Is Everything Going According to Plan? Expectations in Goal Reasoning Agents

Héctor Muñoz-Avila,[1] **Dustin Dannenhauer,**[2] **Noah Reifsnyder**[1]
[1]Lehigh University, [2]Naval Research Laboratory

Abstract

In part motivated by topics such as agency safety, there is an increasing interest in goal reasoning, a form of agency where the agents formulate their own goals. One of the crucial aspects of goal reasoning agents is their ability to detect if the execution of their courses of actions meet their own expectations. We present a taxonomy of different forms of expectations as used by goal reasoning agents when monitoring their own execution. We summarize and contrast the current understanding of how to define and check expectations based on different knowledge sources used. We also identify gaps in our understanding of expectations.

Introduction

Over the past few years there has been an increasing interest in safety for autonomous systems; ensuring that the agent behaves in the way that was intended (Conn 2017). Part of the interest on AI safety is the realization that as systems increase in sophistication they may behave in ways that are counterproductive towards the agent's designer's intent.

Motivated in part by these concerns, there is an increasing interest in goal reasoning, a form of agency where the system formulates its own goals (Aha 2018). For example, agents formulate goals as a reaction to conditions in the environment where the agent is operating. This can include anomalies such as unexpected exogenous events that the agent has not planned for. Cox (1996) presents a taxonomy of potential failures an agent might encounter; the taxonomy identifies four categories of failures: domain knowledge, goal, processes and environmental. This work focuses on two of these:

- *Environmental*. Consider for instance, an autonomous vehicle that is navigating in an environment and the execution of a move action produces no results because the vehicle is stuck in mud.
- *Domain knowledge*. Same example as before, but in this case the vehicle is not moving due to a mechanical failure, which causes a mismatch between the action model and the physical model of the vehicle.

Goal reasoning architectures have been proposed to address the problem of (1) identifying the reasons for the discrepancy, including generating explicit goals to ascertain the reason for the discrepancy, and (2) take the corresponding actions to deal with this discrepancy (Cox 2007; Molineaux, Klenk, and Aha 2010a). For instance, the vehicle may self-assess that it is stuck (e.g., by mud), backtrack to move out of the mud and then replan to continue to its original destination (e.g., avoiding the mud). The question of what the agent should expect seems deceivingly simple: at first glance it would seem sufficient to expect that the effects of the agent's most recently executed action matches the observed changes in the state. As it turns out the answer to the question of what are the agent's expectations while executing its course of action (e.g., a plan) is more complex. For instance, not every change in the state is relevant for the current plan and hence merits attention from the agent.

Table 1: A taxonomy of different notions of expectations.

	STRIPS	Ontol.	HTN	Numeric	ND
Immediate	✓	✓	✓	✓	✓
State	✓	✓	✓	✓	✓
Informed	✓	?	✓	?	✓
Regression	✓	?	?	?	✓
Goldilocks	✓	?	?	?	✓

In this paper we present a taxonomy answering this question in the context of goal reasoning agents. We present six forms of expectations from the goal reasoning literature (Table 1 lists five of these) and contrasts them with the following assumptions about the underlying domain model:

- Actions are defined as (STRIPS) *(name, preconditions, effect)* triples.
- Ontological information about the objects in the environment is provided.
- Hierarchical task network methods are used to generate the plans.
- Arguments in the actions may be numerical.
- Actions having non-deterministic effects.

We also discuss gaps in the formulation of expectation for goal reasoning systems that can be the subject of future research.

Related Work

The study of expectation failures has a long-standing tradition. Mechanisms to enhance a domain description when planning failures occur have been proposed (e.g., (Birnbaum et al. 1990; Sussman 1975)). Plan execution monitoring systems check if the current state satisfies the effects of the action just executed and the preconditions of the actions to be executed next. When this does not happen, a so-called expectation failure or discrepancy occurs (Cox 2007). Wilkins (1985) presents possible execution failures including: goals not achieved and action's preconditions held to be true at planning time that are no longer true during execution. Interestingly, for some of the failures, the agent will generate new goals. The agent is built using the highly expressive SIPE HTN planning and execution system (Wilkins 1988). Other kinds of failures have been studied where even though they are not execution failures, the current execution exhibits conditions that are not desirable (Myers 1999). For instance, when the execution of the plan doesn't meet quality considerations, such as execution taking longer than expected (Fritz and McIlraith 2007).

Some planning systems relax the requirement that the plan must achieve all of its goals. For example, oversubscription planners attempt to satisfy a maximal subset of the goals instead of all of the goals (Van Den Briel et al. 2004). In goal reasoning, goals might change as a reaction to changes in the environment. A primary difference from oversubscription planning vs goal reasoning is that in goal reasoning agents there is a notion that some goals are committed and others may be pending. A pending goal indicates one that is not currently pursued by planning and acting, instead it is a goal the agent may pursue at a later time. The decision to move a goal between pending or committed is considered a goal reasoning problem (Roberts et al. 2016). This is different from oversubscription planning which does not have a notion of pending goals, only a given set of goals for which the planner attempts to generate a plan to achieve a maximal subset. Since goal reasoning may cause a change in the current goals, it is related to plan repair, which aims at modifying the current plan when changes in the environment make actions in the plan invalid (Van Der Krogt and De Weerdt 2005). The main difference between plan repair and goal reasoning is that in the latter the goals might change whereas plan repair sticks with the same goals while searching for alternative plans.

Immediate Expectations

Immediate expectations directly borrow ideas of the plan execution monitoring literature (see related work discussion). Given a domain planning model consisting of a collection of actions $\mathcal{A}$, where each action a is defined as a *(name, preconditions, effect)* triple, $a = (name^a, pre^a, \mathit{eff}^a)$, and plans are generated from $\mathcal{A}$ (Fikes and Nilsson 1971). That is, each $a_k \in \mathcal{A}$. Suppose that a_i in a plan $\pi = (a_1 ... a_n)$ is the next action to be executed. Immediate expectations check that the effects of the previously executed action a_{i-1} currently hold in the environment and the preconditions of a_i also currently hold in the environment (Cox 2007).

$$Player(p1) \wedge Player(p2) \wedge DifferentFrom(p1, p2) \wedge Region(r) \wedge hasPresenceIn(p1, r) \wedge hasNoPresenceIn(p2, r) \rightarrow ControlledRegion(r)$$

Figure 1: Inference Rule for *ControlledRegion(?r)*

If an ontology Ω is available, immediate expectations can be defined to check if $(\Omega \cup s_i \cup \mathit{eff}^{a_{i-1}})$ (respectively $(\Omega \cup s \cup pre^{a_i})$) is consistent, where s is the observed state. The ontology Ω can be used to define richer preconditions or effects; conditions that may not be directly observable but that are inferred to be true by reasoning over the ontology and the current state. For example, in a real-time strategy game setting, using an ontology[1] we define a richer precondition *ControlledRegion*(r) to be a region controlled by the player $p1$ if $p1$ has a presence in the region but the other player $p2$ has no such a presence (Figure 1). This rule is defined for an ontology of a game-playing agent in a strategy game where a player manages armies aiming at defeating the opponent. Rules such as the above are used to plan a strategy. In this case an action can be taken under the precondition that some region is controlled by $p1$; a discrepancy will be detected if $hasPresenceIn(p2, r)$ can be inferred from the current state. When this happens a goal reasoning process is triggered (Dannenhauer and Munoz-Avila 2013).

In HTN planning, plans are generated by a hierarchical task decomposition process in which higher level tasks are recursively decomposed into simpler tasks (Currie and Tate 1991; Erol, Hendler, and Nau 1994).[2] The process continues until so-called primitive tasks are generated. These primitive tasks are defined by actions. Hence, the result of the HTN planning process is a plan π and immediate expectations can be defined as before. However, the immediate expectations concept can be extended to the applicability conditions of each of the tasks decomposed in the echelons of the hierarchy H_π that resulted in π. Specifically, in HTN planning, in addition to the actions, a collection of methods $\mathcal{M}$ is provided. A method m is a *(task, preconditions, subtasks)* triple, $m = (task^m, pre^m, subts^m)$. Examining H_π, we can determine for each action a_i if it was the first primitive task resulting from applying a method m decomposing a task $t \in H_\pi$. Hence, we can check if pre^m is valid in the state s_i when checking if pre^{a_i} is valid. Recursively, we can check if t is the first subtask of another method m' decomposing some task $t' \in H_\pi$ and then also check if $pre^{m'}$ is valid in s_i. At higher levels in the hierarchy, the preconditions will have a higher level of granularity than in the action level. For example, in our work combining HTN and ontologies (Dannenhauer and Munoz-Avila 2013; 2015a), the actions check preconditions about units such as

[1] This ontology is available in OWL format, see Dannenhauer, West, and Hatalis (2013).

[2] In the discussions on this paper we focus on Simple Ordered Task Decomposition as defined for the SHOP 2 HTN planner (Nau et al. 2003). This is the current dominant variant of HTN planning used by those goal reasoning systems that are built on top of an HTN planner.

presence of units in regions whereas some methods have preconditions about the control of a region. This means that in some situations the expectation of an action a can be satisfied in the observed state but the expectations for a method decomposing one of its ancestor tasks in H_t is not valid. For instance, the unit might move out of a region because it has presence but a parent task's expectations are violated because the region is no longer controlled by the player. Thus, an agent checking task's expectations could react to violations that an agent checking only actions' expectations would not. The anomaly will be detected when checking expectations of a latter action in π but by that time the agent would have committed to actions that could have been avoided.

Numeric conditions can be checked to see if numeric conditions in the actions are valid in the state. For example, in the minecraft domain, there is an action to build a pickaxe that requires three stones (Nguyen et al. 2017). It can be checked directly in the state if the action is applicable.

In nondeterministic (ND) domains, an ND action a is defined as a triple $a = (name^a, pre^a, \{\mathit{eff}^a\})$ where pre^a are the preconditions, $\{\mathit{eff}^a\} = \{\mathit{eff}_1^a \dots \mathit{eff}_n^a\}$ are the possible effects of the action. ND planners generate a policy $\pi : S \rightarrow \mathcal{A}$, a mapping from the possible states in the world S to actions $\mathcal{A}$, indicating for any given state $s \in S$, what action $\pi(s) \in \mathcal{A}$ to take (Fu et al. 2011). When executing a policy π from a starting state, an action trace $(\pi(s_0)...\pi(s_{i-1}))$ is generated. For immediate expectations the agent checks that the conditions of the state match any of the ND effects in $\pi(s_{i-1}))$ and check if the preconditions in $\pi(s_i)$ are valid (Reifsnyder and Munoz-Avila 2018).

State Expectations

Some agents propagate forward the state by using the action model $\mathcal{A}$ enabling the agent to maintain the plan trace, $s_0\ a_1\ s_1\ a_2\ \dots\ a_n\ s_n$ (Molineaux, Klenk, and Aha 2010a). That is, the actions in π are annotated with the intermediate state after executing that action. Each state is a collection of atoms. The effects eff^a of an action a are defined as a pair (add^a, del^a) of atom collections, where add^a is the add-list and del^a is the delete list. They indicate the resulting state s' when a is applied to state s: $s' = apply(s, a) = (s - del^a) \cup add^a$. Thus, in the plan trace, $s_i = apply(s_{i-1}, a_i)$ holds. We called these state expectations; they are defined as the state s_i generated by the last action a_i executed.

A discrepancy occurs if $s \neq s_i$, where s is the observed state. Empirical evaluations have consistently shown an improvement in performance defined as the agent satisfying the goal conditions when the agent terminates execution using state expectations compared to immediate expectations. Unlike agents checking immediate expectations, agents checking state expectations guarantee that the goals are satisfied when the agent reaches a terminal state (and stops its execution).

State expectations generalize immediate expectations in the sense that if the state expectation for an action a in a plan π are valid in a state s then the immediate expectation of a is valid. But the opposite is not true. For instance, suppose that an agent is tasked with turning on some beacons. Suppose that the agent turns on a beacon at some action a_i but the beacon is turned off by some opponent agent afterwards. An agent checking state expectations will discover this. In contrast, an agent checking action expectations will not detect this anomaly.

State expectations are perhaps the most frequently used in the goal reasoning literature. For instance, in the introductory articles of goal-driven autonomy, a form of goal reasoning, the expectations defined are state expectations (Klenk, Molineaux, and Aha 2013). Other goal reasoning systems frequently use state expectations as well. For example, (Pozanco, Yolanda, and Borrajo 2018) project the goal states that can be reached by an opponent and formulate goals to impede the opponent from reaching those states.

If an ontology Ω is available, state expectations can be checked by inferring if $(\Omega \cup s \cup s_i)$ is consistent, where s is the observed state and s_i is the expected state (Dannenhauer and Munoz-Avila 2013). The difference versus the corresponding case for immediate expectations is that the agent checks consistency on the projected state s_i as opposed to the effects $\mathit{eff}^{a_{i-1}}$ or the preconditions pre^{a_i} in immediate expectations.

For HTN planning, the intermediate states can be propagated upwards into echelons of the hierarchy H_π that resulted in π (Dannenhauer and Munoz-Avila 2015b). Specifically, a_i if it is the last primitive task resulting from applying a method m decomposing a task $t \in H_\pi$. Then we define the state expectation of t to be s_i. Analogously, we can define the state expectation for a task $t' \in H_\pi$ if t is the last subtask in the method decomposing t'. Unlike with immediate expectations, for state expectations there is no benefit of checking (non primitive) tasks' expectations since they are identical to state expectations in actions in π.

For states with numeric information, bounded expectations have been defined, which is a form of state expectations as the state is carried over (Karneeb et al. 2016; Floyd et al. 2017; Wilson, McMahon, and Aha 2014). They project forward the expected numerical values: for example, if an action consumes $f(d)$ gasoline for a numeric value d having a distance between locations, then the current fuel level v will be decreased to $v - f(d)$. These agents maintain an interval $[l, u]$ by projecting it forward: $[l - f(d), u - f(d)]$. A discrepancy occurs if the observed value in the state is out of the expected bounds.

For ND domains, where as explained before planners generate policies $\pi : S \rightarrow \mathcal{A}$, the state expectation is computed by keeping track of the action trace, $s_0\ \pi(s_0)\ s_1\ \pi(s_1)\ \dots\ \pi(s_n)\ s_{n+1}$, resulting from executing π from state s_0 (Reifsnyder and Munoz-Avila 2018).

Informed Expectations

Informed expectations accumulate forward the effects $\emptyset\ a_1\ \mathit{eff}_1\ a_2\ \dots\ a_n\ \mathit{eff}_n$, where $\mathit{eff}_i = apply(\mathit{eff}_{i-1}, a_i)$ holds (Dannenhauer, Munoz-Avila, and Cox 2016). The difference versus state expectations is that it maintains the accumulated effects (removing those deleted by actions in the trace) as opposed to state expectations that project forward the starting state s_0. In particular the state expectation of a_1 is $s_1 = apply(s_0, a_1)$ whereas the informed expectation of

a_1 is $\mathit{eff}_1 = pos^{a_1}$. Thus, if the state expectation for an action a_i are met in the observed state s, then the informed expectation for a_i are also met; but the other way around is not true.

Like state expectations, informed expectations guarantee that the goals are reached when the agent reaches a terminal state (unless the goal was already achieved in the starting state). However, state expectations may trigger unnecessary replanning and/or goal reasoning processes. For instance, when navigating in the environment the agent may encounter unexpected mud resulting in a discrepancy. However, if the mud is not along the trajectory of the agent, there is no need to change its course of action. In addition, if sensing the environment has associated costs, then checking for state expectations will have larger costs compared to informed expectations as the latter is a subset of the former. Indeed in empirical evaluations agents checking for informed expectations incurred less sensing costs than those checking state expectations while satisfying all the goals in the terminal states.

We know of no work performing informed expectations given an ontology Ω. Presumably the expectation for action a_i could be defined as $(\Omega \cup s \cup \mathit{eff}_i)$ (i.e., a discrepancy occurs if this set is inconsistent).

Like with state expectations, it suffices to compute informed expectations on a_i; if a_i is the last primitive task resulting from applying a method m decomposing a task $t \in H_\pi$, then the informed expectations of i are the same as the informed expectations of a_i (Dannenhauer and Munoz-Avila 2015b). We don't know of work performing informed expectations on numeric values and how to define these is unclear. Unlike state expectations where a variable will have a numeric value in s_0 and this can be propagated forward based on the action definitions (e.g., reducing the fuel level when gasoline is consumed), in informed expectations, as defined, the values of s_0 are not taken into account; only the projected changes. With list of atoms with symbolic values (i.e., when using the state-variable action representation) it is simple to project values forward by adding/removing atoms or keeping track of the newly assigned symbolic values. But when the values are numeric. it is not clear what to project forward.

For ND domains, when applying an action a_i we know what are its potential effects $\{\mathit{eff}^{a_i}\} = \{\mathit{eff}_1^{a_i} \dots \mathit{eff}_n^{a_i}\}$ so we can check against the current state which $\mathit{eff}_k^{a_i}$ of these hold (otherwise a discrepancy is detected). Thus, the informed expectations are maintained by accumulating effects of the action trace generated so far when following policy π from s_0: $\emptyset$ $\pi(s_0)$ eff_1 $\pi(s_1)$ $\dots$ $\pi(s_n)$ eff_n and $\mathit{eff}_i = \mathit{apply}(\mathit{eff}_{i-1}, \mathit{eff}_k^{a_i})$, where $\mathit{eff}_k^{a_i}$ are the ND effects of a_i that hold in the environment (Reifsnyder and Munoz-Avila 2018).

Goal Regression Expectations

The use of goal regression has been investigated to determine the weakest preconditions needed to execute a plan and use these to compute similarity between plans in case-based planning (Veloso and Carbonell 1993). Goal regression has also been used to avoid unnecessary replanning in the context of optimal planning (Fritz and McIlraith 2007). Goal regression expectations borrow directly from these works.

Given a plan $\pi = (a_1 ... a_n)$ achieving g, a collection of atoms, the goal regression expectation of an action $a_i \in \pi$ is its regress condition on $(reg_0\ a_1\ reg_1 ... reg_{n-1}\ a_n\ reg_n)$, such that $reg_{i-1} = regression(a_i, reg_i) = (reg_i - add^{a_i}) \cup pre^{a_i}$ and $reg_n = g$.

If the regression expectations reg_{i-1} for action a_i are met in the observed state then the remaining actions $\pi_i = a_i ... a_n$ in π are executable. Furthermore, reg_{i-1} is the smallest set of conditions that must be satisfied in the environment to guarantee the execution of π_i; that is, if we remove an atom from reg_{i-1}, the preconditions pre^{a_j} of an action in π_i will not be applicable or a goal condition will not be achieved. Goal regression is a powerful form of expectations when the goals are known and a plan achieving those goals has been generated a priori. This might not be the case for agents interleaving planning and execution. Another possibility is when the plans are generated using HTN planning. In that case, tasks, not goals are given as input. If the tasks have not been defined in terms of goals, then it is not possible to ascertain which conditions in the final state s_n are the desired conditions satisfying the tasks. In such cases informed expectations can be used checking that the portion of the plan executed so far meet the expected accumulated goals (Dannenhauer and Munoz-Avila 2015b).

We know of no work performing regression expectations given an ontology Ω. A possible definition for the regression expectation for action a_i could be defined as $(\Omega \cup s \cup reg_i)$, where s is the observed state. That is, a discrepancy occurs if this set is inconsistent. We also know of no work on goal reasoning involving goal regression of numerical values. Prediction techniques such as the ones used for machine learning regression might be of use here since these techniques predict future numerical values.

We also don't know of work defining goal regression expectations for goal reasoning agents using HTN planning. The primary obstacle is that in HTN planning, tasks do not have explicit semantics other than the methods that decompose them. While one could regress the conditions over tasks in H_π using the preconditions of the methods, methods don't have effects. If only preconditions are propagated backwards, without effects, it will lead to inflated regression expectations on the tasks: a discrepancy may be detected for tasks' regression expectations, but no such discrepancy exists for the regression expectations of the actual actions in the plan π. Therefore, it may trigger unnecessarily a goal reasoning process, trying to formulate new goals, for anomalies that do not actually exists. The problem of determining if no new goals need to be formulated when a discrepancy is detected is an open problem in goal reasoning research (Kondrakunta et al. 2018). Goal regression has been defined for tasks when learning HTN planning knowledge in (Hogg, Muñoz-Avila, and Kuter 2008), but in that work tasks are defined as *(preconditions,effects)* pairs as proposed in (Murdock 2001).

Goal regression over nondeterministic domains pose an interesting challenge since (1) there might be multiple paths in a policy π that yield a terminal state and (2) there might

be multiple terminal states. (Reifsnyder and Munoz-Avila 2018) provides a definition. We avoid the details but there are three basic cases:

- If the state is a terminal node then the regressed expectations for that state is g, the collection of achieved goals.
- Suppose that in a state s, when applying the action $\pi(s)$ it yields a single state s' and we have inductively defined its regression expectation, $reg_{s'}$. In that case the regression expectation for s' is computed exactly as in the deterministic case.
- Suppose that in a state s, when applying the action $\pi(s)$ indicated by the policy, it can yield two different states s' and s''. Suppose that, inductively, we have defined their regression expectations $reg_{s'}$ and $reg_{s''}$. Suppose that there is an atom α that occurs at both $reg_{s'}$ and $reg_{s''}$ and another atom β that occurs in $reg_{s'}$ but not $reg_{s''}$. In this case, and assuming α is not added by an effect of $\pi(s)$, then $(\alpha, 1.0) \in reg_{s'}$. The 1.0 signifies that that there is a 100% likelihood that α is required after applying $\pi(s)$. Again assuming β is not added by an effect of $\pi(s)$, then $(\beta, 0.5) \in reg_s$. The 0.5 signifies that there is a likely hood of 50% that β is required. A discrepancy is detected if an atom having at least 50% likelihood in the regressed expectations is not matched in the state. This means, assuming an equiprobable distribution of the ND action's effects that there is at least 50% chance that the execution of the policy will fail after executing $\pi(s)$.

In the conditions above we informally defined the regressed expectations for a state s; the regressed expectation for an action $\pi(s)$ is the regressed expectation for s.

Goldilocks Expectations

Goldilocks expectations combine informed and goal regression expectations. They are designed for situations when the plan π is known but the goals satisfied by π are not known (Reifsnyder and Munoz-Avila 2018). As explained in the previous section, plans generated with HTN planning are an example of such situations. Goldilocks expectations are computed in two steps:

1. The informed expectations are computed as discussed before; namely, the effects are accumulated over π: $\emptyset \ a_1 \ \mathit{eff}_1 \ a_2 \ \ldots \ a_n \ \mathit{eff}_n$, where each $\mathit{eff}_i = apply(\mathit{eff}_{i-1}, a_i)$.
2. Goal regression is performed but instead of regressing from the goals g, we regress from the informed expectations of a_n, eff_n, as follows: $(gold_0 \ a_1 \ gold_1 ... gold_{n-1} \ a_n \ gold_n)$, where $gold_n = \mathit{eff}_n$ and $gold_{i-1} = regression(a_i, gold_i)$.

When Goldilocks expectations are satisfied in a_i then the remaining of the plan π_i is satisfied. They are a superset of the regression expectations.

We don't know of goal reasoning agents combining Goldilocks expectations and ontologies. Although it is possible that given an ontology Ω, to define the Goldilocks expectations of action a_i as $(\Omega \cup s \cup gold_{i-1})$ where s is the observed state. We also don't know of any work combining Goldilocks expectations with numeric variables.

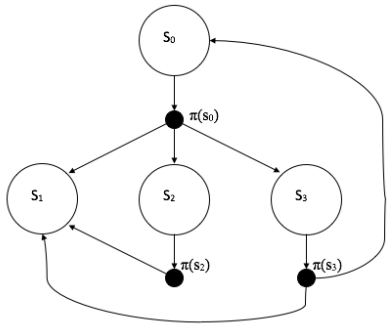

Figure 2: A graph representation G_π of a policy π. s_0 is the starting state. s_1 is the only terminal state. The vertices are the states $s_0, .., s_3$ and their corresponding actions $\pi(s_0), .., \pi(s_3)$. An example of an edge is: $(\pi(s_0), s_2)$

We also don't know of any work defining Goldilocks expectations for HTNs. While, we can compute informed expectations for any task $t \in H_\pi$ as discussed earlier. The issue remains on how to do regression unless either we ignore the fact that tasks have no effects or we assume the effects of the tasks are given.

Defining Goldilocks expectations for policies requires to extend the notion of informed expectations. Previously, we defined informed expectations by projecting forward the accumulated effects over the trace resulting from executing the policy π from s_0. For computing the informed expectations of a policy π, the policy π is viewed as a graph G_π, where the states and actions in π are the vertices and transitions from states to actions and from actions to states are the edges. Figure 2 shows an example. Then, the following steps are performed:

1. Classical depth-first search (DFS), with s_0 as the source, is used to transform G_π into a tree T_π (called the search tree in the terminology of DFS search). T_π is annotated with the edges in G_π that are not explicitly represented in T_π (Cormen et al. 2001).
2. All paths in T_π from s_0 to leaves, taking into account the annotated edges, are computed. Each leaf in T_π is a terminal state in π.
3. Informed expectations are computed on each path computed in Step 2.

These steps result in the leaves of T_π annotated with the informed expectations. The same procedure described before to compute goal regression on the policies is used to compute goal regression on T_π, which are defined as the Goldilocks expectations for T_π. So regression is performed in the informed expectations of the leaf vertices. Since every action in π may occur more than once, the Goldilocks expectation for each action a in π is defined as the Goldilocks expectation for the first time a appears in T_π during the DFS search procedure. This is done so that it regresses the most preconditions compared to other (later) occurences of a.

Expectation Header:
$EX^M_{D \rightarrow E} : EX^M(s_i^M, Explanation, s_{i+1}^M)$

Informal Description: *When the agent encounters a discrepancy, then the agent will generate an explanation for this discrepancy*

Formal Description: If $v_{discrepancies} \neq \emptyset$ in s_i^M and $v_{explanations} \neq \emptyset$ in s_{i+1}^M, return true, otherwise return false.

Figure 3: Metacognitive Expectation Definition for the *Explanation* Mental Action

Metacognitive Expectations

The expectations we have discussed thus far are concerned with the external environment in which the agent is operating (Table 1). However in certain situations, failure may lie within an agent's own cognitive processes rather than in the external environment. For example, consider an agent operating in an environment where the model of one of its actions has a wrong effect (i.e. a self-driving car has a mechanical problem causing it to stop much sooner than expected when applying the brakes). While the behavior could be detected as a discrepancy in world states, how does the agent know that the root problem is that its model is incorrect?

Dannenhauer, Cox, and Munoz-Avila (2018) introduce metacognitive expectations that can be used to detect discrepancies on the agent's own cognitive processes. To do this, cognitive processes (i.e. planning, goal selection, explanation, discrepancy detection, etc) are treated as mental actions that transform the agent's current mental state. Mental states are essentially snapshots of the agent's current knowledge and memory. Metacognitive expectations are run at the meta-level that is constantly monitoring the cognitive processes. One of the metacognitive expectations demonstrated in that work is shown in Figure 3. This metacognitive expectation is domain independent since it only requires obtaining abstract notions of the agents previous and post mental states. The first requirement is that the agent has identified some discrepancies in the world; shown as $v_{discrepancies} \neq \emptyset$ in s_i^M, where s_i^M is the mental state before the mental action of explanation was processed. The second requirement is that the mental state after the explanation cognitive step has generated some explanation; shown as $v_{explanations} \neq \emptyset$ in s_{i+1}^M where s_{i+1}^M is the mental state after the explanation step.

In the example of the autonomous car operating with a mechanical failure, the agent would identify the discrepancy but be unable to explain it. The lack of explanation is detected via the metacognitive expectations which then triggers diagnostic and remedial processes to update the agent's model of the world. The metacognitive expectations are powerful knowledge artifacts that are domain independent, however they currently must be supplied by a human expert. An important area of future work is how an agent might learn for itself when cognitive behavior is anomalous.

Summary and Future Work

In this paper we described 6 forms of expectations from the goal reasoning literature and discuss them based on five different knowledge sources. Here are some highlights:

- Immediate expectations do not guarantee that the agent's goals are fulfilled when the agent's execution terminates.
- In contrast, state, goal regression, informed and Goldilocks expectations guarantee that goals are fulfilled.
- From these four, goal regression performs the least sensing, checking if every atom is valid in the environment. State performs the most sensing. Informed and Goldilocks are in between these two. This means that all but goal regression may flag some discrepancies that won't interfere with the current plan's execution.
- Informed expectations can be used in situations where either the goals or a complete plan/policy is not known.
- Goldilocks expectations can be used when the plan or policy are known but not the goals.
- Cognitive expectations reason about the cognitive processes internal to the agent when acting in the environment as opposed the other forms of expectations which reason about the (external) environment.

In addition to the various entries with question marks in Table 1, there are a number of possible research directions: so far expectations described in this paper have neither been defined for plans with continuous actions (Coles et al. 2012; Molineaux, Klenk, and Aha 2010b) nor for situations where the execution of concurrent actions are allowed. So, for example, it is unclear how to define goal regression expectations for an action a in a plan or policy when a is expected to cool down an engine for 5 minutes and within the last minute of this cool down period another action to start removing a clamp holding the engine must begin its execution. Another layer of complexity is in partially observable environments.
Acknowledgements. This research is supported by ONR under grants N00014-18-1-2009 and N68335-18-C-0427.

References

Aha, D. W. 2018. Goal reasoning: foundations emerging applications and prospects. *AI Magazine. Under review*.

Birnbaum, L.; Collins, G.; Freed, M.; and Krulwich, B. 1990. Model-based diagnosis of planning failures. In *AAAI*, volume 90, 318–323.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2012. Colin: Planning with continuous linear numeric change. *Journal of Artificial Intelligence Research* 44:1–96.

Conn, A. 2017. Artificial intelligence: The challenge to keep it safe.

Cormen, T.; Leirson, C.; Rivest, R.; and Stein, C. 2001. *Introduction to Algorithms*. MIT Press.

Cox, M. T. 1996. Introspective multistrategy learning: Constructing a learning strategy under reasoning failure. Technical report, Georgia Inst of Tech Atlanta.

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

Currie, K., and Tate, A. 1991. O-Plan: The open planning architecture. *Artificial Intelligence* 52(1):49–86.

Dannenhauer, D., and Munoz-Avila, H. 2013. Luigi: a goal-driven autonomy agent reasoning with ontologies. *Advances in Cognitive Systems (ACS-13)*.

Dannenhauer, D., and Munoz-Avila, H. 2015a. Goal-driven autonomy with semantically-annotated hierarchical cases. In *International Conference on Case-Based Reasoning*, 88–103. Springer.

Dannenhauer, D., and Munoz-Avila, H. 2015b. Raising expectations in gda agents acting in dynamic environments. In *IJCAI*, 2241–2247.

Dannenhauer, D.; Cox, M. T.; and Munoz-Avila, H. 2018. Declarative metacognitive expectations for high-level cognition. *Advances in Cognitive Systems*.

Dannenhauer, D.; Munoz-Avila, H.; and Cox, M. T. 2016. Informed expectations to guide gda agents in partially observable environments. In *IJCAI*, 2493–2499.

Dannenhauer, D.; West, W.; and Hatalis, K. 2013. Ontology for Starcraft used in LUiGi and LUiGi-H. https://doi.org/10.5281/zenodo.804944.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *National Conference on Artificial Intelligence (AAAI)*, 1123–1128.

Fikes, R. E., and Nilsson, N. J. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2:189–208.

Floyd, M. W.; Karneeb, J.; Moore, P.; and Aha, D. W. 2017. A goal reasoning agent for controlling uavs in beyond-visual-range air combat. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 4714–4721. AAAI Press.

Fritz, C., and McIlraith, S. A. 2007. Monitoring plan optimality during execution. In *ICAPS*, 144–151.

Fu, J.; Ng, V.; Bastani, F. B.; Yen, I.-L.; et al. 2011. Simple and fast strong cyclic planning for fully-observable nondeterministic planning problems. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, 1949.

Hogg, C.; Muñoz-Avila, H.; and Kuter, U. 2008. HTN-MAKER: Learning HTNs with minimal additional knowledge engineering required. In *Conference on Artificial Intelligence (AAAI)*, 950–956. AAAI Press.

Karneeb, J.; Floyd, M. W.; Moore, P.; and Aha, D. W. 2016. Distributed discrepancy detection for bvr air combat. In *Proceedings of the IJCAI Workshop on Goal Reasoning, New York*.

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

Kondrakunta, S.; Gogineni, V. R.; Molineaux, M.; Munoz-Avila, H.; Oxenham, M.; and Cox, M. T. 2018. Toward problem recognition, explanation and goal formulation. In *6th Goal Reasoning Workshop at IJCAI/FAIM-2018*.

Molineaux, M.; Klenk, M.; and Aha, D. W. 2010a. Goal-Driven Autonomy in a Navy Strategy Simulation. In *AAAI*.

Molineaux, M.; Klenk, M.; and Aha, D. W. 2010b. Planning in dynamic environments: Extending htns with nonlinear continuous effects. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*.

Murdock, J. W. 2001. *Self-improvement through self-understanding: Model-based reflection for agent adaptation*. Ph.D. Dissertation, Georgia Institute of Technology.

Myers, K. L. 1999. A continuous planning and execution framework. *AI Magazine* 63–69.

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

Nguyen, C.; Reifsnyder, N.; Gopalakrishnan, S.; and Munoz-Avila, H. 2017. Automated learning of hierarchical task networks for controlling minecraft agents. In *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*, 226–231. IEEE.

Pozanco, A.; Yolanda, E.; and Borrajo, D. 2018. Counterplanning using goal recognition and landmarks. In *International Joint Conference in AI (IJCAI)*.

Reifsnyder, N., and Munoz-Avila, H. 2018. Expectations of gda agents acting in nondeterministic domains. In *6th Goal Reasoning Workshop at IJCAI/FAIM-2018*.

Roberts, M.; Shivashankar, V.; Alford, R.; Leece, M.; Gupta, S.; and Aha, D. 2016. Goal reasoning, planning, and acting with actorsim, the actor simulator. In *Poster Proceedings of the Fourth Annual Conference on Advances in Cognitive Systems*.

Sussman, G. J. 1975. *HACKER, a Computer Model of Skill Acquisition*. Elsevier.

Van Den Briel, M.; Sanchez, R.; Do, M. B.; and Kambhampati, S. 2004. Effective approaches for partial satisfaction (over-subscription) planning. In *AAAI*, 562–569.

Van Der Krogt, R., and De Weerdt, M. 2005. Plan repair as an extension of planning. In *ICAPS*, volume 5, 161–170.

Veloso, M. M., and Carbonell, J. 1993. Derivational analogy in PRODIGY: Automating case acquisition, storage and utilization. *Machine Learning* 10(3):249–278.

Wilkins, D. E. 1985. Recovering from execution errors in sipe. *Computational Intelligence* 1(1):33–45.

Wilkins, D. E. 1988. *Practical Planning: Extending the Classical AI Planning Paradigm*. San Mateo, CA: Morgan Kaufmann.

Wilson, M. A.; McMahon, J.; and Aha, D. W. 2014. Bounded expectations for discrepancy detection in goal-driven autonomy. In *AI and Robotics: Papers from the AAAI Workshop*.