MIDCA: A Metacognitive, Integrated Dual-Cycle Architecture for Self-Regulated Autonomy

Michael T. Cox, Zohreh Alavi, Dustin Dannenhauer,*
Vahid Eyorokon, Hector Munoz-Avila,* and Don Perlis§

Wright State University
Dayton, OH 45435
michael.cox@wright.edu

*Lehigh University
Bethlehem, PA 18015
hem4@lehigh.edu

§University of Maryland
College Park, MD 20742
perlis@cs.umd.edu

Abstract

We present a metacognitive, integrated, dual-cycle architecture whose function is to provide agents with a greater capacity for acting robustly in a dynamic environment and managing unexpected events. We present MIDCA 1.3, an implementation of this architecture which explores a novel approach to goal generation, planning and execution given surprising situations. We formally define the mechanism and report empirical results from this goal generation algorithm. Finally, we describe the similarity between its choices at the cognitive level with those at the metacognitive.

Introduction

Research on cognitive architectures have made significant contributions over the years including the ability to reason with multiple knowledge modes (Laird 2012), to introspectively examine the rationale for a decision (Forbus, Klenk and Hinrichs 2009), and the ability to learn knowledge of varied levels of abstraction (Langley and Choi 2006). Comparatively less research efforts examine the metacognitive contributions to effective decision-making and behavior. We have worked to develop a comprehensive theory of cognition and metacognition by proposing a *metacognitive, integrated dual-cycle architecture (MIDCA)* and applying an implementation of this architecture to various problem-solving domains. The dual-cycle architecture integrates a problem-solving and comprehension loop at the cognitive level with a control and monitoring loop at the metacognitive level. We use a standard planner for both problem-solving and control functions, while we concentrate most of our work on the comprehension and monitoring processes.

For each of the object and meta-level cycles, we use a *note-assess-guide procedure* (Anderson, Oates, Chong and Perlis 2006; Paisner, Maynord, Cox and Perlis 2013). The *note* phase detects discrepancies, the *assess* phase hypothesizes causes for discrepancies, and the *guide* phase performs a suitable response. This research will enable more robust behavior in autonomous intelligent systems because the capabilities lead both to recovery in the face of surprise and to more effective learning (i.e., by triggering opportunities to fill gaps in the agent's own knowledge).

Autonomy has been studied from many perspectives, and the literature is replete with differing approaches to this topic. The results of autonomy are often some mechanism by which we automate system behavior and decision-making computationally. We claim that for a system to exhibit self-regulated autonomy, however, it must have a model of itself in addition to the usual model of the world. Like self-regulated learning (e.g., Bjork, Dunlosky and Kornell 2013), whereby a learner manages the pace, resources, and goals of learning, *self-regulated autonomy* involves a system that similarly manages the parameters of problem-solving and behavior by understanding itself and its actions in context of its overall environment and the goals it is trying to achieve.

Here we introduce the MIDCA architecture within which we examine issues of autonomy and reasoning about goals. We provide a new formalism that places this work within the formal context of AI planning research and goal reasoning. We then report empirical results from the MIDCA cognitive cycle and early results from the metacognitive cycle. After a summary of related research, we conclude with brief comments and a pointer to future research directions.

MIDCA and Goal Reasoning

Recent work on *goal reasoning* (Aha, Cox, and Munoz-Avila 2013; Hawes, 2011) has started to examine how intelligent agents can reason about and generate their own goals instead of always depending upon a human user directly. Broadly construed, the topic of goal reasoning concerns cognitive systems that can self-manage their goals (Vattam, Klenk, Molineaux, and Aha 2013). MIDCA's decision-making process has significant focus on goal formulation and goal change, and as such, draws much from the work on goal reasoning.

MIDCA consists of "action-perception" cycles at both the cognitive (i.e., object) and the metacognitive (i.e., meta) levels. The output side (i.e., problem-solving) of each cycle consists of intention, planning, and action execution, whereas the input side (i.e., comprehension) consists of perception, interpretation, and goal evaluation (see Figure 1). A cycle selects a goal and commits to achieving it. MIDCA then creates a plan to achieve the goal and subsequently executes the planned actions to make the domain match the goal state. The agent perceives changes to the environment resulting from the actions, interprets the percepts with respect to the plan, and evaluates the interpretation with respect to the goal. At the object level, the cycle achieves goals that change the environment. At the meta-level, the cycle achieves goals that change the object level. That is, the metacognitive "perception" components introspectively monitor the processes and mental state changes at the cognitive level. The "action" component consists of a meta-level controller that mediates reasoning over an abstract representation of the object level cognition.

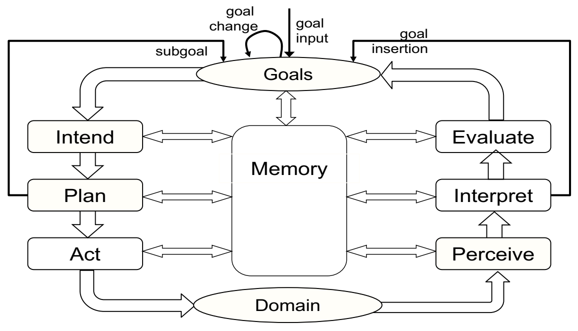

Figure 1. Schematic action-perception cycle for both object and meta-levels

MIDCA Version 1.3

The MIDCA_1.3 model includes a complete planning-acting and perception-interpretation cycle at the cognitive level, and it incorporates with this an interface to the environment.[1] This interface interacts with either a simple world simulator (e.g., blocksworld) or the ROS middleware for robotic applications (Quigley *et al*. 2009). The planning component integrates the SHOP2 hierarchical network planner (Nau *et al*. 2003). We have also integrated the XPLAIN/Meta-AQUA (Burstein *et al*. 2008; Cox and Ram 1999) multistrategy explanation system into the interpretation component. Our standard simulator takes actions from the planner, calculates the changes to the world, and then passes the resulting state to the comprehension side of the cycle. The ROS interface is similar and briefly explained below. Comprehension examines the input for anomalies and formulates new goals for the planner as warranted. Figure 2 diagrams the basic implementation.

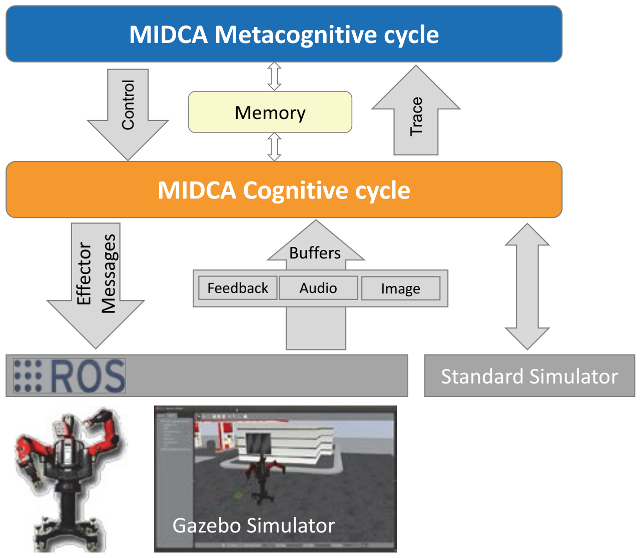

Figure 2. Block diagram of overall MIDCA_1.3 system

MIDCA ROS Interface

We added an application programming interface (API) to MIDCA_1.3 to communicate with ROS and a Baxter (Fitzgerald 2013) humanoid robot. This interface coheres to the principles outlined in Scheutz, Harris and Schemerhorn (2013).[2] It is responsible for sending messages to ROS as requested by MIDCA, and for placing messages received in appropriate queues for MIDCA to process (again, see Figure 2). We created other ROS nodes which are responsible for doing specific actions, such as moving the Baxter's arms, and for getting object representations. These communicate with MIDCA through the API.

In this API, the types of ingoing and outgoing messages on each ROS topic and their meaning is specified. As these messages are asynchronously received, a set of MIDCA handlers put them in appropriate buffers within a partition

[1] The full source code with documentation and examples is available at https://github.com/mclumd/MIDCA

[2] See http://tinyurl.com/oauahz4 for a video with an example interaction.

of MIDCA memory. During the Perceive phase (Figure 1), these messages will be accessed and stored in MIDCA's main memory.

In the Plan phase, after it creates a high level plan for the selected goal, it operationalizes each action using a mapping between high-level actions and directly executable methods for the robot. For example, the high level action *reach(object)* is instantiated in a method which sends out a ROS message to the node which operates the arm, then repeatedly checks for feedback indicating success or failure. Once all actions in a plan are complete, the plan itself is considered complete. If any action fails, the plan is considered failed.

MIDCA Metacognitive Interface

The metacognitive cycle in MIDCA_1.3 receives a trace of activity accumulated from each phase of the object-level. The trace is a linked-list like data structure storing records per phase in the order that they occur during execution at the object level. The first metacognitive phase on the input side of the cycle, *monitor* (Perceive in Figure 1), receives the trace and selects the relevant part of the trace to reason over. Nodes in the trace correspond to a single phase and record all inputs and outputs of that phase. For example, the record for the object-level Plan phase includes the state and goal[3] given to SHOP2 and the resulting plan. After the monitor phase receives and refines the trace, the metacognitive cycle advances to the *interpret* phase where discrepancy detection reasons over the refined trace.

MIDCA and Goal-Driven Autonomy (GDA)

MIDCA's cycle at the object level is related to *goal-driven autonomy (GDA)* (Aha *et al*. 2010; Cox 2013; Klenk, Molineaux and Aha 2013), a kind of goal reasoning that focusses on initial formulation of goals. In GDA, an agent formulates goals as a result of discrepancies between the agent's internal expectations to the outcome of its actions and the actual outcome of the actions. When such a discrepancy occurs, the agent generates an explanation of the discrepancy and based on the explanation, it formulates a new goal to achieve. The crucial difference between MIDCA and GDA is that MIDCA reasons both at the object-level and at the meta-level, whereas GDA research has focused on reasoning at the object level only.

Formal Representations for Goal Reasoning

We now formally define the notion of interpretation for goal reasoning. We use classical planning as the basic formalism. Despite some well-known assumptions (e.g., actions are deterministic), classical planning provides an ideal formalism to define these complex notions because of its simplicity and clear semantics.

Classical Planning

A *classical planning domain* is defined (Ghallab, Nau, and Traverso 2004) as a finite state-transition system in which each state $s \in S = \{s_1, \ldots, s_n\}$ is a finite set of ground atoms of a function-free, first-order language $\mathcal{L}$. A *planning operator* is a triple o = (head(o), pre(o), eff(o)), where pre(o) and eff(o) are o's *preconditions* and *effects*. Each *action* is a ground instance of a planning operator. An action $\alpha \in A$ is *executable* in a state s if $s \vDash \text{pre}(\alpha)$, in which case the resulting state is $(s - \text{eff}^{-}(\alpha)) \cup \textit{eff}^{+}(\alpha)$, where $\text{eff}^{+}(\alpha)$ and $\textit{eff}^{-}(\alpha)$ are the atoms and negated atoms, respectively, in eff(α).

For a classical planning domain, the *state-transition system* is a tuple $\Sigma = (S, A, \gamma)$, [4] where S is the set of all states, and A is the set of all actions as above. In addition, *gamma* is a state transition function $\gamma: S \times A \rightarrow S$ that returns the resulting state of an executable action given a current state. Thus from any given state and action, one can infer the subsequent state $\gamma(s, \alpha) \rightarrow s'$ that follows action execution.

A *classical planning problem* is a triple $P = (\Sigma, s_0, g)$, where Σ is a state transition system, s_0 is the initial state, and g (the *goal formula)* is a conjunction of first-order literals. A goal state s_g satisfies a goal if $s_g \vDash g$. A *plan* π represents a (possibly empty) sequence of plan steps (i.e., actions) $\langle \alpha_1 \alpha_2 \ldots \alpha_n \rangle$ that incrementally changes the state of the world. Here we will use a notation that enables indexing of the individual steps or sub-sequences within the plan. In equation 1 we use the subscript g to indicate a plan that achieves a specific goal. A plan is composed of the first action α_1 followed by the rest of the plan $\pi_g[2\,..n]$.

$$\pi_g[1..n] = \alpha_1 \mid \pi_g[2\,..n] = \langle \alpha_1 \alpha_2 \ldots \alpha_n \rangle \qquad (1)$$

Now we recursively redefine gamma as mapping either single actions or plans to states. Hence π_g is a solution for P if it is executable in s_0 and $\gamma(s_0, \pi_g) \vDash g$. Recursively from the initial state, execution of the plan results in the goal state (see expression 2). The function γ corresponds to Act in Figure 1.

$$\gamma(s_0, \pi_g) = \gamma\left(\gamma(s_0, \alpha_1), \pi_g[2..n]\right) \rightarrow s_g \qquad (2)$$

Interpretation and Goal Reasoning

Goal reasoning has recently extended the classical formulation by relaxing the assumption that the goal is always given by an external user (Cox 2007; see also Ghallab, Nau and Traverso 2014). Although the planning process may start with an exogenous goal, a dynamic environment may present unexpected events with which the system must contend.

[3] Technically, SHOP2 receives tasks as inputs. In MIDCA, we use the standard conversion of encapsulating any goal g into a task (e.g., *achieve(g)*).

[4] Note that we are ignoring in the classical model the set of exogenous events E that are similar to actions but are outside the control (and possibly the observation) of the reasoning system.

In response a goal reasoner must be able to generate new goals at execution time as situations warrant.

Formally, the function β (see expression 3a) returns a (possibly) new goal g' given some state s and a current goal g. As such, β represents a state interpretation process that perceives the world with respect to its goals (i.e., Interpret in Figure 1). It is central to goal formulation and management.

$$\beta(s, g) \rightarrow g' \tag{3a}$$

Goal Transformation

Unlike classical planning models that assume goals to be static and given externally, the goal reasoning model views goals as malleable and subject to change (Cox and Zhang 2007). For example, a chess player may start out with the goal to achieve checkmate (g_{ch}). But given a series of unsuccessful opening moves (i.e., $\pi_{g_{ch}}[1..k]$ where $k < n$), the player may change the goal to draw (g_{dr}). See expression (3b).

$$\beta\big(\gamma(s_0, \pi_{g_{ch}}[1..k]), g_{ch}\big) \rightarrow g_{dr} \tag{3b}$$

Goals can undergo various transformations including priority shifts and goal abandonment (Cox and Veloso 1998). Over time goals follow arcs or trajectories through a space of goals (Bengfort and Cox 2015). Most importantly goals can be created or formulated given a problem state.

Goal Formulation

From some initial state s_0 and no goal state, an agent formulates a new goal as shown in expression 3c.

$$\beta(s_0, \emptyset) \rightarrow g \tag{3c}$$

In one sense, this can still entail user-provided goals. If the input state is one resulting from a speech act whereby a human requests a goal to be achieved, the function of β is to interpret the intention of the human and to infer the goal from the utterance. In another sense, however, this significantly differs from the classical formulation of a problem. For goal reasoning in its simplest form, a planning problem can be cast as the tuple $P = (\Sigma, s_0)$. Given a state transition system and an initial state, the goal-reasoning task is to formulate a goal (if a problem indeed exists in the initial state) and create (and execute) a plan to achieve it. Under most planning schemes, the system halts when the goal state is achieved (or even when a plan is simply found). In goal reasoning, an agent can search for new problems once all goals are achieved by interpreting the final goal state s_g. In this case, expression 3c becomes as in expression 3d. In general, goals can be formulated from any state.

$$\beta(s_g, \emptyset) \rightarrow g' \tag{3d}$$

A Model of Plans, Actions, and Interpretation

A plan to achieve a goal $\beta(s, \emptyset)$ can now be written as $\pi_{\beta(s,\emptyset)}$. Using this notation, we combine plans, action (plan execution), and interpretation in equation 4.

$$\gamma\big(s_0, \pi_{\beta(s_0,\emptyset)}\big) = \gamma\Big(\gamma(s_0, \alpha_1), \pi_{\beta(\gamma(s_0,\alpha_1),\beta(s_0,\emptyset))}[2..n]\Big) \tag{4}$$

When beta generates an exogenous initial goal g_1 from the initial state s_0 and simply returns the input goal from all other states (i.e., g' = g in 3a), the formalization reduces to classical planning with a user-given goal. That is, equation 4 is equivalent to equation 2 because expression 3a represents a trivial boundary case. However when goals change (or new ones are added), plans may need to change as well.

The problem with this formalization is that, in the recursive right-hand side of equation 4 above, the plan is not static as defined in equation 1. That is, it is not necessarily of size $n - 1$. Instead, because the goal may change due to beta, the goal reasoner may need to re-plan and thereby alter the length and composition of the remainder of the plan. To cover this contingency, we define a (re)planning function *phi* that takes as input a state, goal, and current plan as in expression 5. The function φ corresponds to the Plan process in Figure 1.

$$\varphi\big(s, g', \pi_g[1..n]\big) \rightarrow \pi_{g'}[1..m] \tag{5}$$

Note that in the general case g' may or may not be equal to g. Inserting the re-planning function into equation 4, we obtain equation 6 below that resolves the anomaly indicated above. Given φ, the formalism is general across different variations of goal reasoning and (re)planning.

$$\gamma\big(s_0, \pi_{\beta(s_0,\emptyset)}\big) = \gamma\,(\gamma(s_0, \alpha_1), \quad (6)$$
$$\varphi\big(\gamma(s_0, \alpha_1), \beta\big(\gamma(s_0, \alpha_1), \beta(s_0, \emptyset)\big), \pi_{\beta(s_0,\emptyset)}[2..n]\big)]])$$

Object Level Performance

We exemplify the object-level cycle applied to a variation of the blocksworld domain implemented in the standard simulator; this variation adds possible plan failures. The world includes both blocks and pyramids. But instead of arbitrary block stacking, the purpose of plan activity is to build houses such that blocks represent the wall structure and pyramids represent house roofs. Within this domain, objects may also catch on fire, an exogenous event, and so impede housing construction. We perform experiments testing house construction performance using a top-down goal generation strategy compared to a statistical approach (see Maynord, Cox, Paisner and Perlis 2013). We describe some of these details and a description of the individual interpretation methods used for comprehension below.

The current scenario implements a house building cycle that transitions through states as shown in Figure 3. If a part of a house (i.e., a block) catches fire, an operator exists that can extinguish the fire. When parts of houses are on fire, they do not count toward rewards for housing construction. Rewards are provided in terms of points, one for each block or pyramid not on fire in each completed house or tower. For example if the agent in Figure 3 (c) was to next pickup D and place it on A, then the result would be a tower worth 3 points. If D was placed from 3(b), it would be 2 points.

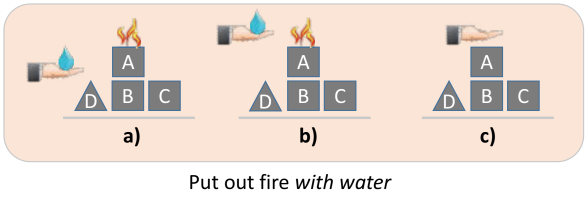

Figure 3. Unexpected event in housing construction cycle

SHOP2 produces various block stacking plans, such as the one for building a tower implied by the goal conjunct (∧(on A B)(on D A)). During plan execution, MIDCA perceives not only the results of performed actions but also states resulting from exogenous events. For example, block A may catch on fire. During interpretation of this new state, MIDCA needs to recognize that the fire represents a problem for its tower building tasks and generate a new goal in response.

Three conditions exist under which MIDCA solves tower construction problems. Under the *exogenous goals condition*, MIDCA only plans for goals given to it by the user. Under this condition it generates no goals of its own and ignores fires. Under the *statistical goal generation condition*, it uses a knowledge structure called a Tilde-FOIL Tree (Maynord *et al*. 2013) to react to fires by generating goals to achieve the negation of blocks on fire. These trees are learned by giving it examples of states under which the goals should be generated and states from which they should not. The final condition is the *GDA goal generation condition*. In this condition, MIDCA uses XPLAIN interpreter to map from the contradiction (i.e., expects A to be clear and observes that it is not; it is on fire instead) to an explanation that the fire was possibly caused by an arsonist. It then generates a goal to find and apprehend the arsonist. This goal represents a problem state that includes the anticipation of future fires and thus addresses the larger context of the situation.

Figure 4 shows the resulting score (y-axis) as a function of the probability that blocks catch fire (x-axis). As the probability of blocks catching fire increases, the performance of statistical goal generation alone converges to that of using only externally provided goals (i.e., no self-generated goals to handle the fires). Using GDA results in near perfect behavior because the cause of the fires (the arsonist) is addressed rather than just reacting to the fires after they occur.

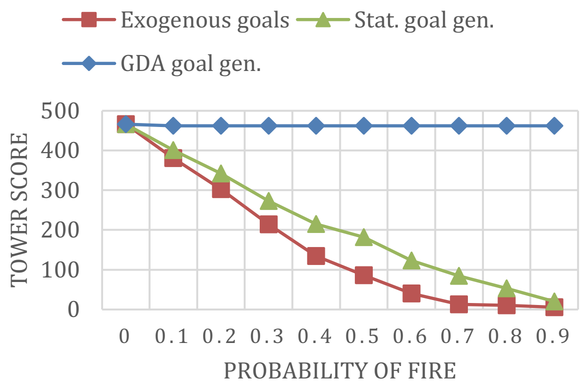

Figure 4. Performance as a function of fire probability

Meta-Level Performance

The metacognitive cycle begins with the monitor phase (Perceive in Figure 1) receiving a reasoning trace τ. Since the trace grows larger with every object level phase executed, the monitor phase also trims the trace into a more reasonable and relevant subtrace τ_1. In the current implementation, the trace is refined to the last object level phase executed. Moving on, the next metacognitive phase (i.e., Interpret) reasons over τ_1 to detect any discrepancies. In the style of goal-driven autonomy, when a discrepancy is found, an explanation is generated, followed by a goal to correct the discrepancy. This concludes the input side (monitor-interpret-evaluate) of MIDCA's metacognitive level. Next, the Intend phase selects which goal to pursue and sends that goal to the plan phase, which stores the plan in memory for the following controller phase to act upon.

In one currently implemented scenario, the cognitive-level planner fails to produce a plan, resulting in an impasse. The metacognitive level begins by receiving the trace containing the planner's failure to produce a plan. In this situation, the metacognitive level expects the planner to produce a non-null plan. Expectations are verified in the Interpret phase by reasoning over τ_1, producing the discrepancy that an impasse has occurred. The Interpret phase then uses this discrepancy as input to generate an explanation which is then used to generate a goal g_n. (we currently map discrepancies directly to goals; generating explanation is a topic of future work). The Intend phase selects this goal, followed by the Plan phase generating a plan to replace the object-level planner. Finally, the meta controller acts upon the plan process and executes the replace(Π) action. Table 1 depicts the similarities with an equivalent process at the cognitive level.

Table 1. Comparison between types of interpretation

Object level interpretation	**Meta-level interpretation**

Expects clear(A);
Observes ¬clear(A)
Symptom = contradiction
	Expects π; Observes ∅
	
	Symptom = `impasse`

Explanation: on-fire(A) → ¬clear(A)	Explanation: flawed-behavior(∏) → impasse
Goal(g) = ¬on-fire(A)	Goal(g) = ¬flawed-behavior(∏)
Plan(π) = extinguish(A)	Plan = replace(∏)

Related Research

MIDCA is related to other cognitive architectures in that it uses varied levels of abstractions. They differ on how these abstractions are represented. SOAR (Laird 2012) represents different levels of abstraction in a hierarchy. Abstract operators are used to refine abstract goals into increasingly concrete ones. SOAR enables the integration of a variety of knowledge forms than can be triggered by production rules at any level of the hierarchy. ACT-R (Anderson 1993) also uses production levels of varied level of abstraction. They generate new goals when rules are triggered at the lower level of the hierarchy. ICARUS (Langley and Choi 2006) also represents knowledge of different level of abstraction by using Horn clauses that define abstract goals. These clauses are used to learn hierarchies of increasing level of abstraction in a bottom-up matter by using teleo-reactive planning techniques. DISCIPLE (Tecuci 1988; Tecuci *et al.* 2002) uses concept hierarchies to represent knowledge of different levels of abstraction. In contrast to these cognitive architectures, MIDCA distinguishes between meta-level and object-level explicitly in the hierarchy.

Two architectures of note have modeled metacognitive components explicitly. CLARION (Sun, Zhang, and Mathews 2006) has a metacognitive module within its structure, but the mechanism is sub-symbolic as opposed to the higher-level symbolic representation of MIDCA. The ACT-R theory was not developed with specific metacognitive effects in mind, but it has recently modeled metacognitive tasks called pyramid problems. The control of reasoning is actually considered to be part of the cognitive function; whereas monitoring of reasoning is classified as metacognitive (see for example Anderson, Betts, Ferris, and Fincham 2011). In any case, many of the existing cognitive architectures have limitations when modeling metacognitive activity and have been modified on an as needed basis to fit some circumstances.

Other cognitive architectures have explored the issue of explanation. For example, Companions (Forbus, Klenk and Hinrichs 2009) uses truth-maintenance techniques to build justifications of the rationale that led to a decision. These explanations particularly target user interaction settings where the system must justify the reasons for actions taken. As discussed earlier in the paper, explanations play a central role in GDA systems, where the explanation serves as a bridge between discrepancies that the agent encounters during execution and the goal to achieve that addresses those discrepancies. In MIDCA, the Interpret and Evaluate steps help it understand the current situation in the environment (which could include a discrepancy with its own expectations).

An alternative formalism (Roberts *et al*. 2015) treats goal reasoning as *goal refinement*. Using an extension of the plan-refinement model of planning, Roberts models goal reasoning as refinement search over a goal memory M, a set of goal transition operators R, and a transition function delta that restricts the applicable operators from R to those provided by a fundamental goal lifecycle. Roberts proposes a detailed lifecycle consisting of goal formulation, goal selection, goal expansion, goal commitment, goal dispatching, goal monitoring, goal evaluation, goal repair, and goal deferment. Thus many of the differential functionalities in β are distinct and explicit in the goal reasoning cycle. However the model here tries to distinguish between the planning and action side of reasoning (φ and γ) and the interpretation and evaluation components inherent in goal reasoning (β).

Conclusion

This work makes the following contributions: (1) It presents a cognitive architecture that includes a metacognitive as well as cognitive layer to model high-level problem-solving and comprehension in dynamic environments. (2) It marks the public release of a documented implementation of the architecture. Although not as extensive as many existing cognitive architectures, MIDCA represents a new take to modeling aspects of autonomy that so far have received little attention. (3) A novel formalization of goal reasoning using well-defined AI planning concepts. (4) We introduced a new interface between MIDCA and physical robot. (5) We present empirical work demonstrating the dual functionality between the object-level and the meta-level.

Future work includes: (1) the important and non-trivial problem of deciding when to run the metacognitive layer. A possible way to tackle this problem is by running the meta-level and the object level concurrently. However, concurrency introduces many synchronization issues in order to effectively modify the cognitive layer without impeding its performance. (2) Making the best use of the reasoning trace, especially for long-duration missions producing very long traces. (3) Exploring which expectations are needed at the metacognitive level, and how to compute/obtain them.

Acknowledgements

This work was supported in part by ONR on grants N00014-15-1-2080 and N00014-12-1-0172 and by DARPA on contract W31P4Q-15-C-0071. This work is also supported in part by the National Science Foundation. We thank the anonymous reviewers for their comments and suggestions.

References

Aha, D. W.; Cox, M. T.; and Munoz-Avila, H. eds. 2013. Goal Reasoning: Papers from the ACS workshop, Technical Report CS-TR-5029, Department of Computer Science, University of Maryland, College Park, MD.

Aha, D. W.; Klenk, M.; Munoz-Avila, H.; Ram, A.; and Shapiro, D. eds. 2010. *Goal-driven Autonomy: Notes from the AAAI Workshop*. Menlo Park, CA: AAAI Press.

Anderson, J. R. 1993. *Rules of the Mind*. Hillsdale, NJ: LEA.

Anderson, J. R.; Betts, S.; Ferris, J. L.; and Fincham, J. M. 2011. Cognitive and Metacognitive Activity in Mathematical Problem Solving: Prefrontal and Parietal patterns. *Cognitive, Affective, & Behavioral Neuroscience* 11(1): 52-67.

Anderson, M. L.; Oates, T.; Chong, W.; and Perlis, D. (2006). The Metacognitive Loop I: Enhancing Reinforcement Learning with Metacognitive Monitoring and Control for Improved Perturbation Tolerance. *Journal of Experimental and Theoretical Artificial Intelligence, 18*(3), 387-411.

Bengfort, B.; and Cox, M. T. 2015. Interactive Reasoning to Solve Knowledge Goals. In D. W. Aha (Ed.), Goal Reasoning: Papers from the ACS Workshop (pp. 10-25). Technical Report GT-IRIM-CR-2015-001. Atlanta, GA: Georgia Institute of Technology, Institute for Robotics and Intelligent Machines.

Bjork, R. A.; Dunlosky, J.; and Kornell, N. 2013. Self-Regulated Learning: Beliefs, Techniques, and Illusions. *Annual Review of Psychology* 64: 417-444.

Burstein, M. H.; Laddaga, R.; McDonald, D.; Cox, M. T.; Benyo, B.; Robertson, P.; Hussain, T.; Brinn, M.; and McDermott, D. 2008. POIROT - Integrated Learning of Web Service Procedures. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence* 1274-1279. Menlo Park, CA: AAAI Press.

Cox, M. T. 2013. Goal-driven Autonomy and Question-Based Problem Recognition. In *Second Annual Conference on Advances in Cognitive Systems 2013, Poster Collection* (pp. 29-45). Palo Alto, CA: Cognitive Systems Foundation.

Cox, M. T. 2007. Perpetual Self-Aware Cognitive Agents. *AI Magazine* 28(1): 32.

Cox, M. T.; and Ram, A. 1999. Introspective Multistrategy Learning: On the Construction of Learning Strategies. *Artificial Intelligence* 112: 1-55.

Cox, M. T.; and Veloso, M. M. 1998. Goal Transformations in Continuous Planning. In M. desJardins (Ed.), *Proceedings of the 1998 AAAI Fall Symposium on Distributed Continual Planning* (pp. 23-30). Menlo Park, CA: AAAI Press / The MIT Press.

Cox, M. T.; and Zhang, C. 2007. Mixed-Initiative Goal Manipulation. *AI Magazine 28*(2), 62-73.

Fitzgerald, C. 2013. Developing Baxter. 2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA): A New Industrial Robot with Common Sense for U.S. Manufacturing. doi:10.1109/TePRA.2013.6556344

Forbus, K.; Klenk, M.; and Hinrichs, T. 2009. Companion Cognitive Systems: Design Goals and Lessons Learned so Far. *IEEE Intelligent Systems* 24: 36-46.

Ghallab, M., Nau, D., and Traverso, P. 2014. The Actor's View of Automated Planning and Acting: A Position Paper. *Artificial Intelligence* 208: 1-17.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory & Practice*. San Francisco: Elsevier.

Hawes, N. 2011. A Survey of Motivation Frameworks for Intelligent Systems. *Artificial Intelligence, 175*(5-6), 1020-1036.

Klenk, M.; Molineaux, M.; and Aha, D. 2013. Goal-Driven Autonomy for Responding to Unexpected Events in Strategy Simulations. *Computational Intelligence* 29(2): 187–206.

Laird, J. E. 2012. *The Soar Cognitive Architecture*. Cambridge, MA: MIT Press.

Langley, P.; and Choi, D. 2006. A Unified Cognitive Architecture for Physical Agents. In *Proceedings of the National Conference on Artificial Intelligence* 21(2): 1469. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Maynord, M.; Cox, M. T.; Paisner, M.; and Perlis, D. 2013. Data-Driven Goal Generation for Integrated Cognitive Systems. In C. Lebiere & P. S. Rosenbloom (Eds.), Integrated Cognition: Papers from the 2013 Fall Symposium (pp. 47-54). Technical Report FS-13-03. Menlo Park, CA: AAAI Press.

Nau, D.; Au, T.; Ilghami, O.; Kuter, U.; Murdock, J.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN Planning System. *Journal of Artificial Intelligence Research* 20: 379–404.

Paisner, M.; Maynord, M.; Cox, M. T.; and Perlis, D. 2013. Goal-Driven Autonomy in Dynamic Environments. In D. W. Aha, M. T. Cox, & H. Munoz-Avila (Eds.), Goal Reasoning: Papers from the ACS Workshop (pp. 79-94). Technical Report CS-TR-5029. College Park, MD: University of Maryland, Department of Computer Science.

Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; and Ng, A. Y. 2009. ROS: An Open-Source Robot Operating System. *ICRA Workshop on Open Source Software* 3(3.2): 5.

Roberts, M., Vattam, S., Alford, R., Auslander, B., Apker, T., Johnson, B., & Aha, D. W. (2015). Goal Reasoning to Coordinate Robotic Teams for Disaster Relief. In A. Finzi, F. Ingrand and A. Orlandini, eds., *Proceedings of ICAPS-15 PlanRob Workshop* (pp. 127-138).

Scheutz, M.; Harris, J.; and Schemerhorn, P. 2013. Systematic Integration of Cognitive and Robotic Architectures. *Advances in Cognitive Systems* 2: 277-296.

Sun, R.; Zhang, X.; and Mathews, R. 2006. Modeling Meta-cognition in a Cognitive Architecture. *Cognitive Systems Research* 7(4), 327-338.

Tecuci, G. 1988. DISCIPLE: A Theory, Methodology and System for Learning Expert Knowledge. Ph.D. diss., Paris 11.

Tecuci, G.; Boicu, M.; Marcu, D.; Stanescu, B.; Boicu, C., and Comello, J. 2002. Training and Using Disciple Agents: A Case Study in the Military Center of Gravity Analysis Domain. *AI Magazine* 23(4): 51.

Vattam, S., Klenk, M., Molineaux, M., and Aha, D. W. 2013. Breadth of Approaches to Goal Reasoning, A Research Survey, Naval Research Lab Washington DC.