Advances in Cognitive Systems 6 (2018) 1–20 Submitted 6/2018; published 8/2018

Declarative Metacognitive Expectations for High-Level Cognition

Dustin Dannenhauer DUSTIN.DANNENHAUER.CTR@NRL.NAVY.MIL
NRC Postdoctoral Fellow, Naval Research Laboratory, Washington D.C., 20375 USA

Michael T. Cox MICHAEL.COX@WRIGHT.EDU
Wright State Research Institute, Wright State University, Dayton, OH 45324 USA

Héctor Muñoz-Avila HEM4@LEHIGH.EDU
Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA 18015 USA

Abstract

Autonomous cognitive agents must adapt with changing environments to remain effective and robust. Expectations about the world enable an agent to identify an anomalous situation and thus provide the foundation for an appropriate response. While significant work has addressed anomalies or discrepancies in the world, few examples exist that address discrepancies within an agent's own cognition by explicitly declaring and reasoning about high-level expectations. Such high-level expectations allow an agent to identify anomalous mental situations, which in turn enables the agent to adapt its own knowledge or cognitive processes to complex problems. This paper introduces a new class of expectations called metacognitive expectations. Our contributions are the following: (1) a general formalism for these expectations; (2) an implementation of metacognitive expectations in an agent embodied within a cognitive architecture; and (3) experimental results indicating that an agent equipped with metacognitive expectations can outperform agents operating without such structures.

1. Introduction

Over the years, the concept of an expectation has played an important functional role in cognitive processes such as the monitoring of plan execution (Pettersson, 2005; Schank, 1982) and in managing comprehension, especially natural language understanding (Schank & Owens, 1987). Expectations are knowledge artifacts that enable cognitive systems to verify they are operating as intended. When performing actions, an agent can check its expectations and takes corrective steps when they are violated. That is, the agent checks if there are *discrepancies* between its expectations and its observations of the state of the world. When such discrepancies are detected, different means of addressing them have been proposed including plan adaptation (i.e., modifying the plan to be executed (Munoz-Avila & Cox, 2008)), learning (i.e., acquiring a new piece of knowledge (Ram & Leake, 1995)), and goal reasoning (i.e., changing the goals to be pursued and generating new plans to achieve them (Aha, 2018; Roberts et al., 2018)).

For example, research in *goal-driven autonomy (GDA) agents* (i.e., a class of goal reasoning agents that may change its goals or formulate a new one in the face of a discrepancy; Cox, 2007;

Klenk et al., 2013; Munoz-Avila et al., 2010) has identified different variants of expectations (Wilson et al., 2014). Dannenhauer et al. (2016) presents a taxonomy of expectations. It notes that some GDA agents compute an expectation based on the effects of its last executed action. For example, an agent expects that after stacking one block on another, the lower block will no longer be clear. If this is not true, then something must have gone wrong (perhaps the top block fell off due to faulty placement). Other GDA agents compute expectations based on the trajectory (i.e., the actions executed so far) Dannenhauer et al. (2016). Regardless of these variants, a common trait is that an agent's expectation is defined as a function of the observed states and the actions previously executed. That is, they are defined in terms of the *ground level* environment consisting of objects, their relationships, and actions in the world.

In contrast to such ground-level (i.e., cognitive) expectations, we introduce a new class of explicit expectations called a *metacognitive expectation* that focuses on metacognition for an intelligent system. When cognition reasons about the world (i.e., the ground level) using its cognitive expectations, we consider the process a kind of mental action operating on or with mental objects (e.g., the expectations). These actions and objects then exist at the *object level.* Analogously, processes at the *metalevel* can use declarative expectations about the object level to benefit metacognition. As such, metacognitive agents operate at three distinct levels (Cox & Raja, 2011):

- *The Ground Level.* An agent executes actions and makes observations of (i.e., perceives) the state of the environment. The world of doing.
- *The Object Level (cognition).* The agent reasons about the actions performed at the ground-level, the physical objects present, the states of those objects, and the relations between them. The world of thinking.
- *The Metalevel (metacognition).* The agent introspectively reasons about the cognition performed at the object level. The world of metareasoning.

There have been a number of works that reason implicitly with what can be considered metacognitive expectations. For example, Cimatti et al. (2017) uses the Action Notation Modeling Language to verify the execution of temporal plans. This verification acts as the expectation. This brings capabilities to (1) verify if an action is executable given the temporal constraints, (2) check if two actions are mutually exclusive, and (3) check if a plan is valid. These capabilities imply that the agent is reasoning about the process. The difference is that our work explicitly represents the object-level expectations and reasons about them within a separate metacognitive process. In the temporal domain, explicit metacognitive expectations would require a representation of the procedure verifying the temporal constraints. This was McCarthy's (1959; 1987) argument about the importance of declarative representations. For us, we have a declarative representation at the meta-level of the agent's goals, one that models processes at the object (cognitive) level. The same analogy can be made with other work that also includes metacognitive expectations although do not explicitly represent them in a declarative manner. This includes research on formulation of intentions, which provides a syntax and semantic representation of the expectations of logical plans (Baral & Gelfond, 2005; Blount et al., 2015). The logical representation of intent can be used to control robotic tasks Gomez et al. (2018). For further examples, see the related research in Section 2.

The primary contributions of this work are (1) a formalization of a declarative notion of metacognitive expectations; (2) an implementation of this formalization in a cognitive architecture; and (3) experiments with two domains from the GDA literature demonstrating the effectiveness of reasoning with expectations at the metalevel. We will use two baselines in our experiment: an agent with replanning capabilities and a GDA agent.

2. Related Research

Much computational research exists in the artificial intelligence community on the topic of metacognitive systems (see Cox, 2005, for a review), but little of this work focuses on the role of expectations. Contrastingly, most of the AI work on expectations relates primarily to knowledge about the world and agents behavior within the world. When an agent encounters a discrepancy, it may perform replanning, modifying the remaining (i.e., not yet executed) steps of the plan (Hammond, 1990; Cushing & Kambhampati, 2005). For example, the agent might observe the current state of the world and generate a new plan that achieves the goals from the current state. Some agents aim to make a more careful modification of the remaining plan by taking into account issues such as estimates to complete the plan (Fox et al., 2006). Regardless of how the plan repair is done, in replanning the goals remain the same. In our work metacognitive agents may change the goals that the agent is pursuing.

In contrast to agents performing plan repair, GDA agents may change their goals (Cox, 2013; Munoz-Avila et al., 2010; Molineaux et al., 2010). For example, consider a GDA agent controlling a vessel carrying some valuable cargo to a certain port. If the agent detects a sonar contact that may likely be an enemy submarine, it may change its goals to head back to the starting port or head to an alternative port thereby circumventing the potential threat and changing its goals in the process. GDA agents select new goals by performing a cognitive cycle in which they detect the discrepancy at the ground level, generate an explanation for the discrepancy and possibly formulate new goals as a result of that discrepancy. GDA subsumes plan repair because the agent might generate the same goal as before in which case the agent generates a new plan from the current state to achieve this goal. In our work, GDA agents can trigger a process generating new goals at the cognitive level akin to most GDA agents. But in addition, our agents run a cycle external to the cognitive level (i.e., metacognition) that supervises the execution of the GDA process. This cycle may trigger changes in the execution at the cognitive level as a result of discrepancies observed at the cognitive level and thereby generate meta-level goals (i.e., those to achieve particular internal states of the agent as opposed to states of the world).

One earlier system that shares many characteristics of metacognitive expectations is the Meta-AQUA story understanding and learning system (Cox, 1996; Ram & Cox, 1994). However, it operates with implicit metacognitive expectations as opposed to explicit declarative structures. The system would expect its explanations to be correct and when they were wrong (or when no explanation was generated), and it would examine a trace of its reasoning to do blame assignment on the object level. And finally it would perform a repair to its knowledge based upon that explanation. Likewise many systems have a capacity to perform metareasoning or model metacognition (e.g., Singh, 2005; Stroulia & Goel, 1995) but have implicit or procedural metacognitive expectations.

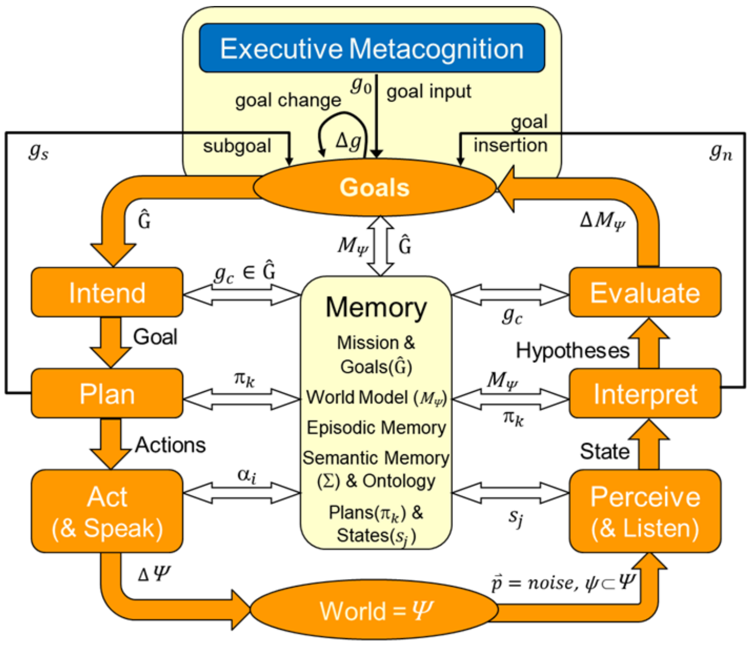


Figure 1. The cognitive action-perception cycle for the object level in MIDCA with meta-level cycle abstracted (adapted from Dannenhauer & Cox, 2018). The six major cognitive processes are perceive, interpret, evaluate, intend, plan, and act.

In our work, these expectations are explicit declarative knowledge artifacts as we will formalize subsequently.

Finally, analysis has shown that expectations drive cognition (Schank, 1982) and influence emotion (Langley, 2017). Furthermore, cognitive psychology research has long demonstrated the potential benefit and general characteristics of reasoning about mental state and mechanisms (Flavell, 1979; Flavell & Wellman, 1977). Humans have and exploit expectations concerning their own mental state (Dunlosky et al., 2013). In compelling examples of this capacity, game show contestants can often decide before they retrieve a memory item whether or not they know answers to given questions and demonstrate this by striking buzzers before their competitors do. Psychologists examined this phenomena and found it to be a robust effect (Reder & Ritter, 1992). More recently research has shown that even very young students can assess how well they have learned material, can predict memory and problem-solving performance, and can use such metacognitive expectations to choose learning and study strategies (Dunlosky, 2013; Dunlosky et al., 2013).

3. Metacognition: Reasoning about Cognition

If cognition is characterized by a system's structured knowledge about rich, complex environments, then metacognition is characterized by the system's knowledge of itself and its own cognitive machinery. This section introduces a metacognitive architecture called MIDCA within which we have

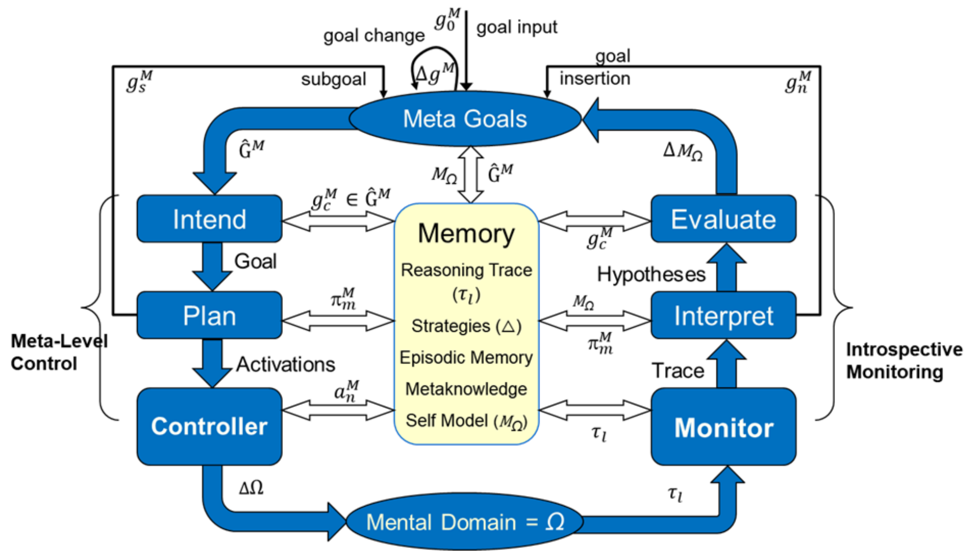


Figure 2. The metacognitive cycle for the meta-level in MIDCA with six analogous processes: monitor, interpret, evaluate, intend, plan, and control.

developed the idea of a metacognitive expectation. We then elaborate this idea formally and discuss its function computationally.

3.1 The MIDCA Metacognitive Architecture

The *Metacognitive Integrated Dual-Cycle Architecture (MIDCA)* (Cox et al., 2016; Paisner et al., 2014) is a three-level architecture composed of a ground level (where actions and perceptions occur), an object level (where problem-solving and comprehension processes exist), and a metalevel (where monitoring and control of the object-level exists). The current implementation incorporates the SHOP2 hierarchical network planner (Nau et al., 2003) and a custom-made heuristic planner (Bonet & Geffner, 2001). In addition to simulation of standard planning domains, it also includes a ROS interface to physical robots (e.g., a Baxter humanoid robot; `rethinkrobotics.com/baxter`) and the Gazebo physics simulator (see `gazebosim.org`).

As shown in Figure 1, object-level problem-solving is composed of *intend*, *plan*, and *act* phases, whereas comprehension consists of *perceive*, *interpret*, and *goal-evaluation* phases. The intend phase chooses a subset of the goal agenda $\hat{G}$ as the current goal g_c. The plan phase generates a sequence of action steps π_k to achieve the goal, and the act phase executes each action α_i in turn. The perceive phase takes a percept and extracts a predicate state s_j; the interpret phase takes the state and induces a model hypothesis or generates a new goal given a discrepancy; and the evaluate phase detects whether the current goal is true in the state, and if so, removes it from the agenda.

At the object level, the cycle achieves goals that change the environment. At the metalevel, a metacognition cycle achieves goals that change the object level (see Figure 2). That is, the metacognitive "perception" components introspectively monitor the processes and mental state changes in cognition. Monitoring occurs by reasoning over traces of object-level behavior (e.g., decision making and inferences). The "action" component consists of a metalevel controller that mediates reasoning over an abstract representation of the object-level cognition. It can manage the activity at the object level by changing the object-level goals, by changing the process executing a phase (e.g., by selecting between planners for the plan phase), or by changing MIDCA's domain knowledge (e.g., SHOP action models). The basis for such control decisions are MIDCA's metacogitive expectations.

3.2 Formalizing Metacognitive Expectations

The AI planning community relies on the notion of a state transition system $\Sigma = (S, A, \gamma)$ for representing planning problems and for reasoning about future situations. In Σ, S is the set of states that the agent can visit, A is the collection of actions that the agent can take, and γ is the state transition function $\gamma : S \times A \rightarrow S$. In its most basic sense, Σ encodes set of immediate expectations implied by the γ function. That is, from some current state, s_i, an agent expects the action α to result in a state $\gamma(s_i, \alpha) = s_{i+1}$.

3.2.1 Self-models, Mental States, and Mental Actions

Here we define an agent's *self-model* (i.e., a model of the cognitive level), Ω, in an analogous manner to Σ. Let $\Omega = (S^M, A^M, \omega)$ where S^M is a set of all possible *mental states*, A^M is the set of *mental actions*, and ω is a *cognitive transition function* defined as $\omega : S^M \times A^M \rightarrow S^M$. Now in a very simple sense we can think of a metacognitive expectation to be implied by ω. Given a mental state, $s_i^M \in S^M$, and mental action, $\alpha^M \in A$, the agent can expect the subsequent mental state $\omega(s_i^M, \alpha^M) = S_{i+1}^M$ to hold. However, many kinds of expectations exist besides the character of the next state in a sequence. Below we will provide some details that highlight their diversity and usefulness in complex situations.

An individual mental state is a vector of variables, $(v_1, ..., v_n)$. The variables that comprise a mental state are dependent on the agent; in MIDCA we use a separate variable for each of the following datums: Selected Goals, Pending Goals, Current Plan, World State, Discrepancies, Explanations, Actions. Thus our mental state is represented as a vector of length $n = 7$ as follows: $(g_c, \hat{G}, \pi, M_\Psi, D, E, \alpha_i)$. We make no constraints on the type of data for a mental state variable.

A mental action may perform reads or updates to variables in a mental state. MIDCA employs the following mental actions: Perceive, Detect Discrepancies, Explanation, Goal Insertion (this updates the set of goals $\hat{G}$ to be achieved), Evaluate (i.e., checks to see if the state M_Ψ entails the current goal g_c, and if so, removes it from $\hat{G}$), Intend (i.e., selects a new g_c from $\hat{G}$), Plan (i.e., calls a planner to generate a plan π to achieve g_c), and Act (i.e., executes the next step α_i from π, where $1 \leq i \leq |\pi|$).

A mental trace, τ, is a sequence of mental states and mental actions interleaved. Specifically $\tau = \langle s_0^M, \alpha_1^M, s_1^M, \alpha_2^M, .., \alpha_n^M, s_n^M \rangle$ where $s_0^M, .., s_n^M \in S^M$ and $\alpha_1^M, ..., \alpha_n^M \in A^M$. When the sequence of mental actions is fixed (as is the case in the current version of MIDCA), a trace may

Table 1. Metacognitive expectations for the mental actions of explanation, plan, and evaluate.

ID	**Expectation Header** / *Informal Description* / *Formal Description*
$EX^M_{D \rightarrow E}$	$EX^M(s_i^M, Explanation, s_{i+1}^M)$
	When the agent encounters a discrepancy, then the agent will generate an explanation for this discrepancy
	If $v_{discrepancies} \neq \emptyset$ in s_i^M and $v_{explanations} \neq \emptyset$ in s_{i+1}^M, return true, otherwise return false.
$EX^M_{I \rightarrow P}$	$EX^M(s_i^M, Plan, s_{i+1}^M)$
	When the agent commits to a goal, then the agent will generate a non-empty plan for that goal.
	If $v_{selectedgoals} \neq \emptyset$ in s_i^M and $v_{currentplan} \neq \emptyset$ in s_{i+1}^M, return true, otherwise return false.
$EX^M_{Eval \rightarrow I}$	$EX^M(s_i^M, Evaluate, s_{i+1}^M)$
	When the agent no longer has any selected goals and has remaining pending goals, the agent will commit to achieving one or more goals.
	If $v_{selectedgoals} = \emptyset$ in s_i^M and $v_{pendinggoals} \neq \emptyset$ in s_i^M and $v_{selectedgoals} \neq \emptyset$ in s_{i+1}^M, return true, otherwise return false.

be represented as only the mental states $\tau = \langle s_0^M, s_1^M, ..., s_n^M \rangle$. However, for the purpose of demonstrating the importance of the interaction of mental actions and mental states, we will use the representation of interleaved mental actions and mental states when describing metacognitive expectations.

We define a metacognitive expectation over a single mental action as $EX^M(s_i^M, \alpha_i^M, s_{i+1}^M)$, for a segment of a mental trace τ, where the length of the segment is always three: prior mental state, s_i^M, mental action, α_i^M, and subsequent mental state, s_{i+1}^M. Since a mental action can read the value of a mental state and update that variable given its original value, the mental state before and after a mental action are relevant to the expectation (thus the τ segment length of three). The metacognitive expectation is a Boolean function, $EX^M : (s_i^M, \alpha_i^M, s_{i+1}^M) \rightarrow \{\top, \bot\}$. If the output is true, the expectation is met. Otherwise, the expectation is not met.

3.2.2 An Example Metacognitive Expectation

We now give an example of a metacognitive expectation, $EX^M_{D \rightarrow E}$, that represents the following notion: *when the agent encounters a discrepancy, then the agent will generate an explanation for this discrepancy*, as shown in Table 1. Formally this is expressed as follows: $EX^M(s_i^M$, $Explanation, s_{i+1}^M)$ is a function that checks $v_{explanations} \neq \emptyset$ in s_{i+1}^M whenever $v_{discrepancies} \neq \emptyset$ in s_i^M.

Table 2. The ω procedure that creates a mental trace τ. It takes as parameters the name of a process $process$ and a list of arguments $args$.

```
1: global $\tau$
2: procedure $\omega(process, args)$
3:     $process.in \leftarrow args$                                        ▷ Process input
4:     $process.out \leftarrow apply(process, args)$     ▷ Application of process to input
5:     if $now(t)$ then
6:         $process.time \leftarrow t$                                  ▷ Set process time
7:     $\tau \leftarrow \tau \circ \langle process.self\ process.out\rangle$   ▷ Set trace to $\langle ...s^M_{i-1}\alpha^M_i s^M_i\rangle$
8:     $return(process.out)$                                          ▷ Process output
```

For example, at the object level when MIDCA detects a discrepancy between the expected outcome of an action and the observed state of the world, then an explanation for this object-level discrepancy must be generated. In the NBeacons domain, if the agent executes the action to move east and MIDCA observes that the agent remains in the same place, MIDCA might generate as an explanation that the agent is stuck and generate a new goal to become $\neg$stuck. In our experiments, at some point during execution, strong winds will affect the agent's move-east action, causing it to move four cells to the east instead of one (the agent is pushed an extra three tiles). When this happens, MIDCA cannot generate any plausible explanation for this discrepancy at the object level because it is lacking relevant knowledge (perhaps the developer could not foresee wind). This triggers a discrepancy with the metacognitive expectation because no explanation at the ground-level was generated.

The previous example is only one of many metacognitive expectations that would be useful to an agent. Table 1 includes two other metacognitive expectations for the mental actions *Plan* and *Evaluate*. The more mental actions an agent has at the cognitive layer, the more metacognitive expectations will be needed. In the next section we describe the MIDCA procedure and indicate how MIDCA deals with this discrepancy of metacognitive expectations.

3.3 Reasoning with Metacognitive Expectations

In the MIDCA architecture, mental traces are automatically generated when cognitive processes are executed at the object level. Table 2 illustrates the procedure ω for capturing mental states and process execution. We assume here that the application of any cognitive process to an input state will output a result s^M in the appropriate format (i.e., a 7-tuple mental state). The input, output and time of invocation are recorded in self-referential object form for the process (Lines 3-6). The function $now(t)$ in Line 5 is a temporal predicate as defined in active logic (Anderson et al., 2008), a kind of step logic. The process and then the output state are subsequently appended to the mental trace τ. Finally Line 8 returns the output to complete the procedure.

Table 3 describes the basic high-level flow of control for the MIDCA architecture. MIDCA initializes the internal state (Lines 3-4) and starts a cognitive cycle (Line 7). The mental trace τ is

Table 3. The MIDCA procedure is indirectly recursive. Cognition calls metacognition which then calls cognition. Many of the symbols used correspond to those of Figures 1 and 2. Note that each invocation of ω returns a cognitive 7-tuple. For brevity, $\top$ indicates an unchanging element.

```
1: global $\tau$
2: procedure MIDCA($\Psi$)
3:     $g_c \leftarrow \emptyset; \hat{G} \leftarrow \emptyset; \pi \leftarrow \langle\rangle; M_\Psi \leftarrow \emptyset; D \leftarrow \emptyset; E \leftarrow \emptyset; \alpha \leftarrow \emptyset$    ▷ Initialize 7 obj-level vars
4:     $s_0^M \leftarrow (g_c, \hat{G}, \pi, M_\Psi, D, E, \alpha)$    ▷ Initial mental state
5:     $cycle \leftarrow \langle perceive, interpret, eval, intend, plan, act\rangle$    ▷ Cognitive cycle
6:     $\tau \leftarrow \langle s_0^M \rangle$    ▷ Initialize trace with initial mental state
7:     cognition($M_\Psi, \pi, cycle, 1$)    ▷ Start thinking
8: procedure COGNITION($M_\Psi, \pi, procs, i$)
9:     $g_c^M \leftarrow \emptyset; \hat{G}^M \leftarrow \emptyset; \pi^M \leftarrow \langle\rangle; D^M \leftarrow \emptyset; E^M \leftarrow \emptyset$    ▷ Initialize 5 metalevel vars
10:    if $procs = \langle\rangle$ then $procs \leftarrow cycle$    ▷ Cycle reset
11:    $next \leftarrow head(procs)$
12:    $procs \leftarrow tail(procs)$
13:    switch $next$ do    ▷ Perform *next* cognitive phase
14:      case $perceive$
15:        $s_i^M = (\top, \top, \top, M_\Psi \cup s_j, \top, \top, \top) \leftarrow \omega$ (perceive, ($\vec{p}(\Psi)$))
16:      case $interpret$
17:        $s_i^M = (\top, \top, \top, \top, D, \top, \top) \leftarrow \omega$ (detect, ($s_j, \pi$))
18:        if $D \neq \emptyset$ then $procs \leftarrow explain|procs$    ▷ Explain added to the head of procs
19:      case $explain$
20:        $procs \leftarrow insertion|procs$    ▷ Insertion added to procs
21:        $s_i^M = (\top, \top, \top, M_\Psi \cup E, \top, E, \top) \leftarrow \omega$ (explain, ($D, \pi$))
22:      case $insertion$
23:        $s_i^M = (\top, \hat{G}, \top, \top, \top, \top, \top) \leftarrow \omega$ (goal-insertion, ($E, \pi$))
24:      case $eval$
25:        $s_i^M = (\top, \hat{G}, \top, M_\Psi, \top, \top, \top) \leftarrow \omega$ (evaluate, ($D, E, g_c$))
26:      case $intend$
27:        $s_i^M = (g_c, \top, \top, \top, \top, \top, \top) \leftarrow \omega$ (intend, ($\hat{G}, \pi$))
28:      case $plan$
29:        $s_i^M = (\top, \top, \pi, \top, \top, \top, \top) \leftarrow \omega$ (plan, ($M_\Psi, g_c, \pi$))
30:      case $act$
31:        $s_i^M = (\top, \top, tail(\pi), M_\Psi \cup \alpha^- \cup \alpha^+, \top, \top, \alpha) \leftarrow \omega$ (act, ($head(\pi)$))
32:    metacognition($M_\Psi, s_i^M, \pi, procs, i$)    ▷ Metacognitive cycle over trace of phase
33: procedure METACOGNITION($M_\Psi, M_\Omega, \pi, procs, i$)
34:    $D^M, E^M, \hat{G}^M \leftarrow$ *meta-interpret*(*monitor* ($M_\Psi, \pi, \tau$), $\pi^M, M_\Omega$)    ▷ *Monitor* then *interpret*
35:    $M_\Omega, \hat{G}^M \leftarrow$ *meta-evaluate*($D^M, E^M, g_c^M$)    ▷ Then *evaluate*
36:    if $\hat{G}^M = \emptyset$ then cognition($M_\Psi, \pi, procs, i + 1$)    ▷ When agenda empty, continue cognition
37:    else
38:        $g_c^M \leftarrow$ *meta-intend* ($\hat{G}^M, \pi^M$)    ▷ *Intend* then *plan* and finally
39:        metacognition ($M_\Psi, M_\Omega, (controller($ *meta-plan* $(M_\Omega, g^M, \pi^M), procs, i))$ )    ▷ *control*
```

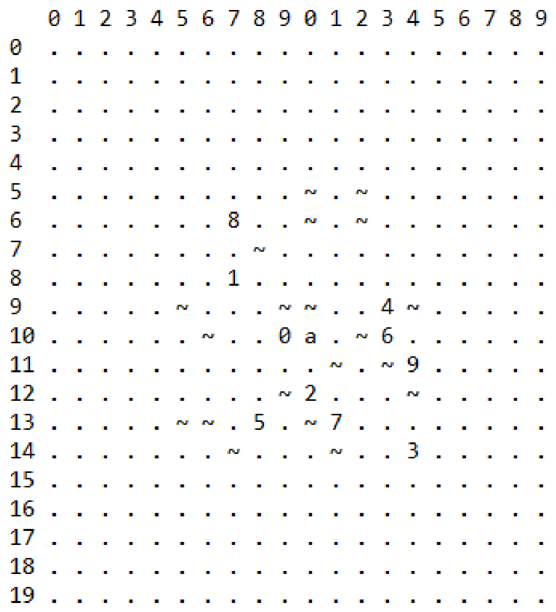
initialized to the sequence having the initial mental state s_0^M (Line 6). The values of s_0^M are empty because no perception has occurred and thus no knowledge of the world yet exists.

After initialization of the basic cognitive state (Line 9 of Algorithm 2), cognition performs one phase (determined by Lines 9-12) of its action-perception cycle (i.e., executes the switch-case control structure in Lines 13-31) and then invokes metacognition (Line 32). Each clause in the switch returns a mental state as defined in the previous section but modifies only selective elements in the state. A $\top$ in any element's position indicates that the value remains unchanged. For example on line 15, $perceive$ extracts a predicate state, s_j, from a percept $\vec{p}$ observed from the environment, Ψ. It assigns the value of the fourth element of the state (i.e., the model of the environment, M_Ψ) the union of the result (s_j) with M_Ψ. The remaining six elements are not effected.

If a metacognitive expectation failure occurs (on Line 34, D^M will not be empty), it attempts to repair itself by creating a meta-level goal and adding it to the meta-level goal agenda, $\hat{G}^M$; otherwise, it calls cognition to continue (on line 36). Interpretation at the metalevel is performed as a three-step process as is the case with the object-level analogue. It is however performed in a single call on line 34, setting a discrepancy D^M if one exists from any metacognitive expectation, explaining D^M, and then formulating one or more new goals $\hat{G}^M$ if necessary to change the object level. Line 35 performs evaluation on any goal from previous metalevel cycles, and then Line 36 checks to see if new meta-level goals were formulated. If so, it performs goal selection with intend (Line 38), plans for the goal at Line 39 and executes the plan with the metalevel controller also on Line 39. Metacognition is called again to perform evaluation on the results (all on Line 39).

Note the parallel control structure between the cognitive and metacognitive cycles (c.f., Figures 1 and 2). Each includes a comprehension sequence of $\langle perceive/monitor, interpret, evaluate\rangle$ and a problem-solving sequence of $\langle intend, plan, act/control\rangle$. In contrast, the former uses ω to record a trace whereas the latter does not. Yet functionally they are very similar. The cognitive level interprets the percepts from the external environment; whereas, the metalevel interprets the current trace of the internal object level. Interpretation is actually composed of a GDA three-step process. It detects discrepancies, explains them, and uses the explanation to generate potential new goals to resolve the discrepancy. This is implemented by having *interpret* add *explain* to the head of the processes to be executed (Line 18) after calling *detect* (Line 17). Explanation then prepends *goal-insertion* to the procedure sequence (Line 20) and calls *explain* (Line 21).

We now reconsider the example at the end of the previous section when MIDCA fails to generate an explanation at the object level because permanent changes in the wind conditions cause the agent to always be pushed an additional three cells whenever it moves east. Since no explanation is generated at the object level, it causes a discrepancy of a metacognitive expectation. In this case Line 34 will evaluate τ, which will contain a sequence of the form $(s_i^M, Explanation, s_{i+1}^M)$ with $v_{explanations} = \emptyset$ in s_{i+1}^M and $v_{discrepancies} \neq \emptyset$ in s_i^M. This will detect the discrepancy and generate a new goal $\hat{G}^M$. Therefore the condition in Line 36 fails and the else code in Lines 38-39 is performed. Line 38 selects the new goal, g^M, to fix the domain theory (the new goal is generated during meta-interpret on Line 34 and is a member of the resulting $\hat{G}^M$). Line 39 generates a plan to modify the domain description (now the effects of the move east will correctly indicate that it will be pushed three additional cells to the right); the controller makes this change in the domain description (also Line 39) and the process is re-started.



```
   0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9
0  . . . . . . . . . . . . . . . . . . . .
1  . . . . . . . . . . . . . . . . . . . .
2  . . . . . . . . . . . . . . . . . . . .
3  . . . . . . . . . . . . . . . . . . . .
4  . . . . . . . . . . . . . . . . . . . .
5  . . . . . . . . . . ~ . ~ . . . . . . .
6  . . . . . . . 8 . . ~ . ~ . . . . . . .
7  . . . . . . . . ~ . . . . . . . . . . .
8  . . . . . . . 1 . . . . . . . . . . . .
9  . . . . . ~ . . . ~ ~ . . 4 ~ . . . . .
10 . . . . . . ~ . . 0 a . ~ 6 . . . . . .
11 . . . . . . . . . . . ~ . ~ 9 . . . . .
12 . . . . . . . . . ~ 2 . . . ~ . . . . .
13 . . . . . ~ ~ . 5 . ~ 7 . . . . . . . .
14 . . . . . . . ~ . . . ~ . . 3 . . . . .
15 . . . . . . . . . . . . . . . . . . . .
16 . . . . . . . . . . . . . . . . . . . .
17 . . . . . . . . . . . . . . . . . . . .
18 . . . . . . . . . . . . . . . . . . . .
19 . . . . . . . . . . . . . . . . . . . .
```

Figure 3. NBeacons Example Scenario. The symbol *a* represents the agent, integers between 0-9 are the locations of beacons, and $\sim$ represents a sand pit.

4. Experimental Study

This section presents experiments with domains from the goal reasoning literature demonstrating the effectiveness of reasoning with metacognitive expectations. We performed an ablation study of three agent types in both the NBeacons and Aggressive Arsonist domains. The experimental results demonstrate that an agent with the metacognitive expectation for the mental action *explanation* is able to successfully trigger metacognition which then resolves a failure of the agent's planning operators.

4.1 Test Domains and Experimental Manipulations

The motivation behind using the NBeacons and Aggressive Arsonist domains (Dannenhauer, 2017) for these experiments is to test robust agents in environments that change over time. The particular kind of changes in which we are interested are those that an agent designer may not have predicted, and as such, these domains are suitable.

4.1.1 NBeacons

NBeacons is a modification of the Marsworld domain used in Dannenhauer & Munoz-Avila (2015). The domain is unbounded, with an area of interest to the agent that is a square grid of 100 tiles (shown in the center of Figure 3). Scattered among the area of interest are a total of 10 beacons and 20 sand pits. Beacons and sand pits cannot be in the same tile. Beacons are deactivated until the agent, which must be on the same tile as the beacon, performs an activate-beacon action. Other actions available to the agent include navigation (move east, west, north, south) and push actions (to free an agent that has fallen into a sand pit). There are five push actions an agent must execute before it will become free to move. Figure 3 shows a part of an NBeacons domain and is centered around the area of interest. Wind may push the agent outside the area of interest, in which case the agent will navigate back.

The primary difference of NBeacons compared to Marsworld is the addition of wind. After a period of time, wind blows causing the agent to be instantaneously pushed extra tiles whenever it attempts to move in a specific direction. If the agent would be pushed through any tiles containing sand pits, the agent will become stuck in the sand pit. In our experiments, wind begins to take effect after 500 ticks at a strength of three (pushing the agent three extra tiles) and then increases to a strength of four after 1500 ticks. Wind causes the effects of the agent's move action to be permanently altered for the rest of its execution. Due to wind, an agent will find itself in a location other than what it expected, sometimes ending up in a sand pit. Wind always blows in one direction. Without wind, an agent never ends up in a sand pit because it will plan around them. All agents acting in the NBeacons domain perform heuristic planning with the heuristic being straight-line distance. Agents are given goals to activate specific beacons. Beacons are deactivated after they are activated to ensure an agent always has an available goal.

4.1.2 Aggressive Arsonist

The second domain, Aggressive Arsonist, is a modification of the arsonist domain first used in Maynord et al. (2013). It is a blocksworld domain with an unseen arsonist lighting blocks on fire. The agent is tasked with randomly generated goals to build towers of height four. In the original Arsonist domain, fire reduces the score the agent gets for completing each goal (here we do not measure score, only goal achievement). When an agent detects fire, they generate goals to put out the fire before returning to block stacking. Each tick of the environment (i.e. every time the agent takes an action) there is 20% chance the arsonist will light a block on fire.

```
  --------
  |  A_  |
  --------
  --------
  | *B_* |
  --------
  --------           / \
  | *C_* |          /   \
  --------         /  F_ \
----------------------------
```

Figure 4. Aggressive Arsonist example scenario. Blocks **B_** and **C_** are on fire. The fire on block **C_** keeps **B_** on fire so that an agent that tries to extinguish **B_** will fail because **C_** is on fire.

The primary difference between Aggressive Arsonist and the original Arsonist domain is that after 400 ticks in Aggressive Arsonist, the domain changes and the arsonist becomes more aggressive. Now every time the arsonist lights a block on fire he/she will instead light two adjacent blocks on fire. The type of fire is stronger, such that the agent can only douse the fires by putting out the bottom fire first (otherwise the bottom fire keeps the block above it lit). Figure 4 shows a scenario with two blocks on fire, where the bottom block keeps the top block on fire.

4.1.3 Agent Types: Replanning (baseline), GDA, Meta + GDA

All three agents, in both domains, are implemented using the MIDCA Cognitive Architecture, albeit with ablated components. The baseline agent, a re-planning only agent, checks to see if the goal was reached at the end of execution of its plan, and if not, re-plans. The second agent, a goal driven autonomy agent, uses expectations to determine when a discrepancy of the world occurs, followed by explanation, and finally goal formulation. Expectations use a state comparison approach described in Dannenhauer & Munoz-Avila (2015) and used in Molineaux et al. (2010). Explanation is implemented as a mapping from discrepancies to goals. Thus our GDA agent in the NBeacons domain generates a goal to become free after it explains that the agent is stuck in a sand pit. When the GDA agent finds it has been blown off course (but not in a sand pit), it inserts the same beacon-activation goal it previously had, which triggers immediate planning in the agent's new state.

The third agent contains the GDA mechanism of the previous agent (GDA at the cognitive level) along with metacognition enabling it to reason with metacognitive expectations. These metacognitive expectations include the expectation $EX^{M}_{D \rightarrow E}$ (see Table 1) mentioned in the example earlier: an agent detecting a discrepancy should also have generated an explanation. When the cognitive GDA component, Explanation, fails to produce an explanation after observing the discrepancy (in NBeacons the discrepancy is being in an unexpected location, in Aggressive Arsonist the discrepancy is failing to put out a fire when there is a fire underneath the block) the metacognitive expectation is triggered and the agent generates a goal to update its domain model. Since the focus of this paper is on modeling metacognitive expectations and using them to detect anomalies of cognition, we do not go into detail regarding operator learning.

Research in learning action models for planning domains has a long history (Čertickỳ, 2014; Wang, 1995, 1996; Yang et al., 2007). The basic premise of these works is to extrapolate the effects of the operator by reasoning on the difference in the state before and after an action is applied. The challenge here is that when multiple actions that are instances of the same operator occur, there can be multiple plausible effects that could be assigned to the operator. Analogously, when observing the states immediately preceding actions that are instances of the same operator, there can be multiple plausible preconditions for the operator. Authors have explored different ways to deal with this problem. For example, Yang et al. (2007) models the problem of learning preconditions and effects of operators as a constraint satisfaction problem. From the input traces it automatically extracts constraints indicating plausible preconditions and effects for each occurrence of an action, which is an instance of the same operator. An example constraint is "atom *p* is a precondition of operator *op*". These constraints are weighted by the number occurrences; for example, if *p* occurs 100 times in the traces in a state immediately preceding actions that are instances of *op*, the constraint is assigned a weight of 100. Other constraints are added to prune the preconditions and effects. One such example

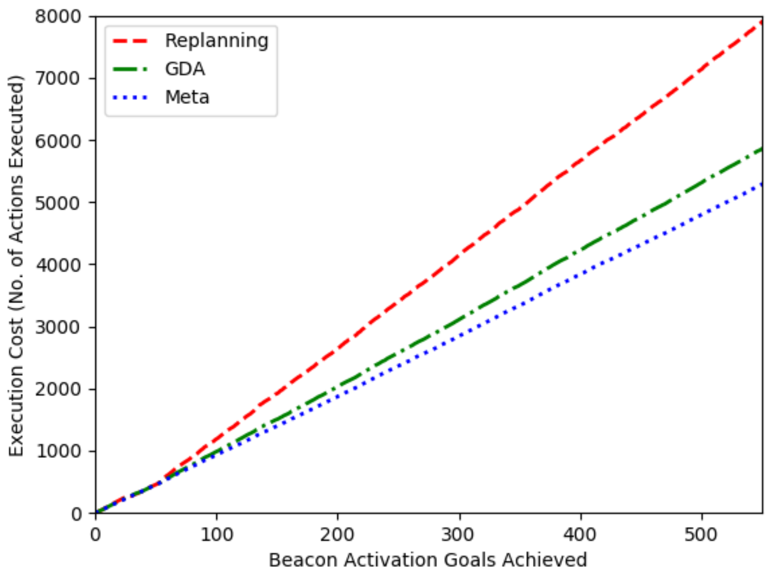


Figure 5. Average execution cost as a function of goal achieved in the NBeacons domain.

is the constraint: "if p is a precondition of *op*, then p cannot be an effect of *op*"; analogously "if p is an effect of *op*, then p cannot be a precondition of *op*". These constraints are given to a MAXSAT solver, which returns a set of constraints that are satisfied. From these the preconditions and effects of operators are extracted. In our work, an action is updated given the distance the agent was pushed by the wind (NBeacons) and using the failed action putoutfire (Aggressive Arsonist). Unlike the other two ablated agents, the metacognitive agent is able to update its model for the action affected by wind (i.e., when wind is blowing east, the move-east action is updated in NBeacons).

Our hypothesis is that the agent equipped with metacognitive expectations will achieve the lowest execution cost compared to the GDA agent and the replanning agent. We expect that the GDA agent will immediately respond to discrepancies while the replanning agent waits for its current invalid plan to run out. The metacognitive agent will detect an explanation failure of its GDA components and update the effects of its actions to take into account the changes in the domain (wind in NBeacons, stronger fires Aggressive Arsonist). For example, the GDA agent will be blown unexpectedly into sand pits while the metacognitive agent will avoid them once it has learned from previous encounters with the wind.

4.2 Empirical Results

The empirical results from both experimental domains are averaged over ten scenarios and shown in Figures 5 and 6. In NBeacons, each scenario is generated by randomly placing the beacons and the sand pits with the agent starting in the center (see Figure 3 for an example). Each agent is given

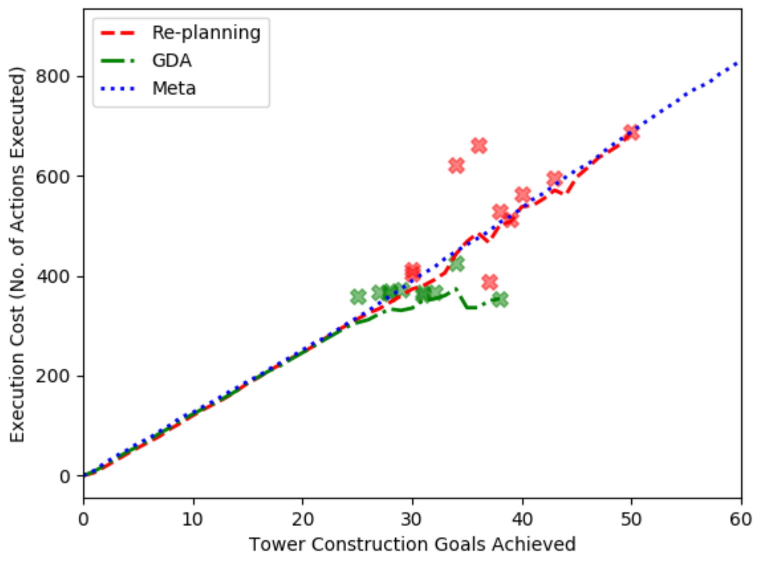


Figure 6. Average execution cost as a function of goals achieved in the Aggressive Arsonist domain. The colored crosses on the graph indicate the last goal achieved in each run before failing.

the same ordered list of goals to achieve. The abscissa (x-axis) of these figures is the cumulative number of goals achieved, and the ordinate (y) is the cumulative execution cost. Execution cost is measured by the number of actions executed, where each action has a unit cost.

4.2.1 NBeacons Results

After 500 actions, wind begins blowing east with a strength of three (the agent will be pushed an additional three tiles), followed with another increase in wind to four (the agent will be pushed an additional four tiles) after 1,500 actions. Prior to wind, the performance of all the agents is the same because they are able to plan around all sand pits. However, once wind is introduced, the agents may be blown into a sand pit and must execute a series of push actions to free themselves, thus raising their execution costs. As shown in Figure 5, all agents exhibit a similar performance for the first 75 goals, which takes an execution cost of about 800 actions. Although wind is introduced after 500 actions, it takes some time to see the benefit because not all actions are affected by wind.[1]

4.2.2 Aggressive Arsonist Results

Each run starts with all blocks on the ground and no fires. After 300 actions, the arsonist becomes aggressive, and lights stronger fires which cause two adjacent blocks to burn (see Figure 4 for an

1. When wind is blowing east, only the move-east action is affected. Thus goals that do not require navigating east will not see any impact from wind).

example). Unlike NBeacons, shortly after this change in the domain, the replanning and GDA agents fail because they are unable to put out fires. The last goal achieved before failing is shown as an $\times$ in Figure 6 with the lines representing the average performance over all ten scenarios. The replanning and GDA agents fail to continue to achieve goals, because they repeatedly fail to put out a fire on a block that has another burning block underneath it. One interesting result is that the GDA agent tends to fail faster than the replanning agent, because it immediately responds to fire. The replanning agent only responds to fire after it achieves its current goal, and so will continue stacking blocks even if lower blocks have caught on fire (while fire prevents stacking operators, it does not affect goal conditions).

4.2.3 Discussion

In these results for both domains, only the metacognitive agent identifies the explanation failure at the GDA level and performs operator learning (in the same fashion as NBeacons). In the Agressive Arsonist domain, the metacognitive agent learns two new operators for putting out fire (one that handles the case of a block on the ground, the other that handles the case of a block stacked on another block). By learning these new operators, the agent is able to put out the fires and continue operating.

As the agents continue to operate in both domains, we see that even when the effects of a single action change in the environment, metacognition improves performance over the replanning and the GDA agents by identifying a cognitive failure and addressing a limitation in the agent's own knowledge. These results indicate that metacognitive expectations (1) are domain-independent: the same metacognitive expectation was used in different domains and (2) lead to both improved performance (as in NBeacons) and the resolution of incapacitating failures that would otherwise prevent the agent from continuing to achieve its goals (as in Aggressive Arsonist). Finally, our hypothesis that an agent with metacognitive expectations would outperform a re-planning agent and GDA agent was met and is supported by both experiments.

5. Conclusion

This work extends the existing research on anticipatory approaches to intelligent agents within the MIDCA cognitive architecture that use expectations to focus problem solving on salient aspects of a problem. We introduce metacognitive expectations, a novel class of expectations that applies the approach to the problem-solving cycle itself rather than, as it is typical in current GDA research, just the environment and actions executed within it. Using dynamic domains, NBeacons and Aggressive Arsonist, that change significantly over time, we report on empirical results showing the benefit of reasoning with metacognitive expectations. These metacognitive expectations are domain-independent; in our experiments with both the NBeacons and Aggressive Arsonist domains, a failure to explain a change in the environment triggers a discrepancy to the same metacognitive expectation, showing the generality of this approach.

We emphasize that the contribution here is the ability to detect a problem has occurred at the cognitive level, independent of the agent's environment. When these problems are addressed there is a measurable effect on the agent's overall performance. While we have introduced the notion of metacognitive expectations, we feel there is much remaining work to create introspective agents robust to changing environments that are capable of evaluating their own processes. An immediate area for future work is to broaden the scope of capabilities at the metalevel. Specifically, we are modifying plan operators. We would like to expand the system's capabilities to modify other components such as the explanation component of cognition. The diagnosis that follows the discrepancy generated from $EX^M_{D \to E}$ may determine a solution is needed in the explanation component instead of the plan operators. The generality of metacognitive expectations such as $EX^M_{D \to E}$ is that they identify problems regardless of the solution type. We leave diagnosing the cause of an anomaly identified from these metacognitive expectations as an area for future work.

As a last thought, we speculate that for a given set of mental actions a cognitive agent possesses, there is a set of metacognitive expectations the agent can employ that applies to any domain. We have introduced three such expectations (Table 1) and empirically demonstrated that an agent equipped with just one metacognitive expectation, namely $EX^M_{D \to E}$, can identify cognitive-level failures, that once fixed, lead to increases in overall agent performance. The metacognitive expectations described here are non-exhaustive. We would like to expand these to include agents with different mental actions. Furthermore, some metacognitive expectations may span more than one mental action. Finally, besides enabling the agent with self-monitoring of the cognitive layer, these expectations can serve as a debugging tool for developers of cognitive agents. While a developer is expected to create the mental actions and knowledge for the agent, metacognitive expectations and other metacognitive processes (when developed in the future) will provide an extra layer of robustness available for free.

Acknowledgements

This research was supported by the Office of Naval Research under grant N00014-18-1-2009, by the Air Force Office of Scientific Research Grant FA2386-17-1-4063, by DARPA/DSO under SPAWAR Contract N6600118C4039 and by National Science Foundation Grant 1217888. We also thank the anonymous reviewers for their comments and suggestions. The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

Aha, D. W. (2018). Goal reasoning: Foundations, emerging applications, and prospects. *AI Magazine*, *39*, 3–24.

Anderson, M., Gomaa, W., Grant, J., & Perlis, D. (2008). Active logic semantics for a single agent in a static world. *Artificial Intelligence*, *172*, 1045–1063.

Baral, C., & Gelfond, M. (2005). Reasoning about intended actions. *Proceedings of the Twentieth National Conference on Artificial Intelligence* (pp. 689–694). Pittsburgh, PA: AAAI Press.

Blount, J., Gelfond, M., & Balduccini, M. (2015). A theory of intentions for intelligent agents. *Proceedings of the Thirteenth International Conference on Logic Programming and Nonmonotonic Reasoning* (pp. 134–142). Lexington, KY: Springer.

Bonet, B., & Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, *129*, 5–33.

Čertický, M. (2014). Real-time action model learning with online algorithm 3*SG*. *Applied Artificial Intelligence*, *28*, 690–711.

Cimatti, A., Micheli, A., & Roveri, M. (2017). Validating domains and plans for temporal planning via encoding into infinite-state linear temporal logic. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (pp. 3547–3554). Palo Alto, CA: AAAI Press.

Cox, M. T. (1996). *Introspective multistrategy learning: Constructing a learning strategy under reasoning failure*. Doctoral dissertation, College of Computing, Georgia Institute of Technology, Atlanta, GA. Technical Report Number GIT-CC-96-06.

Cox, M. T. (2005). Metacognition in computation: A selected research review. *Artificial Intelligence*, *169*, 104–141.

Cox, M. T. (2007). Perpetual self-aware cognitive agents. *AI Magazine*, *28*, 32–45.

Cox, M. T. (2013). Goal-driven autonomy and question-based problem recognition. *Poster Collection of the Second Annual Conference on Advances in Cognitive Systems* (pp. 29–45). Baltimore, MD: Cognitive Systems Foundation.

Cox, M. T., Alavi, Z., Dannenhauer, D., Eyorokon, V., Munoz-Avila, H., & Perlis, D. (2016). MIDCA: A metacognitive, integrated dual-cycle architecture for self-regulated autonomy. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (pp. 3712–3718). Phoenix, AZ: AAAI Press.

Cox, M. T., & Raja, A. (2011). Metareasoning: An introduction. In *Metareasoning: Thinking about thinking*, 3–14. Cambridge, MA: MIT Press.

Cushing, W., & Kambhampati, S. (2005). Replanning: A new perspective. *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling* (pp. 13–16). Monterey, CA: AAAI Press.

Dannenhauer, D. (2017). *Self monitoring goal driven autonomy agents*. Doctoral dissertation, Department of Computer Science and Engineering, Lehigh University, Bethlehem, PA.

Dannenhauer, D., & Munoz-Avila, H. (2015). Raising expectations in GDA agents acting in dynamic environments. *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence* (pp. 2241–2247). Buenos Aires, Argentina: AAAI Press.

Dannenhauer, D., Munoz-Avila, H., & Cox, M. T. (2016). Informed expectations to guide GDA agents in partially observable environments. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (pp. 2493–2499). New York: AAAI Press.

Dannenhauer, Z. A., & Cox, M. T. (2018). Rationale-based perceptual monitors. *AI Communications*, *31*, 197–212.

Dunlosky, J. (2013). Strengthening the student toolbox: Study strategies to boost learning. *American Educator*, *37*, 12–21.

Dunlosky, J., Rawson, K. A., Marsh, E. J., Nathan, M. J., & Willingham, D. T. (2013). Improving students' learning with effective learning techniques: Promising directions from cognitive and educational psychology. *Psychological Science in the Public Interest*, *14*, 4–58.

Flavell, J. H. (1979). Metacognition and cognitive monitoring: A new area of cognitive-development inquiry. *American Psychologist*, *34*, 906–911.

Flavell, J. H., & Wellman, H. M. (1977). Metamemory. In J. R. V. Kail & J. W. Hagen (Eds.), *Perspectives on the development of memory and cognition*, 3–33. Lawrence Erlbaum Associates.

Fox, M., Gerevini, A., Long, D., & Serina, I. (2006). Plan stability: Replanning versus plan repair. *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling* (pp. 212–221). Cumbria, UK: AAAI Press.

Gomez, R., Sridharan, M., & Riley, H. (2018). Representing and reasoning with intentional actions on a robot. *Proceedings of the Workshop on Planning and Robotics at the Twenty-Eighth International Conference on Automated Planning and Scheduling*. Delft, The Netherlands: AAAI Press.

Hammond, K. J. (1990). Explaining and repairing plans that fail. *Artificial Intelligence*, *45*, 173–228.

Klenk, M., Molineaux, M., & Aha, D. W. (2013). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence*, *29*.

Langley, P. (2017). A cognitive systems analysis of personality and conversational style. *Advances in Cognitive Systems*, *5*, 1–12.

Maynord, M., Cox, M. T., Paisner, M., & 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*, 47–54. Menlo Park, CA: AAAI Press.

McCarthy, J. (1959). Programs with common sense. *Symposium Proceedings on Mechanisation of Thought Processes* (pp. 77–84). London, England: Her Majesty's Stationary Office.

McCarthy, J. (1987). Generality in artificial intelligence. *Communications of the ACM*, *30*, 1030–1035.

Molineaux, M., Klenk, M., & Aha, D. (2010). Goal-driven autonomy in a Navy strategy simulation. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence* (pp. 1548–1554). Atlanta, GA: AAAI Press.

Munoz-Avila, H., Aha, D. W., Jaidee, U., Klenk, M., & Molineaux, M. (2010). Applying goal driven autonomy to a team shooter game. *Proceedings of the Twenty-Third International FLAIRS Conference* (pp. 465–470). Daytona Beach Shores, FL: AAAI Press.

Munoz-Avila, H., & Cox, M. T. (2008). Case-based plan adaptation: An analysis and review. *IEEE Intelligent Systems*, *23*, 75–81.

Nau, D. S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, *20*, 379–404.

Paisner, M., Cox, M. T., Maynord, M., & Perlis, D. (2014). Goal-driven autonomy for cognitive systems. *Proceedings of the Thirty-Sixth Annual Conference of the Cognitive Science Society* (pp. 2085–2090). Quebec City, Quebec, Canada: Cognitive Science Society.

Pettersson, O. (2005). Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, *53*, 73–88.

Ram, A., & Cox, M. T. (1994). Introspective reasoning using meta-explanations for multistrategy learning. In R. S. Michalski & G. Tecuci (Eds.), *Machine learning: A multistrategy approach IV*, 349–377. San Francisco, CA: Morgan Kaufmann.

Ram, A., & Leake, D. (1995). *Goal-driven learning*. Cambridge, MA: MIT Press.

Reder, L. M., & Ritter, F. (1992). What determines initial feeling of knowing? Familiarity with question terms, not with the answer. *Journal of Experimental Psychology*, *18*, 435–451.

Roberts, M., Borrajo, D., Cox, M. T., & Yorke-Smith, N. (2018). Special issue on goal reasoning. *AI Communications*, *31*, 115–116.

Schank, R., & Owens, C. (1987). Understanding by explaining expectation failures. In R. G. Reilly (Ed.), *Communication failure in dialogue and discourse*. Elsevier Science.

Schank, R. C. (1982). *Dynamic memory: A theory of reminding and learning in computers and people*. Cambridge, MA: Cambridge University Press.

Singh, P. (2005). *EM-ONE: An architecture for reflective commonsense thinking*. Doctoral dissertation, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA.

Stroulia, E., & Goel, A. (1995). Functional representation and reasoning for reflective systems. *Journal of Applied Intelligence*, *9*, 101–124.

Wang, X. (1995). Learning by observation and practice: An incremental approach for planning operator acquisition. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 549–557). Tahoe City, CA: Morgan Kaufmann.

Wang, X. (1996). Planning while learning operators. *Proceedings of the Third International Conference on Artificial Intelligence Planning Systems* (pp. 229–236). Edinburgh, Scotland: AAAI Press.

Wilson, M. A., McMahon, J., & Aha, D. W. (2014). Bounded expectations for discrepancy detection in goal-driven autonomy. *Papers from the 2014 AAAI Workshop on AI and Robotics*. Quebec City, Quebec, Canada: AAAI Press.

Yang, Q., Kangheng, W., & Yunfei, J. (2007). Learning action models from plan examples using weighted MAX-SAT. *Artificial Intelligence*, *171*, 107–143.