Goal Reasoning with Goldilocks and Regression Expectations in Nondeterministic Domains

Noah Reifsnyder and Héctor Muñoz-Avila
Lehigh University
Bethlehem, PA 18015
{ndr217, hem4}@lehigh.edu

Abstract

Goal-driven autonomy (GDA) is a reflective model of goal reasoning that controls the focus of an agents actions by dynamically resolving unexpected discrepancies to the agent's expectations. Until now, GDA agents define expectations assuming they are executing plans. This paper presents different formalizations of the notion of expectations for GDA agents executing policies. These are necessary when acting in nondeterministic domains, where executing actions might have multiple outcomes. We present an algorithm called Goldilocks, computing these expectations and report on a comparative study between Goldilocks, Informed, Immediate, and Regression expectations and how they affect an agents goal reasoning.

1 Introduction

Goal-driven autonomy (GDA) is a goal reasoning model for controlling agents acting in the environment. GDA agents perform a 4-step cycle in which (1) actions are executed in the environment from a solution (e.g., a plan) generated for achieving previously formulated goals; (2) expectations on the agent's actions are computed and checked against the actual outcomes of the actions; (3) when there is a discrepancy between the outcome and the expectations, the agent formulates an explanation for this discrepancy; (4) as a result of this explanation, new goals to pursue are formulated.

GDA has been the subject of research in a number of topics, including research on GDA agents for naval operations [Molineaux *et al.*, 2010], GDA agents for computer games [Weber *et al.*, 2010; Jaidee *et al.*, 2013], automated learning of GDA components including expectations [Weber *et al.*, 2012], explanations [Molineaux and Aha, 2014], and goal-formulation knowledge [Jaidee *et al.*, 2011].

Researchers have observed that the notion of expectations plays a key role in the overall goal reasoning of GDA agents [Dannenhauer *et al.*, 2016; Dannenhauer and Munoz-Avila, 2015]. If the expectations are too narrow, GDA agents might fail to detect discrepancies when they are needed leading to agents continuing to pursue a goal that will lead to a failure. If the expectations are too general, the agent might trigger the GDA cycle and potentially change its goals in situations when it is not required to do so. A common characteristic of these works is that they compute the expectations over sequences of actions, typical of deterministic domains. We re-examine the notion of expectations when the solutions achieving the goals are policies, which are mappings from states to actions. Policies are needed in nondeterministic domains, where actions may have multiple possible outcomes.

The following are the main contributions of this paper:

- A formalization of GDA expectations for policies.
- An algorithm computing these expectations.
- A comparative study in a domain from the GDA literature, Arsonist [Paisner *et al.*, 2013].

2 Preliminaries

We have a collection of variables V. Each variable $v \in V$ can take a value from C, a set of constants. A state s is a mapping $s : V \rightarrow C$, instantiating each variable to a constant. S denotes the collection of all states.

In a nondeterministic (ND) domain, actions have multiple possible effects: An ND action $a = (name^a, pre^a, Eff^a)$ where pre^a is a partial mapping $pre^a : V_{pre^a} \subseteq V \rightarrow C$. An action a is applicable in a state s if for every variable $v \in V_{pre^a}$, $pre^a(v) = s(v)$. Eff^a is a finite set of possible effects. Specifically, if $Eff^a = \{eff_1^a \dots eff_n^a\}$, then each eff_i^a is a partial mapping $eff_i^a : V_{eff_i^a} \subseteq V \rightarrow C$. Applying eff_i^a to a state from s, results in an s' as follows: if $v \in V_{eff_i^a}$ then $s'(v) = eff_i^a(v)$; otherwise, $s'(v) = s(v)$ (i.e., the variable's value remains unchanged). Applying a to s will nondeterministically choose one of these effects, say eff_i^a, and applies it resulting in a state s'. In this case, we say that a ND choice is made and that $s' \in ND(a(s))$.

A **planning problem** P is defined as a triple $(S_0, g, \mathcal{A})$, indicating the initial state, the goals and the actions respectively. The goals g are a partial mapping $g : V_g \subseteq V \rightarrow C$. The goals g are **satisfied** in a state s if for every variable $v \in V_g$, $g(v) = s(v)$.

A solution is represented as a policy $\pi : S \rightarrow \mathcal{A}$, a partial mapping from the possible states in the world S to actions A, indicating for any given state s, what action $\pi(s)$ to take. Given a policy π, an **execution trace** is any sequence $s_0\ \pi(s_0)\ s_1\ \pi(s_1)\ \dots\ \pi(s_n)\ s_{n+1}$, where s_i is a state that can be reached from state s_{i-1} after applying action $\pi(s_{i-1})$.

A solution policy π is **weak** if there exists an execution trace from s_0 to a state satisfying g. Weak solutions guarantee that a goal state can be successfully reached sometimes. A solution π is either **strong cyclic** or **strong** if for every state s that the agent might find itself in after executing $\pi(s_0)$, there exists an execution trace from the state s to a state satisfying

g. The difference is that in strong cyclic solutions the same state might be visited more than once whereas in strong solutions this never happens. Strong solutions are ideal since they never visit the same state more than once but in some domains they might not exists.

3 GDA Agents and their Expectations

GDA agents generate a solution π achieving some goals g using a planning domain Σ. They continually monitor the environment to check if the agent's own expectations X are met in the environment s. When a discrepancy d is found between the agent's expectations X and the environment s, the agent generates plausible explanations for d; as a result of these explanations new goals $\hat{g}$ are generated, thereby restarting the cycle with $g = \hat{g}$.

We now define the expectations X when the solution π is a policy. We define an expectation X as a partial mapping $X : V_x \subseteq V \rightarrow C$. Hence, a discrepancy occurs in a state s if there exists a variable $v \in V_x$, $X(v) \neq s(v)$.

There are a variety of expectations from the GDA literature [Cox, 2007; Dannenhauer and Munoz-Avila, 2015; Dannenhauer *et al.*, 2016]. All of these have been defined for the deterministic case, when actions have a single outcome. Hence, we adapt these definitions for the nondeterministic (ND) case. We use the following conventions:

1. $\mathcal{A}_{prefix} = (a_1...a_{n+1})$ is the sequence of actions executed so far.
2. $S_{prefix} = (s_0...s_n)$ is the sequence of states visited so far.
3. $G_{prefix} = (g_0...g_n)$ is the sequence of goals that the agent has been pursuing so far. This includes the special case when the goals have not changed: e.g., $g_0 = ... = g_n$. Goals may change as a result of the GDA process.

The relation between $\mathcal{A}_{prefix}$, S_{prefix} and G_{prefix} is as follows: $a_{i+1} = \pi_k(s_i)$, where π_k is a policy achieving g_k. This again includes the case when $\pi_0 = ... = \pi_n$ (i.e., the agent has been executing the same policy. That is, when the goals have not changed). s_{i+1} is the state resulting from applying a_{i+1} to s_i at execution time (i.e., an effect of a_{i+1} is nondeterministically chosen). Under these conventions, s_n is the current state and $a_{n+1} = \pi_n(s_n)$ is the next action to be executed. We now define the expectations at time n, X_n:

1. *Immediate expectations:* $X_n = pre_n$ where pre_n are the preconditions of $\pi(s_n)$.
2. *Informed expectations:* $X_{Inf}(\mathcal{A}_{prefix}, s_0)$ moves forward all valid conditions effects so far in $\mathcal{A}_{prefix}$. Informed expectations are formally defined as follows: $X_{inf}(\mathcal{A}_{prefix}, s_0) = X_{inf}(\mathcal{A}_{prefix}, s_0, \{\})$
   - $X_{inf}((), s, X) = X$.
   - $X_{inf}((a_k a_{k+1}...a_n), s_k, X) = X_{inf}((a_{k+1}...a_n), s_{k+1}, comp)$, where s_{k+1} is the state that resulted from the ND of a_{k+1} to s_k by applying $eff_i^{a_{k+1}} : V_i \subseteq V \rightarrow C$, the ND-chosen effect from $Eff^{a_{k+1}} = \{eff_1^{a_{k+1}} \dots eff_n^{a_{k+1}}\}$. Let $X : V_X \subseteq V \rightarrow C$. We define $comp : V_{comp} \subseteq V \rightarrow C$ is the composite function $eff_1^{a_{k+1}} \bullet X$ defined as follows:
     - if $v \in V_x - V_i$ then $comp(v) = X(v)$.
     - if $v \in V_i$ then $comp(v) = eff_i^{a_{k+1}}(v)$.
     - for all other variables $comp$ is undefined (i.e., $V_{comp} = V_i \cup V_x$)

4 Policy Goal Regression Expectations

In contrast to the deterministic case whereby solutions to planning problems are sequences of actions, in ND domains, policies, consisting of states and actions taken in those states, are the solutions to planning problems. Hence, we define expectations for both states and actions. More specifically, given a policy $\pi : S \rightarrow \mathcal{A}$ that is a strong solution for the planning problem $(s_0, g, \mathcal{A})$, we now define expectations $X_s : V_{x_s} \subseteq V \rightarrow 2^{C \times [0,1]}$ and $X_{\pi(s)} : V_{x_{\pi(s)}} \subseteq V \rightarrow 2^{C \times [0,1]}$ for a state s, partial mappings.

Second, we define the **graph representation** of π, as a graph $G(\pi) = (V, E)$ where $V = S \cup \{\pi(s) | s \in S\}$ and $E = \{(s, \pi(s)) | s \in S\} \cup \{(\pi(s), s') | s' \in ND(\pi(s))\}$. For example, in Figure 1, $V = \{s_0, s_1, s_2, s_3, \pi(s_0), \pi(s_2), \pi(s_3)\}$ and $E = \{(s_0, \pi(s_0)), (\pi(s_0), s_1), (\pi(s_0), s_2), \dots, (\pi(s_3), s_0)\}$. We are assuming that the state s_1 is a goal state; when s_1 is reached, π computation ends successfully.

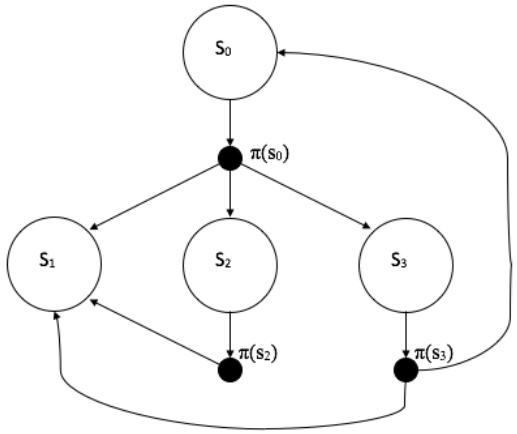

Figure 1: A graph representation $G(\pi)$ of a policy π. s_1 is the goal state.

We now define the **Plan Tree** $\mathcal{T} = (s_0, E_0, V_0)$ with root s_0 for the digraph $G(\pi) = (E, V)$. $\mathcal{T}$represents all paths from the initial state to a goal state, and is constructed using DFS as follows. In DFS, there are four types of edges between nodes:

- **Tree edges**: Edges of either the form $(s, \pi(s))$ or of the form $(\pi(s), s')$ such that s' was visited whenever DFS visited $\pi(s)$ and visited every non-visited child node of $\pi(s)$. For example, $(\pi(s_0), s_1)$ is a tree edge.
- **Back edges**: $e = (\pi(s), s')$ is a back edge if there is path P of tree edges from s_0 to $\pi(s)$ of the form:

$(s_0, \pi(s_0)), (\pi(s_0), s_1) \ldots (s', \pi(s')) \ldots (s, \pi(s))$ and $(\pi(s), s') \in E$. For example, $(\pi(s_3), s_0)$ is a back edge because there is a path of tree edges from s_0 to $(\pi(s_3))$.

- **Cross edges**: $e = (\pi(s), s')$ is a cross edge if there are paths P and P' of tree edges from s_0 to vertices $\pi(s)$ and s' respectively and $e = (\pi(s), s') \in E$. For example, $(\pi(s_2), s_1)$ is a cross edge.
- **Forward edges**: Those would be edges of the form $e = (\pi(s), s')$ such that there is a path of tree edges from $(s_0, \pi(s_0)), \ldots (s, \pi(s)) \ldots (\pi(s''), s'), (\pi(s), s') \in E$ and $s \neq s''$. Forward edges cannot exist in the expansion, as whenever a vertex $\pi(s)$ is visited we visit all of its children that have not been visited. Thus, any possible Forward edge would have been already expanded as a Tree Edge from $\pi(s)$.

To construct $\mathcal{T}$with DFS on $G(\pi)$, we handle each of these edges in a different way. Tree edges are handled as normal within the DFS. Cross edges, forward edges, and back edges are handled as follows:

1. If $e = (\pi(s), s')$ is a cross edge, we expand s' with the subtree rooted in s'.
2. If $e = (\pi(s), s')$ is a back edge, we expand s' with the subtree rooted in s' but excluding the new $e = (\pi(s), s')$ and its children in the subtree.

Computing $G(\pi)$ is linear on the number of actions and states in π. DFS is linear on the number of edges and vertices in $G(\pi)$. Because every $s \in V$ is reachable from s_0, then every $s \in V$ is visited during DFS. Therefore, because every edge is expanded at most once (we ignore the expanded edge in the expansion), computing $\mathcal{T}$is polynomial, $((V + E) * E)$.

Figure 2 showcases an example of expanding a cross edge $e = (\pi(s_2), s_1)$ (as seen in Figure 1). We make a copy of the sub tree rooted at s_1 and point the edge from $\pi(s_2)$ towards the copy. Forward edges are handled similarly.

Figure 3 showcases an example of expanding the back $e = (\pi(s_3), s_0)$ (as seen in Figure 1). We make a copy of the sub tree rooted at s_0 and point the edge from $\pi(s_3)$ to the copy. The other expanded cross edges have been copied over in the sub tree.

We use the plan tree $\mathcal{T}$ to define the expectations X_s and $X_{\pi(s)}$ for every vertex in $G(\pi)$. If s (or $\pi(s)$) occurs more than once in $\mathcal{T}$, then we take the final expectations X in the node that was visited first in the DFS procedure, and set all s (or $\pi(s)$) expectations to X. This is because if it is visited first by DFS, it will be visited last while calculating the expectations therefore having the final value (As shown later). We define **policy goal regression** X in $\mathcal{T}$ for actions in the policy as follows:

1. If $s = s_0$ in $\mathcal{T}$, for every variable $v \in V_{X_{\pi(s)}}$, $X_s(v) = X_{\pi(s)}(v)$.
2. If $s \neq s_0$, we define X_s as follows. Let $eff_s^{\pi(s_j)}$ be the effects of the preceding action $\pi(s_j)$, the parent of s in $\mathcal{T}$, making s_j the grandparent of s in $\mathcal{T}$. If s is not a leaf in $\mathcal{T}$, for all $v \in V_{X_{\pi(s)}}$ - $V_{eff_s^{\pi(s_j)}}$, $X_s(v)$=$X_{\pi(s)}(v)$. If s is a leaf in $\mathcal{T}$, for all $v \in V_g$ - $V_{eff_s^{\pi(s_j)}}$, $X_s(v) = \{(g(v), 1.0)\}$

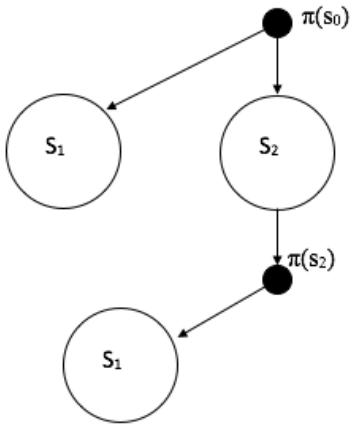

Figure 2: An example of an expanded a cross edge in $G(\pi)$ from Figure 1

3. If $\pi(s) \in \mathcal{T}$, let $\{s_1, \ldots, s_n\}$ be the children of $\pi(s)$ in $\mathcal{T}$and $pre^{\pi(s)}$ be the preconditions of $\pi(s)$. We define $X_{\pi(s)}$ as follows:
   (a) For all $v \in V_{pre^{\pi(s)}}$, $X_{\pi(s)}(v) = \{(pre^{\pi(s)}(v), 1.0)\}$.
   (b) if v is found k times in $\{V_{X_{s_1}}, ..., V_{X_{s_n}}\}$-$V_{pre^{\pi(s)}}$ (i.e., $k \leq n$) with associated (value, probability) pairs (c, p_1),..., (c, p_k). Then, $(c, (p_1 + ...p_k)/n) \in X_{\pi(s)}(v)$. $X_{\pi(s)}(v)$ will consist of a set of $(c, (p_1 + ...p_k)/n)$ pairs, one for each possible value of v among the expectations of its children.

To show an example of Policy Goal Regression Expectations, we define the following actions on $\mathcal{T}(name^a, pre^a, Eff^a)$:

- $(\pi(s_0), \{(C, 1)\}, \{\{(D, 1)\}, \{(A, 1)\}, \{(B, 1)\}\})$;
- $(\pi(s_2), \{(A, 1)\}, \{\{(D, 1)\}\})$;
- $(\pi(s_3), \{(B, 1)\}, \{\{(D, 1)\}, \{(B, 0)\}\})$.

The starting state is s_0=$\{(A, 0), (B, 0), (C, 1), (D, 0)\}$ and the goal is $g = \{(D, 1)\}$.

Figure 4 shows $\mathcal{T}$with the regression expectations. For example, to calculate X_{s_1}, the parent of s_1 is $\pi(s_0)$, with the grandparent being s_0. Since s_1 is a leaf, and V_g - $V_{eff_s^{\pi(s_j)}}$ is $\{\}$, X_{s_1}=$\{\}$. We calculate $X_{\pi(s_0)}$ as follows: for $\pi(s_0)$, $pre^{\pi(s_0)}$=$\{(C, 1)\}$. Thus we add $(C, 1, 1.0)$ into $X_{\pi(s_0)}$(i.e., $X_{\pi(s_0)}(C) = (1, 1.0)$). Now we examine the children $\{s_1, s_2, s_3\}$ of $\pi(s_0)$. Since only C is defined in the expectations of the children, and C was in the preconditions, we do nothing. Therefore $X_{\pi(s_0)}$=$\{(C, 1, 1.0)\}$. We calculate X_{s_2} as follows: The parent of s_2 is $\pi(s_0)$, with the grandparent being s_0. The effects of $\pi(s_0)$ along the edge ($\pi(s_0)$,s_2)=$\{(A, 1)\}$. Thus, A is ignored from $X_{\pi(s_2)}$. With no other variables in $X_{\pi(s_2)}$, $X_{(s_2)}$=$\{\}$. When all is finished, we have the expectations for the actions in the policy π:

- $\pi(s_0) : X_{\pi(s_0)} = \{(C, 1, 1.0)\}$
- $\pi(s_2) : X_{\pi(s_2)} = \{(A, 1, 1.0)\}$

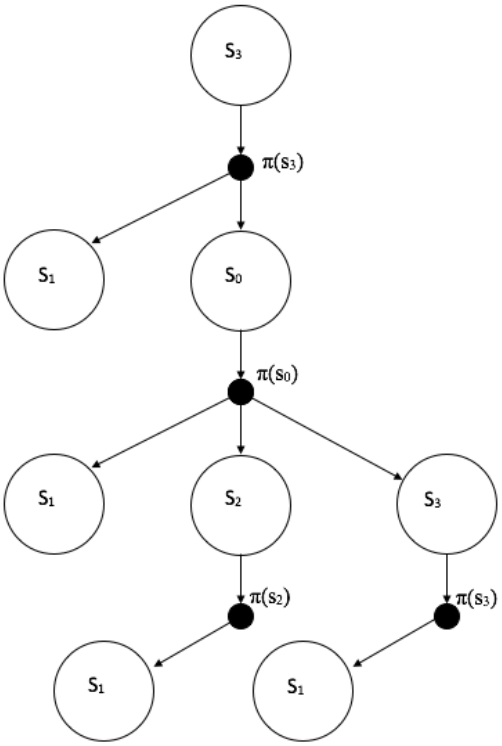

Figure 3: An example of an expanded a back edge in $G(\pi)$ from Figure 1

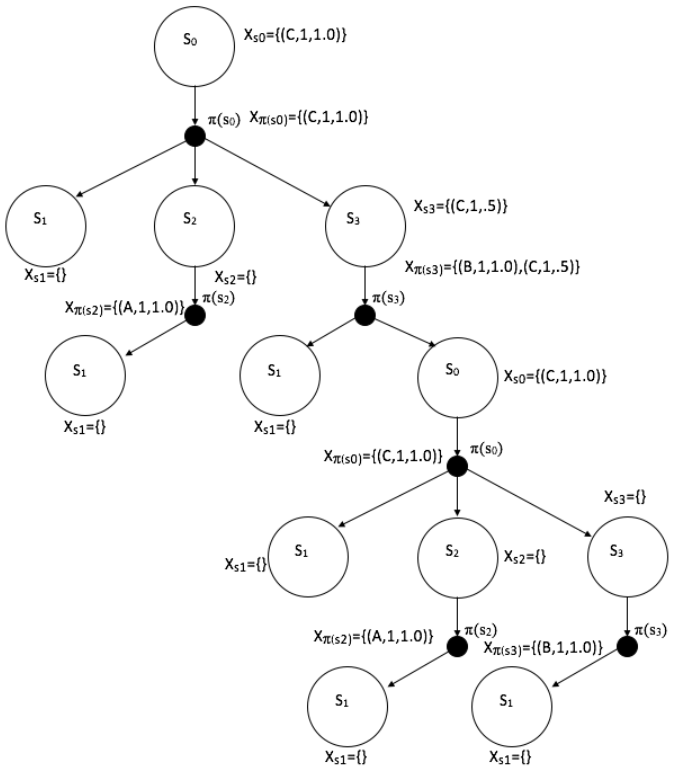

Figure 4: The plan tree $\mathcal{T}$ with calculated Policy Goal Regression Expectations

- $\pi(s_3) : X_{\pi(s_3)} = \{(B, 0, 1.0), (C, 1, .5)\}$

We define **policy goal regression** X in $\mathcal{T}$ for states in the policy as follows: For every non-terminal state s in the policy, we set X_s=$X_{\pi(s)}$. For all terminal states s, which in this case are goal states, we set the expectations as follows: For all $v \in V_g$, $X_s(v) = (g(v), 1.0)$.

5 Goldilocks Expectations and Detecting Discrepancies

The Goldilocks expectations take into account effects of actions (as in informed expectations) while also incorporating regression expectations. To compute Goldilocks expectations, we modify case 2 for Policy Goal Regression Expectations as follows.

2. If $s \neq s_0$, we define X_s as follows. Let $eff_s^{\pi(s_j)}$ be the effects of the preceding action $\pi(s_j)$, the parent of s in $\mathcal{T}$, making s_j the grandparent of s in $\mathcal{T}$.
   - For all $v \in V_{eff_s^{\pi(s_j)}}$, $X_s(v)$=$\{(X_{inf_{s_j}}(v), 1.0)\}$.
   - If s is not a leaf in $\mathcal{T}$: For all $v \in V_{X_{\pi(s)}}$ - $V_{eff_s^{\pi(s_j)}}$, $X_s(v)$=$X_{\pi(s)}(v)$.
   - If s is a leaf in $\mathcal{T}$: For all $v \in V_g$ - $V_{eff_s^{\pi(s_j)}}$, $X_s(v)$=$\{(g(v), 1.0)\}$.

We re-examine the same example as before, but this time using Goldilocs Expectations (Figure 5). The actions, the starting state and goals are the same. For example, to calculate X_{s_1}, the parent of s_1 is $\pi(s_0)$, with the grandparent being s_0. The effects of $\pi(s_0)$ along the edge ($\pi(s_0)$,s_1)=$\{(D, 1)\}$. Thus, $(D, s_0(D), 1.0)$ is added to X_{s_1} where $s_0(D)$=0. Since s_1 is a leaf, X_{s_1}=$\{(D, 0, 1.0)\}$ (i.e., $X_{s_1}(D)$=$\{(0, 1.0)\}$). We calculate $X_{\pi(s_0)}$ as follows: for $\pi(s_0)$, $pre^{\pi(s_0)}$=$\{(C, 1)\}$. Thus we add $(C, 1, 1.0)$ into $X_{\pi(s_0)}$. Now we examine the children $\{s_1, s_2, s_3\}$ of $\pi(s_0)$: $(D, 0)$ is defined 3 times in the children expectations; adding p from each $(D, 0, p)$ in the children expectations (1+1+1) and dividing by the number of children (3), we get 1.0, and add $(D, 0, 1.0)$ in $X_{\pi(s_0)}$. $(B, 0)$ is defined once with $p = 1.0$ dividing by 3, we get .333 and add $(B, 0, .333)$ in $X_{\pi(s_0)}$. $(A, 0)$ is defined twice with $p = 1.0$ and $p = .167$ added together and divided by 3, we get .583. So we add $(A, 0, .583)$ in $X_{\pi(s_0)}$. After appending all of these (v, c, p) triples to the expectations, we get $X_{\pi(s_0)} = \{(A, 0, .583), (B, 0, .333), (C, 1, 1.0), (D, 0, 1.0)\}$. We calculate X_{s_2} as follows: The parent of s_2 is $\pi(s_0)$, with the grandparent being s_0. The effects of $\pi(s_0)$ along the edge ($\pi(s_0)$,s_2)=$\{(A, 1)\}$, where $s_0(A)$=0. Thus, $(A, 0, 1.0)$ is added to X_{s_2}. Next we look to the child of s_2, $\pi(s_2)$. We ignore A since it was in $V_{eff_{s_2}^{\pi(s_0)}}$, which leaves $(D, 0, 1.0)$ to be added to X_{s_2}. Thus X_{s_2}=$\{(A, 1, 1.0), (D, 0, 1.0)\}$ When all is finished, we have our policy with expectations π:

- $\pi(s_0)$:
  $X_{\pi(s_0)} = \{(A, 0, .583), (B, 0, .333), (C, 1, 1.0), (D, 0, 1.0)\}$

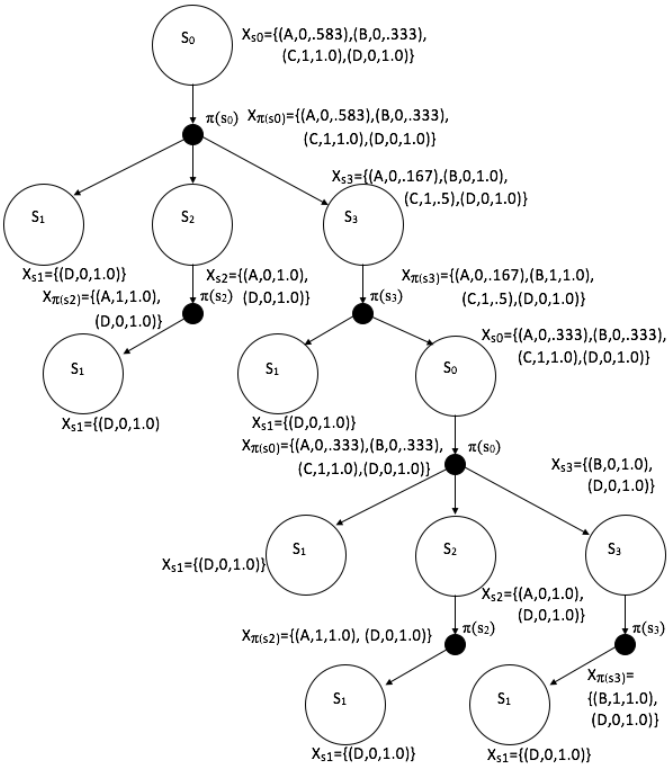

Figure 5: The plan tree $\mathcal{T}$ with modified Policy Goal Regression Expectations for Goldilocks Expectations

- $\pi(s_2) : X_{\pi(s_2)} = \{(A, 1, 1.0), (D, 0, 1.0)\}$
- $\pi(s_3) : X_{\pi(s_3)} = \{(A, 0, .167), (B, 0, 1.0), (C, 1, .5), (D, 0, 1.0)\}$.

Discrepancies. We define a discrepancy as follows: if in the observed state s, $s(v) = c$, for all $(c', p) \in X_s(v)$ where $c' \neq c$ we sum the p's to obtain the probability P that the plan will fail. This is because P is the percentage of descendant leaf vertices that are no longer reachable if $X_s(v) = c$. A discrepancy occurs if $P > 0.5$. We choose .5 as a threshold as it denotes that more than half the remaining tree is no longer reachable and therefore the execution of the policy is more likely to fail than not assuming equal distribution of the nondeterministic outcomes.

Informed expectations are necessary in cases where the goals achieved by the policies are unknown. For example, in our case, we used an ND HTN planner to generate the policies. In HTN planning tasks not goals are used to specify the problems. Using informed expectations infer the possible goals achieved at every terminal state. Regression expectations on these possible goals ensure the ability of the plan to reach the terminal states. By combining both forms of expectations, Goldilocks assures the policy will reach a terminal state and the possible goals will be achieved. We call these Goldilocks after the character of the Goldilocks and the Three Bears fairy tale. The Goldilocks expectations are the best fit when we don't know the goals compared to regression and informed expectations. Our empirical evaluation demonstrates this. In cases where the goals g are explicitly known, policy goal regression is sufficient. Goldilocks works when the goals g are unknown, because of the incorporation of Informed expectations. Informed expectations accumulates all effects created by the agent, some subset of which must be the goals.

6 Empirical Evaluation

In our experiments, we computed 4 expectation types: regression, immediate, informed, and Goldilocks. Aside from the different expectations, the GDA agent was the same. The performance metric is the same as in [Dannenhauer *et al.*, 2016]: the cost of solving the problem. These agents executed actions, until they reach a terminal node. At that point we check if the goals are satisfied, and if they are not the execution is considered a failure.

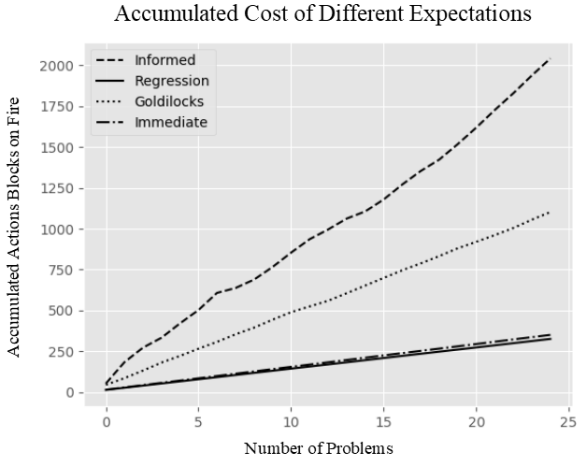

Figure 6: Expectation Costs for Arsonist Domain

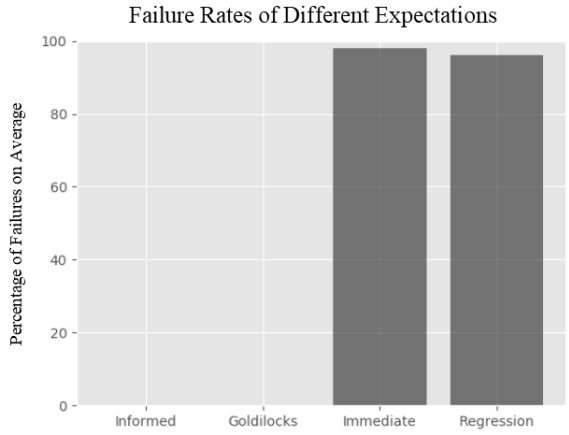

Figure 7: Failure rates for expectations in Arsonist Domain

Our tests were conducted in a variant of the Arsonist domain [Paisner *et al.*, 2013], which is itself a modified version of Blocks World. The goal is to make a tower of 10 blocks. There is an arsonist that is arbitrarily setting blocks on fire. The cost of a problem is the cumulated number of actions where a block is on fire (each block adds 1 point for every action it's on fire). The possible actions for the agent in the Arsonist domain are the usual `stack` and `unstack` a block, both requiring the block not to be on fire, with an added action to `douse` a fire. In our variant, actions have nondeterministic effects: stacking a block on top of another block has an equal probability of being successful or knocking the entire tower over. A failure occurs if any block in the tower is on fire or if the tower is not 10 blocks tall when the agent finishes executing actions. It is possible even with an ND plan for the agent to finish executing with a tower less than 10 blocks tall, because the agents knowledge of the state is limited by the form of expectation that the agent is using.

Figure 6 compares costs between Immediate, Regression, Informed, and Goldilocks expectations. Informed expectations had by far the highest cost. This is because blocks were allowed to burn until they were taken to stack onto the tower; the agent using informed expectations had no way of knowing which blocks it would eventually use. Regression and Immediate expectations performed similarly, with Regression barely outperforming Immediate. They have such low costs because their expectations have no knowledge of previously stacked blocks, and thus do not know if the last `stack` action knocked over the tower. They each performed the necessary `stack` and `unstack` actions to build the tower deterministically, resulting in 10 actions per problem. This was not enough time for many blocks to catch on fire therefore they could not accumulate a large cost. Both expectations had near 100% failure rates (as seen in Figure 7), however, due to the same lack of knowledge in the expectations. Goldilocks had a cost well below Informed expectation, and a failure rate of 0%. This is because it had the knowledge of which blocks it would eventually use in the tower and could put the fires out immediately. Goldilocks also knew about which blocks had been stacked in the tower already allowing it to tell if the tower fell over or if a placed block caught on fire.

7 Related Work

The GDA framework is general allowing a variety of planning paradigms. GDA research has used both STRIPS planning [Molineaux *et al.*, 2010] and MDP-based planning [Jaidee *et al.*, 2013]. However, until now, GDA works compute expectations by examining the action sequence executed so far, what we dubbed π_{prefix}. In contrast, our work extends the notion of expectations to explicitly consider policies.

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]. In both of these works regression is performed for deterministic planning, when the solutions to planning problems are sequences of actions. In our work, we are defining goal regression for nondeterministic planning when solutions are policies.

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, and as a result the action is inapplicable, this is called an expectation failure [Cox, 2007]. Other kinds of failures have been proposed where even though they are not execution failures, the current execution exhibits conditions that are not desirable [Myers, 1999]. Others have suggested failures associated with quality conditions [Fritz and McIlraith, 2007].

GDA focuses on the meta-process of goal reasoning. 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 GDA goals might change as a reaction to changes in the environment. In that sense GDA 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; Warfield *et al.*, 2007]. The main difference between plan repair and GDA is that in the latter the goals might change whereas plan repair sticks with the same goals while searching for alternative plans.

Other works have explored goal reasoning capabilities in environments were the outcomes of the actions maybe uncertain [Karneeb *et al.*, 2016; Floyd *et al.*, 2017; Wilson *et al.*, 2014]; however, those works model solutions as sequence of actions, unlike policies as in our work. These works use what we call state expectations: the agent maintains the expected resulted state following each action and these are matched against the partially-observed state. [Wilson *et al.*, 2014] projects forward continuous variables with their expected variables that are expected to be within an interval defined by a lower and upper bound on the linear sequence of actions. This work also maintains state expectations.

8 Conclusions

In this paper we re-examine the notion of expectations in the context of GDA: immediate and informed expectations, both of which are based on the sequence of actions π_{prefix} executed so far. We introduce two new form of expectations: (1) regression which is defined based on the possible trajectories from the starting state to goal states; (2) Goldilocks which combines the informed and regression expectations We report on a comparative study of these four forms of expectations on the Arsonist domain. Goldilocks and informed expectations are the only ones reaching terminal states without failures (i.e., goals are achieved). However, informed expectations do so having the higher costs than Goldilocks.

For future work, we plan to extend our work for situations where probability distributions on the ND effects are known. Hence, these will need to be aggregated over the regressed

and Goldilocks expectations. We expect that defining these will benefit from Bayesian inferencing mechanisms. We will also explore defining Goldilocks expectations for domains with continuous variables.

Acknowledgements. This research was supported by ONR under grant N00014-18-1-2009 and by NSF under grant 1217888.

References

[Birnbaum *et al.*, 1990] Lawrence Birnbaum, Gregg Collins, Michael Freed, and Bruce Krulwich. Model-based diagnosis of planning failures. In *AAAI*, volume 90, pages 318–323, 1990.

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

[Dannenhauer and Munoz-Avila, 2015] Dustin Dannenhauer and Hector Munoz-Avila. Raising expectations in gda agents acting in dynamic environments. In *IJCAI*, pages 2241–2247, 2015.

[Dannenhauer *et al.*, 2016] Dustin Dannenhauer, Hector Munoz-Avila, and Michael T Cox. Informed expectations to guide gda agents in partially observable environments. In *IJCAI*, pages 2493–2499, 2016.

[Floyd *et al.*, 2017] Michael W Floyd, Justin Karneeb, Philip Moore, and David W Aha. A goal reasoning agent for controlling uavs in beyond-visual-range air combat. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4714–4721. AAAI Press, 2017.

[Fritz and McIlraith, 2007] Christian Fritz and Sheila A McIlraith. Monitoring plan optimality during execution. In *ICAPS*, pages 144–151, 2007.

[Jaidee *et al.*, 2011] Ulit Jaidee, Héctor Muñoz-Avila, and David W Aha. Integrated Learning for Goal-Driven Autonomy. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three*, pages 2450–2455. AAAI Press, 2011.

[Jaidee *et al.*, 2013] Ulit Jaidee, Héctor Muñoz-Avila, and David W Aha. Case-based goal-driven coordination of multiple learning agents. In *International Conference on Case-Based Reasoning*, pages 164–178. Springer, 2013.

[Karneeb *et al.*, 2016] Justin Karneeb, Michael W Floyd, Philip Moore, and David W Aha. Distributed discrepancy detection for bvr air combat. In *Proceedings of the IJCAI Workshop on Goal Reasoning, New York*, 2016.

[Molineaux and Aha, 2014] Matthew Molineaux and David W Aha. Learning unknown event models. In *AAAI*, pages 395–401, 2014.

[Molineaux *et al.*, 2010] Matthew Molineaux, Matthew Klenk, and David W Aha. Goal-Driven Autonomy in a Navy Strategy Simulation. In *AAAI*, 2010.

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

[Paisner *et al.*, 2013] Matt Paisner, Michael Maynord, Michael T Cox, and Don Perlis. Goal-driven autonomy in dynamic environments. In *Goal Reasoning: Papers from the ACS Workshop*, page 79, 2013.

[Sussman, 1975] Gerald J. Sussman. *HACKER, a Computer Model of Skill Acquisition*. Elsevier, 1975.

[Van Den Briel *et al.*, 2004] Menkes Van Den Briel, Romeo Sanchez, Minh Binh Do, and Subbarao Kambhampati. Effective approaches for partial satisfaction (oversubscription) planning. In *AAAI*, pages 562–569, 2004.

[Van Der Krogt and De Weerdt, 2005] Roman Van Der Krogt and Mathijs De Weerdt. Plan repair as an extension of planning. In *ICAPS*, volume 5, pages 161–170, 2005.

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

[Warfield *et al.*, 2007] Ian Warfield, Chad Hogg, Stephen Lee-Urban, and Héctor Munoz-Avila. Adaptation of hierarchical task network plans. In *FLAIRS conference*, pages 429–434, 2007.

[Weber *et al.*, 2010] Ben George Weber, Michael Mateas, and Arnav Jhala. Applying goal-driven autonomy to starcraft. In *AIIDE*, 2010.

[Weber *et al.*, 2012] Ben George Weber, Michael Mateas, and Arnav Jhala. Learning from demonstration for goal-driven autonomy. In *AAAI*, 2012.

[Wilson *et al.*, 2014] Mark A Wilson, James McMahon, and David W Aha. Bounded expectations for discrepancy detection in goal-driven autonomy. In *AI and Robotics: Papers from the AAAI Workshop*, 2014.