Toward a formal model of planning, action, and interpretation with goal reasoning

Michael T. Cox MICHAEL.COX@WRIGHT.EDU
Wright State Research Institute, Wright State University, Dayton, OH 45435 USA

Abstract

Many algorithms have been presented in artificial intelligence for problem solving and planning. Given a goal, these algorithms search for solutions that achieve a goal state by actions or interactions with an environment. However a major assumption is that goals are given, usually by a user directly as input or as part of the problem definition. Furthermore, once given, the goals do not change. Here we formalize the notion that goal specification and goal change are themselves major parts of the problem-solving process. We apply this model to learning in a goal reasoning context.

1. Introduction

In virtually all AI systems, goal states are predefined and exogenously provided by an external user. But to have a continuing existence in time, agents must be flexible to survive and to continue to be useful. Recent work on goal reasoning (Aha, Cox, & Munoz-Avila, 2013) has started to examine how intelligent agents can reason about and generate their own goals instead of always depending upon a human user directly. Much of this previous work has been situated within the context of the automated planning community and has borrowed some of their formal notations as a theoretical framework. This paper will further extend the standard planning formalism to account for mechanisms of goal-reasoning and goal-driven autonomy. The result integrates notations for planning, action, and interpretation within the scope of goal reasoning.

As an example, consider a package delivery domain (e.g., see Figure 1) having the following characteristics. There is a network of locations connected by roads; and from time to time, vehicles may need to transport objects from one location to another as requests come in. Deliveries may be accomplished, for example, by picking up and delivering packages (e.g., in Figure 1, delivering pallets of bottles to a cola bottling plant). AI planning research has considered similar but simplified domains in which the world is static (i.e., no states change unless the agent performs an action), there is a given fixed goal to achieve (e.g., the delivery of certain specified packages), and the planning problem ends at any world state in which

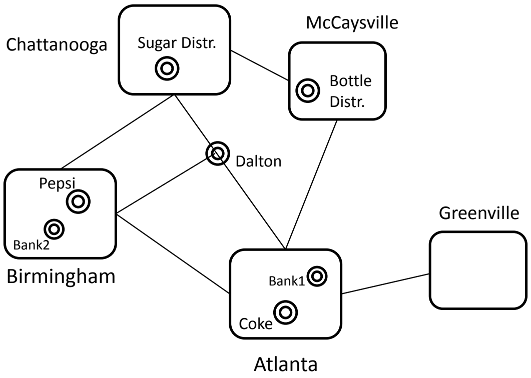

Figure 1. Package delivery domain

the goals are satisfied. In some sense, routine delivery of packages on a route is hardly a problem requiring intelligence. These problems omit any consideration of why those goals need to be achieved and what the agent should do after they have been achieved. In a more realistic model, bottles need to be delivered to the bottling plant when the inventory is low, and it might be necessary for an agent to explain and hence understand this and to generate a new goal when inventory is unexpectedly low.

We begin this paper with a section that provides our formal representations for goal reasoning and shows how it frames the research using a simple blocks world example. The subsequent section examines the formalism within a kind of goal reasoning called goal-driven autonomy. Then we look at learning within this context and apply it to the logistics example above. This section is followed by related research. A brief summary concludes the article.

2. Formal Representations for Goal Reasoning

Much of the research in AI planning has focused on a restricted case called *classical planning*, in which all actions have deterministic effects, and the task is to generate a plan that reaches any of a predefined set of goal states.

2.1 Classical Planning Theory

A *classical planning domain* is typically defined (Ghallab, Nau, & Traverso, 2004) as a finite state-transition system in which each state $s \in S = \{s1, \dots, sn\}$ is a finite set of ground atoms of a function-free, first-order language $\mathcal{L}$.[1] A *planning operator* is a triple o = (head(o), pre(o), eff(o)), where pre(o) and eff(o) (o's *preconditions* and *effects*, respectively) are sets of *literals* (logical atoms and negated logical atoms), and head(o) is a syntactic term of the form *name*(*args*), where *name* is the operator's name and *args* is a list of the variables in pre(o) and eff(o). Each *action* is a ground instance of a planning operator.

An action $\alpha \in A$ is *executable* in a state s if $s \vDash$ pre(α), in which case the resulting state is (s − eff$^{-}(\alpha)$) ∪ *eff*$^{+}(\alpha)$, where eff$^{+}(\alpha)$ and *eff*$^{-}(\alpha)$ are the atoms and negated atoms, respectively, in eff(α). A plan $\pi = \langle \alpha_1, \dots, \alpha_n \rangle$ is executable in s if each α_i is executable in the state produced by α_{i-1}.

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 the resulting state of an executable action given a current state. Thus from any given state and action, one can infer the subsequent state $\gamma(s, \alpha) \rightarrow s'$ that follows after the action is executed.

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

[1] Most classical planning implementations use the language PDDL (Fox & Long, 2003) based loosely on function-free first order languages.

$$\pi_g[1..n] = \alpha_1 \mid \pi_g[2..n] = \langle \alpha_1 \alpha_2 \dots \alpha_n \rangle \tag{1}$$

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).

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

2.2 Planning with Nondeterminism

In the research literature on planning, uncertainty about the possible outcomes of actions has been dealt with mainly in two different ways: using Markov Decision Processes (MDPs) (Kaelbling, Littman, & Cassandra, 1995; Dean, Kaelbling, Kirman, & Nicholson, 1995; Boutilier, Dean, & Hanks, 1999) and using model-checking techniques (Cimatti, Roveri, & Traverso, 1998; Aiello *et al.*, 2001; Bertoli, Cimatti, and Traverso, 2006) like those used for program verification. Which approach is best depends on the situation: MDPs are useful when the transition probabilities are important (e.g., when it is desired to maximize expected utility), and model-checking is useful when the transition probabilities are unknown or unimportant (e.g., if it is necessary to achieve a goal regardless of which nondeterministic outcome occurs). In both approaches, an action with multiple possible outcomes is often (though not always) represented as a *nondeterministic classical operator* o = (head(o), pre(o), eff$_1$(o), eff$_2$(o), …,eff$_k$(o)) that has multiple possible sets of effects. When this representation is used with MDPs, each set of effects eff$_i$(o) will have a probability $p_i(o)$, where $p_1 + p_2 + \dots + p_k = 1$. Instead of a sequential plan π that achieves a goal, MDPs learn a policy π (i.e., a mapping from states to actions) that maximizes expected utility. In this paper we will use the classical definition for clear illustration of the extension to interpretation and goal reasoning.[2]

2.3 Interpretation and Goal Reasoning

Broadly construed, the topic of *goal reasoning* concerns cognitive systems that can self-manage their goals (Vattam, Klenk, Molineaux, & Aha, 2013). Goal reasoning has recently extended the classical formulation by relaxing the assumption that the goal is always given by an external user (Cox, 2007; see also Ghallab, Nau & Traverso, 2014). Although the planning process may start with an exogenous goal, a dynamic environment may present unexpected events with which the system must contend. In response a goal reasoner must be able to generate new goals at execution time as situations warrant.

Formally, the function *beta* (see 3a) returns a (possibly) new goal formula g' given some state s and a current goal g. As such, beta is a state interpretation process that perceives the world with respect to its goals. It is central to goal formulation and goal management.

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

2.3.1 Goal transformation

Unlike classical planning models that assume goals to be static and given externally, the goal reasoning model views goals as malleable and subject to change (Cox & Zhang, 2007). For example

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

a chess player may start out with the goal to achieve checkmate (g_{ch}). But given a series of unsuccessful opening moves (i.e., $\pi_{g_{ch}}[1..k]$ where $k < n$), the player may change the goal to draw (g_{dr}). We represent this goal transformation as a specialization of (3a) shown in (3b).

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

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

2.3.2 Goal formulation

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

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

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

$$\beta\big(s_g, \emptyset\big) \rightarrow g' \qquad \text{(3d)}$$

Of course, and as we will see, goals can potentially be formulated from any current state.

2.4 A Model of Plans, Actions, and Interpretation

A plan to achieve a goal $\beta(s, \emptyset)$ can now be written as $\pi_{\beta(s_0,\emptyset)}$. Using this notation, we combine planning, action (plan execution), and interpretation in equation (4).

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

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

The problem with the current formalization then is that, in the recursive right-hand side of (4) above, the plan is not static as defined in (1). That is, it is not necessarily of size $n - 1$. Instead, because the goal may change due to beta, the goal reasoner may need to re-plan and alter the length and composition of the remainder of the plan.[3] To cover this contingency, we define a (re)planning function *phi* that takes as input a state, goal, and current plan as in (5).

[3] I thank Don Perlis for pointing out this anomaly.

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

Note that in the general case g' may or may not be equal to g. Inserting the re-planning function into (4), we obtain equation (6) below that resolves the anomaly indicated above.

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

Given phi, the formalism is general across different variations of goal reasoning and (re)planning.

$$\gamma\left(s_0,\ \overbrace{\varphi(s_0, \underbrace{\beta(s_0,\emptyset)}_{g_1}, \emptyset)}^{\pi_{g_1}}\right) = \gamma\left(\underbrace{\gamma(s_0,\alpha_1)}_{s_1},\ \overbrace{\varphi\left(\underbrace{\gamma(s_0,\alpha_1)}_{s_1}, \underbrace{\beta\left(\underbrace{\gamma(s_0,\alpha_1)}_{s_1}, \underbrace{\beta(s_0,\emptyset)}_{g_1}\right)}_{g_2}, \underbrace{\underbrace{\varphi(s_0,\beta(s_0,\emptyset),\emptyset)}_{\pi_{g_1}}[2..n]}_{\pi_{g_1}[2..n]}\right)}^{\pi_{g_2}}\right)$$

2.5 A Blocks World Example

Many details are buried within the beta interpretation function. Just consider a situation where goals are given to the agent by a human user in natural language. In this case, beta does not just input a goal in first-order predicate form. Instead it must translate an utterance or the results of a speech act into a predicate representation. That it, it must infer the intent of the user from the current state of the world and what was said. Indeed when a user both states imperatives and asks questions, the illocutionary act intended is often a request for the agent to assume a goal.[4] Consider the utterance "Put block-B on block-C." Although spoken as an action, the intent is actually for the agent to assume the goal on the behalf of the speaker to achieve the state of the block B on top of C. Although we will not discuss the specific details here, Figure 2 shows how the formalism organizes the context of a dialog between human and agent so that intent of the last utterance "Add one more" is possible.

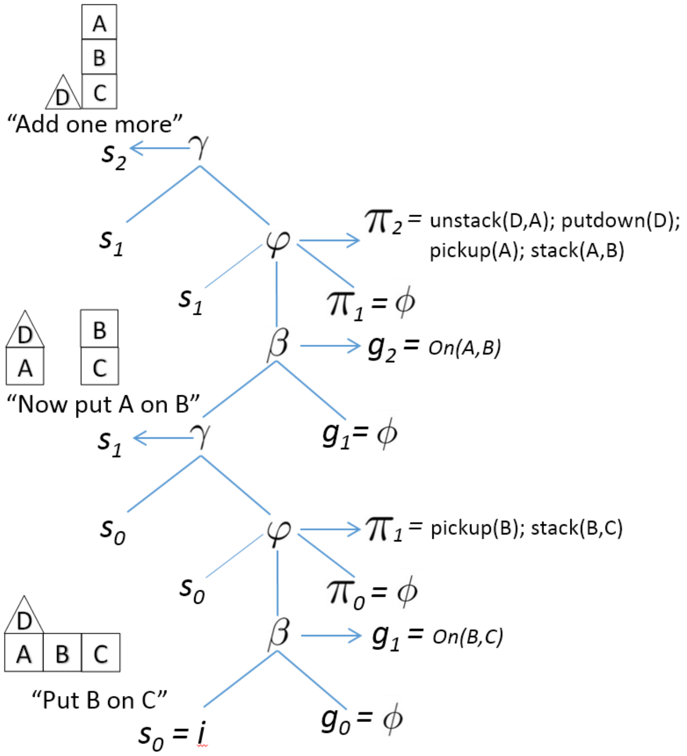

Figure 2. Partial context for blocks example (read from the bottom up)

[4] For questions, the intended goal is a *knowledge goal* (Bengfort & Cox, in press; Ram, 1990).

Figure 2 hides many details as well. Plan π_1 has two actions that are executed sequentially, not concurrently in one invocation of gamma as the figure suggests. A more realistic depiction of the plan execution is shown in Figure 3. Here we see that the execution of the first action results in an intermediate state. From that state no new goal is formulated, and subsequently no new re-planning is performed. The goal is achieved in the next state s_2 and beta evaluates the goal achievement, returning the null goal.

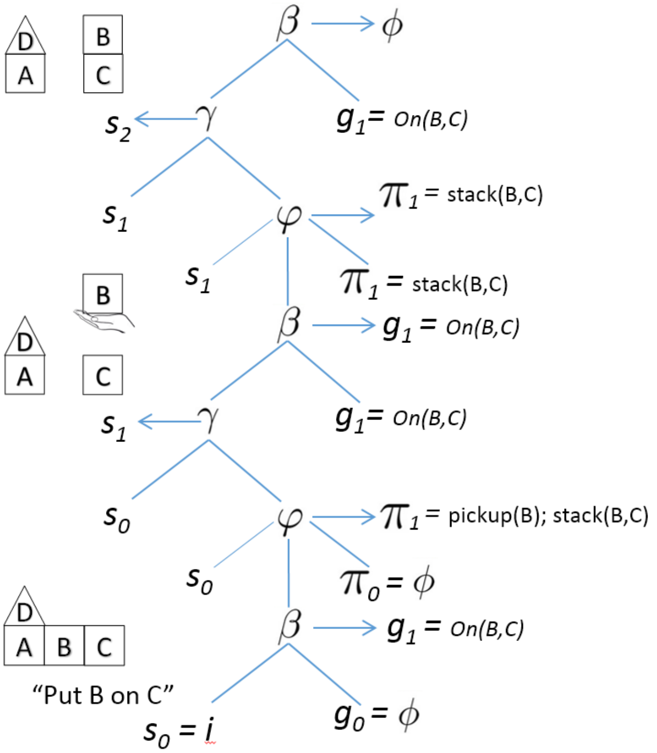

Figure 3. The execution of π_1. Here we drop the user imperative in Figure 2 to put A on B.

3. Goal-Driven Autonomy

Goal-driven autonomy (GDA) (Aha, et al., 2010; Cox, 2007; 2013; Klenk, Molineaux, Aha, 2013; Munoz-Avila, et al., 2010) is a kind of goal reasoning that focuses on goal formulation and explanation. In the GDA framework, the current goal g_c is a distinguished member of the set of all goals the agent currently desires (equation 7), but differs in that the agent has selected it for planning. Also in the GDA framework, the goal reasoner produces not only a plan, π, but also a set of *expectations*, $X = \{x1, \ldots, xk\}$ that represent constraints on the states resulting from each of the i=1..k actions α_i. When any currently observed state, s_c, diverges from the system's expected state, $s_e = x_c \in X$, a *discrepancy*, $d \in D$, is said to occur.

$$\mathcal{G} = \{ g_1, g_2, \ldots g_c, \ldots g_n\} \tag{7}$$

Given a discrepancy, a new goal may be warranted, so the system will retrieve an *abductive explanation* χ of d to determine a cause in the context of s_c. The explanation is performed to resolve the discrepancy. Finally given d, χ, and s_c, goal generation seeks to determine a new current goal, g_c, that seeks to remove the discrepancy d by either changing the state s_c to s_e or by learning a new expectation that justifies s_c. As such we now define a GDA planning problem as a 5-tuple in (8).

$$P_{gda} = (\Sigma, s_c, g_c, s_e, \mathcal{G}) \tag{8}$$

In the case where plan execution outcomes equal expectations (i.e., $s_c = s_e = s_0$) and thus no explanation is necessary (i.e., $\chi = \emptyset$) and the current goal is given (i.e., $g_c = g$), P_{gda} devolves into a classical planning problem P.

According to Cox (2007), the explanation[5] $\chi : \delta \rightarrow \ldots \rightarrow s_c$ contains a *salient antecedent* δ that represents the root cause of the problem signaled by the discrepancy. The goal then is to remove the cause of the problem, hence $g_c = \neg\delta$. *Goal insertion* is thus a search for an explanation that operationalizes the problem in terms of root causes. See Table 1 for the algorithm and Cox (2013) for a cognitive architecture that instantiates the algorithm.

Table 1: GDA algorithm for adding new goal to current goal set

```
function goal-insertion ($s_c$ : STATE; $\alpha_i$ : OPERATOR; $\mathcal{G}$: SET): SET;
    $s_e \leftarrow expects(\alpha_i)$
    Detect discrepancy, $d: s_c \neq s_e$
    If $\nexists d$, then $return(\mathcal{G})$
    Find an explanation, $\chi: \delta \rightarrow \cdots \rightarrow s'$,
    such that $covers(s', s_c)$
    $\sigma \leftarrow unify(s', s_c)$
    $subst(\sigma, precond(\chi))$
    if $satisfied(precond(\chi))$
    then $\mathcal{G} \leftarrow \mathcal{G} \cup \neg subst(\sigma, \delta)$
    $return(\mathcal{G})$
```

So consider again the blocks world example. When told to put A on B, the system easily creates a plan to clear A, pick it up, and place it on B. During the planning it had to solve the subgoal of having A clear before picking up the block. But B is already clear and it expects it to stay that way.

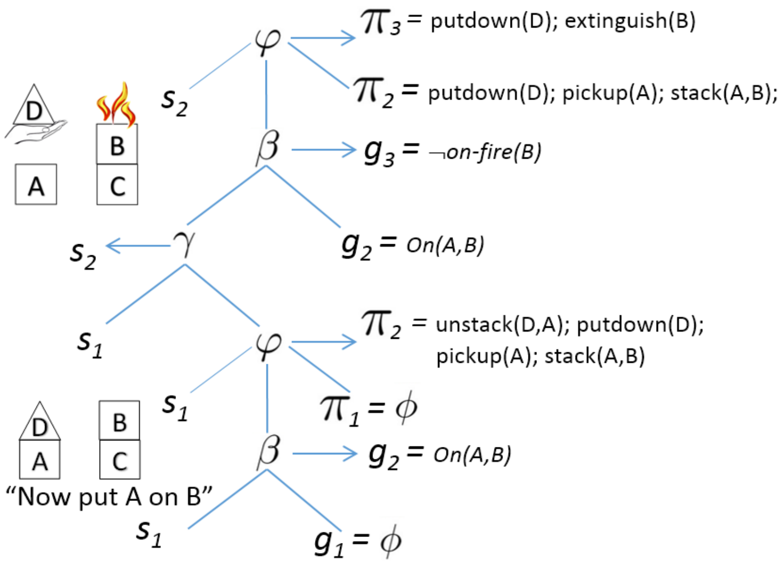

Figure 4. Explanation-based goal formulation

[5] The explanation is actually a graph $\chi = (V, E)$ with $\delta \in V$ an element of the source nodes and $s_c \in V$ a distinguished element of the sink nodes. See also Cox (2011).

Now suddenly assume that block B catches on fire and is no longer clear (see Figure 4 and Paisner, Cox, Maynord & Perlis, 2014 for an extended domain description).

Figure 4 shows state s_2 after the agent unstacked pyramid D from block A during the execution of plan π_2. The agent is still holding D, and B is on fire and hence no longer clear. Here the expectation s_e is clear(B), whereas the current observed state s_c is ¬clear(B). A very simple explanation is that a block being on fire causes it not to be clear.

$$\chi\colon on\_fire(y) \rightarrow \neg\, clear(y)$$

Given the substitution set $\sigma = \{y \mapsto B\}$, the goal $\neg subst(\sigma, \delta)$ becomes the achievement of $\neg on\_fire(B)$ and is added to the goal set $\mathcal{G}$=$\{g_2\}$. Planning then can create a sequence of actions to put out the fire before continuing. The planning function represented by phi needs to be intelligent enough to infer that the putdown action is duplicated in both plans π_2 and π_3. However for beta to be equally intelligent, it should have been informed of the current plan. Therefore, we redefine (3a) as follows.

$$\beta(s, g, \pi) \rightarrow g' \qquad \text{(3a')}$$

In (3a'), beta can use the current plan for expectations. Indeed, we stated in the example above that s_e, the expectation that B will remain clear, occurred during planning for π_2. With such input, beta can also generate a goal to retry a failed plan step during execution (e.g., vacuum cleaner did not remove all the dirt during a particular sweep). Furthermore, goal reasoning can be clear when trying to determine under unexpected and dynamic contexts (e.g., sudden resource reductions) whether plan adaptation or goal adaptation is the most rational choice. In this way too, planning is not only informed by goals from interpretation, interpretation is informed by planning.

4. GDA and the Learning of Goals

Beta is less about planning per se than about understanding the larger context within which a planning agent finds itself. If planning and interpretation are to be successful, learning must also be taken into consideration. Technically learning in this context is about inducing a set of goal states and applicability conditions from observed behavior of an execution system; and the main role for planning is to enable a GDA agent to infer which of the possible goal states are achievable within the expected costs and rewards of achieving those goals.

In the simplest sense, goal formulation can be cast as a novel classification task. An agent is given a state transition system, $\sum$, and a current state, s_c, and it must infer the current goal, g_c. Using a naïve supervised learning approach, one can present the system with many positive and negative examples of the tuple (s,g). From these examples, a learning algorithm can then learn a mapping in the form of a decision tree with goals at the leaves (see Maynord, Cox, Paisner, & Perlis, 2013 for an implementation of this approach). However the time it takes to learn such relations and the effort it takes to prepare suitable examples may not be available. Instead another approach is to exploit failure in performance by explaining why expectations do not apply in new situations and then using these explanations to learn appropriate goal orientations (i.e., to learn G, the set of useful goal states).

But we do not limit ourselves to goal generation alone, because if so the human burden of providing an external goal is shifted to the human specification of the complete range of goal representations. Instead if it is to be effective, the agent should also learn the classes of states in the world that constitute useful goals. This general approach represents a new conception of autonomous behavior.

Reconsider the planning domain from the introduction section and Figure 1. If a GDA-based agent interacts with other agents performing actions, it may observe a sugar distributor make a delivery to an Atlanta cola plant and then watch the manufacturer produce Coca-Cola at that location. The Coca-Cola is then delivered to Greenville where an advertising campaign has concluded. Sales then are observed to produce profits. Given these exogenous-event observations from the execution of π_{Coke} and π_{SDistr}, the GDA-agent can understand the causal relations by performing a goal regression (Veloso, 1994; Waldinger, 1981) upon the goal-subgoal graph entailed by $\sum$ and the observations (see Figure 5). Each node in the graph is an instantiated action-operator whose preconditions must be satisfied before the action can be executed. A tree of conjuncts can then be extracted to give the causal explanation structure shown in Figure 6. For example the state at(Cola-5, Coke) exists because the conjunct has(Coke, \$) $\wedge$ at(Bottles-2, Coke) $\wedge$ at(Sugar-1, Coke) has been satisfied. If any term is unsatisfied, then the explanation structure collapses.

Now a subsequent series of observations may violate the expectations present in this causal structure. For example the system may expect Coke to have further profits in Greenville, but Coke might lack the sales. Sugar may be delivered and promotions may occur, but without bottles, no cola is produced and hence no deliveries arrive for sale. In this case, the goal insertion algorithm (Table 1) would detect the discrepancy between the expectation has(Coke, sales) and the observation $\neg$has(Coke, sales). An explanation process would then use the prior explanation (from Figure 6) to derive a relational sequence responsible for the impasse. The resulting causal chain is shown in Figure 7.

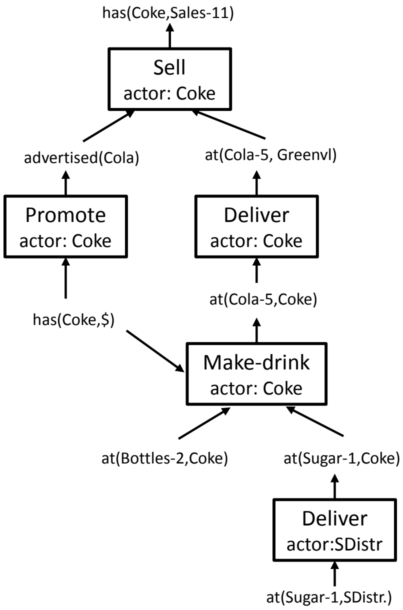

Figure 5. Goal-subgoal graph

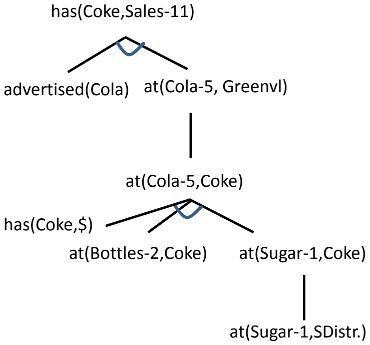

Figure 6. Causal explanation structure

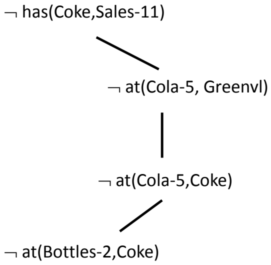

Figure 7. Failure causal chain

The goal insertion process would use the extracted failure chain to generate a goal and to add it to the agent's current set of desired goals $\mathcal{G}$. In Figure 7 the salient causal antecedent δ of the explanation is ¬at(Bottles-2, Coke), that is, the bottles are not at the plant (and that is ultimately why Coke lost profits). The new goal, g_c, is therefore to obtain the state of bottles being at this location (equation 9).

$$g_c = \neg\,(\neg \mathrm{at(Bottles\text{-}2, Coke)}) = \mathrm{at(Bottles\text{-}2, Coke)} \qquad (9)$$

Given a cooperative agent domain where the GDA agent represents a bottle distributor (i.e., BDistr) that coordinates with the sugar distributor and the cola manufacturers, the GDA agent can solve the new goal by generating a plan to perform the delivery method. That is, it will load the bottles at the McCaysville location, drive the load to Atlanta, go to Coke, and unload the shipment there. However a number of technical problems remain. Two in particular that we will discuss now concern issues of goal generalization and of goal anticipation.

4.1 Goal Generalization

First the goal above must be *generalized*. It is not useful to learn an overly specific goal of getting a particular shipment to a destination. Rather the agent must generalize the shipment to any available instance of type bottles. Furthermore, and this is a bit more subtle, the agent must eventually generalize the destination to any cola manufacturer in the domain (and not to just any location). That is Pepsi as well as Coke require bottle stock. By doing so, this simple explanation-based generalization rules out many irrelevant potential goal states such as at(bottles,bank) or at($,Dalton). Here we intend to induce the goal expression g_c in equation (10) and to add $g_c \in G$.

$$g_c = \mathrm{at(b,m)} \wedge \mathrm{bottles(b)} \wedge \mathrm{cola\text{-}manufacturer(m)} \qquad (10)$$

4.2 Goal Anticipation

Second the agent needs to *anticipate* the conditions under which the causal chain in Figure 7 might re-occur and plan to prevent it in the future. It should not wait until Coke sales fail each time. Thus in this example the GDA agent learns to generate the goal, g_c, when the bottles are low in inventory, not waiting until bottles run out and sales plummet. The agent learns a characterization of the state, s_c, in tuples (s_c, g_c), that is a rule $g_c \leftarrow s'$. This is the problem of learning goal-selection criteria (Powell, Molineaux, & Aha, 2011).

Unlike Powell and colleagues who use human guidance to acquire goal-selection knowledge, we suggest that an agent can learn this through unsupervised means. For example the state of inventory at manufacturers may include the predicates surplus, low, very-low, and out-of. The GDA agent should learn that bottles should be delivered when the supplies are low or very low, rather than waiting until they are out or trying to deliver when a surplus exists. In another instance of failure driven learning, the GDA agent would learn to rule out the expression surplus(m,b) ∧ bottles(b) ∧ cola-manufacturer(m) after it attempts to deliver bottles to a manufacturer having a bottle surplus. In this case we assume that the unload operation would not be given permission at the destination. Essentially the learned rule asserts the goal g_c when bottles at the manufacturer are low. This is what we meant when we stated previously that the agent should be able to explain that bottles need to be at the bottling plant *because the inventory is low*.

4.3 Deciding Which Goals Are Worth Generating

Just because a goal state s_g would be a desirable state to be in, there are several situations in which g may not be a useful goal for a goal reasoning agent to formulate. For example, s_g may be impossible to achieve; or in a nondeterministic domain, it may be impossible to guarantee that s_g will always (or usually) be achieved. Even if s_g is achievable, there may be another goal state $s_{g'}$ that is nearly as desirable as s_g and can be achieved at much lower cost than s_g. In such a situation it may be better for GDA to insert g' into its agenda rather than g. In general it is an open research question as to the best way to learn which goals are worth formulating and which to commit to first.

5. Related Research

An alternative formal model (Roberts et al., 2014; in press) treats goal reasoning as *goal refinement*. Using an extension of the plan-refinement model of planning, Roberts and colleagues model goal reasoning as refinement search over a *goal memory M*, a set of *goal transition operators R*, and a transition function *delta* that restricts the applicable operators from R to those provided by a fundamental *goal lifecycle*. Unlike the formalism here that represents much of the goal reasoning process with the single function beta, Roberts proposes a detailed lifecycle consisting of goal formulation, goal selection, goal expansion, goal commitment, goal dispatching, goal monitoring, goal evaluation, goal repair, and goal deferment. Thus many of the differential functionalities in beta are distinct and explicit in the goal reasoning cycle. However the model here tries to distinguish between the planning and action side of reasoning (i.e., phi and gamma) and the interpretation and evaluation components inherent in goal reasoning (i.e., beta). We can roughly note, however, that phi relates to Robert's goal expansion and repair, whereas beta relates to goal formulation, monitoring, and deferment. Missing in my current analysis is the processes of goal dispatching, commitment, and evaluation.

Additionally Roberts proposes a more complex goal structure. A *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. We have been more circumspect regarding the syntactic structure of a goal. In section 2.1 on classical planning, we defined the goal formula to be a conjunction of first-order literals. More generally we would also allow universally quantified predicate conjuncts as in Cox & Veloso (1998; Veloso, Pollack & Cox, 1998). Thus in a package delivery domain we allow $\forall p \mid (package\ p) \land \exists\ d | (destination\ p\ d) \land (location\ p\ d)$ as a goal to deliver all packages to their destination. Under the scope of this goal expression, new goals $(location\ p\ d)$ may arise during planning or plan execution time as additional packages arrive at the warehouse for delivery. See also Talamadupula, Benton, Kambhampati, Schermerhorn, & Scheutz (2010) for their concept of *open world quantified goals*. But overall, much exists in common with the representations here and with Roberts including an association with learning (see Roberts & Aha, this volume).

The two problems of goal generalization and goal anticipation make the learning task as we discuss it here considerably different from related research. Planning research is concerned with generating an action sequence to carry out a goal that is given to a system by an external user (see survey in Ghallab, Nau, & Traverso, 2004). In the larger scope of integrated cognitive systems, we intend to better understand autonomy as the capacities (1) to recognize novel problems and (2) to independently form the desire to remove these problems. Goal anticipation and goal generalization are instrumental processes associated respectively with these two aspects of autonomy. However unlike bottom-up, data-driven learning approaches, the top-down explanation of anomalies is a central pivot in our learning approach (Cox, 2011).

The explanation-based approach we take to learning should not be confused with the older, mainly deductive approach called explanation-based learning (DeJong & Mooney, 1986; Mitchell, Keller, & Kedar-Cabelli, 1986). We use explanation techniques (Cox, 2011; Cox & Ram, 1999; Roth-Berghofer, Tintarev, & Leake, 2011) that depart significantly from earlier work, being more abductive, knowledge rich, and case-based. The idea is to retrieve a graph structure called an explanation pattern given a discrepancy and a context, to instantiate it and to bind it to the discrepancy, and then to adapt it to the context. This also departs from other GDA systems (e.g., ARTUE, Molineaux, Klenk & Aha, 2010) which use abductive causal inference in the form of an assumption-based truth maintenance system or ATMS (de Kleer, 1986) for explanation.

As mentioned previously, goal formulation or generation has been an instrumental task in research on goal reasoning and goal-driven autonomy. In the ARTUE system, goal formulation occurs as a response to discrepancy detection and is based upon knowledge structures called principles (Klenk, Molineaux, & Aha, 2013). Other approaches rely upon links between specific states and a priori goal candidates (Dill & Papp, 2005) or triggering schemas based on the current state (Talamadupula, Benton, Schermerhorn, Kambhampati, & Scheutz, 2009). In the EISBot interactive game-playing system, association rules called goal formulation behaviors link discrepancy-explanation types to goal types (Weber, Mateas & Jhala, 2010). Here we discussed the use of the negation of explanation root causes to generate new goals. On the other hand, the learning of goal state information and its generalization is novel. The closest research is that of learning goal selection knowledge (Jaidee, Munoz-Avila, & Aha, 2011; Powell, Molineaux, & Aha, 2011).

6. Conclusion

This research attempts to reconcile some of the existing research on goal reasoning with theoretical frameworks in the automated planning and the intelligent agents communities. The resulting goal-reasoning formalism augments notions of planning and plan execution with formal models of re-planning and both goal generation and goal change. In situations where goals are given at initialize time and do not change throughout the process and where plans execute as expected, the model devolves into a classical planning formalism. We have extended the model to a kind of goal reasoning called goal-driven autonomy, and we have applied the model to the explanatory learning of those goals worth pursuing.

Acknowledgements

This material is based upon work supported by ARO grant W911NF-12-1-0471 and by ONR grants N00014-15-1-2080, N00014-12-1-0430, and N00014-12-1-0172. I thank the anonymous reviewers for their insights and comments. I also wish to acknowledge the previous input of David Aha, Michael Maynord, and Mark Wilson on an earlier formulation of these ideas during a poster presentation (Maynord, Aha, Wilson & Cox, 2013).

References

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 Mary-land, Department of Computer Science.

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.

Aiello, L. C., Cesta, A., Giunchiglia, E., Pistore, M., & Traverso, P. (2001). Planning and verification techniques for the high level programming and monitoring of autonomous robotic devices. In *European Space Agency Workshop on On-Board Autonomy*, Noordwijk, Netherlands. ESA.

Bengfort, B., & Cox, M. T. (this volume). Interactive reasoning to solve knowledge goals. To appear in *Proceedings of the 2015 Annual Conference on Advances in Cognitive Systems: Workshop on Goal Reasoning* (IRIM Tech Report). Atlanta: Georgia Institute of Technology.

Bertoli, P., Cimatti, A., Roveri, M., & Traverso, P. (2006). Strong planning under partial observability. *Artificial Intelligence*, *170*(4): 337–384.

Boutilier, C., Dean, T. L., & Hanks, S. (1999). Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research (JAIR)*, *11*:1–94.

Cimatti, A., Roveri, M., & Traverso, P. (1998). Strong planning in non-deterministic domains via model checking. In R. Simmons, M. Veloso, & S. Smith (Eds.), *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems* (pp. 36–43). Menlo Park, CA: AAAI Press.

Cox, M. T. (2013). Question-based problem recognition and goal-driven autonomy. *Goal Reasoning: Papers from the ACS workshop* (pp. 10-25). (Tech. Rep. No. CS-TR-5029). College Park, MD: Univ. Maryland, CS Dept.

Cox, M. T. (2011). Metareasoning, monitoring, and self-explanation. In M. T. Cox & A. Raja (Eds.) *Metareasoning: Thinking about thinking* (pp. 131-149). Cambridge, MA: MIT Press.

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

Cox, M. T., & Ram, A. (1999). Introspective multistrategy learning: On the construction of learning strategies. *Artificial Intelligence, 112*, 1-55.

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

Cox, M. T., & Zhang, C. (2007). Mixed-initiative goal manipulation. *AI Magazine 28*(2), 62-73.

Dean, T., Kaelbling, L. P., Kirman, J., Nicholson, A. E. (1995). Planning under time constraints in stochastic domains. *Artificial Intelligence 76*(1-2): 35-74.

DeJong, G., & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning*, *1*(2), 145-176.

de Kleer, J. (1986). An assumption-based TMS. *Artificial Intelligence, 28*(2), 127-162.

Dill, K., & Papp, D. (2005). A goal-based architecture for opposing player AI. In *Proceedings of the First Artificial Intelligence for Interactive Digital Entertainment Conference* (pp. 33-38). Menlo Park, CA: AAAI Press.

Fox, M., & Long, D. (2003). PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR)*.

Ghallab, M., Nau, D., & Traverso, P. (2014). The actor's view of automated planning and acting: A position paper. *Artificial Intelligence 208,* 1–17.

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated planning: Theory and practice*. San Francisco: Morgan Kaufmann.

Jaidee, U., Munoz-Avila, H., & Aha, D.W. (2011). Integrated learning for goal-driven autonomy. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*. Barcelona, Spain.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1995). Partially observable Markov decision processes for artificial intelligence. In *Proceedings of Reasoning with Uncertainty in Robotics* (pp. 146-163).

Klenk, M., Molineaux, M., & Aha, D. (2013). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence, 29*(2), 187–206.

Maynord, M., Aha, D. W., Wilson, M. & Cox, M. T. (2013, December). *On the definition and desirability of goal reasoning*. Invited poster, 2013 Annual Conference on Advances in Cognitive Systems: Workshop on goal reasoning. University of Maryland, Baltimore, MD.

Maynord, M., Cox, M. T., Paisner, M., & Perlis, D. (2013). Data-driven goal generation for integrated cognitive systems. In C. Lebiere & P. S. Rosenbloom (Eds.), *Integrated Cognition: Papers from the 2013 Fall Symposium* (pp. 47-54). Technical Report FS-13-03. Menlo Park, CA: AAAI Press.

Mitchell, T. M., Keller, R., & Kedar-Cabelli, S. (1986). Explanation-based generalization: A unifying view, *Machine Learning, 1*(1), 47-80.

Molineaux, M., Klenk, M., & Aha, D. (2010). Goal-driven autonomy in a navy training simulation. In *Proceedings of Twenty-Fourth AAAI Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Munoz-Avila, H., Aha, D.W., Jaidee, U., Klenk, M., & Molineaux, M. (2010). Applying goal driven autonomy to a team shooter game. *Proceedings of the Twenty-Third Florida Artificial Intelligence Research Society Conference* (pp. 465-470). Menlo Park, CA: AAAI Press.

Paisner, M., Cox, M. T., Maynord, 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.

Powell, J., Molineaux, M., & Aha, D. W. (2011). Active and interactive discovery of goal selection knowledge. In *Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference*.

Ram, A. (1990). Knowledge goals: A theory of interestingness. *Proceedings of the 12th Annual Conference of the Cognitive Science Society* (pp. 206–214).

Roberts, M., & Aha, D. W. (this volume). Cleaning efficiently: A case study in modeling goal reasoning and learning. To appear in *Proceedings of the 2015 Annual Conference on Advances in Cognitive Systems: Workshop on Goal Reasoning* (IRIM Tech Report). Atlanta: Georgia Institute of Technology.

Roberts, M., Vattam, S., Alford, R., Auslander, B., Apker, T., Johnson, B., & Aha, D. W. (in press). Goal reasoning to coordinate robotic teams for disaster relief. To appear in *Proceedings of ICAPS-15 PlanRob Workshop.*

Roberts, M., Vattam, S., Alford, R., Auslander, B., Karneeb, J., Molineaux, M., Apker, T., Wilson, M., McMahon, J., & Aha, D. W. (2014). Iterative goal refinement for robotics. In *Proceedings of ICAPS 2014*.

Roth-Berghofer, T., Tintarev, N., & Leake, D. B. (Eds.) (2011). *Proceedings of the IJCAI-11 Workshop on Explanation-aware Computing ExaCt 2011*, July 2011.

Talamadupula, K., Benton, J., Kambhampati, S., Schermerhorn, P., & Scheutz, M. (2010). Planning for human-robot teaming in open worlds. *ACM Transactions on Intelligent Systems and Technology*, *1*(2).

Talamadupula, K., Benton, J., Schermerhorn, P., Kambhampati, S., & Scheutz, M. (2009). Integrating a closed world planner with an open world robot: A case study. In M. Likhachev, B. Marthi, C McGann, & D. E. Smith (Eds.) *Bridging the Gap between Action and Motion Planning: Papers from the ICAPS Workshop*. Thessaloniki, Greece.

Vattam, S., Klenk, M., Molineaux, M., & Aha, D. (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* (pp. 111-126). Tech. Rep. No. CS-TR-5029. College Park, MD: University of Maryland, Department of Computer Science.

Veloso, 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 Fourth International Conference on Artificial Intelligence Planning Systems* (pp. 171-179). Menlo Park, CA: AAAI Press.

Waldinger, R., (1981). Achieving several goals simultaneously. In N. J. Nilsson & B. Webber (Eds.), *Readings in Artificial Intelligence* (pp. 250-271). Palo Alto, CA: Tioga.

Weber, B. G., Mateas, M., & Jhala, A. (2010). Applying goal-driven autonomy to StarCraft. In *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment.* Menlo Park, CA: AAAI Press.