Adaptive Goal Driven Autonomy*

Héctor Muñoz-Avila

Computer Science & Engineering
Lehigh University
19 Memorial Drive West
Bethlehem, PA 18015-3084 USA munoz@cse.lehigh.edu

Abstract. Goal-driven autonomy (GDA) is a reflective model of goal reasoning combining deliberative planning and plan execution monitoring. GDA's is the focus of increasing interest due in part to the need to ensure that autonomous agents behave as intended. However, to perform well, comprehensive GDA agents require substantial domain knowledge. In this paper I focus on our work to automatically learn knowledge used by GDA agents. I also discuss future research directions.

1 Introduction

Goal-driven autonomy (GDA) is a reflective model of goal reasoning combining deliberative planning and plan execution monitoring. The key aspect of GDA is the intermix between the agent's observations (e.g., from sensors), goal reasoning, including formulating new goals, and acting according to the goals it is pursuing. GDA is related to fields such as the actor's view of planning [15] (i.e., a framework for interleaving planning and execution), online planning [20] (i.e., refining a planning solution while executing it in the environment), cognitive systems [21] (e.g., agents endowed with self-reflection capabilities) and general agency [4]. The key distinctive feature of goal reasoning agents is their capability to adjust their goals, including changing their goals altogether while acting in an environment. This in contrast to approaches such as replanning [14, 36] where the plan generated is modified due to changes in the environment while still aiming to achieve the same goals. For example, in [26], we use guiding principles we called motivators to select which goals to achieve. As indicated in [2], there is an increasing interest in goal reasoning, in part, because of applications such as UUVs' control [29], air combat [13] and disaster response [33].

To perform well, comprehensive GDA agents require substantial domain knowledge to determine expected states [24], identify and explain discrepancies [22], formulate new goals [39], and manage pending goals [37]. This requires, for example, the agent's programmers to anticipate what discrepancies can occur, identify what goals can be formulated, and define their relative priority.

In this paper, I will focus on our work to automatically learn knowledge used by GDA agents performed over the past years. This includes:

* This work is supported in part under ONR N00014-18-1-2009 and NSF 1217888 .

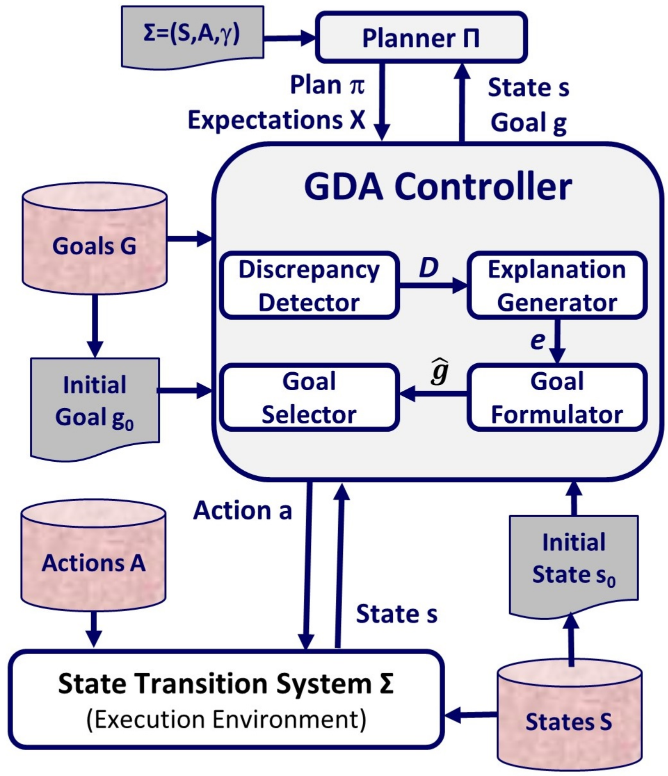

Fig. 1. The goal-driven autonomy model [25, 24]

- Using reinforcement Learning to acquire planning, explanation and goal formulation knowledge (Section 3).
- Using case-based reasoning techniques for learning expectations, and goals (Section 4).
- Formulating a taxonomy of automatically generated expectations using learned hierarchical task network representations (Section 5).

In Section 6, I will discuss ongoing research directions including computing expectations beyond a linear sequence of actions.

2 The goal driven autonomy model

GDA agents generate a plan π achieving some goals g using a planning domain Σ [25, 24, 5] (see Figure 1). They continually monitor the environment to check if the agent's own expectations x are met in the state s. When a discrepancy D is found between the agent's expectations X and the environment s, the agent generate a plausible explanation e for d; as a result of this explanation, a new goal $\hat{g}$ is generated, thereby restarting the cycle with $g = \hat{g}$.

If A denotes all actions in the domain and S denotes all states, the following are the knowledge artifacts needed by GDA agents:

- Σ, which contains a function $\gamma : S \times A \rightarrow S$, indicating for each action $a \in A$, and every state $s \in S$, what is the resulting state $\gamma(s)$ when applying a in s.
- Π, a planning system. In addition to Σ, it may include additional knowledge to generate plans such as HTN planning knowledge.
- G, the collection of all goals that can be pursued by the agents.
- Given an action a in a plan π, what are the expectations of when executing a. This is dependent on where a occurs in π as well as the initial state from which π started its execution.
- The explanation knowledge. For example, explanation knowledge may consist of a collection of rules of the form $D \rightarrow e$, indicating for a discrepancy D what is its explanation e.
- Goal formulation knowledge. For example, goal formulation knowledge may consist of a collection of rules of the form $e \rightarrow \tilde{g}$, indicating for each explanation, what is the next goal to achieve.
- Goal selection knowledge. In its most simple form it always selects to last goal generated by Goal Formulator (i.e., $g = \tilde{g}$).

3 Learning plan, explanation and goal formulation Knowledge with reinforcement learning

We explored the use of reinforcement learning (RL) techniques [35] to learn planning knowledge for GDA. The assumptions are that the agent knows the plausible actions $A_s \subset A$ that can be taken in an state $s \in S$. But the problem is that the agent doesn't know, among those actions in A_s, which is the best or even a good action to take. Typical of RL, the agent aims at learning the value of each action $a \in A_s$ in such a way that it maximizes its rewards. The result of RL's learning process is a policy $\pi : S \rightarrow A$, indicating for every state, which action $\pi(s)$ to take [17].

We also took advantage of RLs capabilities to both select plausible explanations for a discrepancy [12] and to select possible goals for a given explanation [18]. In both cases we formulate the selection problem as a Markov Decision Problem (see Table 1). The aim is to choose the best explanation among the possible discrepancies. Thus we view each discrepancy as an state and each explanation as an action. All other aspects of the GDA process being equal, if the overall performance of the GDA agent improves, we will assign a positive reward to the explanation selected and a negative reward otherwise. This means that the agent could consider multiple plausible explanations for the same discrepancy and over time it learns the value of each explanation (i.e., higher value means more plausible explanation).

We did a similar approach to learn goal selection knowledge but this time modeling the possible explanations as the states and the choice of a goal as the action. These were separate works and we never tried to simultaneously learn explanation selection and goal formulation knowledge. We will retake this point in the future work section.

Table 1. Modeling explanation selection and goal formulation as an MDP

MDP	**Actions**	**States**
Explanation generation	All possible explanations	All possible discrepancies
Goal Formulation	All possible goals	All possible explanations

We tested our approach in adversarial games. Specifically real-time strategy (RTS) games. In these games players control armies to defeat an opponent. These games add elements that make game playing challenging: armies consists of different types of units. The outcome of unit versus unit combat follows a paper-rock-scissors design: archers can easily defeat gryphons but are themselves easily defeated by knights. In turn knights can be easily defeated by gryphons. Thus, the best players are those who maintain a well balanced army and know how to manage it during battle. Another aspect that makes RTS games even more challenging is the fact that players make their moves synchronously thereby rewarding players that can make decisions quickly.

In our experiments [18, 17, 12], the GDA agent is controlling a player that is competing against various opponents, one at the time in each game instance. These opponents used a variety of hard coded strategies. For testing we did the typical leave-one-out strategy: given N opponents, the GDA agent trained by playing against $N-1$ opponents and was tested against the remaining opponent not used in training. The results were compelling showing a significant improvement by the learning GDA agent. The reward function was the difference in score between the GDA player and its opponent. Thus, a positive score means the GDA is defeating the opponent whereas a negative reward indicates that the opponent is winning.

4 Learning expectations and goals with case-based reasoning

Case-based reasoning (CBR) is a problem solving method in which cases, instances of previous problem-solving episodes, are reused to solve new problems [1]. One of the situations when it is desirable to use CBR is when there are gaps in the knowledge preventing the formulation of a complete theory, one that can be reasoned by using first-principles. This is precisely the situation we encountered when computing the expectations after taking an action a in state s, the agent does not have any knowledge about the expected state. Furthermore, since the gaming domains are nondeterministic, meaning that after applying an action a to an state s there could be many possible resulting states a_s and the agent initially does not know which are those states and the probability that anyone of them will be reached after applying action a in s. To solve this problem we created a case base, CB_X of expectations [18]:

$$CB_X : S \times A \rightarrow 2^{S\times[0,1]}$$

This case base maps for each state-action pair (s, a) a probability distribution of pairs (q, p) indicating the probability p of reaching state q. In a typical CBR manner, each time the controller selects an action a from state s (i.e., as indicated by a policy π as described in the previous section), then one of the following two steps is performed:

– If q was never reached before when action a was selected from state s, then (q, p) is stored in the CB_X with $p = 1/N_{(s,a)}$, where $N_{(s,a)}$ is the number of times a has been selected when reaching state s.
– If q has been selected $N_{(s,a,q)}$ times then the value of p is updated to $p = (N_{(s,a,q)} + 1)/N_{(s,a)}$.

CBR is also used to learn to associate a goal state s_π to a policy π [18, 19]. Following a policy from an starting state s_0 produces an execution trace $s_0\ \pi(s_0)\ \cdots\ \pi(s_n)\ s_{n+1}$. If an state q appears with at least some predefined frequency f on the last k steps in the trace then q is assigned to be the goal of the policy (f and k are parameters set by the user). Otherwise s_{n+1} is assigned to be the goal. Since different chains may be generated for the same initial state and policy, we keep a probability distribution of goals akin to the way we kept a provability distribution of expectations described before. Although we only annotate the policy with the goal having the highest probability.

The goals are used as described in the previous section in the goal formulation procedure. When a goal g_π is selected, its associated policy π is executed. Since our agents perform on-line learning, π changes over time, which may result in g_π itself changing.

5 A taxonomy of expectations with HTN planning

For this research thrust, we leverage on our previous work for automatically learning hierarchal task networks (HTN) [16, 40]. The learning problem can be succinctly described as follows: given a collection of plans generated by some (possibly unknown) agent and a collection of tasks T, learn HTN planning knowledge in a manner that is consistent with the given collection of plans. The learned HTN indicate how to decompose a task $t \in T$ into subtasks $t_1\ \cdots\ t_m$ in T and the applicability conditions (i.e., the preconditions) under which this task decomposition, $t \rightarrow t_1\ \cdots\ t_m$, is valid. The learned HTNs can be used by the HTN planner SHOP [27] to generate plans. That is, we set $\Pi = SHOP$ and π is an HTN plan.

Under the premise of using HTN planning for plan generation, we examine the notion of expectations for GDA agents. Expectations play an important role in the performance of GDA agents [8]: if the expectations are too general,

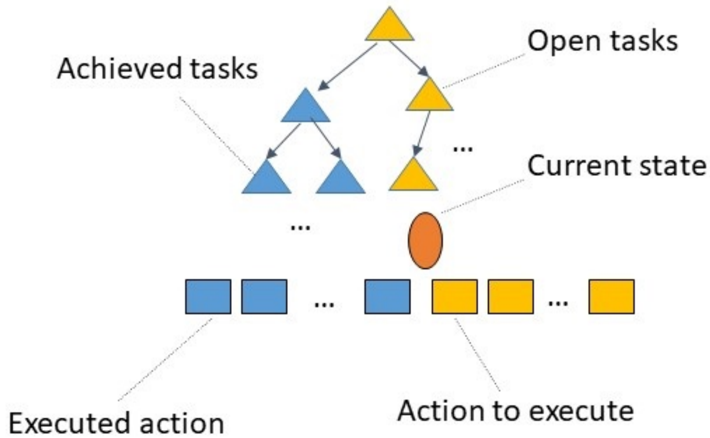

Fig. 2. Generic HTN; actions are divided between those already executed and those to execute; analogously, tasks are divided between those accomplished and those that remain open

many unnecessary discrepancies will be found triggering unnecessary and possible detrimental GDA process. An example of a detrimental situation is the following: suppose in an RTS game that the GDA is very close to destroying an opponent's town hall, thereby defeating the opponent. Suppose that at the same time the player unexpectedly detects a large opponent's force poised to take over the player's resource gathering operation (resources are needed to increase a player's army). This discrepancy may trigger a new goal to defend the resource gathering operation, which in turn may deviate part of the army from attacking the town hall and miss the chance of winning the game. If the expectations are too narrow, discrepancies can be missed in situations were formulating a goal would have been beneficial.

We studied the problem of eliciting expectations from plans [9] and from HTN plans [8]. The basic premise is to divide the plan into two portions: the actions already executed and the actions that remain to execute 2. SHOP propagates forward the state and the next action to execute is checked against the current state. Correspondingly, the task hierarchy can be divided between those tasks that have been achieved because their underlying portion of the plan is executed and those that are open (i.e., remain to be achieved) because some of its underlying actions have not been achieved.

We explored 5 forms of expectations used in by GDA agents and more generally in goal reasoning agents:

– **None**. It checks the preconditions of the next action to execute. That is, if the preconditions are applicable in the current state. This is useful, for example, in the early stages of an RTS game when the player is focusing

on building its own economy. It typically focus on a predefined plan (this is called the build order [28]). As soon as the preconditions of the next action become applicable, it executes it. For example, building a peasant requires a certain amount of gold; the player needs to wait until that amount is reached before actually proceeding to build the peasant. All the following expectations check the preconditions of the next action plus some additional conditions.
– **Immediate**. It checks if the effects of the last executed action has been completed. This can be useful in highly reactive environments such as first-person shooter games were for example the agent is checking if after jumping it reaches the targeted platform.
– **State**. This is the most common form of expectations used in GDA. The agent checks if the conditions in the state obtained by projecting from the starting state after each action executed so far match the observations in the environment. This is a natural way for GDA agent using a variety of conditions in the state including symbolic [25], ontological [6] and numerical [38]. It ensures that the part of the plan that remains to be executed, it is still executable from the current state.
– **Informed**. This is a form we created [9]. Instead of projecting the whole state, it projects only the accumulated effects from the actions executed so far. The basic idea is that only the accumulated effects are the necessary conditions that should be checked against the actual state. There might be observations in the actual state that do not match the projected state, but if those observations are not propagated conditions, there is no need to trigger the GDA process. For example, if in an RTS game, the plan π calls for taking a resource area in the center of the map, and while executing this plan it detects some neutral units in a nearby location that pose no threat to the plan, they can simply ignore them, without triggering an unnecessary GDA cycle. In contrast, state expectations will trigger an iteration of the GDA cycle every time any observation doesn't match the expected state.
– **informed-k**. Informed-k expectations are informed expectations that are checked every k steps, where k is an input parameter to the agent. These expectations were created for situations when there are costs associated with sensing if a condition in the expectations matches an actual observation in the state [7]. For example, a vehicle might need to stop to turn the sensors, in which case the cost can be measured in terms of the time it needs to remain immobile.

Various experiments were made using variations of benchmark domains from the literature such as the Mudworld [23] and the Arsonist [30] domains. In these experiments. all other components of the GDA agent were the same and the only change was the type of expectations they were computing [8, 9, 7]. The following observations were made: none and immediate expectations had no or very low sensing costs but GDA agents frequently failed to achieve the goals it expected to have achieved. GDA agents using state expectations achieved all goals that the agent expected to achieve but had the highest sensing costs compared to agents using other forms of expectations. Using informed expectations also achieve all goals but at a much lower costs than when using state expectations. Depending on the parameter k, Informed-k expectations also achieved all goals while having the lowest cost.

6 Current and future research directions

We are currently exploring a variety of directions. First, existing work on expectations (ours and others) assume the plan π as a sequence of actions (possibly augmented with a hierarchy of tasks). When the solution π is a policy, existing research is looking at the execution trace from following the policy [17]; that is, a sequence of actions. We are currently exploring goal regression for policies, which unlike for the case of sequences of actions, cannot guarantee the minimal set of necessary conditions as opposed to the deterministic case [32]. We are exploring combining informed and goal regression expectations for the non-deterministic case [31].

Second, we also will like to explore the definitions of different forms of expectations for domains that include a combination of numeric and symbolic variables and outcomes are dictated by (a possible unknown) probability distribution. This includes situations when, for example, the agent's gasoline consumption is determined using a probability distribution while at the same time it is ascertaining if a message was sent or not.

Third, we will like to explore the interaction between the different components of the GDA process. For example, how the learned planning knowledge affects the expectations; in turn how discrepancies generated as a result of expectations elicited in this particular way affect the kinds of explanations generated and how this in turn determine the goals formulated. For this purpose we need a common plan representation formalism. We recently adopted the hierarchical goal networks (HGNs) formalism [34] instead of hierarchical task networks (HTN) for the planning formalism used by the planner Π. Unlike HTNs, HGNs represent goals, not tasks, at all echelons of the hierarchy. This has been shown to be useful to learn the hierarchies [3]; it also been shown to be particularly suitable for execution monitoring [11, 10] as the agent can determine directly if a goal is achieved; unlike tasks in HTNs that do not have explicit semantics. In HTNs, the only way to fulfill tasks is by executing the underlying plan generated from the HTN planning process. HGNs have the same expressiveness as total-oder HTN planning [34].

Acknowledgements. This research is supported by ONR under grant N00014-18-1-2009 and by NSF under grant 1217888. No work of this scope can be done by a single person; I will like to thank the following external collaborators: David W. Aha and David Wilson (Naval Research Laboratory), Michael T. Cox and Matthew Molineaux (Wright State University); I will also like to thank the following (current and former) students: Dustin Dannenhauer, Chad Hogg, Sriram Gopalakrishnan, Morgan Fine-Morris, Noah Reifsnyder and Ulit Jaidee.

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI communications **7**(1), 39–59 (1994)
2. Aha, D.W.: Goal reasoning: foundations emerging applications and prospects. AI Magazine. *Under review* (2018)
3. Choi, D., Langley, P.: Learning teleoreactive logic programs from problem solving. In: International Conference on Inductive Logic Programming. pp. 51–68 (2005)
4. Coddington, A.M., Luck, M.: A motivation-based planning and execution framework. International Journal on Artificial Intelligence Tools **13**(01), 5–25 (2004)
5. Cox, M.T.: Perpetual self-aware cognitive agents. AI magazine **28**(1), 32 (2007)
6. Dannenhauer, D., Munoz-Avila, H.: Luigi: a goal-driven autonomy agent reasoning with ontologies. Advances in Cognitive Systems (ACS-13) (2013)
7. Dannenhauer, D.: Self monitoring goal driven autonomy agents. Ph.D. thesis, Lehigh University (2017)
8. Dannenhauer, D., Munoz-Avila, H.: Raising expectations in gda agents acting in dynamic environments. In: IJCAI. pp. 2241–2247 (2015)
9. Dannenhauer, D., Munoz-Avila, H., Cox, M.T.: Informed expectations to guide gda agents in partially observable environments. In: IJCAI. pp. 2493–2499 (2016)
10. Dvorak, D.D., Ingham, M.D., Morris, J.R., Gersh, J.: Goal-based operations: An overview. JACIC **6**(3), 123–141 (2009)
11. Dvorak, D.L., Amador, A.V., Starbird, T.W.: Comparison of goal-based operations and command sequencing. In: Proceedings of the 10th International Conference on Space Operations (2008)
12. Finestrali, G., Munoz-Avila, H.: Case-based learning of applicability conditions for stochastic explanations. In: International Conference on Case-Based Reasoning. pp. 89–103. Springer (2013)
13. Floyd, M.W., Karneeb, J., Aha, D.W.: Case-based team recognition using learned opponent models. In: International Conference on Case-Based Reasoning. pp. 123–138. Springer (2017)
14. Fox, M., Gerevini, A., Long, D., Serina, I.: Plan stability: Replanning versus plan repair. In: ICAPS. vol. 6, pp. 212–221 (2006)
15. Ghallab, M., Nau, D., Traverso, P.: The actor??s view of automated planning and acting: A position paper. Artificial Intelligence **208**, 1–17 (2014)
16. Hogg, C., Muñoz-Avila, H., Kuter, U.: HTN-MAKER: Learning HTNs with minimal additional knowledge engineering required. In: Conference on Artificial Intelligence (AAAI). pp. 950–956. AAAI Press (2008)
17. Jaidee, U., Muñoz-Avila, H., Aha, D.: Learning and reusing goal-specific policies for goal-driven autonomy. Case-Based Reasoning Research and Development pp. 182–195 (2012)
18. Jaidee, U., Muñoz-Avila, H., Aha, D.W.: Integrated Learning for Goal-Driven Autonomy. In: Proceedings of the Twenty-Second international joint conference on Artificial Intelligence-Volume Volume Three. pp. 2450–2455. AAAI Press (2011)
19. Jaidee, U., Muñoz-Avila, H., Aha, D.W.: Case-based goal-driven coordination of multiple learning agents. In: International Conference on Case-Based Reasoning. pp. 164–178. Springer (2013)
20. Keller, T., Eyerich, P.: Prost: Probabilistic planning based on uct. In: ICAPS (2012)
21. Langley, P.: The cognitive systems paradigm. Advances in Cognitive Systems **1**, 3–13 (2012)

22. Molineaux, M.: Understanding What May Have Happened in Dynamic, Partially Observable Environments. Ph.D. thesis, George Mason University (2017)
23. Molineaux, M., Aha, D.W.: Learning unknown event models. In: AAAI. pp. 395–401 (2014)
24. Molineaux, M., Klenk, M., Aha, D.W.: Goal-Driven Autonomy in a Navy Strategy Simulation. In: AAAI (2010)
25. Muñoz-Avila, H., Jaidee, U., Aha, D., Carter, E.: Goal-Driven Autonomy with Case-Based Reasoning. In: Case-Based Reasoning. Research and Development, pp. 228–241. Springer (2010)
26. Muñoz-Avila, H., Wilson, M.A., Aha, D.W.: Guiding the ass with goal motivation weights. In: Goal Reasoning: Papers from the ACS Workshop. pp. 133–145 (2015)
27. Nau, D.S., Cao, Y., Lotem, A., Muñoz-Avila, H.: SHOP: Simple hierarchical ordered planner. In: Dean, T. (ed.) International Joint Conference on Artificial Intelligence (IJCAI). pp. 968–973. Morgan Kaufmann (Aug 1999)
28. Ontanón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., Preuss, M.: A survey of real-time strategy game ai research and competition in starcraft. IEEE Transactions on Computational Intelligence and AI in games **5**(4), 293–311 (2013)
29. Oxenham, M., Green, R.: From direct tasking to goal-driven autonomy for autonomous underwater vehicles. In: 5th Goal Reasoning Workshop at IJCAI-2017 (2017)
30. Paisner, M., Maynord, M., Cox, M.T., Perlis, D.: Goal-driven autonomy in dynamic environments. In: Goal Reasoning: Papers from the ACS Workshop. p. 79 (2013)
31. Reifsnyder, N., Munoz-Avila, H.: Goal reasoning with goldilocks and regression expectations in nondeterministic domains. In: 6th Goal Reasoning Workshop at IJCAI/FAIM-2018 (2018)
32. Reiter, R.: The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In: Lifschitz, V. (ed.) Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy, (Ed.). Academic Press (1991)
33. Roberts, M., Vattam, S., Alford, R., Auslander, B., Apker, T., Johnson, B., Aha, D.W.: Goal reasoning to coordinate robotic teams for disaster relief. In: Proceedings of ICAPS-15 PlanRob Workshop. pp. 127–138. Citeseer (2015)
34. Shivashankar, V.: Hierarchical Goal Network Planning: Formalisms and Algorithms for Planning and Acting. Ph.D. thesis, Dept. of Computer Science, University of Maryland (2015)
35. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press Cambridge (1998)
36. Warfield, I., Hogg, C., Lee-Urban, S., Munoz-Avila, H.: Adaptation of hierarchical task network plans. In: FLAIRS conference. pp. 429–434 (2007)
37. Weber, B.G., Mateas, M., Jhala, A.: Applying goal-driven autonomy to starcraft. In: AIIDE (2010)
38. Wilson, M.A., McMahon, J., Aha, D.W.: Bounded expectations for discrepancy detection in goal-driven autonomy. In: AI and Robotics: Papers from the AAAI Workshop (2014)
39. Wilson, M.A., Molineaux, M., Aha, D.W.: Domain-independent heuristics for goal formulation. In: FLAIRS Conference (2013)
40. Zhuo, H.H., Muñoz-Avila, H., Yang, Q.: Learning hierarchical task network domains from partially observed plan traces. Artificial intelligence **212**, 134–157 (2014)