Goal Transformation and Goal Reasoning

Michael T. Cox
Wright State Research Institute, Beavercreek, OH
michael.cox@wright.edu

Dustin Dannenhauer
Lehigh University, Bethlehem, PA
dtd212@lehigh.edu

Abstract

Agents operating in complex and dynamic domains may observe changes that affect the agent's ability to achieve its goals. Goal transformations allow unachievable goals to be converted into similar achievable goals. Previous work has examined transformation of goals within the state-spaced planner PRODIGY. This paper examines goal transformation within the MIDCA architecture. We introduce goal transformation at the metacognitive level as well as goal transformation in a Hierarchical Task Network planner and discuss the costs and benefits of each approach. We evaluate goal transformations in MIDCA using a modified, resource limited version of the classical blocksworld planning domain, demonstrating the benefit of achieving higher scoring goals due to goal transformations.

1 Introduction

Effective performance in highly dynamic environments requires the discharge of many classical-planning assumptions held in the artificial intelligence commmunity. For example, the closed world assumption is not a practical strategy. The world is under continual change, and planning is often a matter of adjusting to the world as new information is discovered, whether during planning or during execution. However, the adjustment that planners classically perform given exogenous events entails change with regard to the knowledge concerning the current state of the world and, in response, adaptation of the current plan. During execution of plans, outcomes may diverge from expectations, so plans are again adjusted accordingly (see as far back as [Tate, *et al*., 1990]). The contention of this paper, however, is that the adjustment of the goals is often required in addition to adjustment of the plans themselves.

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. Here we examine the idea that goals also represent malleable knowledge structures that an agent can adapt and change as the situation warrants; they are not static.

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 planner 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 show how such a procedure can be incorporated into various mechanisms of a cognitive architecture.

The rest of this paper is organized as follows. Section 2 introduces the goal transformation formalism, and 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.

2 Goal Transformations

Early work by Cox and Veloso [1998] argue 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*.

The goal arguments and predicates may be moved along an abstraction hierarchy, an enumerated set, a number line, or a component partonomy. For example, *concretion* and *specialization* are downward movements through an abstraction hierarchy on either goal arguments or predicates respectively; *abstraction* and *generalization* are their inverses. *Escalation* and *erosion* move the goal up or down enumerated (or numerical) ordered sets of argument values. *Insertion* and *retraction* either adds or deletes a goal from the current set of states the planner must achieve. *Substitution* replaces one goal with an equivalent (either logical, e.g., DeMorgans Law, or semantic). The *identity* transformation makes no change.

A number of such goal changes are inherent in the classical planning process. For example, subgoaling on the preconditions of a planning operator can be considered a kind of goal insertion transformation. Also, when a rationale-based sensing monitor suggests a plan-based cut [Alavi 2016], this is equivalent to a goal retraction.

2.1 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 $s \models \text{pre}(\alpha)$.

For a classical planning domain, the *state-transition system* is a tuple $\Sigma = (S, A, \gamma)$, [1] 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 $s_g \models g$. 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 $\pi_g[2..n]$.

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

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 $\gamma(s_0, \pi_g) \models g$. Recursively from the initial state, execution of the plan results in the goal state (see equation 2).

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

[1] We ignore the set of exogenous events E that are outside the control (and possibly the observation) of the reasoning system.

2.2 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 must be able to generate 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 [Cox 2016]. 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, changing them or formulating new ones 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. Furthermore as shown by the specification of β in Table 1, insertion is treated differently than others (see [Cox, 2016] for further details 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 result shown subsequently.

2.3 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].

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

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.

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

Table 1. The *beta* and *choose* functions. Note that although Δ is an ordered set, $\widehat{\Delta}$ is a sequence where *in* is treated similar to the set operator $\in$ and "–" like set difference. The function *reverse* maintains the order of Δ since *choose* will change it.

$\beta(s{:}S; g_c{:}G){:}G$
$\widehat{\Delta} \leftarrow reverse(choose(s, g_c, \Delta))$
if δ^* in $\widehat{\Delta}$ then
  if $\widehat{\Delta} = \langle \delta^* \rangle$ then
    $\hat{G} \leftarrow \{g_1, g_2, \ldots g_c, \ldots g_n\} \cup \delta^*()$
    $\beta \leftarrow g_c \wedge \delta^*$
  else $\langle \delta_1, \delta_2, \ldots \delta_m \rangle = \widehat{\Delta} \leftarrow \widehat{\Delta}$ - $\delta^*()$
    $\hat{G} \leftarrow \{g_1, g_2, \ldots \delta_m(\ldots \delta_2(\delta_1(g_c))), \ldots g_n\} \cup \delta^*()$
    $\beta \leftarrow \delta_m(\ldots \delta_2(\delta_1(g_c))) \wedge \delta^*()$
else $\hat{G} \leftarrow \{g_1, g_2, \ldots \delta_m(\ldots \delta_2(\delta_1(g_c))), \ldots g_n\}$
  for $\langle \delta_1, \delta_2, \ldots \delta_m \rangle = \widehat{\Delta}$
  $\beta \leftarrow \delta_m(\ldots \delta_2(\delta_1(g_c)))$

$choose(s{:}S, g_c{:}G, \Delta = \{\delta_1, \delta_2, \ldots\}{:}poset){:}sequence$
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 $choose\ (\Delta - \{\delta_1\})$

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.

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

3 MIDCA

The Metacognitive Integrated Dual-Cycle Architecture (MIDCA) [Cox et al., 2016; Paisner et al., 2013] is an architecture for an agent that has both cognitive and metacognitive capabilities. Figure 1 shows the two reasoning cycles. The cognitive cycle directly reasons about and interacts with the environment; it consists of the following phases: Perceive, Interpret, Evaluate, Intend, Plan, Act. Perceive receives input from the environment. Interpret processes the input, noting any anomalies or opportunities and may generate new goals (i.e., the goal insertion transformation, $\beta(\vec{p}_j, \emptyset) \rightarrow g_n$, shown in Figure 1). 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 $\hat{G}$. Plan generates a plan π_k to achieve the agent's current goals; it corresponds to the function φ of expression 4. Finally, the Act phase executes the next action in the agent's current plan; it corresponds to the function γ of equation 2.

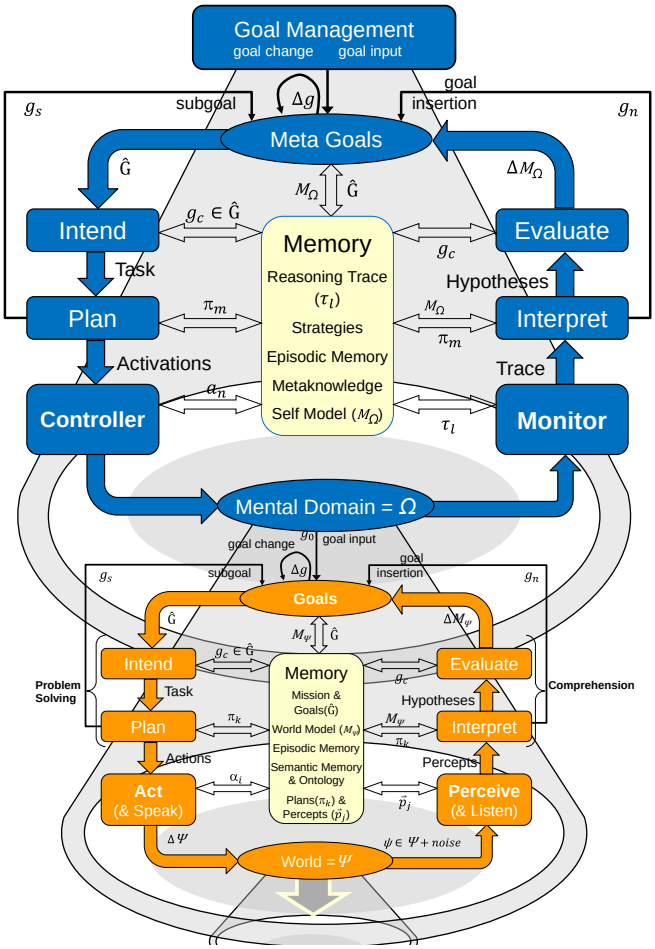

Figure 1. MIDCA Architecture (adapted from [Cox et al., 2016])

Much like the cognitive cycle, the metacognitive cycle (abstractly represented at the top of Figure 1) goes through multiple phases. The primary difference between the metacognitive and cognitive cycles is the source of perception and reception of actions. Metacognitive actions are applied to the cognitive layer instead of the environment. Regarding input, instead of receiving input from the environment, like the cognitive Perceive phase, the metacognitive layer receives input (i.e., in the form of a cognitive trace: data reported from the cognitive phases, see Figure 2) from the cognitive layer. The trace is then processed via phases during metacognition, in a similar fashion to the cognitive layer. From the trace, any anomalies or opportunities will be noted, and new goals introduced. Updates are performed on the agent's metacognitive goals, ensuring they are still applicable. A metacognitive Intend phase chooses a metacognitive 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 the metacognitive Control phase

(e.g., change the goal at the cognitive level, $\Delta g = \beta(s, g) \rightarrow g'$).

4 Goal Transformations in MIDCA

Goal transformations are 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; (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.

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

Table 2. Definition of *goal generalization* and its inverse, *goal specialization*. Bergmann [2002] provides the notation for the class hiearchy $\boldsymbol{CL}$ having leaf classes $L_C \subseteq \boldsymbol{CL}$ and whose root class C has the 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 outside the scope of both operations and instead within the scope of *choose* from Table 1.

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

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

Now assuming operations $\Delta = \{\delta^{ge}, \delta^{sp}, \delta^{I}\}$, an agent may start with the goal to achieve a stable tower of blocks, i.e., $g_1 = stable\text{–}on(A, B)$, 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, that is, $g_2 = on(A,\ B)$, where on is the parent of *stable-on* within a class hierarchy. We then represent an application of the goal generalization $\delta^{ge}(g_1) = g_2$ in the expression $\beta\big(\gamma(s_0, \pi_{g_1}[1..k]), g_1\big) \rightarrow g_2$.

4.1 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 transformation to occur without any additional cognitive cycles.

4.2 Metacognitive Goal Transformations

Alternatively, goal transformation can occur at the metacognitive level. 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 (input to metacognitive layer; 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 metacognitive layer could decide to transform the previous goal to a new goal that did not require the resource, and perform a goal retraction followed by insertion. Then, during the next cognitive Intend phase, the cognitive layer will choose to pursue the recently inserted goal.

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 {*stable-on(A,D), stable-on(C,A), stable-on(B,C)*}. *Stable-on* is a predicate similar to *on* with the additional requirement that a piece of mortar is used to cement the blocks into place. In this example, only 1 piece of mortar is available instead of the needed 3 (one for each *stable-on*). 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)* and *selected goals (SG)* ($\hat{G} = PG \cup SG$). Here we see pending goals 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 in time everything at the cognitive level is proceeding as expected.

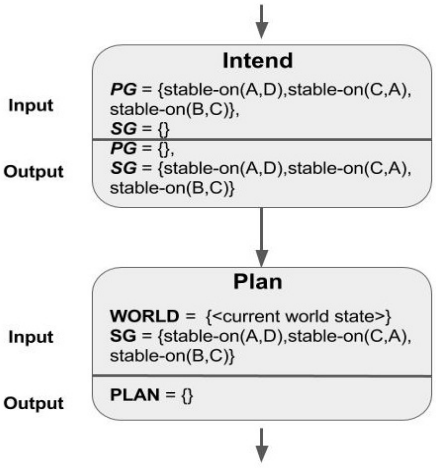

Figure 2. Trace τ of cognitive activity

The next phase, Plan, is tasked with using the newly selected goals (SG) along with the current world state (WORLD) to produce a plan. At this point in MIDCA's execution, the world state contains only a single piece of mortar, not enough for all three of the stable-on 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, cg_1 = {*stable-on(A,D), stable-on(C,A), stable-on(B,C)*} and the meta goal be *achieveable*(cg_1). Next, the metacognitive Intend phase commits to achieving *achieveable*(cg_1). The metacognitive Plan phase produces a single action plan consisting of the action *transform-goal*(cg_1). Finally, the meta Control phase performs the action *transform-goal*(cg_1) (which uses a *generalization* transformation) and yields the new transformed goal cg_1^T= {*on(A,D), on(C,A), stable-on(B,C)*}. cg_1^T is then updated in MIDCA's memory and metareasoning finishes and cognitive reasoning continues. On the following cognitive Intend phase the goal cg_1^T will be selected and the cognitive Plan produces a non-empty plan.

The cost of the metacognitive approach is the requirement for more cognitive cycles: first the cognitive level Plan phase must fail (or issue a warning in the cognitive trace) before the metacognitive layer performs a transformation and inserts a new goal. Then the cognitive layer can continue and in the next Intend phase the new goal will be selected. The benefits of this approach are a planner-independent mechanism for goal transformation, and explicit transformation of goals.

5 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 *stable-on(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.

5.1 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.

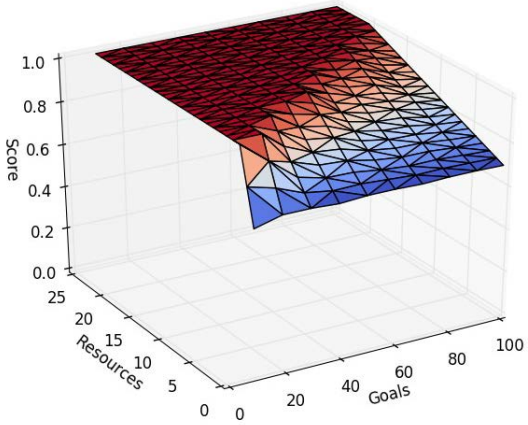

Figure 3. Performance with goal transformation

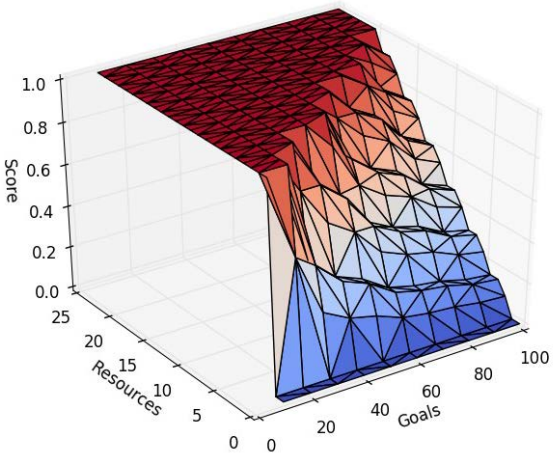

Figure 4. Performance without goal transformation

The volumetric graphs illustrate additional trends when sliced by orthogonal planes. Figure 5 shows the performance score when the number of goals is held constant at 100 (g=100 defines a plane). It directly compares performance with goal transformations and without.

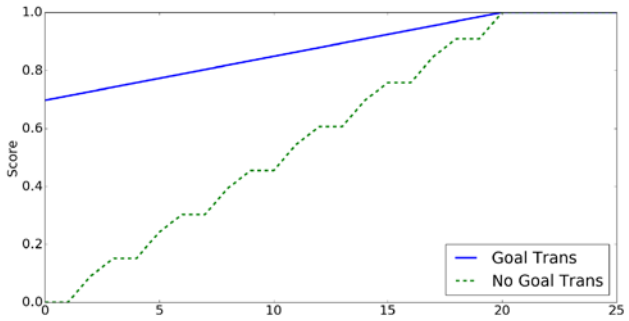

Figure 5. Performance as a function of resource availability given 100 goals

Likewise, Figure 6 shows the performance when the number of mortar resources is held at five and compares both with goal transformations and without. As problems become ever more complex with larger numbers of goals, performance degrades more without goal transformations.

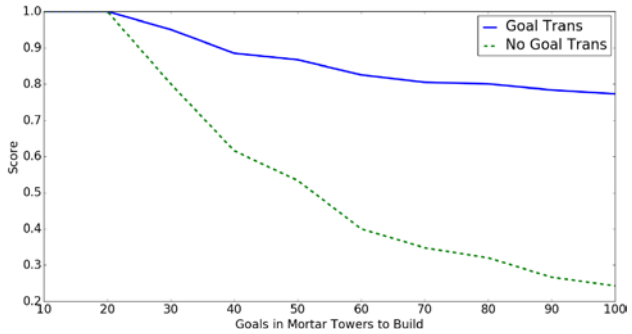

Figure 6. Performance as a function of problem complexity (i.e., number of goals) given 5 resources

6 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; 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 the 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 transformation (i.e., goal insertion) out of many changes a goal reasoner may select.

The work here is an extension of the previous effort by Cox and Veloso [1998], although a number of differences exist. The goal transformations implemented previously were for the state-space planner PRODIGY [Veloso et al., 1995], whereas the current research implents them for the hierarchical planner SHOP [Nau et al., 2003] embedded within MIDCA. Thus we have shown that the concpet is more general than either and placed it firmly within the context of goal reasoning. Additionally, the work described here is situated within a cognitive architecture, and although we explicitly chose to demostrate the same empirical trend, we chose 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 transformations 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, 1993], 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. For example, if an agent observes truck(a) and color(a,blue) and in the absence of other facts about other trucks it might hypothesize that (F1) for all ?x truck(?x) ⇒ color(?x, blue) (question marks represent variables).

Concretion is the opposite process whereby general knowledge is made more specific. For example, assuming F1 is true and if the agent observes truck(a), it will infer color(a,blue). In our work, we are using similar ideas, that of transmutations applied to desired goals.

In the context of 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 a plan trace 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 trace 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, in our work, transmuted goals at the meta-level refer to the process of generating the plan rather than the planning domain itself.

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 (preconditions,effects) pairs as in Hogg [2008] or Horn clauses as in Nejati [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.

7 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 it 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 transformations 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.

Acknowledgements

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

References

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

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

[Alavi 2016] Alavi, Z., and Cox, M. T. (in press). Rationale-based visual planning monitors. To appear in M. Roberts (Ed.), Working Notes of the 4th Workshop on Goal Reasoning. New York, IJCAI-16.

[Bergmann 2002] Bergmann, R. *Experience managemen*t. Berlin: Springer, 2002.

[Bergmann and Wilke 1995] Bergmann, Ralph, and Wolfgang Wilke.Building and refining abstract planning cases by change of representation language. *JAIR* 3 (1995): 53-118.

[Cox 2016] Cox, M. T. in press, A model of planning, action and interpretation with goal reasoning. To appear in *Fourth Annual Conference on Advances in Cognitive*

Systems 2016. Palo Alto, CA: Cognitive Systems Foundation.

[Cox 2007] Cox, M. T. 2007, Perpetual self-aware cognitive agents. *AI Magazine* 28(1), 32-45.

[Cox et al., 2016] 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 30th AAAI Conference on Artificial Intelligence, Vol. 5* (pp. 3712-3718). Palo Alto, CA: AAAI Press.

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

[Dannenhauer and Munoz-Avila, 2015a] Dustin Dannenhauer and Hector Muñoz-Avila. Goal-Driven Autonomy with Semantically-annotated Hierarchical Cases. *International Conference on Case-based Reasoning (ICCBR-15).* Springer. 2015.

[Dannenhauer and Munoz-Avia, 2015b] Dannenhauer, Dustin and Hector Munoz-Avila. Raising expectations in GDA agents acting in dynamic environments. *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI-15).* AAAI Press, 2015.

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

[Hogg, et al., 2008] Hogg, C.; Munoz-Avila, H.; Kutur, U. HTN-MAKER: Learning HTNs with minimal additional knowledge engineering required. In Proceedings of AAAI 2008.

[Ghallab, et al., 2014] 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, et al., 2004] Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory & Practice*. San Francisco: Elsevier.

[Klenk et al., 2013] 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, 1990] Craig A Knoblock. Learning abstraction hierarchies for problem solving. In *AAAI-90*, pages 923–928, 1990.

[Michalski, 1993] Ryszard S. Michalski. *Inferential theory of learning: Developing foundations*. 1993.

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

[Nejati, N., et al., 2006] Nejati, N.; Langley, P.; and Konik, T. 2006. Learning hierarchical task networks by observation. In Proc. of ICML'06, 665–672. New York, NY, USA: ACM.

[Paisner et al., 2013] Paisner, M., Maynord, M., Cox, M. T., & Perlis, D. (2013). Goal-driven autonomy in dynamic environments. In D. W. Aha, M. T. Cox, & H. Munoz-Avila (Eds.), *Goal Reasoning: Papers from the ACS workshop* (pp. 79-94). Tech. Rep. No. CS-TR-5029. College Park, MD: University of Maryland, Department of Computer Science.

[Smith, 2004] Smith, David E. "Choosing Objectives in Over-Subscription Planning."*ICAPS*. Vol. 4. 2004

[Tate, *et al*., 1990] Tate A., Hendler, J, and Drummond, M. 1990. A Review of AI Planning Techniques. In J. Allen, J. Hendler, and A. Tate, eds., Readings in Planning, 26-49. San Fran- cisco: Morgan Kaufmann

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

[Veloso et al., 1995] Veloso, M. M., Carbonell, J., and et al., Integrating Planning and Learning: The PRODIGY Architecture, *Journal of Experimental and Theoretical AI 7*, 81–120, 1995.