Molineaux, M., Aha, D., & Kuter, U. (2012). Efficiently explaining deterministic exogenous events in partially observable environments (Technical Note AIC-12-081). Washington, DC: Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence.

# Efficiently Explaining Deterministic Exogenous Events in Partially Observable Environments

**Matthew Molineaux[1], David 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]Smart Information Flow Technologies, 211 North 1st Street Minneapolis, MN 55401, USA
[1]matthew.molineaux@knexusresearch.com [2]david.aha@nrl.navy.mil [3]ukuter@sift.net

**Abstract**

We consider the problem of continual planning (DesJardins et al. 1999) in hazardous partially-observable dynamic environments, where deterministic exogenous events that cannot be directly observed affect the state of the world and no plan can be guaranteed to succeed. In these environments, limited observability makes state transitions ambiguous and difficult to predict. To resolve this ambiguity, we have developed two versions of DiscoverHistory, an algorithm that understands its environment by abductively explaining changes in state through reference to event models. We provide an analysis of their computational complexity and an empirical comparison of their performance in terms of execution time and success rate at accomplishing goals. We show that use of explanation generation increases the success rate of a continual planning agent and provide an initial benchmark for efficiency of continual planning with explanation in standard domains.

## Introduction

Typical modern planning algorithms are designed to create plans that achieve goals in a range of modeled environments, but not to execute them. Contingent planners for partially observable environments such as Contingent-FF (Hoffmann and Brafman 2005), CLG (Albore, Palacios, and Geffner 2009), and SDR (Shani and Brafman 2011), create plans that succeed in a range of possible initial states, but fail to produce a plan if any of those initial states leads to failure. We call such environments where an agent does not know success is possible *hazardous*. We contend that achieving goals in hazardous environments requires a continual planning agent capable of interleaving planning, execution, and monitoring, because any initial plan must be faulty, and therefore require revision. Our approach to this problem requires the agent to make assumptions about the environment that allow it to proceed, then revise those assumptions as the agent obtains new information. We provide an algorithm that monitors the environment using an *explanation generation* procedure, which performs abductive reasoning to increase its understanding of the environment as execution proceeds.

Explanations are generated by the DiscoverHistory algorithm (DH), introduced in prior work (Molineaux, Kuter, and Klenk in press), which dynamically constructs a history of occurrences (i.e., actions, events, and observations) the agent understands to have happened in the environment. By ensuring that the explanation is consistent with observations, DH reasons about possible histories of the environment, which impact hidden state. To do so, DH requires the world to be modeled in terms of *deterministic exogenous events*, which cannot be predicted with certainty or observed. This is a significant difference from most planning models, which represent uncertainty with conditional or nondeterministic effects of actions. Deterministic exogenous events naturally describe transitions that occur with or without an agent's intervention when a set of conditions are met, such as an alarm going off or an apple falling from a tree. They have previously been studied in planning by Fox et al. (2005) and Gerevini et al. (2006), but their work covers "predictable" exogenous events, which occur at a known time, and they do not consider partially observable environments. Recent work in diagnosis treats exogenous events as actions, meaning that they are non-deterministic (Sohrabi, Baier, and McIlraith 2010) or ignores them entirely, as in traditional fault diagnosis. This is a simplifying assumption, because deterministic exogenous events are constrained to happen immediately when caused.

Novel contributions of this paper include an in-depth description of two versions of DH, an analysis of its computational complexity, and an empirical evaluation of DH's efficiency. To our knowledge, this is the first empirical comparison of a continuous planning algorithm for hazardous, partially observable domains; therefore, no existing systems are compared.

## Definitions and Notation

**Basics.** We use the standard definitions from classical planning for variable and constant symbols, logical predicates and atoms, literals, groundings of literals, propositions, planning operators and actions (Ghallab, Nau, and Traverso 2004, Chapter 2).

Let $\mathcal{P}$ be the finite set of all possible propositions that describe a planning environment. A planning environment is partially observable if an agent only has access to the environment through *observations* that do not cover the complete state. We let $\mathcal{P}_{obs}$ be the set of all propositions that the agent will observe when true. An observation associates a truth value with each of these propositions. We refer to other propositions as *hidden*.

An *event template* is syntactically identical to a classical planning operator: $(\mathsf{name}, \mathsf{preconds}, \mathsf{effects})$, where name, the *name* of the event, and preconds and effects, the *preconditions* and *effects* of the event, are sets of literals. An *event* is a ground instance of an event template. Unlike actions, events are triggered immediately when all of their preconditions are met.

**Explanations.** We formalize the planning agent's knowledge about recent 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 $(\mathsf{obs}, t)$ where obs is an observation, and $t$ is an occurrence point. An *action occurrence* is a pair of the form $(a, t)$ where $a$ is an action. Finally, an *event occurrence* is a pair $(e, t)$ where $e$ is an event. We define occ as a function such that $\mathsf{occ}(o) \mapsto t$; that is, occ refers to the occurrence point $t$ of any occurrence $o$. $R$ is a partial ordering over occurrence points that comprises relationships of the form $t_i \prec t_j$ where $t_i, t_j \in \mathcal{T}$. As a shorthand, we sometimes will speak of the ordering between two occurrences: $o_i \prec o_j$, which means that $\mathsf{occ}(o_i) \prec \mathsf{occ}(o_j)$.

An *execution history* is a finite sequence of observations and actions $\mathsf{obs}_0, a_1, \mathsf{obs}_1, a_2, \ldots, a_k, \mathsf{obs}_{k+1}$. A planning agent's *explanation* of the world given an execution history is denoted as $\chi$, where $\chi$ is a finite set of zero or more event occurrences that are considered as having occurred in some possible world.

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

We say that an occurrence $o$ is *relevant* to a proposition $p$ if the following holds:

$$\begin{aligned}\mathsf{relevant}(p, o) \equiv \\ &\mathsf{knownafter}(p, o) \vee \mathsf{knownafter}(\neg p, o) \vee \\ &\mathsf{knownbefore}(p, o) \vee \mathsf{knownbefore}(\neg p, o).\end{aligned}$$

**Plausibility.** A *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) $p \in \mathsf{preconds}(e)$, (2) $\mathsf{knownafter}(p, o)$, 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, so every event occurrence must occur immediately after its preconditions are satisfied.

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

An explanation $\chi$ is *plausible* iff the following hold:

1. There are no inconsistencies in $\chi$;
2. Every event occurrence $(e, t) \in \chi$ has a proximate cause in $\chi$;
3. For every pair of simultaneous occurrences such that $o, o' \in \chi$ and $occ(o) = occ(o')$, there may be no conflicts before or after: for all $p$, $\mathsf{knownafter}(p, o) \implies \neg\mathsf{knownafter}(\neg p, o')$, and $\mathsf{knownbefore}(p, o) \implies \neg\mathsf{knownbefore}(\neg p, o')$.
4. If $\mathsf{preconds}(e)$ of an event $e$ are all satisfied at an occurrence point $t$, $e$ is in $\chi$ at $t$;

## Algorithms

Molineaux, Kuter, and Klenk (in press) previously described the DH algorithm, which performs an abductive search to find plausible explanations, or possible histories, of a partially-observable dynamic environment. At each level of search, DH applies a refinement to an existing, implausible explanation to resolve one of its inconsistencies. To increase the efficiency of this time-intensive search, we have studied two versions of DH ($\mathrm{DH}_1$ and $\mathrm{DH}_2$). In this section, we describe them in a higher level of detail than previously provided, and give an analysis of their computational complexity.

$\mathrm{DH}_1$ and $\mathrm{DH}_2$ pursue different strategies for finding inconsistencies and maintaining the partial ordering $R$ over all occurrence points. $\mathrm{DH}_1$ creates and maintains a list of inconsistencies. As it makes each refinement to an explanation, $\mathrm{DH}_1$ removes a targeted inconsistency from the list, and examines events relevant to the refinement to find any new inconsistencies; these are added to the list. To efficiently find these inconsistencies, the global ordering $R$ is represented by ordered lists of occurrences relevant to each proposition $p \in \mathcal{P}$, which must be updated during each refinement.

Instead of incrementally adjusting the set of inconsistencies and the global ordering, $\mathrm{DH}_2$ fixes the partial ordering $R$ in advance, and calculates the set of inconsistencies on demand. To fix the partial ordering, $\mathrm{DH}_2$ enumerates the set of possible event occurrences ($E_{poss}$) in advance. This relieves the burden of creating new occurrences and occurrence points during explanation. This makes the process of finding refinement simpler and removes the computational demand of maintaining $R$. However, the enumeration of all occurrences and on-demand calculation of inconsistencies add additional complexity. As a consequnce of pre-enumerating the possible occurrences, the branching factor of $DH_2$ is smaller; it considers only the set $E_{poss}$ as possible added events, instead of the larger set $E$.

In this section, we examine $DH_1$ and $DH_2$ by considering the types of refinements that are common to both, the FINDEXTRAEVENTS subroutine both use, and analyze their computational complexity.

### Inconsistency Refinements

**Adding an occurrence.** Let $\chi$ be an explanation with an inconsistency $i = (p, o, o')$. One way to resolve $i$ is to show that the value of a literal changed between the preceding event $o$ and the following event $o'$. This change can only occur due to some occurrence $o''$ relevant to $p$ such that $o \prec o'' \prec o'$. For example, if a robot issues an action to move itself from room A to room B, then observes that it is in room A, it might postulate that an event occurred after its action and before the observation, moving the robot back from room B to A.

To perform this refinement, $DH_1$ must enumerate all possible event occurrences $o''$ that add the proposition $p$. In contrast, $DH_2$ needs only to consider its pre-generated list of events $E_{poss}$ and filter that list based on the requirements enumerated above. Therefore, $DH_1$ considers more (unnecessary) refinements than $DH_2$.

**Removing an occurrence.** Another possible way to resolve an inconsistency $i = (p, o, o')$, where $o$ and/or $o'$ is an event occurrence, is to generate new explanations with either $o$ or $o'$ is removed. For example, if a robot issues an action to move itself from room A to room B, then observes that it is in room A, it is possible that the actual event of moving from A to B never happened. This raises the possibility that one of that event's conditions, such as the door being open, was not met.

Both DH algorithms handle this identically. First, a new explanation $\chi_{new}$ is generated with the event $o$ removed. Then, to explain why the occurrence does not happen, some $p' \in \mathsf{preconds}(o)$ is constrained not to hold at the occurrence point $\mathsf{occ}(o)$. This constraint takes the form of a *removal occurrence* $o_r$ where $\mathsf{occ}(o_r) = \mathsf{occ}(o)$. This occurrence has no effects and satisfies $\mathsf{preconds}(o_r) = \{\neg p'\}$. This removal occurrence prevents any future addition that would cause $o$ to occur. The maximum number of possible removal refinements is $|\mathsf{preconds}(o)| + |\mathsf{preconds}(o')|$.

**Hypothesizing an initial value.** Given an inconsistency $i = (p, o, o')$, where $o$ refers to the initial observation $\mathsf{obs}_0$, and $p$ is hidden, a new initial value for $p$ may be hypothesized. To do so, DH generates a new explanation by adding to $\chi$ an *initial value occurrence* $o_p, \mathsf{occ}(o_p) = t_0$. This occurrence has no preconditions, and satisfies $\mathsf{effects}(o_p) = \{p\}$. For example, if the robot discussed earlier was assuming the door was open, its closed state would be inconsistent. Correcting that assumption would resolve the inconsistency.

### The FINDEXTRAEVENTS Subroutine

The refinement methods of DH remove and add events as necessary to remove inconsistencies, but do not consider how these changes cause new events to happen. The FINDEXTRAEVENTS subroutine is responsible for this. When an explanation has no inconsistencies, FINDEXTRAEVENTS checks it for missing events and missing causes. This ensures that explanations meet requirements 2 and 4 of a plausible explanation (see Plausibility above), which implement deterministic event execution by requiring that events fire immediately when their conditions are met.

According to requirement 2, an explanation that includes an event for which no proximate cause exists is implausible (because the event should have happened earlier). To ensure that such an explanation is not returned, FINDEXTRAEVENTS iterates over each event occurrence $\mathsf{occ}_i$ in $\chi$, and attempts to find its proximate cause. To do so, it iterates over all occurrences in the explanation and execution history to find the set of prior occurrences $PO = [\mathsf{occ}_0 \ldots \mathsf{occ}_n]$ where for every $\mathsf{occ}_j \in PO$, $\mathsf{occ}_j \prec \mathsf{occ}_i$ and there is no $\mathsf{occ}_k$ such that $\mathsf{occ}_j \prec \mathsf{occ}_k \prec \mathsf{occ}_i$. If for any $\mathsf{occ}_j \in PO$ the set $\mathsf{effects}(\mathsf{occ}_j) \cap \mathsf{preconds}(\mathsf{occ}_i)$ is non-empty, $\mathsf{occ}_i$ has a proximate cause. If there is an $\mathsf{occ}_i$ without a proximate cause in the explanation, then the explanation is faulty and FINDEXTRAEVENTS returns a null explanation.

According to requirement 4, all events that are possible must occur. FINDEXTRAEVENTS guarantees this by adding events that are caused but are not in $\chi$. For each occurrence point $t_i \in \mathcal{T}$, it enumerates all events $e_j$ whose preconditions are met at $t_i$. If an occurrence $occ_{ij} = (e_j, t_i)$ is not present, it is added to $\chi$.

### Computational Complexity

Here we discuss differences between the high-level structures of $DH_1$ and $DH_2$ that affect their practical and theoretical performance.

Algorithm 1 lists pseudocode for $DH_1$, which performs a recursive search for plausible explanations. When an explanation has no inconsistencies (line 3), FINDEXTRAEVENTS is called to search for missing events (line 4). It returns null in the case of a faulty explanation, and no explanations are returned (line 5). If no inconsistencies are left, the explanation is plausible and is returned as the result of $DH_1$ (line 6). Otherwise, the explanation is implausible and search continues. First, an inconsistency is selected (line 7) and all possible refinements are found to resolve it (line 8). Each refinement is then applied in turn, modifying the global ordering relation $R$, and $DH_1$ is called recursively to continue the search (lines 9-13). After search ends, the ordering changes are reverted (line 19).

Algorithm 2 describes $DH_2$. In line 1, all events are pre-enumerated before the search begins. Lines 4-9 describe the same "no inconsistency case" seen before for $DH_1$, with the addition of a function call to find the in-

**Algorithm 1:** DiscoverHistory$_1$

```
1  Procedure DiscoverHistory$_1$ ($\chi$)
2  begin
3      if Inconsistencies$_\chi = \emptyset$ then
4          $\chi \leftarrow$FindExtraEvents ($\chi$)
5          if $\chi = \emptyset$ then return $\emptyset$
6          if Inconsistencies$_\chi = \emptyset$ then return $\{\chi\}$
7      $i \leftarrow$ Select(Inconsistencies$_\chi$)
8      $X \leftarrow \emptyset$   $\Theta \leftarrow$ FindRefinements($\chi$, $i$)
9      foreach $\theta \in \Theta$ do
10         $\chi_{new} \leftarrow$UpdateEvents($\chi, \theta$)
11         $R \leftarrow$UpdateRelations($R, \theta$)
12         Inconsistencies$_{\chi_{new}} \leftarrow$
               UpdateInconsistencies ($\chi, \theta$)
13         $X \leftarrow X \cup$ DH$_1$ ($\chi_{new}$)
14         $R \leftarrow$RevertRelations($R, \theta$)
15     return $X$
```

**Algorithm 2:** DiscoverHistory$_2$

```
1  $R, E_{poss} \leftarrow$FindPossibleEvents()
2  Procedure DiscoverHistory$_2$ ($\chi$)
3  begin
4      Inconsistencies$_\chi \leftarrow$
           FindInconsistencies($\chi$)
5      if Inconsistencies$_\chi = \emptyset$ then
6          $\chi \leftarrow$FindExtraEvents ($\chi$)
7          if $\chi = \emptyset$ then return $\emptyset$
8          Inconsistencies$_\chi \leftarrow$
               FindInconsistencies($\chi$)
9          if Inconsistencies$_\chi = \emptyset$ then return $\{\chi\}$
10     $X \leftarrow \emptyset$   $X_{best} \leftarrow X_{init}$
11     foreach $i \in$ Inconsistencies$_\chi$ do
12         $X_i \leftarrow$ Refine($\chi, i$)
13         if $X_{best} = X_{init}$ or $|X_i| < |X_{best}|$ then
14             $X_{best} \leftarrow X_i$
15     foreach $\chi_{new} \in X_{best}$ do
16         $X \leftarrow X \cup$ DH$_2$ ($\chi_{new}$)
17     return $X$
18
19 Procedure Refine($\chi, i$)
20 begin
21     $X \leftarrow \emptyset$   $\Theta \leftarrow$ FindRefinements($\chi$, $i$)
22     foreach $\theta \in \Theta$ do
23         $\chi_{new} \leftarrow$UpdateEvents($\chi, \theta$)
24         $X \leftarrow X + \chi_{new}$
25     return $X$
```

consistencies present; in DH$_2$, the inconsistencies must be computed when needed. This is because the set of inconsistencies are not updated for each invocation, a procedure which is significantly more expensive without the specialized proposition-organized ordering relation used by DH$_1$. Lines 10-16 handle the recursive case. Instead of selecting a single inconsistency to refine as in DH$_1$, DH$_2$ attempts all refinements of each inconsistency, and recursively searches only the smallest set of explanations found. This either reduces or does not change the branching factor, without changing the set of explanations found. DH$_1$ does not do this because it enumerate events during the search, which makes finding refinements far more expensive.

Because testing all inconsistencies is too expensive for DH$_1$, Select (line 7, Algorithm 1) attempts to predict which inconsistency will lead to the fewest refinements. Each time an inconsistency $i = (p, o', o'')$ is refined, DH$_1$ records the inconsistent literal $p$ and the number of observed refinements. When selecting a new inconsistency, DH$_1$ predicts that the number of refinements will be equal to the average of the last $n$ observed values for inconsistencies with the same literal $p$. Then DH$_1$ selects the inconsistency with the minimum prediction. This naive form of learning for increased efficiency is in the same vein as research in learned heuristic functions (Arfaee, Zilles, and Holte 2011) or search control rules (Minton 1988). While this has performed well in testing, future work may show that more principled search control techniques improve efficiency.

The size of the space of explanations searched by DH$_1$ is infinite; DH$_2$ searches a space that is *merely* exponential in the number of possible events ($E_{poss}$). Therefore, it reduces search time by searching for only the *most plausible* explanation (Leake 1992), using a plausibility metric to bound the search [1]. We consider two such metrics, the choice of which has a practical effect on efficiency: (1) the number of *changes* (i.e., events added to or removed from) made to an initial explanation, or (2) the number of *decisions* made (i.e., branching nodes expanded) in finding a new explanation from an initial explanation. These metrics were selected because they prefer explanations that are "simpler", in the sense that they are closer to what was believed initially. The major differences between the change and decision metrics is that some changes do not entail decisions. When only one refinement is possible, no decision is made, so applying that refinement results in an explanation of the same cost. While FindExtraEvents subroutine may make several changes to an explanation, no choices must be made, so the resulting explanation has the same cost under the decision metric and a higher cost under the change metric. These metrics are used to bound the search: a low value for maximum plausibility is chosen initially, and increased each time search fails, resulting in an iterative deepening search. Successive search depths can become more expensive quickly, so we employ a maximum plausibility bound $M$ to ensure a timely result.

To compute the computational complexity of DH$_1$ and DH$_2$, we start with the complexity of iterative deepening search: $O(b^d)$, where $b$ is the branching factor and $d$ the search depth. The maximum branching factor for DH$_1$ is equal to the number of possible refinements

[1] Omitted from pseudocode to save space.

of an inconsistency: $|E| + (2 * N_p) + 1$, where $E$ is the finite set of all event model instantiations (maximum number of add event refinements for an inconsistency) and $N_p$ is the maximum number of preconditions of an event (maximum number of remove event refinements for an inconsistency). At most one initial value hypothesis can be made for a single inconsistency. Substituting this into the iterative deepening equation, we get the complexity of DH$_1$: $O((|E| + (2 * N_p) + 1)^M)$. The maximum branching factor for DH$_2$ is similar: $|E_{poss}| + (2 * N_p) + 1$. This is always less than or equal to the branching factor for DH$_1$. The advance enumeration step made by DH$_2$ expends no more than a constant amount of effort for each event in $E$; it is therefore bounded by $c|E|$, and falls out of the complexity equation for DH$_2$: $O((|E_{poss}| + (2 * N_p) + 1)^M)$.

## DHAgent

DHAgent is an agent capable of using the explanations generated by DH. It uses a classical planner to create plans under the closed-world assumption (which is not generally correct in partially observable worlds), and constructs a default explanation as it executes a plan. This explanation includes actions executed by the agent, observations received from the environment, and expected exogenous events predicted by running FIND-EXTRAEVENTS on the current explanation after executing each action. After each observation is received, the explanation is checked for inconsistencies. At this point, DH is called to refine the existing explanation. It then selects an arbitrary explanation (the first) returned by DH, and queries it to find the current state, which consists of all facts explained to be true at the most recent occurrence point, both observable and hidden. It then uses a planner to replan for the current state, and resumes execution. This procedure stops when all goals are accomplished or no plan can be generated.

## Experimental Evaluation

We claim that both DH$_1$ and DH$_2$ increase the goal achievement performance of DHAgent in partially-observable hazardous domains. The efficiency of DH$_2$ scales well with increasingly complex explanations. To analyze these claims, we provide empirical results analyzing efficiency and goal achievement performance on four sets of 25 scenarios. Each scenario specifies a set of tasks to accomplish and an initial simulation state without hidden information. To our knowledge, no related systems provide support for deterministic exogenous events, so we do not compare against existing systems. Instead, we compare DHAgent using DH$_1$, DH$_2$ with the decision count plausibility metric, DH$_2$ with the change count plausibility metric, and an ablated DH that does not refine explanations. In these experiments, plans were provided by a version of SHOP2 (Nau et al. 2003) extended with the ability to project the occurrence of events (Molineaux, Klenk, and Aha 2010a).

Two domains were used in these experiments: a Hazardous Rovers domain and a Hazardous Satellites domain. These domains are called hazardous because no plan will succeed in some initial states, unlike in standard planning domains. These domains were introduced in (Molineaux, Kuter, and Klenk in press) for early DH investigations, and require the accomplishment of multiple goals using multiple simulated robotic effectors over the course of a single trial.

There are 3 levels of difficulty in the Hazardous Rovers domain, each corresponding to a different frequency of hidden obstacles in the initial state. All scenarios are pre-generated, and no probabilities are given to the agents, but a higher probability of obstacles translates to a more difficult scenario in which explanation is of higher importance.

To examine the behavior of the DH algorithms in these domains, we ran the four agents on each of the four sets of scenarios (3 Hazardous Rovers and 1 Hazardous Satellites) using 5 different maximum plausibility bounds, recording both execution time and the number of goals achieved. The results of these evaluations are shown in Table 1 shows that DH$_2$ achieves the highest performance at every maximum plausibility bound, and while DH$_2$ scales well as the bound increases, the time required by DH$_1$ tends to increase dramatically. DH$_2$ takes less than 20 seconds on average to generate all explanations at maximum performance, and scales well, with explanation times increasing less than 40% from maximum search depth 1 to 9 on each set of scenarios. However, this does not indicate that DH$_2$ is always more efficient than DH$_1$, because DH$_1$ sometimes attains high performance at lower plausibility bounds.

To compare equivalent results for each system, we found the minimum value of maximum search depth at which each agent reached maximum goal achievement performance, except the ablated agent, which never does. Maximum performance was determined by matching results against the highest absolute performance on that set of scenarios using a 1-tailed paired sample t-test with a 95% confidence threshold. When the null hypothesis of equal means could not be rejected, the agent was said to have reached maximum goal achievement performance. Then we compared the time required for that achievement across all agents. The ablated agent is statistically outperformed on each set by each other agent. In the Hazardous Rovers domain, DH$_2$ is clearly more efficient on each test condition. DH$_1$ is statistically more efficient than DH$_2$ in the Hazardous Satellites domain, but the percentage difference between DH$_1$ and DH$_2$ is small. We conjecture that the higher performance level of DH$_1$ on the Hazardous Satellites domain can be attributed to higher success of branch prediction in DH$_1$ and fewer events in that domain; in future research, we will examine what environment characteristics impact efficiency.

| **Domain** | **Ablated** | $DH_1$ | | | | | $DH_2$ **using change metric** | | | | | $DH_2$ **using decision metric** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Max Search Depth** | 0 | 1 | 3 | 5 | 7 | 9 | 1 | 3 | 5 | 7 | 9 | 1 | 3 | 5 | 7 | 9 |
| **Hazardous Rovers** | 0.65 | 0.65 | 0.71 | 0.71 | **0.80** | 0.80 | 0.65 | 0.71 | 0.71 | **0.83** | 0.84 | 0.68 | ***0.80*** | 0.84 | 0.84 | 0.84 |
| $\lambda = 0.1$ | 0.77 | 0.76 | 1.93 | 11.91 | **156.75** | 180.05 | 12.69 | 14.81 | 17.89 | **15.07** | 15.39 | 11.46 | ***13.52*** | 14.45 | 14.50 | 14.52 |
| **Hazardous Rovers** | 0.25 | 0.25 | 0.28 | 0.29 | **0.53** | 0.56 | 0.25 | 0.31 | 0.33 | ***0.55*** | 0.57 | 0.31 | 0.47 | ***0.59*** | 0.60 | 0.61 |
| $\lambda = 0.2$ | 1.49 | 1.48 | 4.66 | 24.92 | **213.36** | 126.03 | 89.01 | 90.28 | 94.76 | ***19.25*** | 20.53 | 89.53 | 14.68 | ***19.61*** | 32.06 | 49.20 |
| **Hazardous Rovers** | 0.20 | 0.20 | 0.25 | 0.29 | **0.36** | 0.36 | 0.20 | 0.25 | 0.29 | 0.35 | ***0.39*** | 0.25 | 0.33 | ***0.39*** | 0.39 | 0.39 |
| $\lambda = 0.3$ | 1.43 | 1.41 | 3.11 | 51.39 | **269.01** | 362.91 | 81.10 | 82.94 | 86.38 | 17.94 | ***19.33*** | 81.17 | 29.45 | ***19.06*** | 22.40 | 26.17 |
| **Hazardous Satellites** | 0.52 | 0.52 | 0.67 | ***0.69*** | 0.69 | 0.69 | 0.52 | 0.67 | 0.67 | **0.68** | 0.68 | 0.53 | 0.67 | **0.68** | 0.68 | 0.68 |
| $\lambda = 0.3$ | 0.30 | 0.33 | 4.98 | ***13.25*** | 25.89 | 26.64 | 12.39 | 13.18 | 13.78 | **14.60** | 15.23 | 12.41 | 13.52 | **14.51** | 15.75 | 16.87 |

*Table 1* Goal achievement performance (top) and execution time in seconds (bottom) for each agent for 5 values of maximum search depth. Boldface indicates the lowest search depth at which maximum goal achievement performance is attained; italics indicate the fastest of those performances.

## Related Work

**Continual Planning.** Other work in planning and execution that involves reconsidering what happened in the past includes that by Molineaux, Klenk, and Aha (2010b) on explanation in the ARTUE system and Shani and Brafman (2011) on the SDR planner. Our work extends the ARTUE explanation generator with a more principled formalism for exogenous events and the capability to reason over a longer history. SDR maintains beliefs by reconsidering facts from the past and the initial state through a regression process. This process finds and eliminates possible worlds, instead of constructing a series of events that explain the observations. While SDR and DHAgent handle similar tasks, DHAgent is designed to work in hazardous environments where outcomes cannot be guaranteed, and SDR is designed for environments in which a contingent plan can be devised to find an infallible solution. Thus, unlike DHAgent, SDR only replans when the next action's preconditions are not known to be true, and will therefore execute actions in a provably incorrect plan before replanning.

Most work on replanning and plan repair during execution in dynamic environments focuses only on execution-time failures as discrepancies (Kambhampati and Hendler 1992; Myers 1996; Wang and Chien 1997; Myers 1999; Yoon, Fern, and Givan 2007; Ayan et al. 2007; Warfield et al. 2007). In these systems, a discrepancy is recognized only when causal links are broken between the effects and preconditions present in a plan. In particular, when the observed state of the world violates causal links in a plan, a discrepancy occurs and triggers a re-planning or plan-repair process. In contrast, our algorithms generate more expressive and informative explanations of the world that cover any type of changes caused by deterministic exogenous events. A few systems, such as CPEF, (Myers 1999) create dynamic plans that evolve in response to the environment through monitors that trigger when the plan needs repair. While more flexible than causal link repair, this strategy is not as comprehensive as explanation, since it ignores unanticipated changes in the environment.

**Real-Time Control and Planning.** Existing research on *real-time control and execution* typically employs a reactive planning foundation, where the agent decides on an action and executes it immediately (Musliner, Durfee, and Shin 1993; Kabanza, Barbeau, and St-Denis 1997; Goldman et al. 2002; Musliner et al. 2008). Some systems select an action to be executed using planning heuristics, while others generate an offline plan that achieves the goal.

CIRCA is an autonomous planning and control system that builds and executes safety-preserving plans in an environment with unpredictable events (Musliner, Durfee, and Shin 1993). CIRCA includes a Reaction Planner that derives a plan to accomplish mission goals while avoiding or preventing failures, even in real-time environments. Like DHAgent, CIRCA reasons about uncontrollable sources of change, and recognizes when the environment has deviated from expectations. However, it does not reason about possible occurrence histories to understand its environment.

**Diagnosis.** Some research in diagnosis (e.g., (Iwan and Lakemeyer 2003; McIlraith 1998; Sohrabi, Baier, and McIlraith 2010)) has focused on finding action histories that resolve contradictions by assuming the presence of faulty actions and/or missing assumptions about the initial state. This work differs from ours in that it does not take place in the context of an execution framework. Other work in diagnosis (e.g. (Iwan 2001; Gspandl et al. 2011)) has added an execution component, but does not support amending explanations by removing events (or actions) previously believed to have happened. Furthermore, existing cannot reason about deterministic exogenous events, which require support for simultaneous event occurrence and the elimination of explanations in which caused events do not occur.

## Conclusions and Future Work

We described the $DH_1$ and $DH_2$ algorithms, calculated their computational complexity, and presented empirical results showing that both achieve high goal achievement performance statistically exceeding that of an agent that does not perform explanation generation, and $DH_2$'s efficiency appears quite scalable.

In future work, we will extend our experiments to a greater diversity of domains, requiring more complex explanations, to determine what environment factors impact efficiency of explanation generation.

## References

Albore, A.; Palacios, H.; and Geffner, H. 2009. A translation-based approach to contingent planning. In *Proc. 21st Int. Joint Conference on AI (IJCAI-09)*, 1623–1628.

Arfaee, S. J.; Zilles, S.; and Holte, R. 2011. Learning heuristic functions for large state spaces. *Artificial Intelligence* 175(16):2075–2098.

Ayan, F.; Kuter, U.; Yaman, F.; and Goldman, R. P. 2007. HOTRiDE: Hierarchical Ordered Task Replanning in Dynamic Environments. In Ingrand, F., and Rajan, K., eds., *ICAPS-07 Workshop on Planning and Plan Execution for Real-World Systems*, 31–36.

DesJardins, M.; Durfee, E.; Ortiz Jr, C.; Wolverton, M.; et al. 1999. A survey of research in distributed, continual planning. *AI Magazine* 20(4):13.

Fox, M.; Howey, R.; and Long, D. 2005. Validating plans in the context of processes and exogenous events. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, 1151.

Gerevini, A.; Saetti, A.; and Serina, I. 2006. An approach to temporal planning and scheduling in domains with predictable exogenous events. *Journal of Artificial Intelligence Research* 25(1):187–231.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

Goldman, R.; Haigh, K.; Musliner, D.; and Pelican, M. 2002. MACBETH: a multi-agent constraint-based planner. In *Proceedings of the 21st Digital Avionics Systems Conference*, volume 2, 7E3–1. IEEE.

Gspandl, S.; Pill, I.; Reip, M.; Steinbauer, G.; and Ferrein, A. 2011. Belief management for high-level robot programs. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*.

Hoffmann, J., and Brafman, R. 2005. Contingent planning via heuristic forward search with implicit belief states. In *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, 71–80.

Iwan, G., and Lakemeyer, G. 2003. What observations really tell us. *KI 2003: Advances in Artificial Intelligence* 194–208.

Iwan, G. 2001. History-based diagnosis templates in the framework of the situation calculus. *KI 2001: Advances in Artificial Intelligence* 244–259.

Kabanza, F.; Barbeau, M.; and St-Denis, R. 1997. Planning control rules for reactive agents. *Artificial Intelligence* 95(1):67–113.

Kambhampati, S., and Hendler, J. A. 1992. A validation-structure-based theory of plan modification and reuse. *Artificial Intelligence* 55:193–258.

Leake, D. 1992. *Evaluating explanations: A content theory*. L. Erlbaum Associates Inc.

McIlraith, S. 1998. Explanatory diagnosis: Conjecturing actions to explain observations. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning*, 167–179. Morgan Kaufmann Publishers.

Minton, S. 1988. *Learning effective search control knowledge: an explanation-based approach*. Ph.D. Dissertation, Pittsburgh, PA, USA.

Molineaux, M.; Klenk, M.; and Aha, D. 2010a. Planning in dynamic environments: Extending htns with nonlinear continuous effects. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Molineaux, M.; Klenk, M.; and Aha, D. 2010b. Goal-driven autonomy in a Navy strategy simulation. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 1548–1554.

Molineaux, M.; Kuter, U.; and Klenk, M. in press. Discoverhistory: Explaining the past in planning and execution. In *Proceedings of the Eleventh International Conference on Autonomous Agents and Multi-Agent Systems*.

Musliner, D. J.; Pelican, M. J. S.; Goldman, R. P.; Krebsbach, K. D.; and Durfee, E. H. 2008. The evolution of CIRCA, a theory-based AI architecture with real-time performance guarantees.

Musliner, D.; Durfee, E.; and Shin, K. 1993. CIRCA: a cooperative intelligent real-time control architecture. *IEEE Transactions on Systems, Man and Cybernetics* 23(6):1561–1574.

Myers, K. L. 1996. Advisable planning systems. In Tate, A., ed., *Advanced Planning Technology*. AAAI Press.

Myers, K. L. 1999. A continuous planning and execution framework. *AI Magazine* 20(4):63.

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

Shani, G., and Brafman, R. 2011. Replanning in domains with partial information and sensing actions. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Sohrabi, S.; Baier, J.; and McIlraith, S. 2010. Diagnosis as planning revisited. *Proceedings of the International Conference on the Principles of Knowledge Representation and Reasoning* 26–36.

Wang, X., and Chien, S. 1997. Replanning using hierarchical task network and operator-based planning. *Recent Advances in AI Planning* 427–439.

Warfield, I.; Hogg, C.; Lee-Urban, S.; and Munoz-Avila, H. 2007. Adaptation of hierarchical task network plans. In *Proceedings of the Twentieth International FLAIRS Conference (FLAIRS-07)*.

Yoon, S.; Fern, A.; and Givan, R. 2007. FF-Replan: A baseline for probabilistic planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, 352–359.