Advances in Cognitive Systems 5 (2017) 57–76 Submitted 9/2017; published 12/2017

A 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

The cognitive systems literature describes many methods for problem solving and planning. Given a goal state, these methods search for solutions that achieve the goal through interactions with some environment. However, a major assumption is that goals are given, usually by a user directly as input or as part of the problem representation. Furthermore, once given, the goals do not change. Here, we formalize the notion that goal formulation and goal change are themselves major cognitive operations. We include in our model not just plan generation and execution but also interpretation of the environment as plans execute, exogenous events occur, and plans change.

1. Introduction

In virtually all intelligent systems, goal states are predefined and exogenously provided by an external user. But to have a continuing existence over time, agents must be flexible to survive and to continue to be useful. Recent work on *goal reasoning* (see Aha, Cox, & Muñoz-Avila, 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 for them. Much of this work has been situated within the context of the automated planning community and has borrowed their formal notations as a theoretical framework. The goal of this paper is to extend the standard planning formalism to account for mechanisms of goal reasoning and to begin to integrate the vocabulary of cognitive systems with that of the larger artificial intelligence (AI) field. The result combines notations for planning, action, and interpretation within the scope of goal reasoning. The intent is to seek a uniform theoretical basis for expressing and communicating work on integrated cognitive systems. That is, cognitive agents interpret a changing world and express their desires in terms of dynamic goals; they do more than just plan for and achieve a set of goals given by an outside party.

As an example, consider a package-delivery domain (Figure 1) that involves a network of locations connected by roads. From time to time, vehicles must transport objects from one location to another as requests arrive. Deliveries may be accomplished by picking up and delivering packages (e.g., 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), a given fixed goal exists to be achieved (e.g., the delivery of specified packages), and the problem ends at any world state in which 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 should be achieved and what the agent should do after achieving them. In a more realistic model of a dynamically changing world, bottles must be delivered to the bottling plant as the inventory changes, and the agent may need to explain and hence understand this to generate new goals when the inventory becomes unexpectedly low.

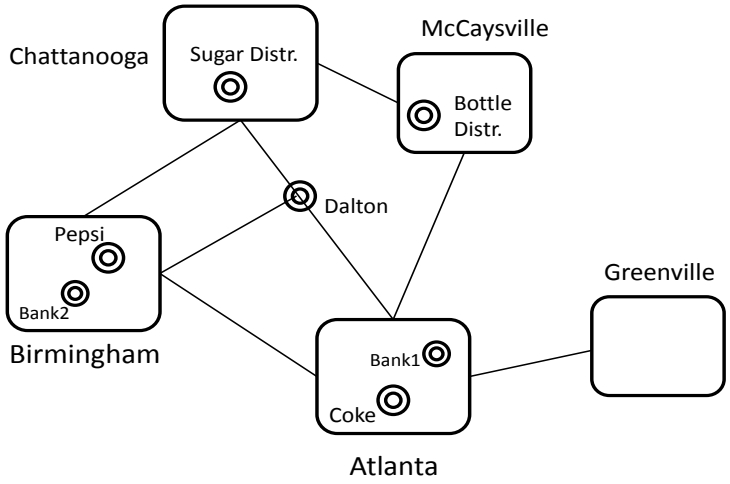

Figure 1. A simple package delivery domain for manufacturing soft drinks. Product ingredients and packaging must be moved through the network of locations and finished products must be delivered, advertised, and sold to consumers.

The first section of the paper provides our formal representations for goal reasoning and 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. After this, we discuss related research and draw some conclusions.

2. A Formalism 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. This section examines the formal model for classical deterministic planning, briefly considers non-deterministic planning, extends the formalism to represent interpretation (including goal change and formulation), and then unifies the individual formalisms into a single model of planning, action, and interpretation. The model is illustrated with a simple example.

2.1 Classical Planning Theory

A *classical planning domain* is typically defined (e.g., Ghallab, Nau, & Traverso, 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 of a function-free, first-order language $\mathcal{L}$. A *planning operator* $o \in O$ is represented as the triple $\big(head(o), pre(o), eff(o)\big)$, where $pre(o)$ and $eff(o)$ (the *preconditions* and *effects* of o, 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* α is a ground instance of a planning operator.

An action $\alpha \in A$ is *executable* in a state s if $s \vDash pre(\alpha)$. The state resulting from the execution of action α is $\big(s - eff^-(\alpha)\big) \cup eff^+(\alpha)$, where $eff^+(\alpha)$ and $eff^-(\alpha)$ are the atoms and negated atoms, respectively, in $eff(\alpha)$. A plan $\pi = \langle \alpha_1 \alpha_2 \ldots \alpha_n \rangle$ is executable in s if each $\alpha_{2 \le i \le n}$ is

executable in the state produced by α_{i-1} and α_1 is executable in s. 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, $\gamma: S \times A \rightarrow S$ is a state transition function that returns the resulting state of an executable action given a current state. Thus, given a state and action, one infers the subsequent state $\gamma(s, \alpha) \rightarrow s'$ that follows after the action is executed.

A *classical planning problem* is the triple $P = (\Sigma, s_0, g)$, where Σ is a state transition system, s_0 is the initial state, and $g \in \mathrm{G}$ (the *goal formula)* is a conjunction of first-order literals.[1] 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 $\langle \alpha_1 \alpha_2 \ldots \alpha_n \rangle$) 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 a 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.

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

Now we recursively recast γ 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 $\gamma(s_0, \pi_g) \vDash g$.

Definition 1: Classical plan execution.

Equation (2) recursively defines plan execution as an invocation of the state transition function. From the initial state, sequential execution of each action results in a state that entails g.

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

2.2 Planning with Nondeterminism

In the AI literature, uncertainty about the outcomes of actions has been dealt with mainly in two different ways: using Markov decision processes (MDPs) (Boutilier, Dean, & Hanks, 1999; Dean et al., 1995; Kaelbling, Littman, & Cassandra, 1995) and using model-checking techniques (Cimatti, Roveri, & Traverso, 1998; Aiello et al., 2001; Bertoli, Cimatti, & 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., to maximize expected utility) and model checking is useful when the transition probabilities are unknown or unimportant (e.g., 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* $o = (head(o), pre(o), eff_1(o), eff_2(o), \ldots, eff_k(o))$ 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 $p_1 + p_2 + \cdots + p_k = 1$. 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.

[1] A goal is often represented as a ground literal or conjunction of ground literals (Ghallab, Nau, & Traverso, 2004) and the literal is an arbitrary state predicate. For the purposes of this paper we will not assume the goal to be grounded (i.e., it can be existentially quantified). Further, the goal is a member of the set of possible goals, not the wider set of all possible states. More generally, it could be a complex expression, as discussed in related research, but here we limit goals to conjuncts.

However, in this paper we will use the classical deterministic definition of planning to illustrate clearly 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 manage their own goals (Vattam et al., 2013). Goal reasoning has recently extended the classical formulation by relaxing the assumption that an initial goal originates independently of the agent. But even if the planning process starts 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 or change old ones at execution time as situations warrant.

We posit a simple model of goal management and change that represents the set $\Delta = \{\delta \mid \delta: G \rightarrow G\}$ of potential transformations on goals an agent may select. An individual change $\delta: G \rightarrow G$ is a function from one goal expression $g \in G$ to another g', where $G \subset S$ is the set of all potential desired states.

Definition 2: Goal-driven interpretation.

Expression (3) defines the general interpretation function $\beta: S \times G \rightarrow G$ as mapping one goal g to a (possibly different) goal g' given some state s:

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

Now given a current contextual state s and goal g, β selects from Δ a sequence of transformations $\langle \delta^1, \delta^2, \ldots \delta^n \rangle$ that results in the output $\delta_n(\ldots \delta_2(\delta_1(g))) = g'$. As such, β is a state interpretation process that perceives the world with respect to its goals. Thus, the function is central to goal reasoning and management. Theoretically, β represents the choice of an agent, given a new state and current goal expression, to determine anew what it wants in the future (i.e., its goal) relative to its ongoing observations.[3]

2.3.1 Goal Change

Unlike classical planning models that assume goals to be static and given externally, a goal reasoning perspective views goals as malleable and subject to change. In our formalism, a particular transformation is represented as $\delta = \big(head(\delta), parameter(\delta), pre(\delta), res(\delta)\big)$, where $pre(\delta)$ and $res(\delta)$ are its preconditions and result. The transformation's identifier is $head(\delta)$, and its input goal argument is $parameter(\delta)$. Among the elements of Δ is the *identity transformation* $\delta^I(g_i) = g_i$ for all $g_i \in G$. As a 4-tuple, δ^I is represented as $(identity, g, \{true\}, g)$. Here, the precondition is simply the set containing only true, because it can always be applied in any sequence. When β selects only identity, the choice captures an agent's decision not to change its current goal given the state s. But more generally, the capability of β to change substantially an agent's goals in the

[2] We ignore for now the set of exogenous events E that are outside the control (and possibly observation) of the reasoning system. Note that the use of the word "interpretation" does not correspond to the notion of interpretation that defines the logical semantics of formal representations. Rather, it denotes a high-level perceptual-like process.

[3] The language we use here implicitly equates the current state with an observation and thus assumes full observability. This simplification is further reinforced by the use of the terms interpretation and perception within the description of β. But we could (and if space allowed should) distinguish the perceptual observation of partial states from the conceptual interpretation of the resulting percepts. We leave this exercise for future work.

face of an uncooperative environment represents a powerful adaptive alternative to plan modification and replanning.

Goals can undergo sequences of transformations (Cox & Dannenhauer, 2016; Cox & Veloso, 1998) including priority shifts (Choi, 2011) and in extreme cases, abandonment (Cox & Dannenhauer, 2017; Harland et al., 2017). As an example, a chess player may start out with the goal to achieve checkmate. Let this be represented by g_{ch}. Given a series of $k < n$ unsuccessful opening moves, $\pi_{g_{ch}}[1..k]$ where $n = |\pi_{g_{ch}}|$, the player may change the goal to draw (i.e., g_{dr}). We denote this through the application of the transformation $\delta(g_{ch}) \rightarrow g_{dr}$ and by specializing expression (3) as $\beta(\gamma(s_0, \pi_{g_{ch}}[1..k]), g_{ch}) \rightarrow g_{dr}$.

While maintaining consistency with automated planning theory, this formalization is broad and generalizable to many types of cognitive processes. Cox, Dannenhauer, and Kondrakunta (2017) enumerate a specific set of transformations Δ that represent many of the operations on goals that exist in the goal reasoning literature. Furthermore, goals can follow arcs or trajectories through a space of goals over time (Bengfort & Cox, 2015; Eyorokon, Panjala, & Cox, 2017). These trajectories result from repeated goal changes through different instances of β interpretation. But most importantly for high-level cognition, goals can be created or formulated given a particular problem state.

2.3.2 Goal Formulation

From some initial state s_0 and an empty goal state, an agent formulates a new goal as in the expression $\beta(s_0, \emptyset) \rightarrow g$. This transformation, designated as $\delta^*() = g$, is called a *goal insertion* (Cox, 2013; Paisner et al., 2013). Here we do not commit to how the transformation might be selected in β. Cost and expected benefit information might be used, but these details are at the discretion of the application (see Section 3.1 for one approach to goal insertion).

In one sense, goal formulation through β can still involve 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 β would be to interpret the intention of the human and to infer the goal from the utterance. In another sense, however, allowing an agent to generate its own goals substantially departs from the classical formulation of a problem. For goal reasoning in its simplest form, a planning problem can be cast as the pair $P = (\Sigma, s_0)$. 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 to create (then execute) a plan to achieve it. Under classical planning and 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. This goal formulation is covered by specializing expression (3) as $\beta(s_g, \emptyset) \rightarrow g'$. As we will show, goals can potentially be formulated from any state.

2.4 A Model of Planning, Action, and Interpretation

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

$$\gamma(s_0, \pi_{\beta(s_0,\emptyset)}) = \gamma\left(\gamma(s_0, \alpha_1),\ \pi_{\beta(\gamma(s_0,\alpha_1),\beta(s_0,\emptyset))}[2..n]\right) \quad (4)$$

When goals change (or new ones are added) through β, plans may need to change as well. The problem with the above formalization is that, in the recursive right-hand side of (4), 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, the goal reasoner may need to replan and alter the length and composition of the remainder of the plan.

Definition 3: Planning and replanning.

To cover the above contingency, expression (5) defines the function $\varphi : S \times G \times 2^O \rightarrow 2^O$, which takes as input a state, goal, and current plan. The expression covers the case of initial planning and of replanning during plan execution time.

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

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). This resolves the anomaly indicated above.

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

Given φ, the formalism is general across different variations of goal reasoning and (re)planning. More fully, we have equation (7) below.

Definition 4: Goal reasoning.

We define goal reasoning as goal-driven interpretation in the context of planning, action, and replanning. On the left-hand side of the equation, φ instantiates the initial planning process from the initial state with a goal provided by β. The right-hand side specifies the recursive structure for interpretation of the state and its associated goal reasoning, together with replanning and plan execution. The annotations above and below the figure indicate intermediate results to help readability. The key portion to examine on the right-hand side is the outer-most invocation of φ from s_1 with a potential new goal g_2 and the remainder of plan π_{g_1}. This is where replanning can occur if the goal has changed.

$$\gamma\left(s_0,\ \underbrace{\varphi(s_0, \underbrace{\beta(s_0,\emptyset)}_{g_1}, \emptyset)}_{\pi_{g_1}}\right) = \gamma\left(\underbrace{\gamma(s_0,\alpha_1)}_{s_1},\ \underbrace{\varphi(\underbrace{\gamma(s_0,\alpha_1)}_{s_1}, \underbrace{\beta(\underbrace{\gamma(s_0,\alpha_1)}_{s_1}, \underbrace{\beta(s_0,\emptyset)}_{g_1})}_{g_2}, \underbrace{\varphi(s_0, \beta(s_0,\emptyset), \emptyset)[2..n]}_{\pi_{g_1}[2..n]})}_{\pi_{g_2}}\right) \quad (7)$$

When β generates an exogenous initial goal $g_1 = g_{user}$ from the initial state s_0 and simply returns the input goal from all other states (i.e., $g' = g$ in 3 using δ^I), the formalization reduces to classical planning with a user-given goal. That is, equation (7) is equivalent to (2) essentially because (3) represents a trivial boundary case.

Theorem: Goal reasoning subsumes classical planning and acting**.**

Proof: It is sufficient to show that goal reasoning as defined by equation (7) is equivalent to classical plan execution as defined by equation (2) when (3) and (5) are restricted in scope. Here we limit the scope by assuming a special case of a goal insertion transformation as $\delta^*(\,) \rightarrow g_{user}$ where $g_{user} = g$ in classical planning and by assuming $\Delta = \{\delta^*, \delta^I\}$.

Restricted functions β (interpretation) and φ (planning) are as follows:

$$\beta(s,g) = \begin{cases} \delta^*(\quad), & (s = s_0) \wedge (g = \emptyset) \\ \delta^I(g), & otherwise \end{cases} \qquad \varphi(s,g,\pi) = \begin{cases} \langle \alpha_1 \alpha_2 \ldots \alpha_n \rangle, & (s = s_0) \wedge (\pi = \emptyset) \\ \pi, & otherwise \end{cases}$$

On the left-hand side of (7), we can now replace $\varphi(s_0, \beta(s_0, \emptyset), \emptyset)$ with $\pi_{\beta(s_0,\emptyset)}$ because of the new output of φ along with equation (1), and then replace $\pi_{\beta(s_0,\emptyset)}$ with π_g given the specialization of β and δ^*. Similar substitutions reduce the right-hand side of (7) to equation (2). From states other than s_0, the functions β and φ simply return their parameters g and π respectively.

2.5 A Blocks World Example

The representational formalism presented in the preceding sections extends the traditional AI planning notation to include a more inclusive scope that incorporates major cognitive functions for an autonomous agent. This includes both action and planning as well as perceptual interpretation and flexible goal reasoning. These representations can be chained together to capture the reasoning and behavior of an agent, and as such, allow an agent to reason about the entire cognitive and environmental context related to what it wants to accomplish (i.e., its goals).

Consider a simple blocks world situation where goals are given to the agent by a human user in natural language. In this case, β does not just input a goal in first-order predicate form. Instead it must translate an utterance or the result of a speech act into a predicate representation. That is, 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 intended illocutionary act is most often a request for the agent to achieve a goal[4] for the speaker. Consider the utterance "Put block B on block C". Although spoken as an action, the intent is actually for the agent to achieve the state of block B being on top of C. Although we will not discuss details here, Figure 2 shows how the formalism organizes the context of a dialog between human and agent so that inferring the intent of the last utterance "Add one more" is possible.

However, Figure 2 hides many details. The plan π_1 has two actions that execute sequentially, not concurrently in one invocation of γ as the figure suggests. A more realistic depiction of the partial plan execution from s_0 to s_1 in Figure 2 is shown in Figure 3. Here we see that the execution of the action $pickup(B)$ results in an intermediate state (now s_1 in Figure 3). From that state no new goal is formulated, and subsequently no new re-planning is performed. The goal $On(B,C)$ (i.e., g_1) is achieved in the next state s_2, and β evaluates the goal achievement and returns the null goal.

Representations such as those depicted in these illustrations provide a clear vocabulary for both the AI planning community and researchers working on integrated cognitive systems. We have shown elsewhere that the combined context as represented in these examples provides a basis for explicit cognitive traces reasoned over by a metacognitive system (Cox & Dannenhauer, 2016; Dannenhauer, 2017). Finally, implementations of such concepts apply to a wide range of applications for robust autonomy, particularly for worlds that change in unexpected ways.

[4] For questions, the intended goal or objective is considered a *knowledge goal* (Bengfort & Cox, 2015; Ram, 1990) as opposed to an attainment goal.

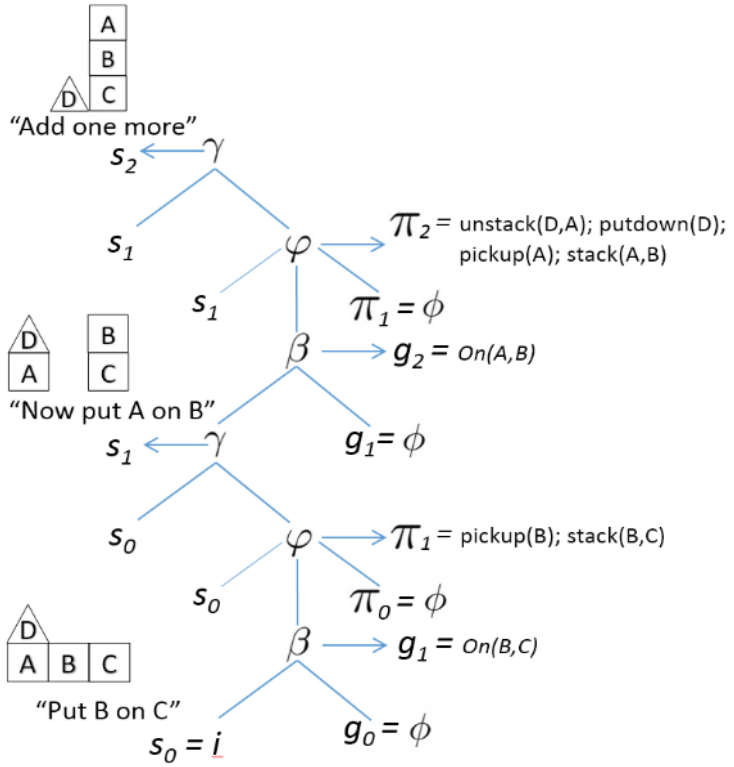

Figure 2. Partial context for simple block stacking example. The structure is read from the bottom upward. Note that horizontal arrows are return values, whereas all other arrows are input arguments. Initially A, B, and C are on the table and D is on A. No explicit goal exists, but the statement to put B on C is part of the initial state. The function β interprets the utterance (stated as an action) as the goal state of B on C. The planning function φ then creates a plan to pick up B and stack it on C. The transition function γ executes the plan to achieve the state pictorially shown on the left and above the initial state.

3. Goal-Driven Autonomy (GDA)

Goal-driven autonomy (GDA) (Aha et al., 2010; Cox, 2007; Klenk, Molineaux, & Aha, 2013; Muñoz-Avila et al., 2010) is a kind of goal reasoning that focuses much of the research on goal formulation and causal explanation. In the GDA framework, goal management is significant and represents a key computational feature. A common framework (Klenk, Molineaux, & Aha, 2013) partitions goal reasoning into a pipeline of (1) discrepancy detection; (2) explanation generation; (3) goal formulation; and finally (4) goal management. In alternative frameworks, goal formulation and management are seen as part of the same computational process; i.e., goal management is considered as combinations of various goal transformations, with formulation simply an instantiation of the goal insertion transformation (Cox, 2013). Regardless, the common assumption is that formulation of goals need not depend directly upon an outside human user, and it represents a central feature of autonomy for intelligent agents. Here we examine GDA to illustrate concrete aspects of the theory put forth in this paper.

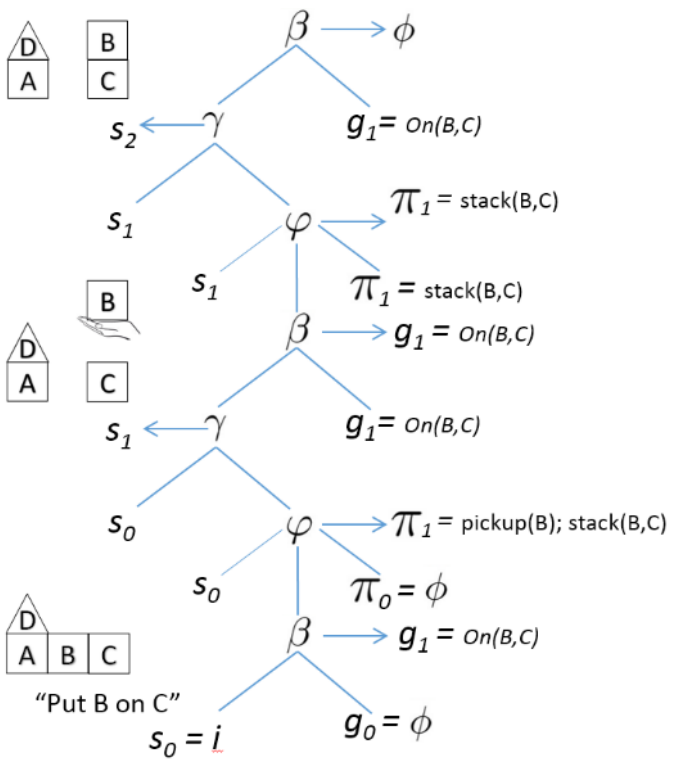

Figure 3. Greater contextual detail shown for the execution of π_1. The structure is read from bottom to top. Here we drop the user imperative in Figure 2 to put A on B. The individual steps of the plan are executed sequentially by γ and shown with earlier steps lower in the figure.

3.1 Explanation and Goal Formulation

The current goal g_c of an agent is a distinguished member of the *goal agenda* $\hat{\mathrm{G}}$. As shown in equation (8), the agenda is the set of all goals the agent currently desires, whereas g_c is the subset of $\hat{\mathrm{G}}$ the agent selected for planning.

$$\hat{\mathrm{G}} = \{g_1, g_2, \ldots g_c, \ldots g_n\} \subseteq \mathrm{G} \tag{8}$$

Many techniques can apply to the goal selection problem; see Kondrakunta and Cox (2017) for the implemented approach we use in practice. Also, in the GDA framework, the goal reasoner produces not only a plan π but also a set of *expectations* $EX = \{x_1, \ldots, x_k\}$, which represents anticipated states associated with each of the $i = 1..k$ actions α_i. When any currently observed state s_c diverges from the agent's expected state $s_e = x_c \in EX$, a *discrepancy* d: $s_c \neq s_e \mid d \in D$ is said to occur (Dannenhauer, 2017; Dannenhauer & Muñoz-Avila, 2015; Dannenhauer et al., 2016).

To elaborate the causal factors responsible for the discrepancy, a GDA agent applies the abductive explanation function in Table 1. A case-based *explanation pattern (XP)*[5] χ: $\omega \rightarrow \ldots \rightarrow s_c$

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

traces a path from some set of antecedents ω through one or more causal factors to the unexpected state. From the expectations and discrepancy, a set of candidate explanations are retrieved from the casebase. Unification computes a substitution set σ from the current state and the explanation's target node. Instantiating role variables from σ produces a context specific explanation. The function returns the first such explanation whose preconditions hold. Dannenhauer et al. (2016) provide implementation details with respect to both *the expectation generator X(s_c, α_i)* and the *check* function that detects a discrepancy.

Table 1. Case-based explanation function for a goal-driven autonomy agent.

Parameters: s_c – current state; α_i – action; X – expectation function; Casebase – explanations.

function *GDA-explanation* (s_c : S; α_i : O; X:S×Π→S; Casebase: $\{\chi\}$): χ

1. $s_e \leftarrow \mathrm{X}(s_c, \alpha_i)$ *;; determine expectations*
2. $d \leftarrow check(\mathrm{s}_e, \mathrm{s}_c, \alpha_i)$ *;; see Dannenhauer et al. (2016)*
3. **if** $\nexists d$, **then return**$(\emptyset)$
4. XPs $\leftarrow \{[\omega \rightarrow \cdots \rightarrow s'] \in \mathrm{Casebase} | covers(s', s_c)\}$ *;; retrieve candidate XPs*
5. **for** $\chi \in$ XPs **do**
6. $\sigma \leftarrow unify(s', s_c)$ *;; unification returns a substitution set*
7. **for** $y \in precond(\chi)$ **do** $subst(\sigma, y)$ *;; instantiate explanation variables*
8. **if** $\forall y | y \in precond(\chi) \wedge satisfied(y)$ *;; do XP preconditions hold ?*
   then return (χ) *;; if so, then return candidate*
9. **return** $(\emptyset)$ *;; no suitable XP found*

The explanation χ: $\omega \rightarrow ... \rightarrow s_c$ contains a *salient antecedent* $\varepsilon \in \omega$ that represents the root cause of the problem signaled by the discrepancy (Cox, 2007). The goal then is to remove the cause of the problem, hence $g_n = \delta(\,) = \neg\varepsilon$. Goal formulation is thus a search for an explanation that operationalizes the problem in terms of root causes. Table 2 presents a procedure defining the goal-insertion transformation that formulates a goal and inserts it into the goal agenda. The details of what constitutes a salient antecedent is outside the scope of this paper. But see Cox (2013) and Cox et al. (2016) for an integrated cognitive system that implements goal insertion.

Consider again the blocks world example from Section 2.5. When told to put A on B, the system 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 the system expects it to stay that way.

Table 2: Goal-insertion function for adding new goal to current goal set $\hat{G}$.

Parameters: XP – explanation pattern; $\hat{\mathrm{G}}$ – goal agenda.

function *GDA-goal-insertion* (XP: χ; $\hat{G}$: S): S

1. XP: $\omega \rightarrow \cdots \rightarrow$ s
2. **if** $\forall \varepsilon | \varepsilon \in precond(XP) \wedge salient(\varepsilon)$
   then $\hat{\mathrm{G}} \leftarrow \hat{G} \cup \neg\varepsilon$ *;; insertion transformation, i.e., $\delta(\,)$ inserted into $\hat{G}$*
3. **return**$(\hat{G})$

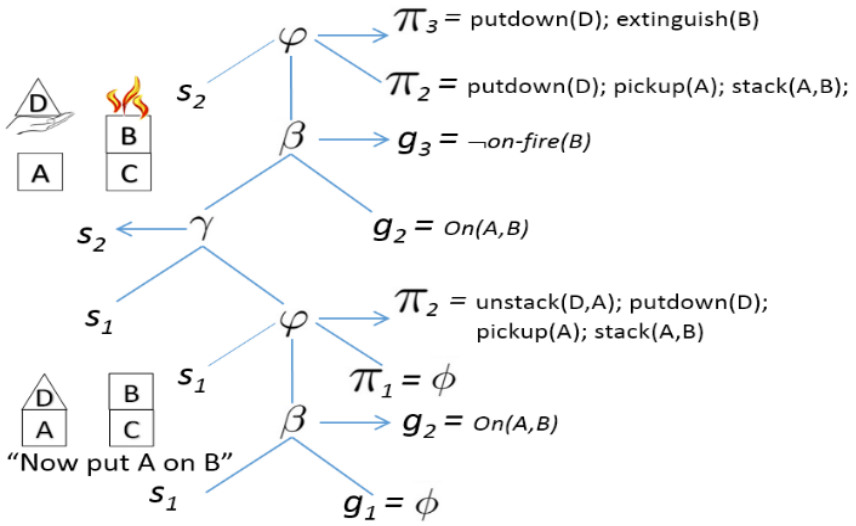

Figure 4. A structure representing processes during explanation-based goal formulation, read from bottom to top. During the execution of plan π_2 to put A on B, a fire breaks out. The interpretation function β returns the goal g_3 of not having B on fire, given the state s_2 and the old goal g_2.

Now assume that block B catches on fire and is no longer clear (see Figure 4 and Paisner et al., 2014, for an extended domain description for this scenario). 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 $\neg clear(B)$. A very simple explanation is that a block being on fire causes it to not be clear:

$$\chi: on\text{-}fire(y) \rightarrow \neg\, clear(y) \tag{9}$$

Given $\omega = \{on\text{-}fire(y)\}$ from (9) and the *substitution set* $\sigma = \{y \mapsto B\}$, the explanation becomes $on\text{-}fire(B) \rightarrow \neg\, clear(B)$. As a result of ω having only one element, the salient antecedent ε is trivially chosen as $on\text{-}fire(B)$. The new goal thus becomes $g_n = \neg on\text{-}fire(B)$, and it is added to the agenda. Hence $\hat{G} = \{g_n, g_2\}$. Planning can subsequently create a sequence of actions to put out the fire before continuing. The planning function represented by φ needs to be intelligent enough to infer that the putdown action is duplicated in both plans π_2 and π_3. However, for β to be equally intelligent, it should have been informed of the current plan. Therefore, we redefine (3) as follows:

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

In (3'), β 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, β 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 (using Δ) is the most rational choice. Planning is not only informed by goals from interpretation, but interpretation is also informed by planning. Alavi and Cox (2016) further discuss the influence planning can have upon interpretation.

Finally, given d, χ, and g_c, goal generation seeks to determine a new top-level goal or subgoal g_n to remove the discrepancy by either changing the state s_c to s_e or by learning a new expectation that justifies s_c. Given this approach, we define a GDA planning problem as the six-tuple in equation (10):

$$P_{gda} = (\Sigma, s_c, g_c, s_e, \hat{\mathrm{G}}, \Delta) \tag{10}$$

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

3.2 GDA and the Learning of Goals

The function β 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 the set of possible goal states along with their applicability conditions from the observed behavior of an execution system. The main role for planning then 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 (see also Pozanco, Fernandez, & Borrajo, 2016).

In the simplest sense, goal formulation can be cast as a novel classification task. An agent is given a state transition system Σ 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 positive and negative examples of the tuple (s, g). From these examples, it can then learn a mapping in the form of a decision tree with goals at the leaves (see Maynord et al., 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 hold in new situations and then using these explanations to learn appropriate goal orientations (i.e., to learn $G \subset S$, the set of useful goal states defined in Section 2.3). We do not limit ourselves to goal generation alone, because this would shift the burden of providing an external goal to 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.

3.2.1 The Package Delivery Example

Consider again the planning domain from the introductory 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 are then 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 en-

tailed by Σ and the observations, as in Figure 5(a). 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, which will give the causal explanation structure that is shown in Figure 5(b). For example, the predicate $at(Cola\text{-}5, Coke)$ holds true because the conjunct $has(Coke, \$) \wedge at(Bottles\text{-}2, Coke) \wedge at(Sugar\text{-}1, Coke)$ has been satisfied. If any term is unsatisfied, then the explanation structure collapses.

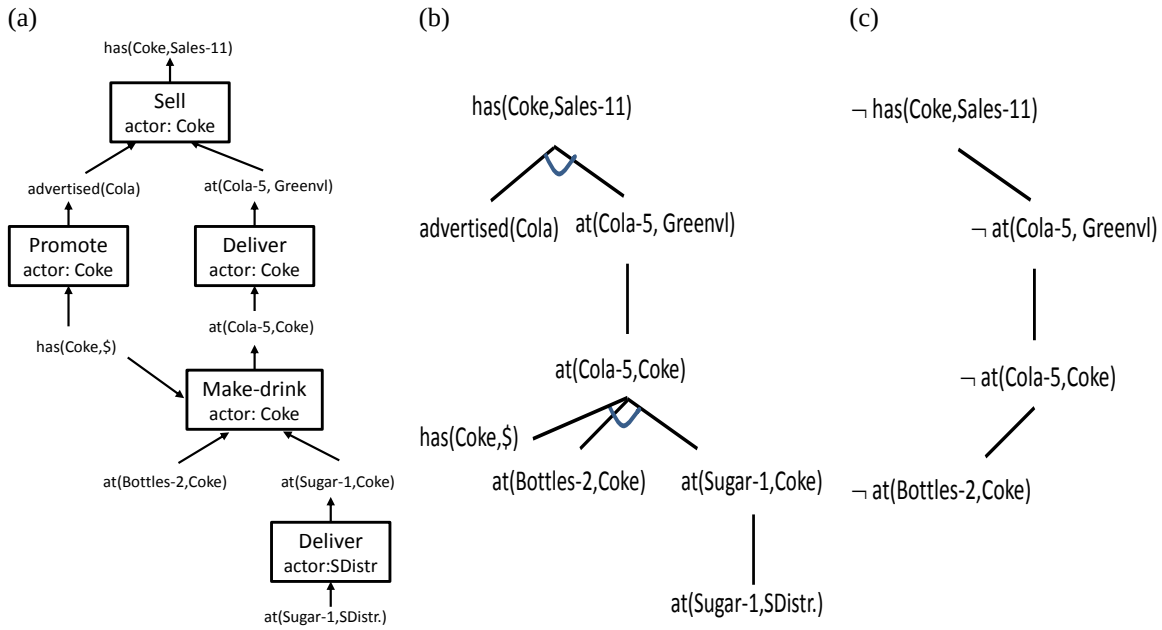

Figure 5. Inducing an explanation from the goal-subgoal structure inherent in the state transition system: (a) Goal-subgoal graph; (b) Causal explanation structure; (c) Failure causal chain.

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 function (Table 2) would detect the discrepancy between the expectation $has(Coke, sales\text{-}11)$ and the observation $\neg has(Coke, sales\text{-}11)$. An explanation process would then use the prior explanation (from Panel b) to derive a relational sequence responsible for the impasse. The resulting causal chain is shown in Figure 5(c).

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 $\hat{G}$. In Panel c, the salient causal antecedent ε of the explanation is $\neg at(Bottles\text{-}2, Coke)$, 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 11):

$$g_c = \neg\left(\neg at(Bottles\text{-}2, Coke)\right) = at(Bottles\text{-}2, Coke) \tag{11}$$

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 concern issues of goal generalization and of goal anticipation.

3.2.2 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, 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 (12) and to add $g_c \in G$.

$$g_c = at(b, m) \wedge bottles(b) \wedge cola\text{–}manufacturer(m) \tag{12}$$

3.2.3 Goal Anticipation

Second, the agent must *anticipate* the conditions under which the causal chain in Figure 5 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 s' of the state, s_c, that predicts when g_c is appropriate. That is, the system needs to acquire 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, $very\text{-}low$, and $out\text{-}of$. 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, after the agent attempts to deliver bottles to a manufacturer having a bottle surplus, it would learn to rule out the expression $surplus(m, b) \wedge bottles(b) \wedge cola\text{-}manufacturer(m)$. In this case, we assume that the $unload$ action would not receive permission at the destination. Essentially, the learned rule asserts the goal g_c when bottles at the manufacturer are low. This is what the introductory section meant when it stated that the agent should be able to explain that bottles need to be at the bottling plant *because the inventory is low*.

Although a goal state s_g would be a desirable state, 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 a much lower cost than s_g. In such a situation, it may be better for the GDA process to insert g' into the agenda rather than g. An open research question is how to learn which goals are worth formulating and which ones to commit to first.

4. Related Research

Many researchers have explored ideas similar to those presented here, especially in the recently constituted goal reasoning community. This body of work includes formalisms for goal life-cycles, frameworks of cognitive architectures, and the concept of goal-driven autonomy. Additionally, researchers have examined the mechanism of explanation as it applies to goal formulation.

Johnson et al. (2016) and Roberts et al. (2014, 2015) proposed an alternative formal model that treats goal reasoning as cyclical goal-refinement. Using an extension of the plan-refinement model of planning, they model goal reasoning as a 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 life cycle*.[6] Unlike the formalism here, which represents much of the goal reasoning process with the single function β, Roberts et al. propose a detailed cycle consisting of goal formulation, selection, expansion, commitment, dispatching, monitoring, evaluation, repair, and goal deferment. Thus, many of the differential functionalities in β are distinct and explicit in the goal reasoning cycle. However, the model here distinguishes between the planning and action aspects of reasoning (i.e., φ and γ) and the interpretation and evaluation components inherent in goal reasoning (i.e., β). We note, however, that φ relates to Roberts et al.'s goal expansion and repair, whereas β relates to goal formulation, monitoring, and deferment. Processes for goal dispatching, commitment, and evaluation are missing in the current analysis.

Additionally, Roberts et al. propose a more complex goal structure. A *goal node* includes not only the desired state but also superordinate and subordinate goal linkages, constraints, quality metrics, and pointers to the associated plan. We have been more circumspect regarding the syntactic structure of a goal. Section 2.1 defined the goal formula to be a conjunction of first-order literals, but goals have been expressed more generally as universally quantified predicate conjuncts (Cox & Veloso, 1998; Veloso, Pollack, & Cox, 1998). Thus, in a package delivery domain they allow $\forall p \mid (package\ p) \wedge \exists\ d | (destination\ p\ d) \wedge (location\ p\ d)$ 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 execution time as additional packages arrive at the warehouse for delivery. Talamadupula et al. (2010) have a similar concept of *open-world quantified goals*. Despite these differences, our framework has much in common with their approach, including an association with learning (Roberts & Aha, 2015).

The two problems of goal generalization and goal anticipation briefly mentioned in Section 3 make the learning task 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 maintain that autonomy includes the capacities to recognize novel problems and 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, top-down explanation of anomalies is a central pivot in our framework.

Our explanation-based approach to GDA and to learning should not be confused with the older, mainly deductive approach used in explanation-based learning (DeJong & Mooney, 1986; Mitchell,

[6] Harland et al. (2014) propose a goal life-cycle for attainment and maintenance goals. The latter, which we have not addressed, seek to maintain a state of the world and thus require monitoring to ascertain if the state departs from a desired range of values. Goals can therefore be pending, active, suspended, aborted, or (in the case of maintenance goals) monitored.

Keller, & Kedar-Cabelli, 1986). We use explanatory techniques (Cox, 2011; Cox & Ram, 1999; Roth-Berghofer, Tintarev, & Leake, 2011) that depart significantly from earlier work, in that they are more abductive, knowledge-rich, and case-based. Given a discrepancy and a context, the idea is to retrieve a graph structure called an explanation pattern, instantiate it, bind it to the discrepancy, and then 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 (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. Using knowledge structures called principles, goal formulation occurs as a response to discrepancy detection in the ARTUE system (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 et al., 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.

Finally, a number of existing cognitive architectures, such as Soar (Laird, 2012; Laird, Rosenbloom, & Newell, 1986) and ICARUS (Choi & Langley, 2018; Langley, Choi, & Rogers, 2009), include an accounting of goal formulation and goal selection. A typical approach is to enumerate all possible goals and the conditions under which they are triggered. Tac-Air Soar (Jones et al., 1999) uses this technique. Operators exist for various goal types and data-driven context-sensitive rules spawn them given matching run-time observations. A similar approach is used by ICARUS (see especially Choi, 2011) and its successor architecture PUG (Langley et al., 2017). Here, rules are defined for top-level goals and the conditions under which the environment triggers them. An extended version of the PUG architecture also lets an agent generate goals during plan execution and assign numeric utility to them. However, even if one could enumerate with operators or rules all the relevant goals for a domain and all the conditions under which they apply, an expectation failure or surprise may occur if the domain shifts (e.g., introduction of novel technology). The general capacity to recognize new problems and explain their causes enables the formulation of goals from first principles, even under conditions not necessarily envisioned by an agent designer.

5. Conclusion

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

Much of this theoretical work does have a specific instantiation in computational implementations, but we have focused on the formalism and the general framework rather than empirical comparisons or analyses. Instead, the theory provides a conceptual vision of high-level cognition and deliberate choice firmly rooted in the goals an agent seeks rather than the correlations it experiences. The framework provides a uniform language for describing how goals originate and change, and understanding the role goal operations serve within an agent. The framework also

situates important aspects of goal-based reasoning within a distinct interpretation process that is separate from planning and action. Finally, the formalism we propose integrates all three cognitive processes within a single recursive expression.

Details about how goal management occurs within the interpretation process and the β function have not been discussed in this paper. For example, the selection procedure that chooses a specific δ from Δ and representations for goal transformations (other than the basic identity function) remain for future publication. Rather, the purpose of this paper has been to enumerate a set of principles of goal reasoning that have a consensus within the cognitive systems community but use the formal notations and representations of the larger AI community in their expression. The hope is to bridge some of the differences in style and method and to encourage synthesis and reflection.

Acknowledgements

This material is based upon work supported by AFOSR Grant FA2386-17-1-4063, as well as ONR Grants N00014-15-C-0077 and N00014-15-1-2080. I also wish to acknowledge the input of David Aha, Pat Langley, Michael Maynord, Dana Nau, Don Perlis, and Mark Wilson on an earlier formulation of these ideas. The suggestions and comments of the anonymous reviewers have been especially valuable.

References

Aha, D. W., Cox, M. T., & Muñoz-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, D. W., Klenk, M., Muñoz-Avila, H., Ram, A., & Shapiro, D. (Eds.) (2010). *Goal-directed 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. *Proceedings of the European Space Agency Workshop on On-Board Autonomy*, Noordwijk, NL: ESA.

Alavi, Z., & Cox, M. T. (2016). Rationale-based visual planning monitors. *Working Notes of the Fourth Workshop on Goal Reasoning*. New York: IJCAI.

Bengfort, B., & Cox, M. T. (2015). Interactive reasoning to solve knowledge goals. *Goal Reasoning: Papers from the ACS Workshop* (pp. 10–25). Atlanta, GA: Georgia Institute of Technology.

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

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

Choi, D. (2011). Reactive goal management in a cognitive architecture. *Cognitive Systems Research, 12,* 293–308.

Choi, D., & Langley, P. (2018). Evolution of the ICARUS cognitive architecture. *Cognitive Systems Research, 48,* 25–38.

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

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

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

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

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

Cox, M. T., & Dannenhauer, D. (2016). Goal transformation and goal reasoning. *Working Notes of the Fourth Workshop on Goal Reasoning*. New York: IJCAI.

Cox, M. T., & Dannenhauer, Z. A. (2017). Perceptual goal monitors for cognitive agents in changing environments. *Poster Collection of the Fifth Annual Conference on Advances in Cognitive Systems* (pp. 1–16). Palo Alto, CA: Cognitive Systems Foundation.

Cox, M. T., Dannenhauer, D., & Kondrakunta, S. (2017). Goal operations for cognitive systems. *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence* (pp. 4385–4391). Palo Alto, CA: AAAI Press.

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. *Proceedings of the AAAI Fall Symposium on Distributed Continual Planning* (pp. 23–30). Menlo Park, CA: AAAI Press.

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

Dannenhauer, D., & Muñoz-Avila, H. (2015) Raising expectations in GDA agents acting in dynamic environments. *Proceedings of the Twenty-Fourth International Conference on Artificial Intelligence* (pp. 2241–2247). Menlo Park, CA: IJCAI.

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

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

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

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

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

Eyorokon, V., Panjala, U., & Cox, M. T. (2017). Case-based goal trajectories for knowledge investigations. *Proceedings of the Thirtieth International FLAIRS Conference* (pp. 477–482). Palo Alto, CA: AAAI Press.

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

Harland, J., Morley, D. N., Thangarajah, J., & Yorke-Smith, N. (2017). Aborting, suspending, and resuming goals and plans in BDI agents. *Autonomous Agents and Multi-Agent Systems, 31*, 288–331.

Harland, J., Morley, D. N., Thangarajah, J., & Yorke-Smith, N. (2014). An operational semantics for the goal life-cycle in BDI agents. *Autonomous Agents and Multi-Agent Systems, 28*, 682–719.

Hawes, N. (2011). A survey of motivation frameworks for intelligent systems. *Artificial Intelligence, 175*, 1020–1036.

Johnson, B., Roberts, M., Apker, T., & Aha, D. (2016). Goal reasoning with information measures. *Proceedings of the Fourth Annual Conference on Advances in Cognitive Systems.* Palo Alto, CA: Cognitive Systems Foundation.

Jones, R. M., Laird, J. E., Nielsen, P. E., Coulter, K. J., Kenny, P., & Koss, F. V. (1999). Automated intelligent pilots for combat flight simulation. *AI Magazine, 20*, 27–41.

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1995). Partially observable Markov decision processes for artificial intelligence. *Proceedings of the International Workshop on Reasoning with Uncertainty in Robotics* (pp. 146–163). Berlin: Springer.

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

Kondrakunta, S., & Cox, M. T. (2017). Autonomous goal selection operations for agent-based architectures. *Working Notes of the Fifth Workshop on Goal Reasoning*. Melbourne: IJCAI.

Laird, J. E. (2012). *The Soar cognitive architecture*. Cambridge, MA: MIT Press.

Laird, J. E., Rosenbloom, P. S. & Newell, A. (1986). *Universal subgoaling and chunking: The automatic generation and learning of goal hierarchies*. Hingham, MA: Kluwer.

Langley, P., Choi, D., Barley, M., Meadows, B., & Katz, E. P. (2017). Generating, executing, and monitoring plans with goal-based utilities in continuous domains. *Proceedings of the Fifth Annual Conference on Advances in Cognitive Systems*. Troy, NY: Cognitive Systems Foundation.

Langley, P., Choi, D., & Rogers, S. (2009). Acquisition of hierarchical reactive skills in a unified cognitive architecture. *Cognitive Systems Research, 10*, 316–332.

Maynord, M., Cox, M. T., Paisner, M., & Perlis, D. (2013). Data-driven goal generation for integrated cognitive systems. *Integrated Cognition: Papers from the AAAI Fall Symposium* (pp. 47–54). Menlo Park, CA: AAAI Press.

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

Muñoz-Avila, H., Jaidee, U., Aha, D. W., Carter, E. (2010). Goal-driven autonomy with case-based reasoning. *Proceedings of the Eighteenth International Conference on Case-Based Reasoning* (pp. 228–241). Berlin: Springer.

Molineaux, M., Klenk, M., & Aha, D. (2010). Goal-driven autonomy in a navy training simulation. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence* (pp. 1548–1554). Menlo Park, CA: AAAI Press.

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

Paisner, M., Maynord, M., Cox, M. T., & Perlis, D. (2013). Goal-driven autonomy in dynamic environments. *Goal Reasoning: Papers from the ACS Workshop* (pp. 79–94). Baltimore, MD: University of Maryland Institute for Advanced Computer Studies.

Powell, J., Molineaux, M., & Aha, D. W. (2011). Active and interactive discovery of goal selection knowledge. *Proceedings of the Twenty-Fourth International Florida Artificial Intelligence Research Society Conference* (pp. 413–418). Menlo Park, CA: AAAI Press.

Pozanco, A., Fernández, S., & Borrajo, D. (2016). On learning planning goals for traffic control. *Working Notes of the Fourth Workshop on Goal Reasoning*. New York: IJCAI.

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

Roberts, M., & Aha, D. W. (2015). Cleaning efficiently: A case study in modeling goal reasoning and learning. *Goal Reasoning: Papers from the ACS Workshop* (pp. 146–155). Atlanta, GA: Georgia Institute of Technology.

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. *Planning and Robotics: Papers from the ICAPS Workshop*. Jerusalem, Israel: AAAI Press.

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. *Planning and Robotics: Papers from the ICAPS Workshop*. Portsmouth, NH: AAAI Press.

Roth-Berghofer, T., Tintarev, N., & Leake, D. B. (Eds.) (2011). *Proceedings of the Sixth International Workshop on Explanation-aware Computing.* Barcelona: IJCAI.

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*, 14:1–14:24.

Talamadupula, K., Benton, J., Schermerhorn, P., Kambhampati, S., & Scheutz, M. (2009). Integrating a closed world planner with an open world robot: A case study. *Papers from the ICAPS Workshop on Bridging the Gap Between Task and Motion Planning*. Thessaloniki, Greece: AAAI Press.

Vattam, S., Klenk, M., Molineaux, M., & Aha, D. (2013). Breadth of approaches to goal reasoning: A research survey. *Goal Reasoning: Papers from the ACS Workshop* (pp. 111–126). College Park, MD: University of Maryland.

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. *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*, 250–271. Palo Alto, CA: Tioga.

Weber, B. G., Mateas, M., & Jhala, A. (2010). Applying goal-driven autonomy to StarCraft. *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* (pp. 101–106). Menlo Park, CA: AAAI Press.