

---

## Perceptual Goal Monitors for Cognitive Agents in Changing Environments

---

**Michael T. Cox**

MICHAEL.COX@WRIGHT.EDU

Wright State Research Institute, Wright State University, Beavercreek, OH 45431 USA

**Zohreh A. Dannenhauer**

ALAVI.3@WRIGHT.EDU

Department of Computing Science and Engineering, Wright State University, Dayton, OH 45435 USA

### Abstract

Autonomous agents are expected to manage their behavior across many complex situations and to solve difficult problems that arise during their tenure. Environments that face such agents are often quite uncooperative, uncertain, and changing during the period an agent plans or executes the actions of its plans. The research here examines a cognitive mechanism to anticipate changes in the environment that allows the dropping of a given goal and the associated plan to achieve that goal. When goals are formulated, the agent creates a perceptual monitor associated with the conditions that represents the reasons the goal is preferred to the initial state. When these conditions no longer hold, the agent is able to release the goal and avoid further effort in service of its achievement. We show how this algorithm outperforms planners that do not use goal monitors or anticipate change.

### 1. Introduction

A cognitive agent represents an intelligent system that uses knowledge structures to achieve the goals to which it is committed in an environment within which it can perceive and act (Huhns & Singh, 1998). Cognitive architectures are fixed infrastructures that organize and manage the knowledge and the functional reasoning processes that interpret the environment and determine action in service of goals for implemented cognitive agents (Langley, Laird & Rogers, 2009). Such agent models span many domains and problem-solving tasks. The kinds of problems cognitive agents face have increasingly become those in which the agent's goals must be flexible given the dynamic nature of the environments within which they operate. Goals are not simply static predicate representations given as input by some external user. The agent itself is expected to recognize situations in which new goals are to be formulated or current goals changed and abandoned. This is the basic conception of *goal reasoning* (Aha, Cox, & Munoz-Avila, 2013; Hawes, 2011).

Rich goal structures facilitate the reasoning required for adaptive behavior in changing environments and provide focus for the application of scarce cognitive resources (Schank & Abelson, 1977; Simon, 1967). However few cognitive agents represent the reasons they pursue the goals they possess. Most often they blindly do what they are told (but see Aha & Coman, 2017;

Coman, Gillespie, & Munoz-Avila, 2015). Humans provide goals as input to a problem statement or as imperative commands to be carried out. Indeed many problems are simply arbitrary initial state and goal state configurations such as having one block on top of another (Cox, 2013). This paper begins to examine the tasks of reasoning about goals in terms of the contextual reasons they are being sought and monitoring that such conditions remain in effect as the agent plans for its goals and executes actions within the plans. If a human tells an agent to achieve a goal or perform a task, the agent should consider the reasons why (or why not) this is desirable. If the reasons are not obvious, then ideally the agent should ask the human to elaborate. If the agent formulates a goal on its own, it should have a reason for doing so. These reasons establish the means for monitoring that the goal is still worth achieving.

In addition to goal monitoring, we recognize a number of operations on goals and distinguish them for operations on plans (Cox, Dannenhauer, & Kondrakunta, 2017). Although the purpose of a plan is to establish a state of the world that satisfies a goal or a set of goals, we argue that the separation of goal and planning operations provides at least an organizational benefit within a cognitive architecture. However, like Roberts and colleagues (Roberts et al., 2016; 2015) who combine both types of operations into a goal life-cycle framework, we acknowledge the close relationship between the two. Table 1 classifies the ten primary operations on goal expressions.

Table 1. Fundamental set of goal operations (extended from Cox, Dannenhauer & Kondrakunta, 2017).

N	Operation	Description
1	Goal formulation	Create a new pending goal
2	Goal selection	Commit to an active goal from the set of pending goals
3	Goal suspension	Pause in pursuit of a currently committed goal
4	Goal resumption	Resume pursuit of a suspended goal
5	Goal change	Change a goal into a similar one that is close to the original goal
6	Goal monitoring	Track that a goal maintains its usefulness
7	Goal delegation	Find another agent willing to pursue a goal for you
8	Goal interpretation	Infer the meaning of a stated intent by another agent
9	Goal abandonment	Remove a pending or committed goal from consideration
10	Goal achievement	Verify that a goal state is satisfied in some environment

This paper addresses the *goal monitoring operation* (i.e., #6 above). Many planning approaches focus on the capability to generate and execute a sequence of actions that achieve a goal. Some planning approaches replan or adapt plans when the world changes or otherwise is uncooperative. When the goal suddenly becomes true in the current state, some agents will gracefully stop planning or cease executing a plan for a goal that is no longer needed. However few if any address the problem that goals are pursued for some reason. When the reason for the goal (as opposed to the goal itself) ceases to hold, the agent should also abandon the goal or otherwise change its behavior. We propose goal monitoring as a kind of cognitive process that oversees the continuing benefit of selective goal expressions and when situations warrant decides whether to abandon its goals.

The contributions of the paper include a declarative representation for goals, an extended enumeration of goal reasoning operations, the formalization of the goal monitor operation, an updated specification of the goal-reasoning function *beta*, and a cognitive systems implementation of the goal monitor and abandonment operations evaluated against a no-monitoring condition. We begin the paper by describing a representation for goal expressions and by formalizing the goal monitoring operation. We then show how a cognitive architecture called MIDCA implements these representations and operations. An empirical evaluation of this technique in a simple domain follows with related research and a conclusion finishing the paper.

## 2. Goals, Goal Monitors, and Goal Abandonment

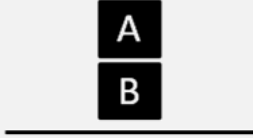
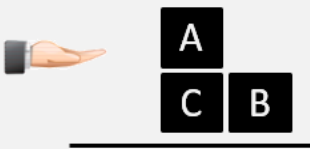
### 2.1 Representations for Goals

In many agent formalizations, the goal state is simply a first-order literal consisting of a truth predicate and a sequence of arguments. For example, the predicate representations  $on(A, B)$  is a goal of the agent desires a block with the label  $A$  to be on top of the block labelled  $B$ . However the agent may have such a goal but not be committed to achieving it yet (Cohen & Levesque, 1991). Thus we require a representation that can identify many of the attributes and modes through which goal traverse before finally being achieved (or not). Additionally, we wish to represent the goal relationship itself as a first-class mental object about which a cognitive system can reason as they can about physical objects in the environment.

In the pictorial illustration of Table 2, the goal is to have block A on B, but in the current state,  $s_0$ , A is on C instead. Here the goal and each block have a specific representation. The left hand side of the table shows them in RDF (resource description framework) triple format (Hayes & Patel-Schneider, 2014); whereas, the right hand side shows them using a frame representation (Fikes & Kehler, 1985; Minsky, 1974). The goal is to have a particular state hold between two arguments, the domain and co-domain. As any predicate, the goal has a current truth value, and it was created by operator  $op.9$ . Here we see that the goal belongs  $agent.8$ , and it is not currently delegated to another agent.  $Agent.8$  is committed to achieving the goal, and it is currently both active and monitored.

The problem with this representation, however, is that the  $on$  relationship for the blocks is an attribute of the blocks, yet the  $on$  goal is itself a different representational object with its own attributes. Representationally, the desired future state is to have the value of the  $on$  attribute of  $block.6$  to be  $block.17$  rather than  $block.10$ . But with classical frames, no mechanism exists with which to refer directly to the  $on$  attribute of  $block.6$ .

Table 2. Goal representations for  $g_c = on(A, B)$  from initial state  $s_0$  using RDF-like triples, pictorial illustrations, and frame structures. Note that the reasons the goal is pursued is captured in the monitor structure discussed later in the paper rather than in the goal shown here.

Subject	Predicate	Object	Pictorial States	Classical Frame
goal.5	Isa	on	 <p><math>g_c = goal.5</math></p>	goal.5:
goal.5	Domain	block.6		( on (domain block.6)
goal.5	co-domain	block.17		(co-domain block.17)
goal.5	truth-value	$\perp$		(truth-value $\perp$ )
goal.5	created-by	op.9		(created-by op.9)
goal.5	delegated-to	nil		(delegated-to nil)
goal.5	goal-of	agent.8		(goal-of agent.8)
goal.5	Mode	committed		(mode committed)
goal.5	Activation	activated		(activation activated)
goal.5	is-monitored	T		(is-monitored T) )
block.6	Clear	T	 <p><math>s_0</math></p>	block.6: (cube (clear T)
block.6	On	block.10		(on block.10)
block.6	Name	A		(name "A") )
block.17	Clear	T		block.17: (cube (clear T)
block.17	Name	B		(name "B") )
block.10	Clear	$\perp$		block.10: (cube (clear $\perp$ )
block.10	Name	C		(name "C") )

So to represent explicitly the relationship between blocks A and C, we reify the on predicate in a frame structure. Then for a frame system to treat this uniformly, another level of nesting provides the necessary representation to differentiate attribute values and the attribute relationship itself. We adapted an approach from the representational formalism in Cox (1996) and show it below.

```

block.6: ( cube (clear (value T))
          (on (value block.10) // value facet of on attribute for block A
              (relation state.21) // explicit representation as frame facet
              (name (value "A")) )

state.21: ( on (domain (value block.6)) // 1st-class relationship between blocks A and C
            (co-domain (value block.10)) )

```

Treating a goal as a first-class, declarative knowledge-structure linked to explicit relations over objects enables a uniform mechanism for multiple cognitive processes in the goal-reasoning framework. The goal operations then serve as executive management functions from one declarative goal structure to another. Many research efforts have recognized the importance of rich representations for goals (e.g., Braubach, Pokahr, Moldt, & Lamersdorf, 2004; Hinrichs & Forbus, 2016), and some have emphasized goals as first-class representational structures (e.g., Hinrichs &

Forbus, 2013). Additionally, many planning systems monitor plan execution in changing environments and adapt plans when conditions prove unfavorable (e.g., Ayan, Kuter, Yaman & Goldman, 2007; Munoz-Avila & Cox, 2008; Pettersson, 2005). Here we focus on the monitoring of the reasons goals are being pursued rather than the plans that accomplish the goals. Replanning adapts the plan when the plan starts to fail; goal monitoring adapts or drops the goal when the goal justification conditions no longer hold.

## 2.2 The Goal Monitor and Abandonment Operations

We posit a model of goal operations that represents the set of transformations on goals an agent may choose (Cox, in press). The agent's *goal agenda*  $\hat{G} = \{g_1, g_2, \dots, g_c, \dots, g_n\}$  contains the current goal  $g_c$ . An individual goal operation,  $\delta: G \rightarrow G$ , is a function from one goal expression  $g \in G \subset S$  to another  $g'$ , where  $S$  is the set of all possible states. An operation is formalized as a transformation represented by  $\delta = (\text{head}(\delta), \text{parameter}(\delta), \text{pre}(\delta), \text{res}(\delta))$ , where  $\text{pre}(\delta)$  and  $\text{res}(\delta)$  are its preconditions and result. The decisions  $\langle \delta^1, \delta^2, \dots, \delta^n \rangle$  result in the goal  $\delta_n(\dots \delta_2(\delta_1(g_c))) = g'$ .

Table 3 shows the goal reasoning function  $\beta$  that perceives the world with respect to its goals, managing them as necessary. More formally, the function  $\beta: S \times G \rightarrow G$  returns a (possibly new) goal  $g'$  given some state  $s$  and a current goal  $g_c$ . The distinguished transformation  $\delta^*$  represents goal formulation and has been detailed in Cox (in press; 2013). Here we assume  $\delta^*$  can also be implemented with goal operators as discussed in section 3.1. Three main cases exist within the beta function: (1) only goal formulation  $\delta^*$  is chosen; (2) the chosen elements consist of goal formulation and one or more goal changes; (3) only goal changes are chosen. For each case, beta updates the goal agenda, and if necessary, changes the goal. An output value for the input goal is finally returned.

Table 3. The *beta* goal-reasoning function (adapted from Cox, Dannenhauer, & Kondrakunta, 2017). Although  $\Delta$  is an ordered set,  $\hat{\Delta}$  is a sequence where *in* is treated like the set operator  $\in$  and “-“ like set difference. Reverse maintains the order of  $\Delta$  (choose inverts it).

---

<b><math>\beta(s: S; g_c: G): G</math></b>	
$\hat{\Delta} \leftarrow \text{reverse}(\text{choose}(s, g_c, \Delta))$	
if $\delta^* \text{ in } \hat{\Delta}$ then	// if new goal formulated
$g_q \leftarrow \delta^{mo}(\delta^*)$	// then goal monitoring applied to new goal
if $\hat{\Delta} = \langle \delta^* \rangle$ then	// case 1: goal formulation only
$\hat{G} \leftarrow \{g_1, g_2, \dots, g_c, \dots, g_n\} \cup g_q$	
$\beta \leftarrow g_c \wedge g_q$	
else $\langle \delta_1, \delta_2, \dots, \delta_m \rangle = \hat{\Delta} \leftarrow \hat{\Delta} - g_q$	// case 2: goal formulation plus goal change
$\hat{G} \leftarrow \{g_1, g_2, \dots, \delta_m(\dots \delta_2(\delta_1(g_c))), \dots, g_n\} \cup g_q$	
$\beta \leftarrow \delta_m(\dots \delta_2(\delta_1(g_c))) \wedge g_q$	
else $\hat{G} \leftarrow \{g_1, g_2, \dots, \delta_m(\dots \delta_2(\delta_1(g_c))), \dots, g_n\}$	// case 3: goal change only
$\beta \leftarrow \delta_m(\dots \delta_2(\delta_1(g_c)))$	

---

Table 4 shows the function *choose* used within Beta. From an input set of goal transformations  $\Delta$ , the recursive choose function returns the sequence of transformations whose preconditions are satisfied in the current state  $s$ . Note that a sequence whose first element is head and whose remaining element are the subsequence tail is written as “head | tail”. In set notation, however, the symbol “|” signifies “such that.”

Table 4. The *choose* function (adapted from Cox, Dannenhauer, & Kondrakunta, 2017). The function ends up reversing the order of the poset  $\Delta$ .

---

```

choose( $s: \mathcal{S}, g_c: \mathbf{G}, \Delta = \{\delta_1, \delta_2, \dots\}: \text{poset}$ ): sequence
return (if  $\Delta = \{\}$  then  $\langle \rangle$ 
      else if  $[\forall x | x \in \text{pre}(\delta_1) \wedge (s \cup g_c) \models x]$  then
         $\delta_1 | \text{choose}(s, g_c, \Delta - \{\delta_1\})$ 
      else choose ( $s, g_c, \Delta - \{\delta_1\}$ ))

```

---

Table 5 formalizes the *goal monitor operation* as a goal transformation,  $\delta^{mo}$ . The content of the goal does not change in this operation. Rather the is-monitored attribute changes. This flag on the goal signals to an implementation that a monitor procedure needs creation. The monitor includes 2 major conditions. First the monitor encapsulates environmental conditions whose change signals the need for goal reconsideration. Second the monitor includes a specification of the response (e.g., goal abandonment) if perceptions detects the first condition. We will examine this next.

Table 5. The goal monitor operation as transformation. Bergmann’s (2002) notation  $\mathbf{CL}$  is a class hierarchy having leaves  $L_C \subseteq \mathbf{CL}$  and whose root class  $C$  has superclass  $\top$ , i.e.,  $C_{\text{superclass}} = \top$ . Precondition  $\text{pre}_2$  of the transformation assures that the goal is not already monitored. State  $s$  in  $\text{pre}_2$  is within scope of  $\beta$  above.

---

```

 $\delta^{mo}(g_c: \mathbf{G}): \mathbf{G}$ 
head( $\delta^{mo}$ ) = monitor
parameter( $\delta^{mo}$ ) =  $g_c = p(\text{obj1}, \text{obj2})$ 
pre1( $\delta^{mo}$ ) =  $\text{obj1} \in \text{Objs} \wedge \text{obj2} \in \text{Objs}$ 
pre2( $\delta^{mo}$ ) =  $\exists p, p', i | p \in \mathbf{CL} \wedge p' \in \mathbf{CL} \wedge p_{\text{superclass}} = p' \wedge p = (p_{\text{name}}, p', (p.A_1, p.A_2, \dots, p.A_m))$ 
   $\wedge 1 \geq i \geq m \wedge A_i = \text{is-monitored} \wedge p.\text{is-monitored} = \perp$ 
pre3( $\delta^{mo}$ ) = needs-monitoring( $s, g_c$ ) // reasoning whether to monitor the goal
pre( $\delta^{mo}$ ) = {pre1( $\delta^{mo}$ ), pre2( $\delta^{mo}$ ), pre3( $\delta^{mo}$ )}
res( $\delta^{ge}$ ) = if  $\forall x | x \in \text{pre}(\delta^{mo}) \wedge (s \models x)$  then  $p.\text{is-monitored} \leftarrow \top$ 
return( $g_c$ )

```

---

### 3. Goal Operations in MIDCA

The *metacognitive integrated dual-cycle architecture (MIDCA)* is a cognitive architecture that models both cognition and metacognition for intelligent agents (Cox et al., 2016; Paisner et al.,

2014). It consists of “action-perception” cycles at both the cognitive level and the metacognitive level (see Figure 1). In general, a cycle performs problem-solving to achieve its goals and tries to comprehend the resulting actions and those of other agents. The output side of each cycle consists of intention, planning, and action execution, whereas the input side consists of perception, interpretation, and goal evaluation.

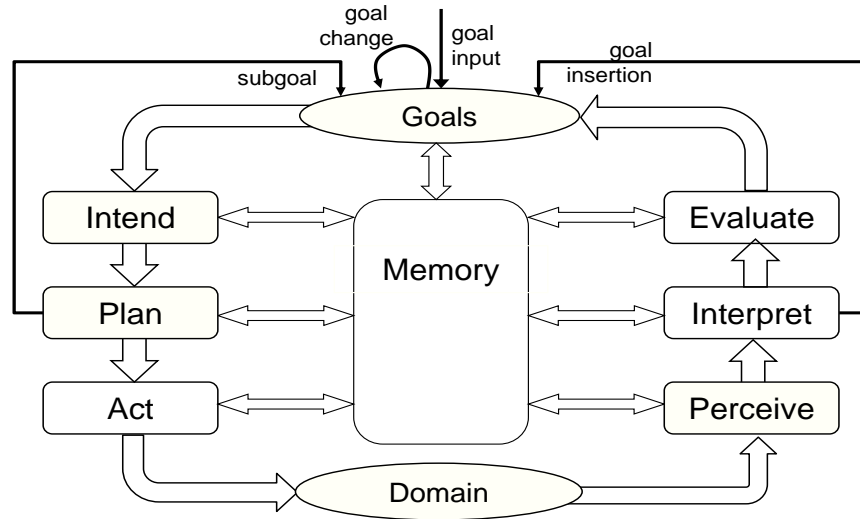


Figure 1. Schematic action-perception cycle for both cognitive and metacognitive levels in MIDCA (Cox et al., 2016). Intend, Plan, and Act compose the problem-solving mechanism in the architecture, and Perceive, Interpret, and Evaluation constitute the comprehension mechanism. Memory is shared between them.

In problem solving, the Intend phase commits to a current goal from those available. The Plan phase then generates a sequence of actions (a hierarchical-task-network plan). The plan is executed by the Act phase to change the actual world through the effects of the planned actions. The agent will then use these expectations in the next cycle to evaluate the execution of the plan.

Comprehension starts with perception of the world through the Perceive phase. The Interpret phase takes as input the resulting predicate relations and the expectations in memory to determine whether the agent is making sufficient progress. It is here that new goals are generated when the environment presents problems and opportunities for the agent. The Evaluate phase incorporates the concepts inferred from Interpret and notes whether existing goals are achieved.

### 3.1 Goal Monitoring

In a cognitive system, goals provide focus for the agent’s reasoning and represent the desired future state it seeks to achieve. Three types of goal monitors can exist for these knowledge structures.

1. **Operator style.** Observing preconditions of the goals;
2. **Explanatory.** Observing the causal justifications of the goal;
3. **Direct.** See if the goal is achieved exogenously.

*Explanatory monitors* assume that the goal was formulated in response to a discrepancy between the agent’s expectations and the observation (Cox, 2007; Dannenhauer & Munoz-Avila, 2015). An explanation provides the antecedents for the discrepancy, and the agent generates a goal from the explanation. The antecedents also provide the environmental conditions that must persist for the goal to still be valid. *Direct monitors* check that the goal itself does not exogenously become true at some point in the planning or in plan execution. If they do, then the goal can be dropped from either the set of pending goals or from the current goal expression.

In this paper, we consider the *operator style monitor* and will leave implementation of the explanatory monitor for future work. In MIDCA, this monitor class currently depends upon goals being generated by *goal operators*.<sup>1</sup> We denote a goal operator,  $o$ , as the tuple  $(name(o), precond(o), result(o))$ . The set of literals  $precond(o)$  represents the operator’s preconditions. They specify what conditions the current state must satisfy in order for  $o$  to be applied. The persistence of these conditions in the future is also the target of the goal monitor. The term  $result(o)$  specifies the goal  $g$ . Tac-Air Soar (Jones, Laird, Nielsen, Coulter, Kenny, & Koss, 1999) takes this approach. Operators exist for various goal types and data-driven context-sensitive rules spawn them given matching run-time observations.

Algorithm 1 shows high-level details in the MIDCA Interpret phase. When a feature being monitored changes and the change is detected, we say that the monitor fires. If a monitor fired, then the goal will be abandoned and removed from pending goals  $\hat{G}$  (steps 2-6 in Algorithm 1). The algorithm next checks to see if a new goal is created (steps 7-10). If a new goal exists and Interpret decides to have it monitored, a monitor will be created for  $g_n$ ’s operator (steps 10-13).

Algorithm 2 shows the details of creating monitors for the preconditions of a given goal operator  $op$ . These preconditions must be monitored, because, should they become false, the goal is not useful in the current state anymore. This indirectly assumes that goal formulation is performed when the operator preconditions hold in the current state.

The function *fired* checks for monitors that trigger (see Algorithm 3). Perceive creates a set of percepts from environmental input ( $\Psi$ ) and induces a predicate representation  $s$  from these percepts (Alavi & Cox, 2016). It then takes a list of monitors and checks if the conditions are still satisfied in  $s$ . If not, it will assemble a list of goals to drop from a list of the agent’s pending goals.

---

<sup>1</sup> Note that goal operators are distinct from planning operators. The former represents an alternative choice for goal formulation; whereas the latter is an action model that represents a potential choice for step in a plan. An agent may create a goal monitor based on the preconditions of the goal operator and may create a plan monitor based on the preconditions of the plan operator (see Alavi & Cox, 2016; Dannenhauer & Cox, in press, for details regarding our use of plan monitors in MIDCA). In such light, the monitors are very similar, but when the monitor condition fires, a goal monitor abandons or otherwise changes the goal; whereas a plan monitor changes the plan. We anticipate a divergence of similarity between the two for explanatory goal monitors.



*Algorithm 1.* Goal monitoring in the MIDCA Interpret phase. Goal formulation and goal abandonment accompany the monitoring procedure.

---

**Input:** list of pending goals  $\hat{G}$ , current state  $s$ , and list of goal monitors  $mnts$

**Output:** list of goal monitors  $mnts$

---

```

1: function goal_monitoring ( $\hat{G}, s, mnts$ )
2:   for  $g$  in fired( $mnts$ )
3:      $\hat{G} \leftarrow \hat{G} - g$            // goal abandoned
4:      $g.is\text{-monitored} \leftarrow \perp$ 
5:      $mnts \leftarrow mnts - \{(p, g) | p = \text{monitored}\text{-states}(mnts, g)\}$  //removing monitor from mnts
6:   end for
7:    $pending \leftarrow \hat{G}$            // temp var
8:   // Three cases of  $\beta$  below:  $g_n = g_c$  (no change);  $g_c$  may change into  $g_n$ ; or  $g_n$  added to  $\hat{G}$ 
9:    $g_n \leftarrow \beta(s, g_c)$ 
10:  if  $|pending| + 1 = |\hat{G}|$       // when new goal formulated in beta,  $\hat{G}$  will be larger by 1
11:     $\wedge g_n.is\text{-monitored}$  then
12:       $mnts \leftarrow \text{generate\_monitors}(g_n.created\text{-by}, s, mnts)$ 
13:    end if
14:  return( $mnts$ )
15: end function

```

---

*Algorithm 2.* Goal-monitor generation. The algorithm assumes that the monitor uses the operator style.

---

**Input:** goal operator  $o$ , world state  $s$ , and list of monitors  $mnts$

**Output:** list of monitors  $mnts$

---

```

1: function generate_monitors ( $o, s, mnts$ )
2:   for  $p$  in precondition( $o$ ) do
3:      $mnts \leftarrow (p, g) \cup mnts$ 
4:   end for
5:   return ( $mnts$ )
6: end function

```

---

*Algorithm 3.* Goal monitor firing. The algorithm checks the state for monitors that fire.

---

**Input:** list of monitors  $mnts$

**Output:** list of goals to drop  $goals\_to\_drop$

---

```

1: function fired ( $mnts$ )
2:    $s \leftarrow \text{perceive}(\Psi)$ 
3:    $goals\_to\_drop \leftarrow \emptyset$ 
4:   for  $(p, g)$  in  $mnts$ 
5:     if  $\neg(s \models p)$  then
6:        $goals\_to\_drop \leftarrow g \cup goals\_to\_drop$ 
7:     end if
8:   end for
9:   return ( $goals\_to\_drop$ )
10: end function

```

---

### 3.2 A Short Goal Monitoring Example

Table 6 illustrates an example goal operator for a logistics delivery task. If MIDCA receives an order to deliver package  $p_1$  to location  $l_{11}$  and  $p_1$  is available in one of the warehouses (e.g.,  $w_2$ ), the *beta* function uses the goal operator to formulate the delivery goal  $g_7 = delivered(p_1, l_{11})$ . If MIDCA decides to monitor this goal, monitors are created to observe the conditions  $obj-at(p_1, w_2)$  and  $order(p_1, l_{11})$ . Now if at a later time,  $p_1$  is stolen or otherwise becomes missing from the warehouse or if the order is canceled, then the monitor will abandon  $g_7$ , removing it from  $\hat{G}$ .

Table 6. Exemplified goal operator for delivering an ordered package.

Attribute	Representation
Goal operator	$o(?p, ?w, ?l)$
Preconditions	$obj-at(?p, ?w), order(?p, ?l), delivered(?p, ?l) \notin \hat{G}$
Result	$g = delivered(?p, ?l)$
Monitor conditions	$obj-at(?p, ?w), order(?p, l)$

## 4. Empirical Performance Evaluation

We claim that using goal monitors in a cognitive architecture like MIDCA increases the performance of the agent. To evaluate this hypothesis, we conducted tests with MIDCA on a simulated logistics domain. We use a simple simulator to model the world state and agent actions that change the state.

### 4.1 Logistics Domain Experiments

The version of the logistics domain (Veloso, 1994) we use includes packages inside different warehouses that are needed to be delivered to their destinations by trucks or airplanes. The agent is tasked to deliver packages for different orders. For example, transporting the package  $p_1$  by truck to location  $l$  and then unloading it achieves the goal  $g_c = delivered(p_1, l)$ . We assume that the agent has full observability and has access to the list of packages in the warehouses.

When there is an order for a package and the package exists in one of the warehouses, Interpret generates a delivery goal for that package. The MIDCA Intend phase selects one warehouse and commits to achieving all goals for the packages in that warehouse. The JSHOP planner<sup>2</sup> (Nau et al. 2003) that implements the MIDCA Plan phase then generates a plan for these packages. If one package is stolen from a warehouse  $w_i$ , the planner fails to generate a plan for all delivery goals in  $w_i$ . However, with goal monitors, the agent will know that a package is missing, and before Plan starts planning for that warehouse, it will drop the goal for the missing package. Planning will now succeed for the current goals. Notice that lost packages are distributed evenly across warehouses, and we assume that packages are stolen from the warehouse that is not planning for currently.

We ran two experiments. In the first, we varied the number of warehouses, and in the second, we varied the number of lost packages. Every goal achieved (each package delivered) by MIDCA

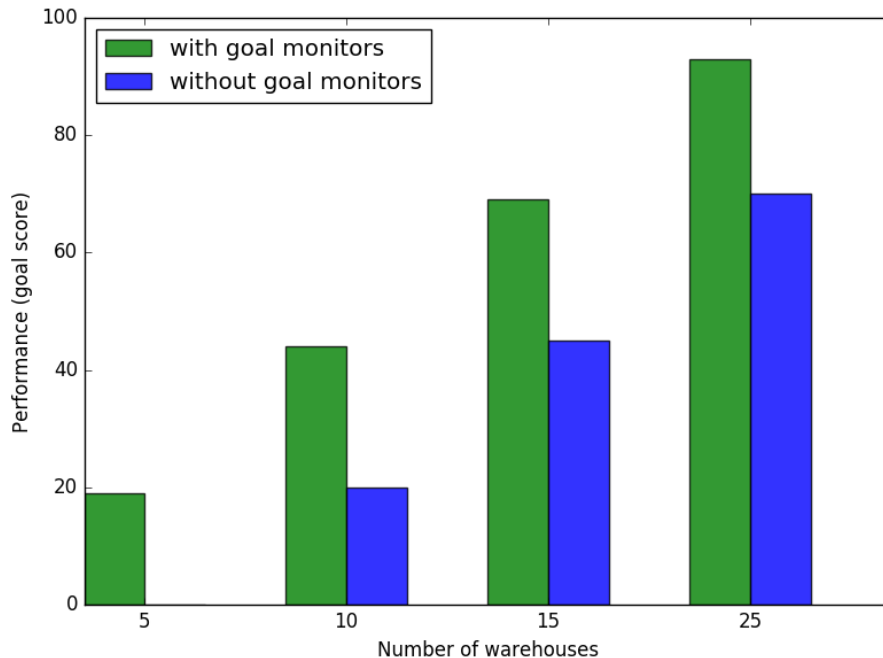
<sup>2</sup> <https://sourceforge.net/projects/shop/files/JSHOP2/>

has a score of one point. In each scenario of the first experiment, we set the initial state to be the one with  $n$  warehouses, five packages in each warehouse, and one order exists for each package. Interpret generates a delivery goal for each package. During runtime, six packages from different warehouses become lost. We varied the number of warehouses from five to twenty in increments of five. Each warehouse has five packages.

In each scenario of the second experiment, the initial state is set to be the one with twenty warehouses with five packages each. During runtime,  $n$  packages are lost. We varied the number of lost packages from one to twenty.

## 4.2 Experimental Results

Figures 2 and 3 summarize the results of MIDCA with and without goal monitors for these two experiments. The y-axis is the goal score that the agent was able to achieve for delivering packages. We plot the score as a function of the amount of warehouses in Figure 2 and in Figure 3 as a function of the number of lost packages. The results show that the performance of MIDCA with goal monitors is better than MIDCA with static goals (e.g., no goal monitors), because goal monitors allow the agent to drop its goals when they are not achievable. In Figure 2, when the number of warehouses is five, MIDCA without goal monitors is not able to achieve any goal (one package is lost from each warehouse causing all plans to fail).



*Figure 2.* Logistics domain performance with goal monitors and without goal monitors. Number of stolen packages is six in each warehouse. Each warehouse has five packages.

Figure 3 shows the result of the second experiment with twenty warehouses. When no package is lost, both approaches show equivalent performance. But when more packages are stolen, MIDCA with goal monitors is able to achieve a higher score by dropping delivery goals of lost packages. The score of MIDCA with static goals converges to zero as more packages are lost. The results support our claim that the goal monitors technique improves the performance of a cognitive architecture in a simulated logistics domain.

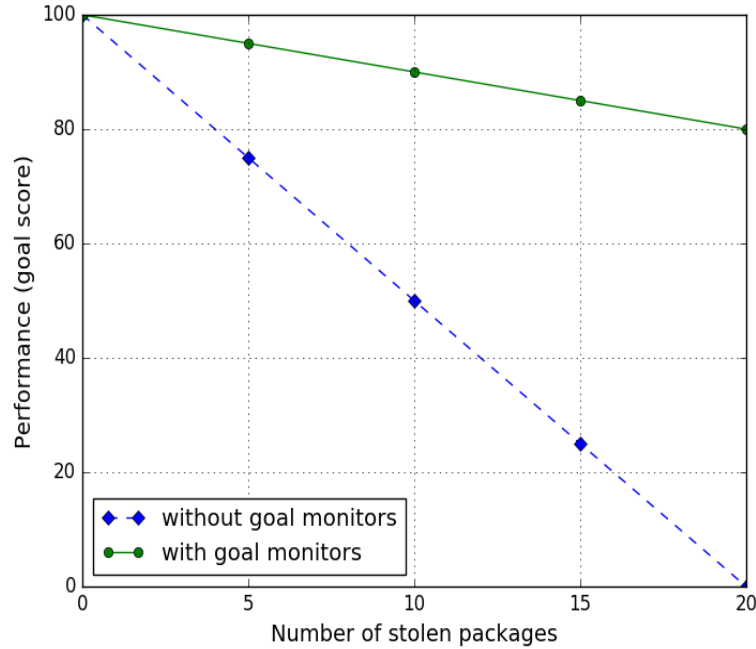


Figure 3. Logistics domain performance in MIDCA for twenty warehouses with five packages in each.

## 5. Related Research

*Goal-driven autonomy (GDA)* (Aha, et al., 2010; Cox, 2013; Klenk, Molineaux & Aha, 2013) is a kind of goal reasoning that focuses on explanation of discrepancies in order to formulate new goals. Our work is firmly situated within this research area. GDA agents generate goals as the agent encounters differences between the agent’s expectations for the outcome of its actions and the actual observed outcomes in each new state (Dannenhauer & Munoz-Avila, 2015). When such a discrepancy occurs, GDA agents generate a causal explanation for the discrepancy, and generate a new goal based on the causal structure.

Other research in the GDA and broader goal reasoning community has formalized the idea of goal change and goal reasoning. Roberts and colleagues (Johnson, Roberts, Apker, & Aha, 2016; Roberts et al., 2016; 2015) has developed the notation of a goal life-cycle where goals transition through modalities that represent goal formulation, goal selection, goal expansion, goal

commitment, goal dispatching, goal monitoring, goal evaluation, goal repair, and goal deferment. Many of these transitions corresponds to our goal operations, but their formalism itself treats goal reasoning as *goal refinement* where our casts it in standard notation similar to that used in the automated planning community. Additionally Roberts proposes a complex goal structure that differs from ours. Their *goal node* includes not only the desired state but also super-ordinate and subordinate goal linkages, goal constraints, quality metrics, and pointers to the current plan associated with the node. Finally we note that the BDI (belief-desire-intention) community has also developed a goal life cycle representation and formalized a set goal operations. See Harland, Morley, Thangarajah, & Yorke-Smith (2014). This work, unlike the goal reasoning community, does not focus on goal change and goal formulation. The BDI community has also developed a sophisticated mechanism for performing goal suspension, resumption, and abandonment (Harland, Morley, Thangarajah, & Yorke-Smith, 2017). Their work differentiates *goal abandonment* (where plans are cleaned up and then the goal is dropped) from the direct *goal drop* operation itself.

The BDI work cited above also characterizes a kind of *goal monitoring* for maintenance goals that assures a particular state holds across an interval of time. These goals contrast with achievement goals that establish a particular state at a point in time. This process monitors the state and (re)activates the maintenance goal whenever the state changes during the interval. Our use of goal monitoring is directed at achievement (i.e., attainment) goals and monitors the reasons goals were formulated in the first place. Although both are called goal monitoring, our work is very different.

Finally, cognitive agents including GDA agents need to adapt to changes in the environment. *Rationale-based plan monitors* (Alavi & Cox, 2016; Veloso, et al., 1998) provide a means of focusing visual attention on features of the world likely to affect the plan during planning time. The MIDCA Plan phase generates these monitors to interact with a vision system and react only to those environmental changes that bear on current planning decisions. When plan monitors detect relevant changes, corresponding plan transformations are executed as needed. Alternatively, the work in this paper concerns reactions to environmental changes that affect the agent's goals. The Interpret phase in MIDCA generates goal monitors associated with the conditions that led the agent to choose that goal. It enables the agent to abandon the goal when the goal is not useful in the current state.

## 6. Conclusion

Autonomous cognitive agents reason about and formulate their own goals and need to adapt to changes in the environment. In this paper, we introduce goal monitors for cognitive systems that observe the justification for goal selection and abandon the goal when justifications are not valid in the environment. This research follows the cognitive systems research paradigm (Langley, 2012) in that it focuses on high-level cognition, represents goals as structured knowledge, is a systems-level research topic, implements a heuristic approach to intelligence, is inspired by human cognition, and despite the notational formalism, is an exploratory rather than formal research endeavor. This research represents an increment in the exploration, specification and further

understanding of the functional roles goal operations contribute to successful high-level reasoning and subsequent robust behavior for cognitive agents in difficult environments.

Much future work remains to be performed. We have implemented a simple form of goal monitors that works similar to existing plan monitor implementations. As mentioned previously, however, the goal monitor process will diverge from plan monitors once we implement explanatory monitors as well. Currently the monitors simply drop the goals if the monitored conditions no longer hold. Yet alternative responses exist in many situations that would be preferred over abandonment. Goals might be changed instead. To make such choice, we need to develop a mechanism to reason about the response once a monitor fires rather than simply carry out a predetermined response. Finally the evaluation presented here is still preliminary. We showed results that were essentially obvious, though as to our knowledge, this had not been previously demonstrated in the literature. A more thorough empirical evaluation rests in the near future.

### Acknowledgements

This research was supported by ONR grants N00014-15-1-2080 and N00014-15-C-0077. We thank the anonymous reviewers for their comments and suggestions.

### References

- Aha, D. W., & Coman, A. (2107). The AI rebellion: Changing the narrative. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (pp. 4826-4830). Menlo Park, CA: AAAI Press.
- Aha, D. W., Klenk, M., Munoz-Avila, H., Ram, A., & Shapiro, D. (Eds.) (2010). *Goal-driven Autonomy: Notes from the AAAI Workshop*. Menlo Park, CA: AAAI Press.
- Aha, D. W., Cox, M. T., & Munoz-Avila, H. (Eds.) (2013). *Goal Reasoning: Papers from the ACS workshop* (Tech. Rep. No. CS-TR-5029). College Park, MD: University of Maryland.
- Alavi, Z., & Cox, M. T. (2016). Rationale-based visual planning monitors. In *Working Notes of the 4th Workshop on Goal Reasoning*. New York, IJCAI-16.
- Ayan, N.F., Kuter, U., Yaman F., & Goldman R. (2007). Hotride: Hierarchical ordered task replanning in dynamic environments. In F. Ingrand, & K. Rajan (Eds.) *Planning and Plan Execution for Real-World Systems – Principles and Practices for Planning in Execution: Papers from the ICAPS Workshop*. Providence, RI.
- Bergmann, R. (2002). *Experience management: Foundations, development methodology, and internet-based applications*. Berlin: Springer.
- Braubach, L., Pokahr, A., Moldt, D., & Lamersdorf, W. (2004). Goal representation for BDI agent systems. In R. H. Bordini et al. (Eds.) *PROMAS 2004, LNAI 3346* (pp. 44–65). Berlin: Spring.
- Cohen, P. R., & Levesque, H. J. (1990). Intention is choice with commitment. *Artificial Intelligence*, 42, 213-261.
- Coman, A., Gillespie, K., & Munoz-Avila, H. (2015). Believable emotion-influenced perception: The path to motivated rebel agents. In D. W. Aha (Ed.), *Goal reasoning: Papers from the ACS workshop*. Tech. Rep. No. GT-IRIM-CR-2015-001. Atlanta, GA: Georgia Institute of Technology, Institute for Robotics and Intelligent Machines.

- Cox, M. T. (in press). A goal reasoning model of planning, action, and interpretation. To appear in *Advances in Cognitive Systems*.
- Cox, M. T. (2013). Goal-driven autonomy and question-based problem recognition. In *2nd Annual Conference on Advances in Cognitive Systems 2013, Posters* (pp. 29-45). Cog. Sys. Foundation.
- Cox, M. T. (2007). Perpetual self-aware cognitive agents. *AI Magazine*, 28(1), 32-45.
- Cox, M. T. (1996). *Introspective multistrategy learning: Constructing a learning strategy under reasoning failure* (Tech. Rep. No. GIT-CC-96-06). Doctoral dissertation, Georgia Tech, Atlanta.
- 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. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence* (pp. 3712-3718). Palo Alto, CA: AAAI Press.
- Cox, M. T., Dannenhauer, D., & Kondrakunta, S. (2017). Goal operations for cognitive systems. In *Proceedings of the Thirty-first AAAI Conf. on Artificial Intelligence*. Palo Alto, CA: AAAI Press.
- Dannenhauer, Z., & Cox, M. T. (in press). Rationale-based visual planning monitors for cognitive systems. To appear in *Proceedings of the 30th International FLAIRS Conf.* Palo Alto, CA: AAAI Press.
- Dannenhauer, D. and Muñoz-Avila, H. (2015) Raising expectations in GDA agents acting in dynamic environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*. AAAI Press.
- Fikes, R., & Kehler, T. (1985). The role of frame-based representation in reasoning. *Communications of the ACM*, 28(9), 904-920.
- Harland, J., Morley, D. N., Thangarajah, J., & Yorke-Smith, N. (2017). Aborting, suspending, and resuming goals and plans in BDI agents. *Autonomous Agents and Multi-Agent Systems*, 31(2), 288-331.
- Harland, J., Morley, D. N., Thangarajah, J., & Yorke-Smith, N. (2014). An operational semantics for the goal life-cycle in BDI agents. *Autonomous Agents and Multi-Agent Systems*, 28, 682-719.
- Hawes, N. (2011). A survey of motivation frameworks for intelligent systems. *Artificial Intelligence*, 175(5-6), 1020-1036.
- Hayes, P. J., & Patel-Schneider, P. F. (2014, February). *RDF 1.1 semantics: W3C recommendation*. W3C. Retrieved February 24, 2017, from <http://www.w3.org/TR/2014/REC-rdf11-mt-20140225>.
- Hinrichs, T. R., & Forbus, K. D. (2013). Beyond the rational player: Amortizing type-level goal hierarchies. In Aha, D. W., Cox, M. T., & Munoz-Avila, H. (Eds.), *Goal Reasoning: Papers from the ACS workshop*. Tech. Rep. No. CS-TR-5029. College Park, MD: University of Maryland, Department of Computer Science.
- Hinrichs, T. R., & Forbus, K. D. (2016). Qualitative models for strategic planning In *Advances in Cognitive Systems*, 4, 75-92.
- Huhns, M. N., & Singh, M. P. (1998, November). Cognitive agents. *IEEE Internet Computing*.
- Johnson, B., Roberts, M., Apker, T., & Aha, D. (2016). Goal reasoning with information measures. In *Fourth Annual Conference on Advances in Cognitive Systems 2016*. Palo Alto, CA: Cognitive Systems Foundation.

- Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine*, 20(1), 27-41.
- Klenk, M., Molineaux, M., & Aha, D. W. (2013). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence* 29(2): 187–206.
- Langley, P., Laird, J. E., & Rogers, S. (2009). Cognitive architectures: Research issues and challenges. *Cognitive Systems Research*, 10(2), 141-160.
- Langley, P. (2012). The cognitive systems paradigm. *Advances in Cognitive Systems*, 1, 3–13.
- Minsky, M. (1974). *A framework for representing knowledge*. MIT AI Memo 306. Cambridge.
- Munoz-Avila, H., & Cox, M. T. (2008). Case-based plan adaptation: An analysis and review. *IEEE Intelligent Systems*, 23(4), 75-81.
- Nau, D., 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., Maynard, M., & Perlis, D. (2014). Goal-driven autonomy for cognitive systems. In *Proceedings of the 36th Annual Conference of the Cognitive Science Society* (pp. 2085–2090). Austin, TX: Cognitive Science Society.
- Petterson, O. (2005). Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems*, 53, 73–88
- Roberts, M., Shivashankar, S., Alford, R., Leece, M., Gupta, S., Aha, D. W. (2016). Goal Reasoning, Planning, and Acting with ACTORSIM, The Actor Simulator. In *Proceedings of the 4th Annual Conference on Advances in Cognitive Systems 2016*. Cog. Systems Foundation.
- 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, & A. Orlandini (Eds.), *Planning and Robotics: Papers from the ICAPS Workshop*. Palo Alto, CA: AAAI Press.
- Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*. Hillsdale, NJ: Lawrence Erlbaum Associates.
- Simon, H. A. (1967). Motivational and emotional controls of cognition. *Psychological Review*, 74(1), 29-39.
- Veloso, M. M. (1994). *Planning and learning by analogical reasoning*. Berlin: Springer.
- Veloso, M. M., Pollack, M. E., & Cox, M. T. (1998). Rationale-based monitoring for continuous planning in dynamic environments. In R. Simmons, M. Veloso, & S. Smith (Eds.), *Proceedings of the 4<sup>th</sup> International Conference on Artificial Intelligence Planning Systems* (pp. 171-179). Menlo Park, CA: AAAI Press.