. The ASR component can be configured to store words that it has not previously heard before, and recognize when it hears them again. This is achieved by adding a one-shot speech recognizer (OSSR) to the LVCSR already present in the ASR component. [64]. Forming a representation at this level is the first step in the process of learning a new word and its meaning. Being able to consistently recognize the word allows the rest of the language understanding components to begin to create a model of its meaning. This representation can also be used by the Speech Production component when the robot must speak about the new word to a human. The ASR component can be configured to connect to the Speech Production component and use its stored acoustic repre-

sentations of novel words to update the models in the production system, similarly to the way in which it updates the models in the recognition system [64].

The performance of the speech recognition mechanism in the ASR component can also be improved through connections with other components in the architecture and the information they can provide. This integration into an embodied system provides the ASR component with types of information that a speech recognizer in isolation could not have. One such integration is a configuration of DIARC in which the social context of the dialogue is used to bias the results of the speech recognizer [77]. Through a connection with the Dialogue Management component, the ASR component receives biasing signals for parts of its lexicon based on the position in the dialogue and the roles of the agents that are speaking. This integration improves the performance of the speech recognition mechanism used in ASR, and results in a system as a whole that models biological mechanisms.

3.3.2 Parsing

, The Parsing component, referred to as the Natural Language Processing (NLP) or Natural Language Understanding component (NLU), grounds the text of an utterance in a form that the rest of the components of DIARC can *understand*. To do this, the component must interpret the syntactic structure of the utterance, as well as its semantics. The semantic representation that is used throughout DIARC is logical predicates, so for a given utterance, the parser must produce a predicate expressions that represents its semantics.

The parsing component uses a parser that is throughly integrated with the rest of the architecture. This integration allows the parser to produce semantics of the correct form, as well as enhancing the capabilities of other components. The parser uses a dictionary of rules to interpret the utterances it receives. Each rule in the dictionary is composed of (1) the word in the lexicon to which it corresponds, (2) a syntactic definition of the word in Combinatory Categorial Grammar (CCG), and (3) a semantic definition of the word in lambda calculus. The lambda function in an entry generates all or part of a predicate whose meaning is grounded in formal expressions the rest of the system can understand. The syntactic rules determine how the lambda functions corresponding to the words in the utterance are applied in relation to each other [22].

The predicate expression created by the parser is the first notion of understanding of an utterance that is generated in DIARC. The predicates produced here are, after potential transformations by the Pragmatics and Reference Resolution components, the input to reasoning components like Dialogue, Action, Affordance and Vision. Accordingly, the representations generated by the parser must be interpretable by these other components. The Parser component can be configured with different sets of rules for different applications. The semantics it produces can be tailored to meet the representational requirements of any of the other components present in a given configuration of DIARC. This allows the system to have a universally understandable internal representation of knowledge, whose implementation can be

varied for the task at hand. Configurations of the system that are used in different tasks may require different semantics for the same utterance. The parser is able to be configured with different sets of rules so that the semantics of an utterance are always *understood*, regardless of the configuration of the architecture.

Like with the ASR component, in task-driven dialogue scenarios in which the lexicon of an interaction and its meaning are mutually understood by all of the participants, the parser can be configured with rules that guarantee understanding of any of the possible utterances the system might receive. In an open world, it is, again, not possible to have rules for every scenario the system may encounter. The Parsing component is equipped with mechanisms to generate representations of novel words as it encounters them. When a word is received from the ASR component that is not in the parser's set of rules, a new rule is generated for it. The syntax of the new word is inferred from its current usage, and is updated based on subsequent usages. The semantic representation of the word is also generated in conjunction with the syntax. The first time the word is heard, the portion of the semantic predicate for the utterance that it corresponds to does not have any meaning for the other components in the system. However, its meaning can be learned through the semantics of subsequent utterances, grounding the new semantic representation in the parser in the rest of the system [64, 17].

The parser performs syntactic and semantic parsing at the same time, which allows utterances to be understood incrementally as they unfold. This incremental understanding allows for the semantics of part of the utterance to begin to be understood before the utterance has been completely received by the system. This incremental parsing and understanding is especially useful in embodied human-robot interaction scenarios in which time is critical for interactions to appear natural. The incremental understanding of the parser component can be utilized by other components to improve their performance. For example, it can be used with the vision component to improve visual search speed [37] or a planner to update plans as new information is received [18]. Additionally, in many open-world settings, a robotic agent may not be able to completely parse an utterance, due to disfluencies in the interlocutors' speech, information loss, or an excess of novel terms that do not allow for successful inference of their meaning. In these cases, the ability to provide a partial parse, on the portion of the utterance that has been understood, and to not have the requirement of a complete utterance, allows the system to at least partially understand the utterance, which may be sufficient in some interactions [39].

The semantic representations that the parser generates are those that are required by the other components within `DIARC`, as they allow other components to perform further inference and understanding on what the system has heard. Some of the first interpretation of the predicate form of an utterance occurs in the Reference Resolution (RR) component. In order to properly understand referring expressions in an utterance, RR must be able to identify them. The semantic representation generated by the Parser component demarcates the portions of an utterance that contain referring expressions, and provides additional semantic information about the nature of the referring expression based on the syntax of the utterance [79].

3.3.3 Open-World Reference Resolution and Referring Expression Generation

DIARC's *Reference Resolution* (RR) and *Referring Expression Generation* (REG) components facilitate, respectively, the understanding and generation of referring expressions in uncertain and open worlds. To enable these capabilities, both components rely on a distributed, cognitively inspired memory model [78]. The base of this model is a set of *Consultants*, each of which provides access to a different architectural component that is viewed as a *distributed, heterogeneous knowledge base* [88] that can (1) provide access to a list of candidate referents; (2) advertise a list of logical predicates it can assess; (3) assess how probable it is that any of the listed candidate referents satisfy any of the advertised properties, and (4) hypothesize and assert knowledge regarding new candidate referents. This architecture is designed to provide access to knowledge of candidate referents regardless of their location and form of representation, facilitating a *domain independent* approach (cf. [83]).

In addition to this distributed model of Long Term Memory, the RR component has access to a set of hierarchically nested caches, inspired by the Givenness Hierarchy's conception of the Focus of Attention, Set of Activated Entities, and Set of Familiar Entities [28]. These caches provide fast access to likely referents during the resolution of anaphoric, deictic, and definite noun phrases [93]. When this *GH-theoretic* reference resolution process [94, 79] is unable to identify sufficiently likely candidate referents, a Long Term Memory query is performed using the *DIST-POWER* algorithm [88]. *DIST-POWER* is an adaptation of the *POWER* algorithm [87] and cognitive model [86], which uses the aforementioned Consultant framework to perform reference resolution (i.e., identify the targets of referring expressions used by the robot's interlocutors) when information is distributed across multiple architectural components. *POWER* performs reference resolution under uncertainty by effecting a search through the space of possible variable-entity assignments, incrementally computing the probability of assignments as they are built up, and pruning branches of the tree of assignments when their probability falls below a given threshold.

POWER improves on previous reference resolution approaches through its ability to handle open-worlds. If *POWER* is unable to find an acceptable set of candidate referents for a query involving n variables, it recurses, trying again using a relaxed query involving $n-1$ variables, with the removed variable selected on the basis of linguistic factors such as prepositional attachment and recency. This process repeats until a sufficiently probable set of candidate referents is found, or until all variables have been removed. Once this process terminates, new entities are hypothesized for all variables removed in this way, using the capabilities provided by the Consultant responsible for each new entity (according to its inferred type).

Just as these consultant capabilities are used to facilitate Reference Resolution, so too are they used to facilitate Referring Expression Generation, in which properties are selected to describe referents to which *the robot* wishes to refer. This is performed by the REG component using the *DIST-PIA* algorithm [91]. *DIST-PIA* is a version of the classic *Incremental Algorithm* [21], which uses the aforementioned

Consultant framework to perform Referring Expression Generation when information is uncertain and distributed across multiple architectural components [90].

3.3.4 Pragmatics

DIARC provides several alternate mechanisms for pragmatic understanding, in which the intentions underlying utterances made to the robot are inferred (e.g., the goals that the robot's interlocutor desires it to uptake [5]), and pragmatic generation, in which an utterance for communicating the robot's own intentions is abduced. These capabilities are crucial in order to understand and generate *indirect speech acts* [69], which we have shown in laboratory experiments to be commonly used in human-robot dialogue [14], especially in contexts with highly conventionalized social norms [95].

These capabilities are facilitated by a set of Speech Act theoretic [68] pragmatic rules, each of which maps a different utterance, under a different environmental or dialogue context, to a different candidate intention. One option is to use these rules directly, without accounting for uncertainty. In this case, during understanding, the first matching rule is used to determine the correct interpretation [11], while during generation, the utterances produced by all matching rules may be ranked and then voted upon [24].

Alternatively, *Dempster-Shafer (DS)* theoretic [70] rules, which are augmented with DS-theoretic uncertainty intervals [85], may be used to perform these tasks under uncertainty, in which case the results of all applicable rules are combined, yielding a *set* of candidate intentions or utterance forms, each of which is augmented with its own DS-theoretic uncertainty interval [80]. If, during the understanding process, the uncertainty intervals associated with the produced candidate intentions reflect a sufficiently high degree of uncertainty, a clarification request can immediately be constructed and issued [89]. We have shown in previous work how this process is able to produce clarification requests that resolve both referential and pragmatic uncertainty, and that align with human preferences [92].

3.3.5 Task-Based Dialogues

Utterances typically do not exist in isolation. Previously spoken utterances in a dialogue, and also an agent's mental model of the world, can permit the agent to deduce meaning from a given utterance beyond its semantics, even when considered in isolation. The Dialogue component and Belief component provide the agent with mechanisms to model the world. They allow the agent to understand/predict how other agents will act based on its own actions, as well as allowing it to better decide how it should act in a given context. The previously described natural language understating components (ASR, Parsing, Reference Resolution, Pragmatics) allow the agent to determine the semantics of an utterance, while the Dialogue and Belief components, allow it to deduce new information given those semantics.

To engage in a dialogue, an agent must be able to model its interlocutor(s), so that it fully understands the utterances it hears, and thus knows how to respond. To do this, the Dialogue and Belief components use explicit rules that represent relationships between groups of utterances, as well as relationships between utterances and the past, present, and future beliefs of agents. For task-based dialogue interactions in which information about the task can be known by the agent before engaging in the dialogue, two types of rules are used: rules about the effects of perceptions, actions and past beliefs on new or updated beliefs, and rules about the effects of types of utterances on beliefs. The Dialogue component uses these rules, in combination with the utterance semantics that it receives, to determine how the agent should respond. The rules relevant to the dialogue are part of the agent's set of beliefs about the world. They are stored in the Belief component, as are the rest of the agent's beliefs. These beliefs may originate from perceptions of the world, like understanding an utterance or performing a visual search, or may follow from the application of rules to perceptual semantics [10].

The Dialogue component determines how it should respond by monitoring the agent's beliefs in the Belief component. When it receives an utterance, it uses the information about the utterance's type (statement, questions, command, etc.), which has been determined by the other natural language understating components, to convert the utterance semantics into expressions that represent the agent's beliefs about the world. The new beliefs are asserted into the Belief component, which updates its state and performs inference based on the new information. Once the agent's beliefs have been updated, the Dialogue component considers its new belief state, which has resulted from the utterance, and determines the agent's response. The response manifests itself as the submission of a goal or goals to the Goal Manger which may, in turn, result in the submission of further goals [54, 62, 80].

4 Two Example Configurations of DIARC

In this section, we briefly describe two different configurations of DIARC: a single-agent configuration for one-shot learning of objects through natural language [64], and a multi-agent configuration that uses shared components to enable shared information and cognition between the agents. [45].

4.1 Learning Object Parts in One-Shot

The first example shows the DIARC configuration for a robot that can learn to recognize a new part of an object and use that knowledge to pick up the object by the newly learned part. Here, we assume that the robot already knows the object ("knife") and how to recognize it. A video of the interaction can be viewed here: http://bit.ly/2cfx3gL.

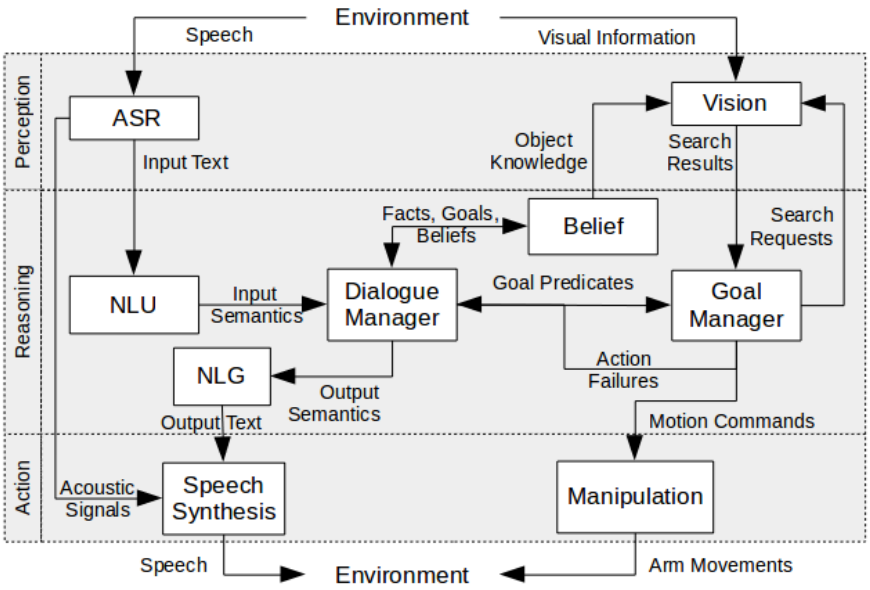

Fig. 1 The DIARC configuration for one-shot learning of objects and actions.

Human: Pick up the knife by the handle. Robot: OK.
Robot: But what is a handle? Human: The orange part of the knife is the handle. Robot: OK.
Human: Pick up the knife by the handle. Robot: OK.

When the ASR component recognizes the unknown word “handle” in the utterance “Pick up the knife by the handle”, it recreates a unique identifier “UT1” for it, which it then passes on to the NLU component. The NLU component infers the proper tag of speech from the lexical context and the semantic requirements for generating a grammatically correct command “pickUp(self,partOf(UT1,knife))”, which it forwards to the Dialogue Manager component. The Dialogue Manager component checks the command for indirect interpretations [11], and once it determines that it is a literal command, it acknowledges it by generating “OK” via the NLG and Speech Synthesis components. At the same time, it forwards the request to both the Belief component, which can generate relevant implications from the command that might, in turn, impact the execution, and then to the Goal Manager component as a new goal “pickUp(self,partOf(UT1,knife),leftArm)”. The Goal Manager component then begins to execute the “pickup” action, which requires it to convert the condensed description “partOf(UT1,knife))” to an expression “on(graspPoint,partOf(UT1,knife))” that it can send to the Vision component for processing. The Vision component, however, does not have any knowledge of “UT1”, hence the visual search for the appropriate grasp points fails, which, in turn,

causes a failure of the pickup action communicated to the Dialogue Manager component, which instructs the NLG and Speech Synthesis components to generate the question “But what is a handle?”. Upon hearing the definition “The orange part of the knife is the handle.”, the ASR component recognizes the word “handle”, which it had previously associated with “UT1”, and thus passes on “The orange part of the knife is the UT1.” to the NLU component, which, in turn, generates the semantics “is(UT1,partOf(orange,knife))”, and sends it again to the Dialogue Manager component. There, the utterance is recognized as a factual statement about a perceivable object and modified according to pragmatic rules to generate the form “looksLike(UT1,partOf(orange,knife))”, which is then passed on to the Belief component, and also acknowledged through the NLG and Speech Synthesis components (“OK”). The Belief component asserts the new fact to its knowledge base, thereby triggering a notification to the Vision component, which has requested to be notified of all facts of the form “looksLike(X,Y)”. When the robot is then instructed again to pick up the knife by the handle, the Vision component is now able to resolve the reference “UT1” (i.e., “handle”), as it has learned the grounding of “UT1” as “partOf(orange,knife))”. It finds a set of grasp points on the handle that it passes on to the Goal Manager component, which forwards the grasp constraints to the Manipulation component. The Manipulation component then selects the best grasp point to plan a trajectory for the robot’s arm to those points (a subsequent lift action then completes the “pickUp”).

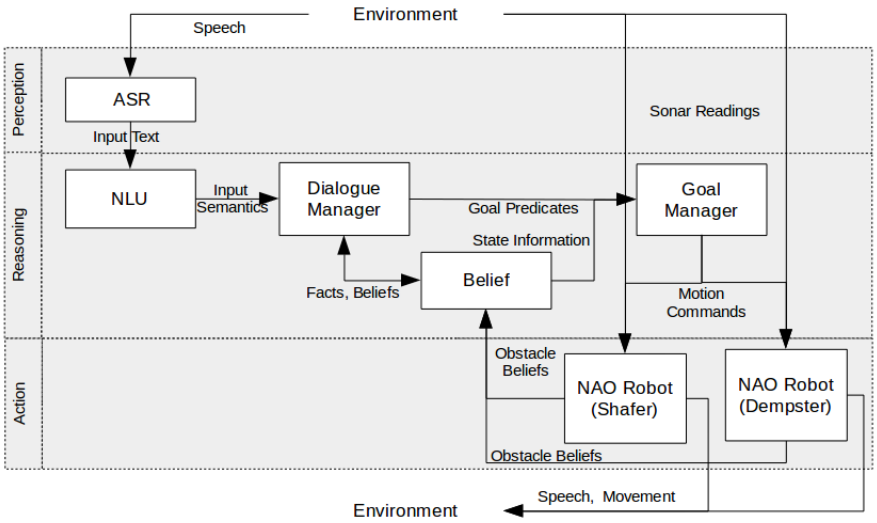

Fig. 2 The DIARC configuration for component-sharing among two Nao robots.

4.2 Sharing Components among Multiple Agents

The second example demonstrates the sharing of various architectural components so as to enable shared cognition, in this case, between two Nao robots called "Dempster" and "Shafer", although it works for any number of heterogeneous agents. A video of the interaction can be viewed here: `https://www.youtube.com/watch?v=JPufmIPHX9Y`.

Human: Hello Shafer. Robot: Hello.
Human: Hello Dempster. Robot: Hello.
Robot: Dempster, tell Shafer to stand. Human: Certainly, I will do that right away.

The human starts by greeting both robots, and the analysis of the message content is used to invoke the subset of components in the joined architecture corresponding to the respective robot. For example, the utterance "Hello Shafer", once transliterated by the ASR component, is analyzed in the NLU component as a greeting addressed to the Shafer robot, and is passed on as such to the Dialogue Manager component, which determines the correct dialogue move for Shafer to say "Hello" as well. As a result, it routes the greeting message through the Goal Manager component directly to the Shafer Nao component, which produces the speech output in the Shafer robot (the robot components include both motion and speech synthesis functionality). Similarly, greeting the Dempster robot will cause it to respond with "Hello". When the human instructor then addresses the Dempster robot with the request for Shafer to stand, the ASR component again passes on the text form of the utterance on to the NLU component, which generates the semantics "want(human,Dempster,do(tell(Dempster,Shafer,do(Shafer,stand)))". This command is then sent to the Dialogue Manager component, which forwards the semantics to the Belief component, where it is asserted while generating an acknowledging dialogue move "Certainly, I will do that right away", which gets forwarded to the Dempster Nao component for speech synthesis (note that the Dialogue Manager component can determine the appropriate robot component from the pragmatic information about the addressee in the dialogue).

When new information is asserted in the Belief component, it may result in new goals being submitted to the Goal Manager component, in which case the Dialogue Manager is also informed of the new goal. In this case, there is a new goal for the Dempster robot: "tell(Dempster,Shafer,do(Shafer,stand))". When the Goal Manager component executes the "tell(X,Y,Z)" action using the bindings of X=Dempster, Y=Shafer, Z=do(Shafer,stand), its generates and submits the new subgoal "do(Shafer,stand)", which has the Shafer robot as the actor and the "stand" as the action. Execution then triggers the "stand" action in the Shafer Nao component, causing the robot to stand up.

Analogous to issuing commands, it is also possible to inquire about an agent's perceptions or knowledge via another agent. Such mediated interactions are automatically enabled by the sharing agent's Belief and Goal Manager components.

5 Applications

DIARC has been used on a variety of virtual and robotic agents in a great variety of contexts (e.g., [84, 51, 53, 88, 96, 13, 87, 19, 80, 74, 38, 61, 83, 10, 8, 62]). Like other cognitive architectures, it has also been integrated with other architecture (e.g., ACT-R, Soar, and Icarus [75] and Vulcan [84]. Most importantly, it has been employed in many human-robot interaction experiments, with both autonomous as well as teleoperated robots (e.g., [79, 82, 7, 16, 15, 12, 73, 97, 9, 57, 20, 58, 6, 67]). More relevant to this chapter, DIARC has also been used to model aspects of human cognition (e.g., [2, 67, 6, 55, 62, 37, 65, 48]). Here, we will only be able to provide a short summary of recent experimental and modeling work using DIARC.

5.1 HRI Experiments with DIARC

Recent empirical investigations of human-robot teams have largely focused on humans' use of indirect language when interacting with robots. This work has demonstrated (1) that humans will regularly use indirect language during the course of human-robot interactions [14], (2) that humans use indirect language when interacting with teleoperated and autonomous robots with a frequency similar to that used when interacting with other humans [4], (3) and that indirect language use is increased in contexts with highly conventionalized social norms [95]. Indirect language is notably used by humans in order to adhere to social norms such as *politeness* [73]. We have also investigated human perception of the use of polite language by robots. This work has shown that politeness not only increases human ratings of robot likability, but that this effect is significantly stronger for women than it is for men [72].

Our empirical work has also demonstrated differences in how robots' morphology and communication style may have significant impact on team dynamics; our investigation of autonomous vs. teleoperated robots suggested that humans perceived teleoperated robots to be less intelligent than co-present human teammates [4]; our investigation of verbally vs. silently communicating robots suggested that humans find silently communicating robots to be significantly creepier than verbally communicating robots (at least for the communication of task-dependent, human-understandable information among robots co-located with human teammates in a cooperative setting) [81, 82].

5.2 Cognitive Modeling with DIARC

Even though DIARC was never designed to be a model of the human mind, and thus was never intended to be a modeling framework for human cognition, it affords unique modeling capabilities due its real-time, embodied nature and integrated nat-

ural language understanding capabilities, and has thus been used for various types of (mostly qualitative) models over the years. Early models of incremental natural language processing demonstrate the incremental integration of perceptual context in the resolution and generation of references with ambiguities due to prepositional attachment (e.g., [60, 6]), as well as models of incremental word substitution for correcting phonetic errors (e.g., [6]). Later models of incremental natural language processing focused on natural language-guided biasing of visual spatial attention (e.g., [37]) and models of human-like task-based dialogues (e.g., [62]). Particularly notable is a model of natural-language guided conjunctive visual search that was fit to human data in a novel way and used to clarify possible explanations of observed empirical data [65].

Additional modeling work investigated the interaction between affect and cognition, in particular, the effect of mood states on goal management and ways to bias goals (e.g., [56]), as well as to modulate affect in the speech (e.g., [67]).

Most recently, DIARC has also been used to demonstrate human infant word learning in a cross-situational embodied context (e.g., [48]), as well as human-like norm-learning (e.g., [52]).

6 Conclusion

DIARC, as an architecture scheme, is neither a finished product nor does it aspire to be one. Rather, its purpose is to provide researchers with an expanding framework for exploring the functional and architectural design trade-offs of different types of autonomous agents. By being flexible in its instantiations, it allows for custom configurations of classes of systems targeted at particular physical platforms and classes of tasks. By being flexible in its component algorithms, it allows for the easy integration of novel algorithms, and thus provides researchers not interested in the development of cognitive systems with an evaluation platform. Ongoing work on DIARC is aimed at its unique contributions compared to classical cognitive architectures: the open-world aspects, one-shot learning, component-sharing, introspection mechanisms and integration with ADE middleware. The goal is not only to improve existing functionality for the investigation of more sophisticated algorithms and involved architectural features, but to also provide a robust implementation platform for future autonomous robot applications that allow for human-like task-based interactions in natural language dialogues.

Acknowledgements The work on DIARC has been supported by various research grants from the US National Science Foundation and the US Office of Naval Research over the years, most recently by NSF grant IIS1316809.and ONR grant N00014-16-1-2278.

References

1. Virgil Andronache and Matthias Scheutz. ADE - a tool for the development of distributed architectures for virtual and robotic agents. In *Proceedings of the Fourth International Symposium "From Agent Theory to Agent Implementation"*, pages 606–611, 2004.
2. Virgil Andronache and Matthias Scheutz. Integrating theory and practice: The agent architecture framework APOC and its development environment ADE. In *Proceedings of AAMAS 2004*, pages 1014–1021. ACM Press, 2004.
3. Virgil Andronache and Matthias Scheutz. ADE - an architecture development environment for virtual and robotic agents. *International Journal of Artificial Intelligence Tools*, 15(2):251–286, 2006.
4. Maxwell Bennett, Tom Williams, Daria Thames, and Matthias Scheutz. Differences in interaction patterns and perception for teleoperated and autonomous humanoid robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.
5. Timothy Brick, Paul Schermerhorn, and Matthias Scheutz. Speech and action: Integration of action and language for mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1423–1428. IEEE, 2007.
6. Timothy Brick and Matthias Scheutz. Incremental natural language processing for hri. In *Proceedings of the Second ACM IEEE International Conference on Human-Robot Interaction*, pages 263–270, Washington D.C., March 2007.
7. Gordon Briggs, Ian McConnell, , and Matthias Scheutz. When robots object: Evidence for the utility of verbal, but not necessarily spoken protest. In *Proceedings of the 7th International Conference on Social Robotics*. 2015.
8. Gordon Briggs and Matthias Scheutz. Facilitating mental modeling in collaborative human-robot interaction through adverbial cues. In *Proceedings of the SIGDIAL 2011 Conference*, pages 239–247, Portland, Oregon, June 2011.
9. Gordon Briggs and Matthias Scheutz. Investigating the effects of robotic displays of protest and distress. In *Proceedings of the 2012 Conference on Social Robotics*, LNCS, pages 238–247. Springer, 2012.
10. Gordon Briggs and Matthias Scheutz. Multi-modal belief updates in multi-robot human-robot dialogue interaction. In *Proceedings of 2012 Symposium on Linguistic and Cognitive Approaches to Dialogue Agents*, 2012.
11. Gordon Briggs and Matthias Scheutz. A hybrid architectural approach to understanding and appropriately generating indirect speech acts. In *Proceedings of the twenty-seventh AAAI Conference on Artificial Intelligence*, 2013.
12. Gordon Briggs and Matthias Scheutz. How robots can affect human behavior: Investigating the effects of robotic displays of protest and distress. *International Journal of Social Robotics*, 6:1–13, 2014.
13. Gordon Briggs and Matthias Scheutz. "sorry, i can't do that:" developing mechanisms to appropriately reject directives in human-robot interactions. In *Proceedings of the 2015 AAAI Fall Symposium on AI and HRI*. 2015.
14. Gordon Briggs, Tom Williams, and Matthias Scheutz. Enabling robots to understand indirect speech acts in task-based interactions. *Journal of Human-Robot Interaction (JHRI)*, 2017.
15. Priscilla Briggs, Matthias Scheutz, and Linda Tickle-Degnen. Reactions of people with parkinson's disease to a robot interviewer. In *Proceedings of the Workshop on Assistive Robotics for Individuals with Disabilities at IROS 2014*. 2014.
16. Priscilla Briggs, Matthias Scheutz, and Linda Tickle-Degnen. Are robots ready for administering health status surveys: First results from an hri study with subjects with parkinsons disease. In *Proceedings of 10th ACM/IEEE International Conference on Human-Robot Interaction*. 2015.
17. Rehj Cantrell, Paul Schermerhorn, and Matthias Scheutz. Learning actions from human-robot dialogues. In *Proceedings of the 2011 IEEE Symposium on Robot and Human Interactive Communication*, July 2011.

18. Rehj Cantrell, Kartik Talamadupula, Paul Schermerhorn, J. Benton, Subbarao Kambhampati, and Matthias Scheutz. Tell me when and why to do it!: Run-time planner model updates via natural language instruction. In *Proceedings of the 2012 Human-Robot Interaction Conference*, Boston, MA, March 2012.
19. Tathagata Chakraborti, Gordon Briggs, Kartik Talamadupula, Yu Zhang1, Matthias Scheutz, David Smith, and Subbarao Kambhampati1. Planning for serendipity. In *Proceedings of IROS*. 2015.
20. Charles Crowell, Matthias Scheutz, Paul Schermerhorn, and Michael Villano. Gendered voice and robot entities: Perceptions and reactions of male and female subjects. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, October 2009.
21. Robert Dale and Ehud Reiter. Computational interpretations of the gricean maxims in the generation of referring expressions. *Cognitive science*, 19(2):233–263, 1995.
22. Juraj Dzifcak, Matthias Scheutz, Chitta Baral, and Paul Schermerhorn. What to do and how to do it: Translating natural language directives into temporal and dynamic logic representation for goal management and action execution. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA '09)*, Kobe, Japan, May 2009.
23. Kathleen Eberhard, Hannele Nicholson, Sandra Kuebler, Susan Gundersen, and Matthias Scheutz. The indiana cooperative remote search task (crest) corpus. In *Proceedings of LREC 2010: Language Resources and Evaluation Conference*, Malta, May 2010.
24. Felix Gervits, Gordon Briggs, and Matthias Scheutz. The pragmatic parliament: A framework for socially-appropriate utterance selection in artificial agents. In *Proceedings of the 39th annual meeting of the Cognitive Science Society (COGSCI)*, 2017.
25. Felix Gervits, Kathleen Eberhard, and Matthias Scheutz. Disfluent but effective? a quantitative study of disfluencies and conversational moves in team discourse. In *Proceedings of the 26th International Conference on Computational Linguistics*, 2016.
26. Felix Gervits, Kathleen Eberhard, and Matthias Scheutz. Team communication as a collaborative process. *Frontiers in Robotics and AI*, 3:62, 2016.
27. J J Gibson. *The ecological approach to visual perception*, volume 39. 1979.
28. Jeanette K Gundel, Nancy Hedberg, and Ron Zacharski. Cognitive status and the form of referring expressions in discourse. *Language*, pages 274–307, 1993.
29. Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature Reviews: Neuroscience*, pages 194–203, March 2001.
30. JAUS. Jaus.
31. J. Kramer, M. Scheutz, J. Brockman, and P. Kogge. Facing up to the inevitable: Intelligent error recovery in massively parallel processing in memory architectures. In Hamid R. Arabnia, editor, *International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 227–233, Las Vegas, 2006.
32. James Kramer and Matthias Scheutz. ADE: A framework for robust complex robotic architectures. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4576–4581, Bejing, China, October 2006.
33. James Kramer and Matthias Scheutz. ADE: Filling a gap between single and multiple agent systems. In *Proceedings of the ACE 2004 Symposium at the 18th European Meeting on Cybernetics and Systems Research*, Vienna, Austria, 2006.
34. James Kramer and Matthias Scheutz. Reflection and reasoning mechanisms for failure detection and recovery in a distributed robotic architecture for complex robots. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, pages 3699–3704, Rome, Italy, April 2007.
35. James Kramer and Matthias Scheutz. Robotic development environments for autonomous mobile robots: A survey. *Autonomous Robots*, 22(2):101–132, 2007.
36. James Kramer, Matthias Scheutz, and Paul Schermerhorn. 'talk to me!': Enabling communication between robotic architectures and their implementing infrastructures. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3044–3049, San Diego, CA, October/November 2007.

37. Evan Krause, Rehj Cantrell, Ekaterina Potapova, Michael Zillich, and Matthias Scheutz. Incrementally biasing visual search using natural language input. In *Proceedings of AAMAS*, pages 31–38, 2013.
38. Evan Krause, Michael Zillich, Tom Williams, and Matthias Scheutz. Learning to recognize novel objects in one shot through human-robot interactions in natural language dialogues. In *Proceedings of Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.
39. S. Kuebler, R. Cantrell, and M. Scheutz. Actions speak louder than words: Evaluating parsers in the context of natural language understanding systems for human-robot interaction. In *Proceedings of RANLP*, pages 56–62, 2011.
40. J. E. Laird, C. Lebiere, and P. Rosenbloom. A standard model of the mind: Toward a common computational framework across artificial intelligence, cognitive science, neuroscience, and robotics. *AI Magazine*, forthcoming.
41. Pat Langley, John E. Pat, and Seth Rogers. Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2):141–160, June 2009.
42. Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. Yarp: Yet another robot platform. *International Journal on Advanced Robotics Systems*, 3:43–48, 2006.
43. H. Nicholson, K. Eberhard, and M. Scheutz. Um...i don't see any: The function of filled pauses and repairs. In *Proceedings of 5th Workshop on Disfluency in Spontaneous Speech*, pages 89–92, 2010.
44. R. C. Nunez, R. Dabarera, M. Scheutz, G. Briggs, O. Bueno, K. Premaratne, and M. N. Murthi. DS-based uncertain implication rules for inference and fusion applications. *16th International Conference on Information Fusion (FUSION)*, pages 1934–1941, 2013.
45. Bradley Oosterveld, Luca Brusatin, and Matthias Scheutz. Two bots, one brain: Component sharing in cognitive robotic architectures. In *Proceedings of 12th ACM/IEEE International Conference on Human-Robot Interaction Video Contest*, 2017.
46. Morgan Quigley, Kenneth Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. Ros: an open-source robot operating system. In *Proceedings of ICRA Workshop on Open Source Software*, 2009.
47. Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.
48. Sepideh Sadeghi, Matthias Scheutz, and Evan Krause. An embodied incremental bayesian model of cross-situational word learning. In *proceedings of the 2017 Joint IEEE International Conference on Development and Learning and Epigenetic Robotics (ICDL-EpiRob)*, 2017.
49. Vasanth Sarathy and Matthias Scheutz. A Logic-based Computational Framework for Inferring Cognitive Affordances. *IEEE Transactions on Cognitive and Developmental Systems*, PP(99):1–1, 2016.
50. Vasanth Sarathy and Matthias Scheutz. Cognitive Affordance Representations in Uncertain Logic. In *Proceedings of the 15th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2016.
51. Vasanth Sarathy and Matthias Scheutz. A logic-based computational framework for inferring cognitive affordances. *IEEE Transactions on Cognitive and Developmental Systems*, 8(3), 2016.
52. Vasanth Sarathy, Matthias Scheutz, Joseph Austerweil, Yoed Kenett, Mowafak Allaham, and Bertram Malle. Mental representations and computational modeling of context-specific human norm systems. In *Proceedings of the 39th Annual Meeting of the Cognitive Science Society*, 2017.
53. Vasanth Sarathy, Jason Wilson, Thomas Arnold, and Matthias Scheutz. Enabling basic normative hri in a cognitive robotic architecture. In *Proceedings of the 2nd workshop on Cognitive Architectures for Social Human-Robot Interactionat the 11th ACM/IEEE Conference on Human-Robot Interaction*, 2016.
54. Paul Schermerhorn and Matthias Scheutz. Natural language interactions in distributed networks of smart devices. *International Journal of Semantic Computing*, 2(4):503–524, 2008.

55. Paul Schermerhorn and Matthias Scheutz. Dynamic robot autonomy: Investigating the effects of robot decision-making in a human-robot team task. In *Proceedings of the 2009 International Conference on Multimodal Interfaces*, Cambridge, MA, November 2009.
56. Paul Schermerhorn and Matthias Scheutz. The utility of affect in the selection of actions and goals under real-world constraints. In *Proceedings of the 2009 International Conference on Artificial Intelligence*, July 2009.
57. Paul Schermerhorn and Matthias Scheutz. Disentangling the effects of robot affect, embodiment, and autonomy on human team members in a mixed-initiative task. In *Proceedings of the 2011 International Conference on Advances in Computer-Human Interactions*, pages 236–241, Gosier, Guadeloupe, France, February 2011.
58. Paul Schermerhorn, Matthias Scheutz, and Charles R. Crowell. Robot social presence and gender: Do females view robots differently than males? In *Proceedings of the Third ACM IEEE International Conference on Human-Robot Interaction*, pages 263–270, Amsterdam, The Netherlands, 2008. ACM Press.
59. M. Scheutz. ADE - steps towards a distributed development and runtime environment for complex robotic agent architectures. *Applied Artificial Intelligence*, 20(4-5), 2006.
60. Matthias Scheutz and Virgil Andronache. Architectural mechanisms for dynamic changes of behavior selection strategies in behavior-based systems. *IEEE Transactions of System, Man, and Cybernetics Part B: Cybernetics*, 34(6):2377–2395, 2004.
61. Matthias Scheutz, Gordon Briggs, Rehj Cantrell, Evan Krause, Tom Williams, and Richard Veale. Novel mechanisms for natural human-robot interactions in the diarc architecture. In *Proceedings of AAAI Workshop on Intelligent Robotic Systems*, 2013.
62. Matthias Scheutz, Rehj Cantrell, and Paul Schermerhorn. Toward humanlike task-based dialogue processing for human robot interaction. *AI Magazine*, 32(4):77–84, 2011.
63. Matthias Scheutz, Jack Harris, and Paul Schermerhorn. Systematic integration of cognitive and robotic architectures. *Advances in Cognitive Systems*, pages 277–296, 2013.
64. Matthias Scheutz, Evan Krause, Brad Oosterveld, Tyler Frasca, and Robert Platt. Spoken instruction-based one-shot object and action learning in a cognitive robotic architecture. In *Proceedings of the 16th International Conference on Autoomous Agents and Multiagent Systems*, 2017.
65. Matthias Scheutz, Evan Krause, and Sepideh Sadeghi. An embodied real-time model of language-guided incremental visual search. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society*, 2014.
66. Matthias Scheutz, Paul Schermerhorn, James Kramer, and David Anderson. First steps toward natural human-like HRI. *Autonomous Robots*, 22(4):411–423, May 2007.
67. Matthias Scheutz, Paul Schermerhorn, James Kramer, and C. Middendorff. The utility of affect expression in natural language interactions in joint human-robot tasks. In *Proceedings of the 1st ACM International Conference on Human-Robot Interaction*, pages 226–233, 2006.
68. John R Searle. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press, 1969.
69. John R Searle. Indirect speech acts. *Syntax and semantics*, 3:59–82, 1975.
70. Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.
71. Aaron Sloman and Matthias Scheutz. A framework for comparing agent architectures. In *UK Workshop on Computational Intelligence*, pages 169–176, 2002.
72. Megan Strait, Priscilla Briggs, and Matthias Scheutz. Gender, more so than age, modulates positive perceptions of language-based human-robot interaction. In *4th International Symposium on New Frontiers in Human-Robot Interaction, AISB*. 2015.
73. Megan Strait, Cody Canning, and Matthias Scheutz. Let me tell you! investigating the effects of robot communication strategies in advice-giving situations based on robot appearance, interaction modality, and distance. In *Human-Robot Interaction (HRI)*, pages 479–486, 2014.
74. Kartik Talamadupula, Gordon Briggs, Tathagata Chakraborti, Matthias Scheutz, and Subbarao Kambhampati. Coordination in human-robot teams using mental modeling and plan recognition. In *Proceedings of IROS*. 2014.

75. Kartik Talamadupula, Gordon Briggs, Matthias Scheutz, and Subbarao Kambhampti. Architectural mechanisms for handling human instructions for open-world mixed-initiative team tasks and goals. *Advances in Cognitive Systems*, 5, 2017.
76. J. Gregory Trafton, Laura M. Hiatt, Anthony M. Harrison, Franklin, P. Tamborello, Sangeet II, S. Khemlani, and Alan C. Schultz. ACT-R/e: An embodied cognitive architecture for human-robot interaction. *Journal of Human-Robot Interaction*, 1(1):78–95, 2012.
77. Richard Veale, Gordon Briggs, and Matthias Scheutz. Linking cognitive tokens to biological signals: Dialogue. In *Proceedings of the 35th Annual Conference of the Cognitive Science Society*, page forthcoming, Austin, TX, 2013. Cognitive Science Society.
78. Tom Williams. A consultant framework for natural language processing in integrated robot architectures. *IEEE Intelligent Informatics Bulletin (IIB)*, pages 10–14, 2017.
79. Tom Williams, Saurav Acharya, Stephanie Schreitter, and Matthias Scheutz. Situated open world reference resolution for human-robot dialogue. In *Proceedings of the 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2016.
80. Tom Williams, Gordon Briggs, Bradley Oosterveld, and Matthias Scheutz. Going beyond command-based instructions: Extending robotic natural language interaction capabilities. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI)*, 2015.
81. Tom Williams, Priscilla Briggs, Nathaniel Pelz, and Matthias Scheutz. Is robot telepathy acceptable? investigating effects of nonverbal robot-robot communication on human-robot interaction. In *Proceedings of 23rd IEEE Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2014.
82. Tom Williams, Priscilla Briggs, and Matthias Scheutz. Covert robot-robot communication: Human perceptions and implications for human-robot interaction. *Journal of Human-Robot Interaction (JHRI)*, 2015.
83. Tom Williams, Rehj Cantrell, Gordon Briggs, Paul Schermerhorn, and Matthias Scheutz. Grounding natural language references to unvisited and hypothetical locations. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence (AAAI)*, 2013.
84. Tom Williams, Collin Johnson, Matthias Scheutz, and Benjamin Kuipers. A tale of two architectures: A dual-citizenship integration of natural language and the cognitive map. In *Proceedings of the 16th International Conference on Autonomous Agents and Multi-Agent Systems*, 2017.
85. Tom Williams, Rafael C Núñez, Gordon Briggs, Matthias Scheutz, Kamal Premaratne, and Manohar N Murthi. A dempster-shafer theoretic approach to understanding indirect speech acts. In *Advances in Artificial Intelligence– Proceedings of the 14th Ibero-American Conference on AI (IBERAMIA)*, 2014.
86. Tom Williams and Matthias Scheutz. A domain-independent model of open-world reference resolution. In *Proceedings of the 37th annual meeting of the Cognitive Science Society (COGSCI)*, 2015.
87. Tom Williams and Matthias Scheutz. Power: A domain-independent algorithm for probabilistic, open-world entity resolution. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.
88. Tom Williams and Matthias Scheutz. A framework for resolving open-world referential expressions in distributed heterogeneous knowledge bases. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI)*, 2016.
89. Tom Williams and Matthias Scheutz. Resolution of referential ambiguity using dempster-shafer theoretic pragmatics. In *Proceedings of the AAAI Fall Symposium on AI for HRI (AI-HRI)*, 2016.
90. Tom Williams and Matthias Scheutz. Referring expression generation under uncertainty: Algorithm and evaluation framework. In *Proceedings of the 10th International Conference on Natural Language Generation (INLG)*, 2017.
91. Tom Williams and Matthias Scheutz. Referring expression generation under uncertainty in integrated robot architectures. In *Proceedings of the Robotics: Science and Systems Workshop on Human-Centered Robotics: Interaction, Physiological Integration and Autonomy*, 2017.

92. Tom Williams and Matthias Scheutz. Resolution of referential ambiguity in human-robot dialogue using dempster-shafer theoretic pragmatics. In *Proceedings of Robotics: Science and Systems (RSS)*, 2017.
93. Tom Williams and Matthias Scheutz. Reference resolution in robotics: A givenness hierarchy theoretic approach. In Jeanette Gundel and Barbara Abbott, editors, *The Oxford Handbook of Reference*. Oxford University Press, 2018 (Forthcoming).
94. Tom Williams, Stephanie Schreitter, Saurav Acharya, and Matthias Scheutz. Towards situated open-world reference resolution. In *Proceedings of the AAAI Fall Symposium on AI for HRI (AI-HRI)*, 2015.
95. Tom Williams, Daria Thames, Julia Novakoff, and Matthias Scheutz. thank you for sharing that interesting fact!: Effects of capability and context on indirect speech act use in task-based human-robot dialogue. In *Proceedings of the 13th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2018.
96. Jason R. Wilson, Evan Krause, Matthias Scheutz, and Morgan Rivers. Analogical generalization of actions from single exemplars in a robotic architecture. In *Proceedings of AAMAS 2016*, 2016.
97. Chen Yu, Paul Schermerhorn, and Matthias Scheutz. Adaptive eye gaze patterns in interactions with human and artificial agents. *ACM Transactions on Interactive Intelligent Systems*, 1(2):13, 2012.
98. Philipp Zech, Simon Haller, Safoura Rezapour Lakani, Barry Ridge, Emre Ugur, and Justus Piater. Computational models of affordance in robotics: a taxonomy and systematic classification. *Adaptive Behavior*, 25(5):235–271, 2017.