Learning Event Models that Explain Anomalies

Matthew Molineaux[1], David W. Aha[2], and Ugur Kuter[3]

[1]Knexus Research Corporation; 9120 Beachway Lane; Springfield, VA 22153; USA
[2]Naval Research Laboratory, Code 5514; Washington DC; USA
[3]University of Maryland, Institute for Advanced Computer Studies,
College Park, MD 20742; USA
[1]matthew.molineaux@knexusresearch.com
[2]david.aha@nrl.navy.mil
[3]ukuter@cs.umd.edu

Abstract. In this paper, we consider the problem of improving the goal-achievement performance of an agent acting in a partially observable, dynamic environment, which may or may not know all events that can happen in that environment. Such an agent cannot reliably predict future events and observations. However, given event models for some of the events that occur, it can improve its predictions of future states by conducting an explanation process that reveals unobserved events and facts that were true at some time in the past. In this paper, we describe the DISCOVERHISTORY algorithm for discovering an explanation for a series of observations in the form of an event history and a set of assumptions about the initial state. When knowledge of one or more event models is not present, we claim that the capability to learn these unknown event models would improve performance of an agent using DISCOVERHISTORY, and provide experimental evidence to support this claim. We provide a description of this problem, and suggest how the DISCOVERHISTORY algorithm can be used in that learning process.

Keywords: Abductive explanation, planning and execution, relational learning

1 Introduction

Recent research in robotics has demonstrated the need for intelligent agents that can recognize dangers to themselves and respond proactively, before humans can communicate their needs. For example, the Mars rovers must navigate for many hours at a time without a human in the loop, and the current generation of autonomous underwater vehicles can operate for similar periods of time. To robustly recognize emerging dangers, these types of agents must reason about the state information that they can observe, and continuously learn new information about how the world works.

Most commonly, agents that reason about many possible states use a *belief state* representation (Russell and Norvig, 2003), in which an agent plans over the set of "belief states", which is the power set of standard states. While effective in small domains, the size of the belief state in use grows exponentially with the number of hidden literals. We consider instead a model in which the agent begins by making assumptions about hidden facts in the world (most often a closed world assumption), and then *explains*

Report Documentation Page

1. REPORT DATE **2011**	2. REPORT TYPE	3. DATES COVERED **00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE **Learning Event Models that Explain Anomalies**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Research Laboratory,Navy Center for Applied Research in Artificial Intelligence,Code 5514,Washington,DC,20375**	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
In T. Roth-Berghofer, N. Tintarev, & D.B. Leake (Eds.) Explanation-Aware Computing: Papers from the IJCAI Workshop. Barcelona, Spain.

14. ABSTRACT
In this paper, we consider the problem of improving the goalachievement performance of an agent acting in a partially observable, dynamic environment, which may or may not know all events that can happen in that environment. Such an agent cannot reliably predict future events and observations. However, given event models for some of the events that occur, it can improve its predictions of future states by conducting an explanation process that reveals unobserved events and facts that were true at some time in the past. In this paper we describe the DISCOVERHISTORY algorithm for discovering an explanation for a series of observations in the form of an event history and a set of assumptions about the initial state. When knowledge of one or more event models is not present, we claim that the capability to learn these unknown event models would improve performance of an agent using DISCOVERHISTORY, and provide experimental evidence to support this claim. We provide a description of this problem and suggest how the DISCOVERHISTORY algorithm can be used in that learning process.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**10**	

what has happened when unexpected state observations occur. This is made possible by modeling the types of exogenous *events* that can occur, and determining what sequences of such events are consistent with the observations. Unlike work in diagnosis (e.g., McIlraith 1998; Iwan and Lakemeyer 2003), we treat these events as *uncontrollable*; that is, the rules of the world *require* them to happen when certain conditions are met. This treatment is more realistic for many environments and, by narrowing the set of possible explanations, it allows an agent to correct initial assumptions about the world. An agent can use these events in planning as well as in explanation, which allows explanations to improve predictions made in the future.

We previously created an algorithm, DISCOVERHISTORY, that corrects assumptions in dynamic, partially observable environments (Molineaux et al., in submission). It assumes that the space of possible events is fully available to the agent. However, this assumption is too strong to meet the needs of real-world agents that must operate on their own. How should a Mars Rover respond, for example, if it unexpectedly encounters a Martian parasite? Clearly its designers cannot plan for all possible events, so the robot must respond without a good description of how interstellar parasites behave. Ideally, it should observe the parasite's behaviors and explain what events cause them, thus learning its own model of environment events.

This paper is organized as follows. Section 2 surveys related work in explanation and diagnosis. Section 3 reviews a formalism for explanation of the past and DISCOVERHISTORY. Section 4 introduces an environment designed to examine its behavior, while Section 5 describes an empirical study that examines the benefits of improving the environment model used by DISCOVERHISTORY. Section 6 sketches an algorithm for learning event models, and Section 7 concludes.

2 Related Work

Our work extends the work described by Molineaux et al (in submission), who introduced DISCOVERHISTORY but, unlike here, assumed that the agent is given knowledge on the complete space of possible events. Here we focus on explaining unexpected states by learning the models of *undetected* events. That is, the role of explanations in this paper is to help an agent to understand its environment, rather than communicate its reasoning to humans. This differs from other visions on the role of explanations in AI systems, such as explanation-aware computing, which concerns software-user interactions (e.g., Atzmeuller and Roth-Berghofer, 2010). Likewise, self-explaining agents (e.g., Harbers et al., 2010) and explainable AI systems (van Lent et al., 2004) can generate explanations of their behavior, again for human consumption. While several methods have been investigated that allow agents to explain their observations (e.g., Schank and Leake (1989) describe a case-based approach), we are not aware of similar approaches that learn event models by abducing explanations from unexpected state changes.

While many automated planners can operate in partially observable dynamic environments, most ignore this challenging learning problem. Rather than attempt to discover the causes of the unexpected state, they instead dynamically replan from it (e.g., Myers, 1999; Ayan et al., 2007; Yoon et al., 2007). Those few exceptions that *do* learn tend to focus on other learning tasks, such as selecting which plans to apply (Corchado

et al., 2008), reducing the time to generate optimal plans when replanning (Rachelson et al., 2010), or reducing the need for future replanning tasks (Jimenez et al., 2008).

Reinforcement learning techniques can be used to learn policies for solving planning problems in partially-observable environments, such as by modeling tasks as Partially Observable Markov Decision Processes (POMDPs) (Russell and Norvig, 2003). Our work differs because we use a relational representation for environment models that permits greater scalability and greater generalization.

Using an extension of the situation calculus, McIlraith (1998) describes an approach that uses goal regression and theorem proving to generate diagnostic explanations, defined as conjectured sequences of actions that caused an observed but unexpected behavior. Iwan and Lakemeyer (2003) addressed related problems using a similar approach, though they did not distinguish actions from events, they focused on single observations, and they did not empirically test an implementation of their formalization. Sohrabi et al. (2010) followed up with an empirical analysis of planning techniques to perform diagnosis, and extended the implementation to also find a consistent set of initial assumptions. Their problem is similar to ours, except that they treat events as actions, which allows them to occur without a strict causal relationship to a prior event. Furthermore, these systems are not incorporated into a plan-executing agent.

ASAS (Josephson and Bharathan 2006) is a "dynamic abducer" that can continuously maintain an explanation of its environment and revise its beliefs. However, it is not clear whether this process of belief revision is used to improve future predictions.

Research on learning goals (Cox and Ram, 1999; Ram and Leake, 1995) is also related in that explanation failures are used to trigger learning. However, these goals are not related to the problem of learning event models or choosing actions.

3 Planning and Explanation

We are interested in algorithms that achieve goals in partially observable, dynamic domains by executing a sequence of actions. We model the task of finding a sequence of actions to accomplish a goal as a planning problem, and employ a Hierarchical Task Network planner, SHOP2, for this purpose (Nau et al, 2003). However, since our domains are partially observable and dynamic, the world is not entirely predictable. Rather than enumerating all possible states resulting from its actions, which is highly computationally expensive, our agent makes assumptions about the environment and creates a plan that will accomplish its goals if those assumptions hold. During execution, if the agent's assumptions are incorrect, the agent will observe facts that do not match its predictions. We call this difference between predictions and observations a *discrepancy* or *anomaly*. Section 3.1 presents our definition of explanations for these observations.

To recover from anomalies, we use DISCOVERHISTORY, which corrects the agent's assumptions about the environment by searching for an explanation of the anomaly. This explanation will include a revised set of assumptions about the world, which can then be used to improve the next planning iteration. Having been informed more about the environment, the planner can now create a better plan. Section 3.2 summarizes the DISCOVERHISTORY algorithm.

3.1 Definitions

We use the standard definitions from classical planning for variable and constant symbols, logical predicates and atoms, literals, groundings of literals, propositions, and actions (Nau et al, 2003). We assume a state is encoded by a set of logical propositions.

Let $\mathcal{P}$ be the finite set of all possible propositions for describing an environment. An environment is partially observable if an agent only has access to it through *observations* that do not cover the complete state, where an observation is encoded as a set of propositions. We let $\mathcal{P}_{\mathsf{obs}}$ be the set of all propositions that the agent can observe in the world. An observation associates a truth value with each of these propositions. Further, let $\mathcal{P}_{hidden}$ be the set of *hidden* (state) propositions that an agent *cannot* observe; for example, a robot's exact location may be hidden to it if it has no GPS contact.

An *event template* is defined syntactically the same as a classical planning operator: (name, preconds, effects), where name, the *name* of the event, is a literal, and preconds and effects, the *preconditions* and *effects* of the event, are sets of literals. We assume that an event always occurs immediately when all of its preconditions are met in the world.

We formalize the planning agent's knowledge about the changes in its environment as an *explanation* of the world. We define a finite set of symbols $\mathcal{T} = \{t_0, t_1, t_2, \ldots, t_n\}$, called *occurrence points*. An *ordering relation* between two occurrence points is denoted as $t_i \prec t_j$, where $t_i, t_j \in \mathcal{T}$.

There are three types of *occurrences*. An *observation occurrence* is a pair of the form (obs, t) where obs is an observation. An *action occurrence* is a pair of the form (a, t) where a is an action (performed by the agent). Finally, an *event occurrence* is a pair (e, t) where e is an event. In all of the occurrence forms, t is an occurrence point. Given an occurrence o, we define occ as a function such that $\mathsf{occ}(o) \mapsto t$; that is, occ refers to the occurrence point t of any observation, action, or event.

An *execution history* is a finite sequence of observations and actions obs_0, a_1, obs_1, a_2, ..., a_k, obs_{k+1}. A planning agent's *explanation* of the world given an execution history is a tuple $\chi = (C, R)$ such that C is a finite set of occurrences that includes the execution history and zero or more event occurrences that happened according to that explanation. R is a partial ordering over a subset of C, described by ordering relations $\mathsf{occ}(o_i) \prec \mathsf{occ}(o_j)$ such that $o_i, o_j \in C$. As a shorthand, we sometimes will say $o_i \prec o_j$, which is true when $\mathsf{occ}(o_i) \prec \mathsf{occ}(o_j)$.

We use the definitions $\mathsf{knownbefore}(p, o)$ and $\mathsf{knownafter}(p, o)$ to refer to the truth of a proposition p before or after an occurrence o occurs. Let o be an action or event occurrence. Then, the relation $\mathsf{knownbefore}(p, o)$ is true when $p \in \mathsf{preconds}(o)$. Similarly, the relation $\mathsf{knownafter}(p, o)$ is true when $p \in \mathsf{effects}(o)$. If o is an observation occurrence and $p \in \mathsf{obs}$, then both $\mathsf{knownbefore}(p, o)$ and $\mathsf{knownafter}(p, o)$ hold.

We say that an occurrence o is *relevant* to a proposition p if the following holds: $\mathsf{relevant}(p, o) \equiv \mathsf{knownafter}(p, o) \vee \mathsf{knownafter}(\neg p, o) \vee \mathsf{knownbefore}(p, o) \vee \mathsf{knownbefore}(\neg p, o)$. We use the predicates $\mathsf{prior}(o, p)$ and $\mathsf{next}(o, p)$ to refer to the prior and next occurrence relevant to a proposition p. That is to say, $\mathsf{prior}(o, p) = \{o' \mid \mathsf{relevant}(p, o') \wedge \neg \exists o'' s.t. \mathsf{relevant}(p, o'') \wedge o' \prec o'' \prec o\}$. Similarly, $\mathsf{next}(o, p) = \{o' \mid \mathsf{relevant}(p, o') \wedge \neg \exists o'' s.t. \mathsf{relevant}(p, o'') \wedge o \prec o'' \prec o'\}$.

The *proximate cause* of an event occurrence (e, t) is an occurrence o that satisfies the following three conditions with respect to some proposition p: (1) $p \in$ preconds(e), (2) knownafter(p, o), and (3) there is no other occurrence o' such that $o \prec o' \prec (e, t)$. Every event occurrence (e, t), must have at least one proximate cause, meaning that by condition 3, every event occurrence (e, t) must occur immediately after its preconditions are satisfied.

An *inconsistency* is a tuple (p, o, o') where o and o' are two occurrences in χ such that knownafter$(\neg p, o)$, knownbefore(p, o'), and there is no other occurrence o'' such that $o \prec o'' \prec o' \in R$ and p is relevant to o''.

An explanation $\chi = (C, R)$ is *plausible* if and only if the following holds:

1. There are no inconsistencies in χ;
2. Every event occurrence $(e, t) \in \chi$ has a proximate cause in χ;
3. Simultaneous occurrences cannot contradict one other: for each pair $o, o' \in C$ such that $occ(o) = occ(o')$, and for all p, knownafter$(p, o) \implies \neg$knownafter$(\neg p, o')$, and knownbefore$(p, o) \implies \neg$knownbefore$(\neg p, o')$.
4. If preconds(e) of an event e are all met at occurrence point t, e must be in χ at t.

3.2 DISCOVERHISTORY algorithm

The DISCOVERHISTORY algorithm (Molineaux et al., 2011) finds a plausible explanation that is consistent with an execution history. At first, our agent makes a set of "optimistic" closed-world assumptions, assuming that the obstacles it knows about are not present, and makes a plan based on those assumptions. Later, when an anomaly occurs, it can revisit these assumptions and replan. An anomaly occurs as a result of one of two problems: (1) there are hidden facts and/or event types present in the environment for which the agent has no model, or (2) the agent's initial optimistic assumptions are false. The first problem requires learning, which we address in Section 6. The second problem means that the agent can improve its environment model by correcting its explanation, which will require changing its assumptions.

The agent maintains a current explanation at all times. As it executes actions, the agent makes predictions about what events will occur, according to its observations and assumptions, and adds those events to its explanation. However, in a partially-observable, dynamic world, events occur that the agent is unable to predict, which cause anomalies. The agent recognizes an anomaly because one or more inconsistencies appear in its explanation. As described in Section 3.1, an inconsistency occurs when a fact is true after a relevant occurrence, and false before the next relevant occurrence. When these inconsistencies appear, DISCOVERHISTORY resolves them, using one of the following three inconsistency resolution methods.

1. **Event addition**. This searches for a situation where an event that was not predicted caused a literal's value to change. The new event must occur before the later occurrence, so as to explain the value at that time, but not before the earlier occurrence.
2. **Event removal**: This searches for a situation where one of the predicted events did not happen as predicted.

3. **Assumption change**. Assumptions are only made about the initial state in our explanations, since all following states can be determined from the initial state in a deterministic world. Any event prior to the inconsistency would provide more information about the truth of the literal in question. Therefore, this resolution method is only applied when no relevant prior event is present in the explanation.

Each of these resolution methods can introduce new inconsistencies. Thus, DISCOVER-HISTORY continues resolving inconsistencies until a consistent explanation is found, or until a maximum number M of changes have been made. This search exhaustively enumerates all explanations within M applications of resolution methods.

When a plausible explanation is found, the assumptions made by that explanation are adopted and can be used in prediction. This results in fewer surprises. Since many such surprises are deleterious, goals can be achieved more often, as shown in Section 5. A more detailed explanation of DISCOVERHISTORY is in (Molineaux et al, 2011).

4   The Hazardous Rover Domain

Inspired by the Mars rovers,[1] which operate in a very hazardous environment, we constructed a domain for testing that is predictable enough to make planning tractable, but requires significant explanation to perform well. In our domain, the only actions are to move in one of the four cardinal directions, and the only goals are to reach a location on a six-by-six grid. However, complications can prevent a rover from reaching its goal:

1. Sandstorms, which obscure visibility and can blow a moving rover off course
2. Hidden sand pits, which trap a rover and prevent it from moving
3. Compass malfunctions, which reverse a rover's navigational system, causing it to move north when it intended to move south

Each of these obstacles provides a different challenge for learning. Consider the problem of a rover that has not been programmed to recognize a sandstorm. When the windy season on Mars starts, the rover's sensor mechanisms become covered in sand, and it can no longer recognize its location. It can predict where it should be, but its predictions may deviate more from reality when strong sandstorm winds blow it off course. When the storm clears, the rover observes that it has not reached its desired location. This is an opportunity to learn; the rover could explain that the unexpected change in its position occurs only *when the rover is covered in sand*.

Next consider the case of a rover that finds itself in a hidden sand pit. The rover observes that it can not move any longer when it reaches a certain location. Later on, another rover gets stuck nearby. Due to the same results occurring to both rovers at the same location, it could explain that rovers get stuck *at a certain location*.

Finally, consider the case of compass malfunctions. In early afternoon one day, the rover attempts to move north, and finds that it ends up in a different location than the one it expected, which could have been reached by going to the south. It tries moving north repeatedly, and repeatedly observes a similar result, a location one to the south

---

[1] http://marsrover.nasa.gov/home/

of its last location. By finding the similarities between the transitions, the rover could explain that an event occurred *in early afternoon* that affected its navigation.

With foreknowledge of these types of events, we have shown in prior work (Molineaux et al, 2011) that an agent employing DISCOVERHISTORY outperforms one that only performs replanning and execution monitoring. We now investigate whether an agent that can learn these event models would outperform one that could not.

5   Experiment: Potential for Event Learning

To examine the potential effects of learning in this domain, we compare the performance of several explanation-forming agents with different levels of knowledge about the domain. We measure performance in this domain as the percentage of goals achieved. As explained earlier, the Hazardous Rovers Domain includes three types of obstacles. For each obstacle, one event related to understanding that obstacle is considered as a target for learning. With regard to sandstorms, the event that causes a rover to blow off course in a sandstorm is *sandstorm-blows*; for sand pits, the event that causes the area around a hidden sand-pit to appear abnormally rough is *pit-causes-rough*; the event that causes a compass to malfunction is called *compass-malfunction-occurs*. Eight agents using the DISCOVERHISTORY algorithm are compared. The first agent has knowledge of none of these events; three agents know about one; three agents know about two; and the last agent has knowledge of all three. If learning is effective in this domain, we expect improvement from an agent with less knowledge to an agent with more.

To compare the agents, we randomly generated 25 scenarios in the hazardous rover domain. In each scenario, three rovers are each assigned a random initial location and a random goal that requires at least four moves to reach from the starting location. Each location on the six-by-six grid contains a sand pit with probability 0.3, and with probability 0.66 the sand pit will be hidden. At each possible time point, a storm may begin with probability 0.2, unless a storm is ongoing at that time. Finally, each rover has a compass malfunction within the first 100 time steps. The time step is chosen by taking the floor of the real number found by squaring a number randomly distributed between 0 and 10. This biases the event to occur early in a scenario execution.

Table 1 shows the difference in the average percentage of goals achieved between each pair of agents which differ only by the knowledge of a single event. When all other events are known, the difference averages 13.8%, meaning that the additional knowledge of a single event was sufficient to achieve success instead of failure in 13.8% of cases. The performance advantage caused by all 3 events together is 32.0%, with statistical significance at p < .0001. 9 out of 12 comparisons showed that knowledge of an additional event improved goal achievement at a statistically significant level, indicating that learning event models is a promising avenue.

6   Discovering Events

Now that we've shown the utility of event learning in improving the performance of an explanation discovery agent in a partially observable, dynamic planning domain, we consider the problem of inducing those events. This can be neatly divided into two

Table 1: Increase in goals achieved after adding event models. (N/S): non-significant.

Known Events	Perf.	Known events	Perf.	Known events	Perf.
compass-malfunctions pit-causes-rough	18.6%, p<.01	storm-blows pit-causes-rough	12.0%, p<.01	compass-malfunctions storm-blows	10.7%, p<.05
compass-malfunctions pit-causes-rough	14.7%, p<.01 16.0%, p<.01	storm-blows pit-causes-rough	5.33%, p<.05 9.32%, p<.05	compass-malfunctions storm-blows	6.66%, (N/S) 4.00%, (N/S)
none	16.0%, p<.01	none	6.66%, p<.05	none	4.00%, (N/S)

(a) Effect of adding storm-blows   (b) Effect of adding compass-malfunctions   (c) Effect of adding pit-causes-rough

separate problems: discovering when something unexplained happens, and finding an effect that explains it. In a first-order logic domain, the first problem results in a set of literal *preconditions,* and the second in a set of literal *effects.* If found, such an event could be used to explain the anomalies encountered by the agent. However, if all of the preconditions of an event are hidden, and no evidence is available to determine their values, then that event cannot be discovered. Therefore, we assume that at least one precondition is observable. Similarly, since hidden effects are difficult to verify, we assume that all events cause at least one observable effect. We discuss these two problems, discovery of preconditions and effects, in Sections 6.1 and 6.2 respectively.

When events are discovered, we need metrics to use in evaluating and adopting those events. We propose to use *predictive power* and *explanatory power*. If an event is usually predicted before it occurs, and rarely predicted before it does not occur, it has high predictive power. Similarly, if knowledge of an event enables explanation after it occurs, and does not lead to false explanations when it has not occurred, it has high explanatory power. While both are useful as metrics of success, they cannot be easily evaluated *in situ* by an agent. However, an agent can approximate them by maintaining a history of the conditions and results of prior predictions and explanations. By re-attempting each prior prediction, the agent can evaluate how many predictions would have been made incorrect by the inclusion of the event in the agent's knowledge and how many would have been made correct. Similarly, explanations can be re-attempted with the new knowledge; failed explanations that become successful are successes, and successful explanations for which no explanation is now possible are failures. By adopting a confidence metric over predictive and explanatory power, an agent can decide when to include a discovered event in its knowledge base for use in future explanation attempts.

6.1 Discovering Event Preconditions

In the simplest case, all anomalies can be attributed to a single event. This may be a useful assumption in many cases (e.g., when conditions gradually change an agent might be exposed to new types of events incrementally). When the windy season starts on Mars, for example, the rover may already be prepared with knowledge of equipment malfunctions and hidden sand pits. Its problem is to determine what is common among all the anomalies that are encountered. The classification of anomaly-causing states is a relational multi-instance learning task (Dietterich et al, 1997) over the complete

history of observations made by the agent; any sequence of occurrences that contains an anomaly is represented as a "bag" of positive examples, and all other sequences are included in a bag of negative examples. Each example consists of a set of literals known to be true at one time.

As an example, suppose the rover moves to an area on Mars that is covered with hidden sand pits, and surrounded by rough terrain. The rover frequently gets caught in these pits, and each time the experience is anomalous, because it doesn't know how to predict. The rover builds up a memory of anomalies and the states that preceded them. The rover solves the learning task described above to discover that each anomaly was preceded by states in which rough ground was present and a sand pit was not nearby. These two facts become the preconditions for a new generated event, G1.

6.2 Discovering Event Effects

After discovering the conditions that caused the recognized anomalies, we can assume an event model exists that is triggered by those conditions. We can now predict when this event will occur, but its effects are still unknown. To find the effects of this event, we can add it to the knowledge base as an event model with no effects, and apply a modified version of DISCOVERHISTORY with a new inconsistency resolution method. This method will resolve inconsistencies whose durations overlap with the new event; it will resolve them by assuming that the event has an effect that causes the inconsistency. All such assumptions made in a plausible explanation constitute one possible set of effects for the event model. This modified DISCOVERHISTORY could be used to find a set of plausible explanations for each recognized anomaly. Therefore, using DISCOVERHISTORY to explain all recognized anomalies will result in a set of possible event models, which can be evaluated in terms of their predictive and explanatory power.

Moving on with the hidden sand pits example (Section 6.1), the rover applies DISCOVERHISTORY to resolve inconsistencies. Suppose that the rover observes a literal (`wheels-stuck`) which prevents the move action from working. The rover's explanation now includes a G1 event that occurred after moving to a rocky area. Now, the inconsistency can be explained: (`wheels-stuck`) is an effect of the G1 event.

7 Conclusion

Our empirical studies showed that learning event models can improve the performance of an agent using DISCOVERHISTORY in a partially-observable dynamic environment. We proposed a framework for acquiring these models from experience. Our next steps will be to implement this framework and evaluate its ability to (1) discover unknown event models and (2) improve its goal-achievement performance over time.

Acknowledgments

References

1. Atzmueller, M., & Roth-Berghofer, T. (2010). Towards explanation-aware social software: Applying the mining and analysis continuum of explaining. In *Proceedings of the Fifth International Workshop on Explanation-aware Computing.* Lisbon, Portugal.
2. Ayan, N.F., Kuter, U., Yaman F., & Goldman R. (2007). Hotride: Hierarchical ordered task replanning in dynamic environments. In F. Ingrand, & K. Rajan (Eds.) *Planning and Plan Execution for Real-World Systems – Principles and Practices for Planning in Execution: Papers from the ICAPS Workshop*. Providence, RI.
3. Bharathan, V., & Josephson, J. (2006). *Logic Jnl IGPL.* 14 (2): 271-285.
4. Corchado, J.M., Glez-Bedia, M., De Paz, Y., Bajo, J., & De Paz, J.F. (2008). Replanning mechanism for deliberative agents in dynamic changing environments. *Computational Intelligence,* 24(2), 77-107.
5. Cox, M.T., & Ram, A. (1999). Introspective Multistrategy Learning: On the Construction of Learning Strategies, *Artificial Intelligence*, 112:1-55.
6. Dietterich, T.G., Lathrop, R.H., & Lozano-Perez, T. (1997). Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1–2) 31–71.
7. Harbers, M., Meyer, J.J., & van den Bosch, K. (2010). Explaining simulations through self explaining agents. *Journal of Artificial Societies and Social Simulation,* 13(1)4.
8. Iwan, G., & Lakemeyer, G. (2003). What observations really tell us. *Proceedings of the Twenty-Sixth Annual German Conference on Artificial Intelligence* (pp. 192-208). Hamburg, Germany: Springer.
9. Jimenez, S., Fernandez, F, & Borrajo, D. (2008). The PELA architecture: Integrating planning and learning to improve execution. *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence* (pp. 1294-1299). Chicago, IL: AAAI Press.
10. van Lent, M., Fisher, W., & Mancuso, M. (2004). An explainable artificial intelligence system for small-unit tactical behavior. *Proceedings of the Sixteenth Conference on Innovative Applications of Artificial Intelligence* (pp. 900-907). San Jose, CA: AAAI Press.
11. McIlraith, S.A. (1998). Explanatory diagnosis: Conjecturing actions to explain observations. *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning* (pp. 167-179). Trento, Italy: Morgan Kaufmann.
12. Molineaux, M., Kuter, U., and Klenk, M. (2011). What Just Happened? Explaining the Past in Planning and Execution. In *Proceedings of the Sixth International Workshop on Explanation-Aware Computing*.
13. Myers, K.L. (1999). A continuous planning and execution framework. *AI Magazine,* 20(4).
14. Nau, D., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). SHOP2: An HTN planning system. JAIR 20:379–404.
15. Rachelson, E., Abbes, A.B., & Diemer, S. (2010). Combining mixed integer programming and supervised learning for fast re-planning. *Proceedings of the Twenty-Second International Conference on Tools with Artificial Intelligence* (pp. 63-70). Arras, France: IEEE Press.
16. Ram, A., & Leake, D. (eds.). (1995). *Goal-Driven Learning*. MIT Press/Bradford Books.
17. Russell, S. J. & Norvig, P. (2002), *Artificial Intelligence: A Modern Approach (2nd Edition)*, Prentice Hall .
18. Schank, R.C., & Leake, D.B. (1989). Creativity and learning in a case-based explainer. *Artificial Intelligence*, 40(1-3), 353-383.
19. Sohrabi, S., Baier, J. A., & McIlraith, S. A. (2010). Diagnosis as planning revisited. In *Proceedings of the 12th International Conference on the Principles of Knowledge Representation and Reasoning*.
20. Yoon, S., Fern, A., & Givan, B. (2007). FF-replan: A baseline for probabilistic planning. *Proceedings for the Seventeenth International Conference on Automated Planning System*s (pp. 352–360). Providence, RI: AAAI Press.