Goal Operations for Cognitive Systems

Michael T. Cox, Dustin Dannenhauer, Sravya Kondrakunta

Wright State Research Institute
Beavercreek, OH 45431
michael.cox@wright.edu

Lehigh University
Bethlehem, PA 18015
dtd212@lehigh.edu

Wright State University
Dayton, OH 45435
kondrakunta.2@wright.edu

Abstract

Cognitive agents operating in complex and dynamic domains benefit from significant goal management. Operations on goals include formulation, selection, change, monitoring and delegation in addition to goal achievement. Here we model these operations as transformations on goals. An agent may observe events that affect the agent's ability to achieve its goals. Hence goal transformations allow unachievable goals to be converted into similar achievable goals. This paper examines an implementation of goal change within a cognitive architecture. We introduce goal transformation at the metacognitive level as well as goal transformation in an automated planner and discuss the costs and benefits of each approach. We evaluate goal change in the MIDCA architecture using a resource-restricted planning domain, demonstrating a performance benefit due to goal operations.

Introduction

Recent work on *goal reasoning* (Aha, *et al.,* 2013; Hawes, 2011) has started to examine how intelligent agents can reason about and generate their own goals instead of always depending upon a human user directly. Broadly construed, the topic concerns complex systems that self-manage their desired goal states (Vattam, *et al.*, 2013). In the decision-making process, goals are not simply given as input from a human, rather they constitute representations that the system itself formulates, changes, and achieves. Here we examine the idea that goal reasoning constitutes a series of fundamental operations on goals that focus cognition on what is important for performance.

When the world changes during planning or during execution (in the real world, a clear chronological line between the two is not always present), goals may become obsolete. For example, it makes little sense to pursue the goal of securing a town center if the battlefield has shifted to an adjacent location. At such a point, a robust agent must be able to alter the goal minimally to compensate; otherwise, a correct plan to secure the old location will not be useful at execution time. We view a goal transformation to be a movement in a goal space and in this paper show how such procedures can be incorporated into various mechanisms of a cognitive architecture.

The rest of this paper is organized as follows. Section 2 introduces the concept of goal operations and formalizes the notion of goal transformation to subsume such operations. Section 3 describes the MIDCA cognitive architecture within which we have implemented such transformations. Section 4 discusses the differences in goal transformation mechanisms (i.e., at the planning level or the metacognitive level). Section 5 presents experiments and discusses the results. Related work is discussed in Section 6, and we conclude in Section 7.

Goal Operations

Work by Roberts and colleagues suggests that agents performing goal reasoning transition sets of goals through a series of modes in a *goal life cycle* (Johnson, et al. 2016; Roberts et al., 2015). To transition between two modes in the cycle, an agent executes a cognitive strategy. For example, a goal may be selected from among a set of pending goals to become the currently active goal using some selection strategy. Many of these transitions correspond to what we conceive of as goal operations as opposed to planning operations. A goal selection strategy is an example of the former; whereas an expansion strategy moving a goal from a committed to expanded mode represents a planning operation that creates a plan for the goal.

Our taxonomy of goal operations includes the following.

- *Goal formulation* – create a new pending goal
- *Goal selection* – choose an active goal from pending
- *Goal change* – change a goal to a similar one
- *Goal monitoring* – watch that a goal is still useful
- *Goal delegation* – give a goal to another agent
- *Goal achievement* – execute action to attain a goal state

Early work by Cox and Veloso (1998) argues that goals can exist in an abstraction hierarchy whereby some goals specify desired state predicates that are more general than others. The concept introduced in their work is that an important strategy for re-planning in dynamic environments is to shift goals along this hierarchy and other goal spaces. Such movement is called a *goal transformation*.

We claim that each goal operation can be thought of as a kind of transformation. In this paper we will show how both goal formulation and change are represented with a formal notation of goal transformations, and we will situate these operations within an implemented cognitive architecture.

Planning Formalism

A *classical planning domain* is defined (Ghallab, *et al.*, 2004) as a finite state-transition system in which each state $s \in S = \{s_1, \ldots, s_n\}$ is a finite set of ground atoms. A *planning operator* is a triple o = (head(o), pre(o), eff(o)), where pre(o) and eff(o) are preconditions and effects. Each *action*, $\alpha \in A$ is a ground instance of some operator o. An action is *executable* in a state s if $s \vDash$ pre(α).

For a classical planning domain, the *state-transition system* is a tuple $\Sigma = (S, A, \gamma)$, where S is the set of all states, and A is the set of all actions as above. In addition, *gamma* is a state transition function $\gamma: S \times A \rightarrow S$ that returns a resulting state of an executed action given a current state, i.e., $\gamma(s, \alpha) \rightarrow s'$.

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

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

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

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

Interpretation and Goal Transformation

Goal reasoning has recently extended the classical formulation by relaxing the assumption that the goal is always given by an external user (Cox, 2007; see also Ghallab, *et al.*, 2014). Although the planning process may start with an exogenous goal, a dynamic environment may present unexpected events with which the system must contend. In response, a goal reasoner generates new goals at execution time as situations warrant. Furthermore, goals themselves may change over time as an adaptive response to a dynamic world.

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. Here we posit a simple model of goal change $\Delta = \{\delta \mid \delta: G \rightarrow G\}$ that represents the set of potential operations or transformations on goals an agent may select. A *goal transformation* is a tuple δ = (head(δ), parameter(δ), pre(δ), res(δ)), where pre(δ) and res(δ) are δ's *preconditions* and *result.* Then given s and g, the agent makes a decision $\langle \delta^1, \delta^2, \ldots \delta^n \rangle$ that results in β output $\delta_n(\ldots \delta_2(\delta_1(g))) = g'$. As such, β represents a state interpretation process that perceives the world with respect to its goals, managing them as necessary.

Goals are dynamic and subject to change. But there exists an element of Δ, the *identity transformation* $\delta^I(g_i) = g_i$ for all $g_i \in G$, i.e., the tuple $(identity, g, \{true\}, g)$, which represents the decision not to change g given s, i.e., the agent's selection of δ^I for β. Further, goals can be created or formulated given any state, including the initial state s_0, and a goal state, including the empty state $\emptyset$. This is represented by the *goal insertion transformation* $\delta^*() = g$. This distinguished operation has been a particular focus of the goal reasoning community (see Klenk et al., 2013). Given that it relaxes the assumption that goals are necessarily provided by a human, this significantly differs from classical planning. Further as shown by the specification of β in Table 1, insertion is treated differently than others (see Cox, 2016, regarding δ^*).

Assuming an agent's goal agenda $\hat{G} = \{g_1, g_2, \ldots g_c, \ldots g_n\}$ containing the current goal g_c, Table 1 shows the details of β. The function (1) uses *choose* to select a sequence of goal operations to apply; (2) alters the goal agenda; and (3) returns a (possibly) new or changed goal. Section 4 will present a simple example of an application of these functions we implemented for the empirical results shown subsequently.

Model of Planning, Acting, and Interpretation

A plan to achieve a goal $\beta(s, \emptyset)$ can now be represented as $\pi_{\beta(s,\emptyset)}$. Using this notation, we combine plans, action (plan execution), and interpretation as in equation 3 (Cox, 2016).

$$\gamma(s_0, \pi_{\beta(s_0,\emptyset)}) = \gamma\left(\gamma(s_0, \alpha_1), \pi_{\beta(\gamma(s_0,\alpha_1),\beta(s_0,\emptyset))}[2..n]\right) \qquad (3)$$

However when goals change (or new ones are added) due to β, plans may need to change as well. Thus, the goal reasoner may need to re-plan and thereby alter the length and composition of the remainder of the plan. To cover this contingency, we define a (re)planning function *phi* that takes as input a state, goal, and current plan as in expression 4.

$$\varphi(s, g', \pi_g[1..n]) \rightarrow \pi_{g'}[1..m] \qquad (4)$$

Note that in the general case g' may or may not be equal to g. Inserting φ into equation 3 in place of π, we obtain equation 5 below. The formalism is general across different variations of goal reasoning and (re)planning.

$$\gamma(s_0, \varphi(s_0, \beta(s_0, \emptyset), \emptyset)) = \gamma(\gamma(s_0, \alpha_1), \qquad (5)$$
$$\varphi(\gamma(s_0, \alpha_1), \beta(\gamma(s_0, \alpha_1), \beta(s_0, \emptyset)), \varphi(s_0, \beta(s_0, \emptyset), \emptyset)[2..n])]])$$

Table 1. Beta and Choose. Although Δ *is an ordered set,* $\widehat{\Delta}$ *is a sequence where in is treated like the set operator* $\in$ *and "–" like set difference.* Reverse *maintains the order of* Δ (choose *inverts it).*

```
$\boldsymbol{\beta(s{:}S;\, g_c{:}\, G){:}G}$
$\widehat{\Delta} \leftarrow reverse(choose(s, g_c, \Delta))$
if $\delta^*$ in $\widehat{\Delta}$ then
        if $\widehat{\Delta} = \langle \delta^* \rangle$ then                                // insertion only
                $\hat{G} \leftarrow \{g_1, g_2, \dots g_c, \dots g_n\} \cup \delta^*()$
                $\beta \leftarrow g_c \wedge \delta^*$
        else $\langle \delta_1, \delta_2, \dots \delta_m \rangle = \widehat{\Delta} \leftarrow \widehat{\Delta} - \delta^*()$     // insertion plus others
                $\hat{G} \leftarrow \{g_1, g_2, \dots \delta_m(\dots \delta_2(\delta_1(g_c))), \dots g_n\} \cup \delta^*()$
                $\beta \leftarrow \delta_m(\dots \delta_2(\delta_1(g_c))) \wedge \delta^*()$
else  $\hat{G} \leftarrow \{g_1, g_2, \dots \delta_m(\dots \delta_2(\delta_1(g_c))), \dots g_n\}$      // no insertion
        $\beta \leftarrow \delta_m(\dots \delta_2(\delta_1(g_c)))$

$\boldsymbol{choose(s{:}\, S, g_c{:}\, G, \Delta = \{\delta_1, \delta_2, \dots\}{:}\, poset){:}\, sequence}$
if $\Delta = \{\}$ then $choose \leftarrow \langle\,\rangle$
else if $\forall x | x \in pre(\delta_1) \wedge satisfied(x)$ then
                $choose \leftarrow \delta_1 | choose(\Delta - \{\delta_1\})$
            *else choose* $(\Delta - \{\delta_1\})$
```

MIDCA: The Metacognitive Integrated Dual-Cycle Architecture

The *Metacognitive Integrated Dual-Cycle Architecture (MIDCA)* (Cox et al., 2016; Paisner, et al., 2013) is an agent architecture that has both cognitive and metacognitive capabilities. Figure 1 shows the two reasoning cycles. The cognitive cycle directly reasons about and interacts with the world through the following phases. *Perceive* receives environmental input (noise + $\psi \subset \Psi$), and *Interpret* processes the input, noting any anomalies or opportunities and generating new goals (i.e., the goal insertion transformation, $\beta(\vec{p}_j, \emptyset) \rightarrow g_n$). Interpret partially corresponds to the function β. *Evaluate* checks to see which goals the agent is pursuing are still relevant. *Intend* chooses which goal g_c the agent should be currently pursuing from the goal agenda $\hat{G}$. As such it implements a *goal selection* operation. *Plan* generates a plan π_k to achieve the agent's current goals, and it corresponds to the function φ of expression 4. Finally, the *Act* phase executes the next action in the agent's current plan, and it corresponds to the function γ of equation 2.

Much like the cognitive cycle, the metacognitive cycle (abstractly represented at the top of Figure 1) goes through cyclical phases. Computationally the primary difference between the metacognitive and cognitive cycles is the source of perception and the locus of actions. The metacognitive cycle introspectively monitors and applies executive control to the cognitive layer instead of the environment.

Instead of perceiving environmental input, a metacognitive *Monitor* phase assembles a trace representation of the cognitive phases. The trace, τ, is then processed via metacognition in a manner similar to the cognitive layer. A metacognitive *Interpret* phase introspects upon τ and notes any anomalies or opportunities introducing new meta-goals as necessary. A meta-level *Evaluate* phase updates the agent's current meta-goals, ensuring they are still applicable. A metacognitive *Intend* phase chooses a meta-goal to pursue and then generates a plan to achieve that goal. Finally, like the cognitive Act phase, the the next action in the meta-level plan will be executed by a metacognitive *Control* phase (e.g., change the goal at the cognitive level, $\Delta g = \beta(s, g) \rightarrow g'$).

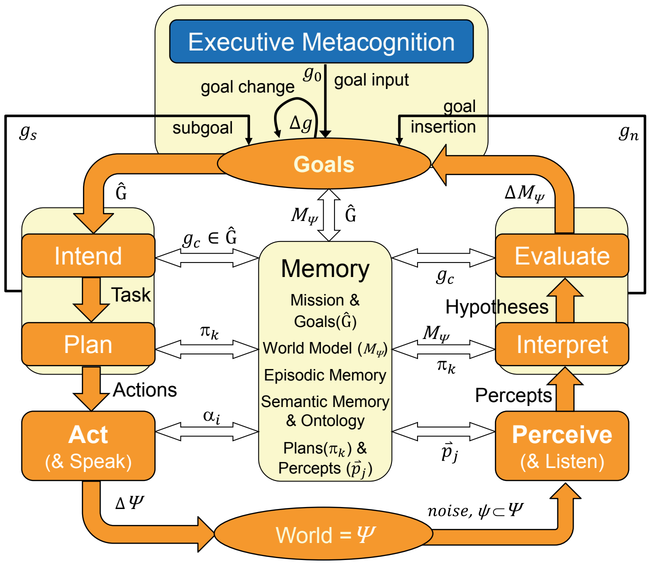

Figure 1. Schematic of the MIDCA Architecture (adapted from Paisner, et al., 2013)

Goal Transformations in MIDCA

Goal change is required in at least two cases: (i) when the agent senses a change in the environment that dictates an adjustment either during the planning process or during the execution of plan steps; and (ii) when the planner cannot solve the current problem because of a lack of known resources. In this paper, we focus on the second case. The agent is presented with a situation where resources needed to achieve an original goal become unavailable. Within the cognitive architecture there are at least two distinct places to perform goal transformation: (1) within the planning system of the Plan phase of the cognitive layer and (2) within the metacognitive reasoning component.

For example, consider the *generalization transformation*. Here, we use a specific transformation δ^{ge} that operates on goals representing predicate relations between objects of type $Objs$ (see Table 2). If goal change occurs during metacognitive reasoning, the state s comes from the cognitive trace τ; otherwise, access to s is direct.

Now assuming operations $\Delta = \{\delta^{ge}, \delta^{sp}, \dots, \delta^{t}\}$, an agent may start with the goal to achieve a stable tower of blocks, i.e., $g_1 = stableOn(A, B)$, etc. But given an unexpected lack of mortar resources (e.g., another agent may have recently used them), the agent might change the goal during execution (i.e., at action α_k in $\pi_{g_1}[1..k]$, where $k < n$) to a less durable tower (i.e., $g_2 = on(A, B)$) where on is the parent of $stableOn$ within a class hierarchy. We then represent an application of the goal generalization $\delta^{ge}(g_1) = g_2$ in the expression $\beta(\gamma(s_0, \pi_{g_1}[1..k]), g_1) \rightarrow g_2$.

Table 2. Goal generalization and its inverse, goal specialization. Bergmann (2002) provides a notation for class hiearchy $\boldsymbol{CL}$ having leaf classes $L_C \subseteq \boldsymbol{CL}$ and whose root class C has superclass $\top$, i.e., $C_{superclass} = \top$. Precondition pre_2 of both transformations assures that a parent or child class p' exists. State s in pre_2 is within the scope of choose from Table 1.

$$\boldsymbol{\delta^{ge}(g_c : G) : G}$$
$$head(\delta^{ge}) = generalization$$
$$parameter(\delta^{ge}) = g_c = p(obj1, obj2)$$
$$pre_1(\delta^{ge}) = p \in \boldsymbol{CL} \land obj1 \in Objs \land obj2 \in Objs$$
$$pre_2(\delta^{ge}) = \exists p, p' | p \in \boldsymbol{CL} \land p' \in \boldsymbol{CL} \land p_{superclass} = p' \land p = \left(p_{name}, p', (p.A_1, p.A_2, \dots p.A_m)\right) \land p' \neq \top$$
$$pre_3(\delta^{ge}) = limitedResourcesForGoal(s, g_c)$$
$$pre(\delta^{ge}) = \{pre_1(\delta^{ge}), pre_2(\delta^{ge}), pre_3(\delta^{ge})\}$$
$$res(\delta^{ge}) = p'(obj1, obj2)$$

$$\boldsymbol{\delta^{sp}(g_c : G) : G}$$
$$head(\delta^{sp}) = specialization$$
$$parameter(\delta^{sp}) = g_c = p(obj1, obj2)$$
$$pre_1(\delta^{sp}) = p \in \boldsymbol{CL} \land obj1 \in Objs \land obj2 \in Objs$$
$$pre_2(\delta^{sp}) = \exists p', p | p' \in \boldsymbol{CL} \land p \in \boldsymbol{CL} \land p'_{superclass} = p \land p' = \left(p'_{name}, p, (p'.A_1, p'.A_2, \dots p'.A_m)\right) \land p' \notin L_C$$
$$pre_3(\delta^{sp}) = surplusResourcesForGoal(s, g_c)$$
$$pre(\delta^{sp}) = \{pre_1(\delta^{sp}), pre_2(\delta^{sp}), pre_3(\delta^{sp})\}$$
$$res(\delta^{sp}) = p'(obj1, obj2)$$

Cognitive Level Goal Transformations

Goal transformations can be implemented within a *hierarchical task network (HTN)* planner's methods. HTN planning is a powerful planning paradigm because of its incorporation of domain knowledge into methods, which guide the plan generation. Assuming goal transformations of the domain are known by the domain engineer, this knowledge can be incorporated into the HTN methods. For example, an extra precondition can be added to the method that checks the state for an available resource. If the resource is available, the method will guide the search in one direction utilizing that resource. If it is unavailable, it will guide the search in another direction (one that may be less desireable but uses no resources). In this way, various goal transformations can be performed during plan generation. This approach contains all of the goal transformation knowledge within the planner. The costs of this approach are the extra knowledge burden and the encoding of it into the HTN methods. The benefit is a self-contained goal-transformation enabled planner that can perform goal transformations during plan generation. This allows a goal change to occur without any additional cognitive cycles.

Metalevel Goal Transformations

Alternatively, goal transformation can occur at the metalevel. No goal transformation knowledge needs to be encoded in the HTN methods. Instead, in the example where a resource is not available yielding the current goal as unachievable, the planner would notify the metacognitive reasoner. For example, the planner could fail to produce a plan (or log a warning) in the cognitive trace (i.e., input to metalevel; see Figure 2). The metacognitive layer would perceive the warning or failure to produce a plan and begin its reasoning mechanism. As a result, the metalevel could transform the previous goal into a new goal that did not require the resource and perform a goal retraction followed by insertion. During the next cognitive Intend phase, the cognitive layer will then choose to pursue the recently inserted goal.

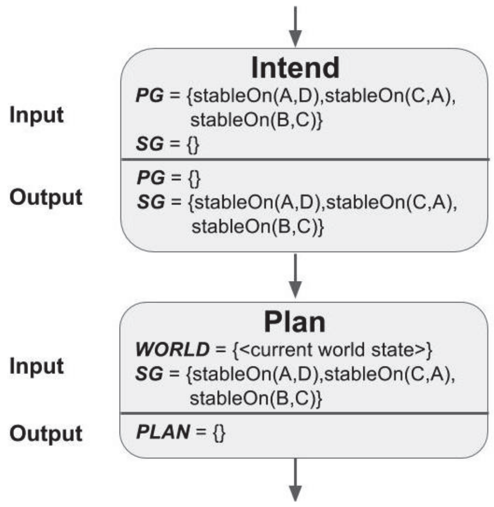
Intend
Input
PG = {stableOn(A,D),stableOn(C,A), stableOn(B,C)}
SG = {}
Output
PG = {}
SG = {stableOn(A,D),stableOn(C,A), stableOn(B,C)}
Plan
Input
WORLD = {<current world state>}
SG = {stableOn(A,D),stableOn(C,A), stableOn(B,C)}
Output
PLAN = {}

Figure 2. Trace τ of cognitive activity

Figure 2 shows the last two phases of the trace when the cognitive Plan phase fails to produce a plan for the goal to achieve a stable tower. The goal for a stable tower is encoded as $\{\ stableOn\ (A, D),\ stableOn\ (C, A),\ stableOn\ (B, C)\ \}$. Predicate *stableOn* is similar to *on* with the additional requirement that a piece of mortar is used to cement the blocks into place. In this example, only one piece of mortar is available instead of the needed three (one for each *stableOn*). The cognitive level Plan phase will attempt to plan for the stable tower goal and fail. Figure 2 shows the segment of the trace used to identify the failure of the cognitive Plan phase (and thus trigger metacognitive reasoning). In the trace (τ) segment we see the results of Intend and Plan phases (this is part of the whole trace produced from every phase executed thus far in MIDCA's execution). The trace segment for Intend records as input the *pending goals* ($PG = \hat{G}$*)* and *selected goals* ($SG = g_c$*)*. Here we see PG contains the goal atoms for the stable tower. Intend, the phase with the primary purpose of selecting which goals to commit to, moves these three goal atoms to SG as its output. At this point, everything at the cognitive level is proceeding as expected.

The next phase, Plan, is tasked with using the newly selected goals along with the current world state (WORLD ≈ *Ψ*) to produce a plan. At the current time in MIDCA's execution, the world state contains only a single piece of mortar, not enough for all three of the stableOn goal atoms. Once the Plan phase finishes and the trace output showing no plan is recorded, the metacognitive Monitor phase retrieves the trace segment and the metacognitive Interpret notes an anomaly: the cognitive level Plan phase produced no plan.

The expectation used to detect this anomaly is that the Plan phase should produce a non-empty plan that achieves the selected goals (given a mental domain model of operators for each phase, this kind of expectation would fall under the *immediate* type of expectations categorized in (Dannenhauer and Munoz-Avila 2015b). We use expert authored expectations here and leave a mental domain model and resulting expectations for future work). The metacognitive *Intend* phase also inserts a new meta-goal which is to have an achievable goal at the cognitive level.

Specifically, let the cognitive goal be $g = \{stableOn\ (A,D),\ stableOn\ (C,A),\ stableOn\ (B,C)\}$ and the meta-goal be $g^m = achieveable(g)$. Next, the metacognitive Intend phase commits to g^m. The metacognitive Plan phase produces an action plan π^m consisting of the single action $\alpha^m = transformGoal(g)$. Finally, the meta-level Control phase executes α^m (which uses the goal transformation δ^{ge}) and yields the modified goal $g' = \{on\ (A,D),\ on\ (C,A),\ stableOn\ (B,C)\}$. The goal g' is updated in MIDCA's memory and metareasoning finishes. Finally cognitive reasoning continues. The following Intend phase will select g', and the Plan phase produces a non-empty plan.

The cost of the metalevel approach is the requirement for more computation. The cognitive level Plan phase must first fail (or issue a warning) before the metalevel performs the goal change operation. Then the cognitive layer can continue, and the next Intend phase will select the new goal. The benefits of this approach are a planner-independent mechanism for goal change and a declarative representation of change that enables reasoning about such operations.

Computational Experiments

We evaluated MIDCA's goal transformation in a modified blocks world domain. The blocks world is a well known planning domain where an agent is tasked with arranging blocks on a table. The actions available to the agent are pickup from and putdown on the table, unstack, and stack. The modification we introduced is a resource called mortar that allows the agent to build sturdier towers, and thus achieve more points for those towers. Specifically, for every two blocks adjoined with mortar the agent receives an extra point. Each block in the tower is also worth one point. New operators are introduced that allow blocks to be stacked using mortar, in addition to the operators to stack without mortar. The former achieves the goal of *stableOn(A,B)*; whereas the latter achieves *on(A,B)*. The change from the fomer to the latter is an example of the generalization transformation.

There is always a finite amount of mortar. If there is not enough mortar to stack blocks, the agent will need to change the goal and resort to stacking without mortar and reduced points rewarded for that tower. This requires a transformation of the goal, because the original goal is always to build sturdy, mortar towers. We ran experiments varying the number of resources (i.e., mortar) and the number of goals. If the agent did not have enough mortar, the solid-tower goals could not be achieved, and without a transformation of the goal, the agent would gain no reward.

Empirical Results

We collected data from 260 instances of MIDCA varying resources and number of goals. Figure 3 shows the results of MIDCA using a goal transformation strategy; whereas Figure 4 shows the results with fixed, static goals (i.e., no goal change). The y-axis is the percentage of the maximum score the agent was able to achieve (i.e. the maximum score is the score received in the case all towers use mortar). By comparing the graphs, it is clear that when the number of resources is sufficient for the complexity of the problem, both graphs show equivalent performance. But when resources relative to the number of goals are scarce, MIDCA is able to achieve a higher score by changing goals. In Figure 4, one side of the graph drops to zero under resource limitations. By changing goals appropriately, MIDCA is able to achieve a higher performance score in Figure 3.

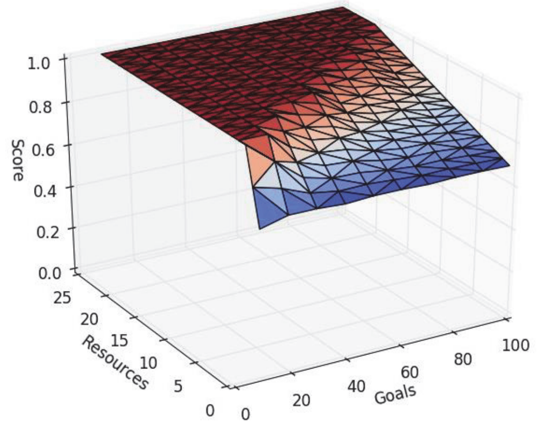

Figure 3. Performance with goal transformation

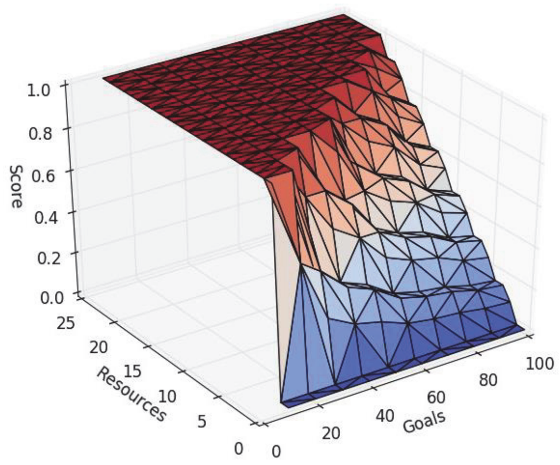

Figure 4. Performance without goal transformation

Related Work

Goal-driven autonomy (GDA) agents (a kind of goal reasoning agent) generate new goals as the agent encounters discrepancies between the agent's expectations of the outcome of its actions and the observed outcomes in the state (Aha, *et al.*, 2010; Cox, 2013; Klenk *et al.*, 2013; Dannenhauer and Munoz-Avila, 2015a). When such a discrepancy occurs, GDA agents generate an explanation for the discrepancy (e.g., a truck is not moving as expected because it is stuck in the mud), and generate a new goal accordingly (e.g., unstuck truck). Goals are not usually transformed other than retraction and insertion; new goals are generated, possibly replacing the previous goals or to aid achieving those goals. To some extent, the work here is more general; goal generation or formulation is one possible operation of the taxonomy described in the introduction; a goal reasoner may perform any of these operations.

The work here is related to the previous effort by Cox and Veloso (1998), although a number of differences exist. The goal operations implemented in their work (referred to as transformations) were for the state-space planner PRODIGY (Veloso et al., 1995), whereas our work implements them for the hierarchical planner SHOP (Nau et al., 2003) embedded within MIDCA. Thus we have shown that the concept of goal operations is more general and placed it firmly within the context of goal reasoning. Additionally, the work described here is situated within a cognitive architecture, as well as being empirically evaluated in a different problem domain (modified blocksworld versus air campaign planning), again showing significant generality.

Oversubscription planning (Smith, 2004) addresses the problem of generating plans for many goals by choosing a subset of goals when all of them cannot be achieved so as to assist classical planners which might otherwise fail. This is similar to a retraction transformation whereby the system drops particular goals. Goal operations (i.e., goal selection) include this type of planning decision and others.

The roots of our work go back to Michalski's early *inferential theory of learning* that views the process of solving a problem as a series of knowledge transmutations (Michalski, 1994), steps taken to modify a body of knowledge to attain a solution. It distinguishes between generalization and concretion transmutations. Generalization transmutations transform specific knowledge into more general knowledge. We are using similar ideas, i.e., transmutations applied to goals.

In the context of automated planning, changing the plan representation has been long explored, for example in the context of abstraction planning (Knoblock, 1990; Bergmann and Wilke, 1995). Analogous to Michalski's generalization and concretion transmutations, plans can be generalized. For example, the plan *⟨move (truck1, locA, locB), load (truck1, lockA, obj1)⟩* can be abstracted into *⟨move (?truck, locA, locB), load (?truck, lockA, obj1)⟩*. Concretion transforms the plan in the opposite direction. Such generalizations are done with the aim of facilitating problem solving. For instance, some generalizations involve not only replacing constants for variables but eliminating some of the conditions in the problem. A key point in abstraction planning is the distinction between generalization and abstraction; in the former the planning language remains the same (e.g., when replacing constant with variables, the actions remain the same). In contrast, in abstraction the language itself changes. For example, the plan *⟨move (truck1, locA, locB), load (truck1, lockA, obj1)⟩* can be replaced by the construct *moveANDload (truck1, locA, locB, obj1)*; *moveANDload* is not a name appearing in the concrete domain but a construct to reason at higher levels. Generalization and abstraction might make the problem easier (i.e., by removing hard constraints; such as requiring a special kind of vehicle to transport some specific goods). When a plan is solved in the generalized or abstracted form, it is transformed into a concrete solution. Nevertheless whether abstracted or generalized, goals generated still refer to the planning domain. In contrast, transmuted goals at the meta-level refer to the process of generating the plan rather than the planning domain itself in our work.

Akin to plan abstraction, work on learning HTNs aims at generating hierarchies that subsume plan traces; although the aim of HTN learning algorithms is to learn the abstractions. Typically, these algorithms use as input some additional knowledge such as task semantics defined as precondition-effects pairs (Hogg, Munoz and Kuter, 2008) or Horn clauses (Nejati, Langley and Konig, 2006) to find sound ways to abstract the plan traces. The generated hierarchies represent knowledge about the domain unlike in our work where the transformed meta-level goals reflect knowledge about the process not the domain itself.

Conclusion

The idea of goal change is a fundamental concept for intelligent systems; people change their mind all the time, and for good reason. A system that reasons about its goals and its ability to achieve them will sometimes have to adapt to changing information and changing environments if it is to act rationally. Here we have argued that adaptation is not always a mattter of plan change, rather sometimes an agent will change its goals instead. But deciding whether to change the plan or the goal is itself a hard decision not addressed here. Instead we have presented a model of goal transformations and introduced the concept within the MIDCA cognitive architecture, showing the effects on performance. A more complete algorithm to select a given choice or ones to implement specific transformations are left to future research. However, goal operations need to be used conservatively and with caution. Otherwise in all instances, the substitution of the current goal set with the empty set by a series of retraction transformations can be satisfied by the null plan, an unsatisfactory proposition.

Acknowledgments

This research was supported by ONR under grants N00014-15-1-2080 and N00014-15-C-0077 and by NSF under grant 1217888. We thank Hector Munoz-Avila and the anonymous reviewers for their comments and suggestions.

References

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

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

Bergmann, R. eds. 2002. *Experience Management: Foundations, Development Methodology, and Internet-based Applications*. Berlin: Springer.

Bergmann, R.; and Wilke, W. 1995. Building and Refining Abstract Planning Cases by Change of Representation Language. *JAIR* 3: 53-118.

Cox, M. T. 2016. A Model of Planning, Action, and Interpretation with Goal Reasoning. In *Proceedings of the fourth Annual Conf. on Advances in Cognitive Systems,* 48-63. Palo Alto, CA: Cognitive Systems Foundation.

Cox, M. T. 2013. Goal-Driven Autonomy and Question-Based Problem Recognition. In *Proceedings of the second Annual Conference on Advances in Cognitive Systems*, 29-45. Palo Alto, CA: Cognitive Systems Foundation.

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

Cox, M. T.; Alavi, Z.; Dannenhauer, D.; Eyorokon, V.; Munoz-Avila, H.; and Perlis, D. 2016. MIDCA: A Metacognitive, Integrated Dual-Cycle Architecture for Self-Regulated Autonomy. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, Vol. 5, 3712-3718. Palo Alto, CA: AAAI Press.

Cox, M. T.; and Veloso, M. M. 1998. Goal Transformations in Continuous Planning. In *Proceedings of the 1998 AAAI fall symposium on distributed continual planning*, 23-30. Menlo Park, CA.: AAAI Press.

Dannenhauer, D.; and Munoz-Avila, H. 2015a. Goal-Driven Autonomy with Semantically-annotated Hierarchical Cases. *In Proceedings of the International. Conference on Case-Based Reasoning*, 88-103. Berlin: Springer.

Dannenhauer, D.; and Munoz-Avila, H. 2015b. Raising Expectations in GDA Agents Acting in Dynamic Environments. *In Proceedings of the International Joint Conference on Artificial Intelligence*. Palo Alto, CA: AAAI Press.

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

Hogg, C.; Munoz-Avila, H.; and Kuter, U. 2008. HTN-MAKER: Learning HTNs with Minimal Additional Knowledge Engineering Required. In *Proceedings of the Twenty third AAAI Conference on Artificial Intelligence*, 950-956. Menlo Park, CA: AAAI Press.

Johnson, B.; Roberts, M.; Apker, T.; & Aha, D.W. 2016. Goal reasoning with information measures. In *Proceedings of the Fourth Conference on Advances in Cognitive Systems*. Evanston, IL: Cognitive Systems Foundation.

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

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

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

Knoblock, C. A. 1990. Learning Abstraction Hierarchies for Problem Solving. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 923-928. Menlo Park, CA: AAAI Press.

Michalski, R. S. 1994. Inferential Theory of Learning: Developing Foundations for Multistrategy Learning. *Machine Learning: A Multistrategy Approach* IV, 3-61. San Francisco: Morgan Kaufmann.

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

Nejati, N.; Langley, P.; and Konik, T. 2006. Learning Hierarchical Task Networks by Observation. In *Proceedings of the Twenty Third International Conference on Machine Learning*, 665-672. NY: ACM.

Paisner, M.; Maynord, M.; Cox, M. T.; and Perlis, D. 2013. Goal-Driven Autonomy in Dynamic Environments. In D. W. Aha, M. T. Cox, & H. Munoz-Avila (Eds.), *Goal Reasoning: Papers from the ACS Workshop*, 79-94. Tech. Rep. No. CS-TR-5029, Department of Computer Science, University of Maryland, College Park, MD.

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, & Andrea Orlandini(Eds.) Planning and Robotics: Papers from the ICAPS Workshop*. Palo Alto, CA: AAAI Press.

Smith, D.E. 2004. Choosing Objectives in Over-Subscription Planning. In *Proceedings of the ICAPS 4:*393. Menlo Park, CA: AAAI Press.

Vattam, S.; Klenk, M.; Molineaux, M.; and Aha, D.W. 2013. Breadth of Approaches to Goal Reasoning: A Research Survey. In D. W. Aha, M. T. Cox, & H. Munoz-Avila (Eds.), *Goal Reasoning: Papers from the ACS Workshop*, 111-126. Tech. Rep. No. CS-TR-5029, Department of Computer Science, University of Maryland, College Park, MD.

Veloso, M.; Carbonell, J.; Perez, A.; Borrajo, D.; Fink, E.; and Blythe, J. 1995. Integrating Planning and Learning: The PRODIGY architecture. *JETAI* 7(1): 81-120.