I N F S Y S

RESEARCH

R E P O R T

INSTITUT FÜR INFORMATIONSSYSTEME

ABTEILUNG WISSENSBASIERTE SYSTEME

DIAGNOSING PLAN EXECUTION DISCREPANCIES IN A LOGIC-BASED ACTION FRAMEWORK

Thomas Eiter Esra Erdem Wolfgang Faber

INFSYS RESEARCH REPORT 1843-04-03

AUGUST 2004

Institut für Informationssysteme
Abtg. Wissensbasierte Systeme
Technische Universität Wien
Favoritenstraße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

TECHNISCHE UNIVERSITÄT WIEN

INFSYS RESEARCH REPORT

INFSYS RESEARCH REPORT 1843-04-03, AUGUST 2004

DIAGNOSING PLAN EXECUTION DISCREPANCIES IN A LOGIC-BASED ACTION FRAMEWORK

Thomas Eiter[1] Esra Erdem[2] Wolfgang Faber[3]

Abstract. This paper introduces a logic-based framework for monitoring plan execution relative to a given set of trajectories. According to this framework, a monitoring agent can tell when things go wrong by checking the compatibility of the plan execution with the given trajectories, considering the current state information. She finds an explanation for these detected discrepancies by examining the given trajectories against possible evolutions of the current state from the initial state. Such an explanation provides information about possible points of failure that may be useful in two ways. First, points of failure can be used to determine some checkpoints (specifying when to check for discrepancies), and this may be useful if the plan is executed many times. Second, they can be used for recovering from discrepancies. For instance, the agent can undo the plan execution until a point of failure, from which the remainder of the plan can be (re)executed, thus avoiding an expensive (re)planning step. In this paper, we present formal specifications for discrepancies and explanations for them in a general action representation framework, which can accommodate nondeterminism, concurrency, and dynamic worlds. We describe two ways of finding explanations for detected discrepancies, and study their properties. Moreover, we analyze the computational complexity of detecting discrepancies and finding explanations for them, and describe how these computational problems can be solved.

---

[1]Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Wien, Austria. E-mail: eiter@kr.tuwien.ac.at

[2]Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Wien, Austria. E-mail: esra@kr.tuwien.ac.at

[3]Institut für Informationssysteme, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Wien, Austria. E-mail: faber@kr.tuwien.ac.at

Acknowledgements: We thank Matthias Fichtner and Mikhail Soutchanski for useful discussions and other colleagues for helpful comments. This work was supported by FWF (Austrian Science Funds) under project P16536-N04.

1 Introduction

In real-world domains, executing a plan without monitoring is a fragile strategy, since non-determinism might cause unintended effects of actions which, for instance, may prevent the execution of the rest of the plan from reaching a goal state. Imagine an agent who goes for shopping and, at some point, realizes that she does not have enough money to pay for the milk because she accidentally picked a similar, but more expensive one. To cope with this problem offline, sensing actions have been introduced and the notions of a conditional plan and a conformant plan, which enforces that the goal is reached under any circumstances, have been defined. However, conditional plans are difficult to generate and storing them might require exponential space, while conformant plans often simply do not exist. Furthermore, frequent or continuous sensing usually comes at a cost, slows down the plan execution, and makes finding a plan more involved because of a larger action repertoire.

In this paper, we introduce a novel logic-based framework for execution monitoring, according to which the agent can tell when things go wrong by detecting discrepancies between the actual world at a certain time stamp (also referred to as stage) and the agent's internal representation of it. Thus, rather than *enforcing a priori* that any possible execution will be successful, *monitoring aims at checking, from time to time, whether the execution is "on track"* and the state reached complies with the intentions. The agent may find out what goes wrong by making diagnoses of the discrepancies, and then she can recover from such discrepancies to achieve her goals, possibly using the information obtained from the diagnoses. For instance, in the previous example, with monitoring, the agent can discover earlier (e.g. just after she picked the milk) that she picked a more expensive milk, and she can pick a cheaper one instead.

In this paper, we study the first two steps of monitoring: detecting discrepancies and generating diagnoses for them. The last step, plan recovery, is not formally addressed.

The main contributions of this paper are summarized as follows.

- We describe execution monitoring in the general logic-based action representation framework of Turner [25]. In this framework, an action description can be represented by a transition diagram—a directed graph whose nodes correspond to states and whose edges correspond to action occurrences. It can accommodate nondeterminism, concurrent actions and dynamic worlds. Such action representations can be obtained from domain descriptions in STRIPS-based or more expressive action description languages, such as $\mathcal{C}+$ [13] or $\mathcal{K}$ [6]. This allows us to use systems such as CCALC [15], CPLAN [12, 8], and DLV$^{\mathcal{K}}$ [6], for reasoning about actions.

- We formalize the notion of a discrepancy between the run-time execution of a plan and a set of trajectories. These trajectories describe some possible executions of the plan that are intended or preferred as in [20], and they can be determined by meta-level constraints which are not represented (or cannot be expressed) in the framework.

- For a diagnosis of discrepancies, we introduce the notion of a "point of failure", at which the execution of the scheduled action yields a state that is not intended. A point of failure can be considered as a checkpoint specifying when to check for discrepancies, and this can be useful if the plan is executed many times. It can also be used for recovering from the detected discrepancies. The idea is, by applying Occam's Razor, recovering from discrepancies with the least amount of effort by undoing action effects to re-establish the predicted state, and to reuse the rest of the plan from there. This kind of plan recovery can be useful when the remaining plan is very long.

- Differentiating between strict and relaxed obedience of the plan execution to the given set of trajectories, we describe a state-oriented and a history-oriented diagnosis of discrepancies. It turns out that in a common setting, these notions surprisingly coincide, while they are different if further information about the past is available.

- On the computational side, we analyze the complexity of the three main computational problems in execution monitoring: detecting a discrepancy, verifying a diagnosis for a discrepancy, and finding a diagnosis for a discrepancy. Then, we introduce a generic algorithm to generate diagnoses for a discrepancy. For each of the three problems above, which can be solved in polynomial time with the help of an NP oracle, we give a precise complexity characterization in terms of completeness for a complexity class. In particular, we show that computing a diagnosis for a discrepancy is complete for the not widely known class FNP//OptP[$O(\log n)$] from [4], which is in between computability in nondeterministic polynomial time and deterministic polynomial time with an NP oracle. Few problems in AI have been shown to be complete for this class so far. We also note that the complexity results for the problems above, derived for Turner's action representation framework, carry over to similar representation frameworks like that of CCALC and that of DLV$^{\mathcal{K}}$.

Among other logic-based frameworks for execution monitoring are [11, 21, 22], and [9]. These frameworks differ from ours with respect to discrepancy detection and diagnosis mainly as follows. In [11, 21, 22], a discrepancy is detected when the remaining plan is not successful, considering all possible trajectories; diagnosis of discrepancies is not considered. In [9], a discrepancy is detected when the action is not executable or when the effects of an action are not as intended; diagnosis of discrepancies is provided in terms of some abnormality predicates. No complexity analysis for discrepancy detection and diagnosis is provided in these work. More discussion on the related work is given in Section 7.

In the following, first we present the action representation framework and the planning framework we consider (Section 2), and briefly discuss our monitoring framework (Section 3). Then, we precisely describe discrepancies (Section 4), and introduce two kinds of diagnosis of discrepancies (Section 5). After we analyze the computational complexities of the problems of detecting discrepancies and finding diagnosis of discrepancies, we present algorithms to solve them (Section 6). Then we conclude with a discussion on the related work (Section 7) and the future work (Section 8).

In order not to distract from the flow of reading, proofs of the technical results have been moved to the Appendix.

2 Preliminaries

We introduce in brief Turner's action representation and planning framework [25].

We begin with a set $\mathcal{A}$ of action symbols and a disjoint set $\mathcal{F}$ of fluent symbols. Let *state*$(\mathcal{F})$ be a formula in which the only nonlogical symbols are elements of $\mathcal{F}$. This formula encodes the set of states that correspond to its models. Let *act*$(\mathcal{F}, \mathcal{A}, \mathcal{F}')$ be a formula in which the only nonlogical symbols are elements of $\mathcal{F} \cup \mathcal{A} \cup \mathcal{F}'$, where $\mathcal{F}'$ is obtained from $\mathcal{F}$ by priming each element of $\mathcal{F}$. Then the models of the formula

$$\mathit{state}(\mathcal{F}) \wedge \mathit{act}(\mathcal{F}, \mathcal{A}, \mathcal{F}') \wedge \mathit{state}(\mathcal{F}') \tag{1}$$

correspond to the set of transitions. That is,

- the start state corresponds to an interpretation of (the symbols in) $\mathcal{F}$,[1]
- the set of actions executed corresponds to an interpretation of $\mathcal{A}$, and
- the end state corresponds to an interpretation of $\mathcal{F}'$.

Formula (1) is abbreviated as *tr*$(\mathcal{F}, \mathcal{A}, \mathcal{F}')$.

Example 1 [13] Putting a puppy into water makes the puppy wet, and drying a puppy with a towel makes it dry. With the fluents

$$\mathcal{F} = \{\mathit{inWater}, \mathit{wet}\},$$

and the actions

$$\mathcal{A} = \{\mathit{putIntoWater}, \mathit{dryWithTowel}\},$$

the states can be described by the formula

$$\mathit{state}(\mathcal{F}) = \mathit{inWater} \supset \mathit{wet}.$$

Since there are three interpretations of $\mathcal{F}$ satisfying *state*$(\mathcal{F})$

$$\{\mathit{inWater}, \mathit{wet}\}, \{\neg \mathit{inWater}, \mathit{wet}\}, \{\neg \mathit{inWater}, \neg \mathit{wet}\}$$

there are three states: $\{\mathit{inWater}, \mathit{wet}\}$, $\{\mathit{wet}\}$, $\{\}$.

The action occurrences can be defined as follows:

$$\begin{aligned} \mathit{act}(\mathcal{F}, \mathcal{A}, \mathcal{F}') = \; & (\mathit{inWater}' \equiv \mathit{inWater} \vee \mathit{putIntoWater}) \wedge \\ & (\mathit{wet}' \equiv (\mathit{wet} \wedge \neg \mathit{dryWithTowel}) \vee \mathit{putIntoWater}) \wedge \\ & (\mathit{dryWithTowel} \supset (\neg \mathit{inWater} \wedge \neg \mathit{putIntoWater})) \end{aligned}$$

The last line of the formula above expresses that $\mathit{dryWithTowel}$ is executable when $\mathit{inWater}$ is false and it is not executable concurrently with $\mathit{putIntoWater}$.

[1] In the rest of the paper, we sometimes say "interpretation of $\mathcal{S}$" to mean an interpretation of the symbols in a set $\mathcal{S}$.

For instance, the interpretation

$$\{\neg inWater, wet, dryWithTowel, \neg putIntoWater, \neg inWater', \neg wet'\}$$

satisfies $tr(\mathcal{F}, \mathcal{A}, \mathcal{F}')$, therefore it describes a transition:

$$\langle\{wet\}, \{dryWithTowel\}, \{\}\rangle. \tag{2}$$

Note that the interpretation of $\mathcal{A}$ above describes the occurrence of $dryWithTowel$.$\Box$

The meaning of a domain description can be represented by a transition diagram—a directed graph whose nodes correspond to states and whose edges correspond to action occurrences. In a transition diagram, a "trajectory" of length n is obtained by finding a model of the formula

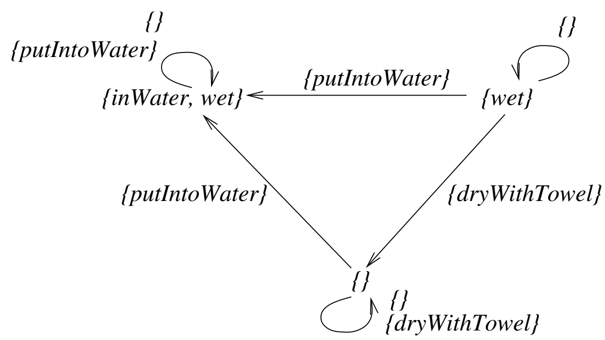
$$\bigwedge_{t=0}^{n-1} tr(\mathcal{F}_t, \mathcal{A}_t, \mathcal{F}_{t+1})$$

where each $\mathcal{F}_i$ (resp., each $\mathcal{A}_i$) is the set of fluents (resp., actions) obtained from $\mathcal{F}$ (resp., $\mathcal{A}$) by adding time stamp i to each fluent symbol (resp., each action symbol). The trajectory is the alternating sequence of states and action occurrences that correspond to the interpretation of fluents and actions respectively. The trajectory is of the form

$$S_0, A_0, S_1, \ldots, S_{n-1}, A_{n-1}, S_n$$

where each S_i is the state that corresponds to the interpretation of $\mathcal{F}_i$, and each A_i is the action occurrences that correspond to the interpretation of $\mathcal{A}_i$.

Example 2 The transition diagram for the action description of Example 1 is as follows:



In this diagram, paths of length 1 describe transitions, and paths of length n describe trajectories. For instance, the path

$$\{wet\}, \{\}$$

describes transition (2), and the path

$$\{wet\}, \{\}, \{inWater, wet\}$$

describes the trajectory

$$\{wet\}, \{dryWithTowel\}, \{\}, \{putIntoWater\}, \{inWater, wet\} \tag{3}$$

where a wet puppy is first dried with a towel and then put into the water. This trajectory is obtained from the following interpretation of $\mathcal{F}_0 \cup \mathcal{A}_0 \cup \mathcal{F}_1 \cup \mathcal{A}_1 \cup \mathcal{F}_2$

$$\begin{aligned}\{ &\neg inWater_0, wet_0, \neg putIntoWater_0, dryWithTowel_0, \\ &\neg inWater_1, \neg wet_1, putIntoWater_1, \neg dryWithTowel_1, \\ &inWater_2, wet_2\}.\end{aligned}$$

that satisfies the conjunction $tr(\mathcal{F}_0, \mathcal{A}_0, \mathcal{F}_1) \wedge tr(\mathcal{F}_1, \mathcal{A}_1, \mathcal{F}_2)$.□

In a planning problem, an initial state is described by the formula $init(\mathcal{F})$ such that $init(\mathcal{F}) \models state(\mathcal{F})$, and the goal is described by the formula $goal(\mathcal{F})$.

A *plan of length* n for the planning problem is obtained from any model of the formula

$$init(\mathcal{F}_0) \wedge \bigwedge_{t=0}^{n-1} tr(\mathcal{F}_t, \mathcal{A}_t, \mathcal{F}_{t+1}) \wedge goal(\mathcal{F}_n); \tag{4}$$

as the sequence

$$\langle A_0, \ldots, A_{n-1} \rangle$$

of action occurrences A_i that correspond to the interpretation of $\mathcal{A}_i$.

A *feasible trajectory for a plan* $P = \langle A_0, \ldots, A_{n-1} \rangle$ is a trajectory

$$S_0, A'_0, S_1, \ldots, S_{n-1}, A'_{n-1}, S_n$$

such that $A_i = A'_i$ $(0 \leq i < n)$, $S_0 \models init(\mathcal{F}_0)$, and $S_n \models goal(\mathcal{F}_n)$. In the following, we will denote by T_P the set of all feasible trajectories for a plan P.

We can talk about more specific states, transitions, or trajectories by applying some "substitutions" to the formulas describing them. Consider, for instance, Example 2. To find states reachable from the state $S = \{wet\}$, we need to find the transitions of the form $\langle S, A, S' \rangle$. These transitions can be described by substituting S for $\mathcal{F}$ in $tr(\mathcal{F}, \mathcal{A}, \mathcal{F}')$, that is, by the formula $tr(S, \mathcal{A}, \mathcal{F}')$ obtained from $tr(\mathcal{F}, \mathcal{A}, \mathcal{F}')$ by replacing every atom $p \in \mathcal{F}$ with $\top$ if $p \in S$, and by replacing every atom $p \in \mathcal{F}$ with $\bot$ otherwise. Similarly, trajectories S_0, A_0, S_1, A_1, S_2 of length 2 with the initial state $S_0 = \{\}$ and the action occurrence $A_1 = \{dryWithTowel\}$ at time stamp 1 can be expressed by $tr(S_0, \mathcal{A}_0, \mathcal{F}_1) \wedge tr(\mathcal{F}_1, A_1, \mathcal{F}_2)$. Here, $tr(S_0, \mathcal{A}_0, \mathcal{F}_1)$ is the formula obtained from $tr(\mathcal{F}_0, \mathcal{A}_0, \mathcal{F}_1)$ by replacing every atom $p_0 \in \mathcal{F}_0$ with $\top$ if $p \in S_0$, and by replacing every atom $p_0 \in \mathcal{F}_0$ with $\bot$ otherwise. Similarly, $tr(\mathcal{F}_1, A_1, \mathcal{F}_1)$ is the formula obtained from $tr(\mathcal{F}_1, \mathcal{A}_1, \mathcal{F}_2)$ by replacing every atom $p_1 \in \mathcal{A}_1$ with $\top$ if $p \in A_1$, and by replacing every atom $p_1 \in \mathcal{A}_1$ with $\bot$ otherwise.

Example 3 Consider a version of the blocks world where the block being moved to a location may end up at a different location because the agent might not grip it properly.

With the fluents $on(B, L)$ ("block B is on location L"), and the actions $move(B, L)$ ("move block B to location L"), the states, i.e., $state(\mathcal{F})$, can be defined by the conjunction of the following formulas:[2]

[2]The symbols $B, B1, B2, L, L1, L2$ are schematic variables: $B, B1, B2$ range over a finite set of block constants, and $L, L1, L2$ range over the set of location constants that consists of the set of block constants and the constant $table$.

- every block should be on some location:

$$\bigvee_{L} on(B, L);$$

- if a block is on some location then it is not anywhere else:

$$on(B, L) \supset \bigwedge_{L \neq L1} \neg on(B, L1);$$

- a block cannot have more than one block on itself:

$$on(B1, B) \supset \bigwedge_{B1 \neq B2} \neg on(B2, B);$$

- every block is "supported" by the table (here $supported(B)$ is an auxiliary propositional variable defined in terms of $on(B, L)$):

$$supported(B).$$

Formula $act(\mathcal{F}, \mathcal{A}, \mathcal{F}')$ can be defined by the conjunction of the following formulas:

- the preconditions of $move(B, L)$:

$$move(B, L) \supset \bigwedge_{B1} \neg on(B1, B);$$

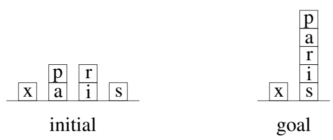
- the effects of $move(B, L)$, and inertia:

$$on(B, L1)' \supset (on(B, L1) \vee \bigvee_{L} move(B, L));$$

- no-concurrency:

$$move(B, L) \supset \bigwedge_{B1 \neq B, L1} \neg move(B1, L1).$$

Consider, in this domain, the blocks a, p, x, i, r, s, and a planning problem $\mathcal{P}$ with the initial state and the goal state as follows:



A solution to the planning problem $\mathcal{P}$ is the following plan of length 6:[3]

$$P = \langle move(r, x), move(i, s), move(r, i), move(p, x), move(a, r), move(p, a)\rangle.$$

[3]We sometimes drop curly brackets from singleton action occurrences in a plan. For instance, in P, singleton action occurrences of the form $\{move(B, L)\}$ are written as $move(B, L)$.

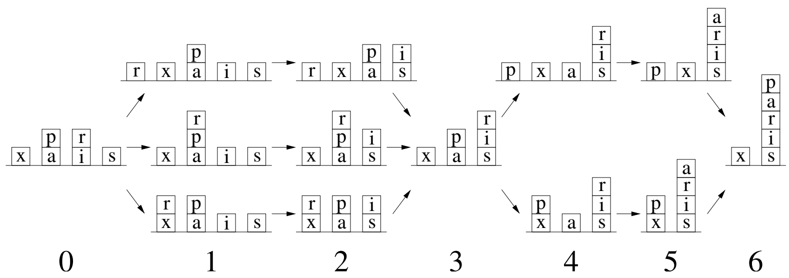
The feasible trajectories T_P are of the form

$$S_0, \{move(r,x)\}, S_1, \{move(i,s)\}, S_2, \{move(r,i)\}, S_3,$$
$$\{move(p,x)\}, S_4, \{move(a,r)\}, S_5, \{move(p,a)\}, S_6$$

where each S_i is a state that corresponds to an interpretation of $\mathcal{F}_i$. The states S_0 and S_6 are the initial state and the goal state, respectively, presented in the figure above; S_2–S_5 are states such that

$$\mathit{tr}(S_0, \{move(r,x)\}, S_1) \wedge \mathit{tr}(S_1, \{move(i,s)\}, S_2) \wedge \mathit{tr}(S_2, \{move(r,i)\}, S_3) \wedge$$
$$\mathit{tr}(S_3, \{move(p,x)\}, S_4) \wedge \mathit{tr}(S_4, \{move(a,r)\}, S_5) \wedge \mathit{tr}(S_5, \{move(p,a)\}, S_6).$$

These feasible trajectories can be represented by the paths from 0 to 6 in the following graph:



□

In the following sections, an expression of the form $\mathcal{F} \equiv \mathcal{F}'$ denotes the conjunction $\bigwedge_{f \in F} f \equiv f'$.

3 Monitoring Framework

According to our framework, in the process of monitoring the execution of a plan relative to a given set of trajectories, the monitoring agent, from time to time,

1. checks whether there is a discrepancy between the current state and the corresponding states of the given trajectories relative to the plan;
2. if no discrepancy is detected then continues with the execution of the plan; otherwise, may try to find a diagnosis of discrepancies by examining the given trajectories against possible evolutions of the current state from the initial state;
3. recovers from the discrepancies, possibly by using the diagnosis of discrepancies, to reach a goal state.

Steps 1–3 above are described further in Figure 1.

Consider, for instance, the blocks world described in Example 3. Suppose that an agent is executing the plan P. At time stamp 3, at a state S_3, the agent does not grip block p properly

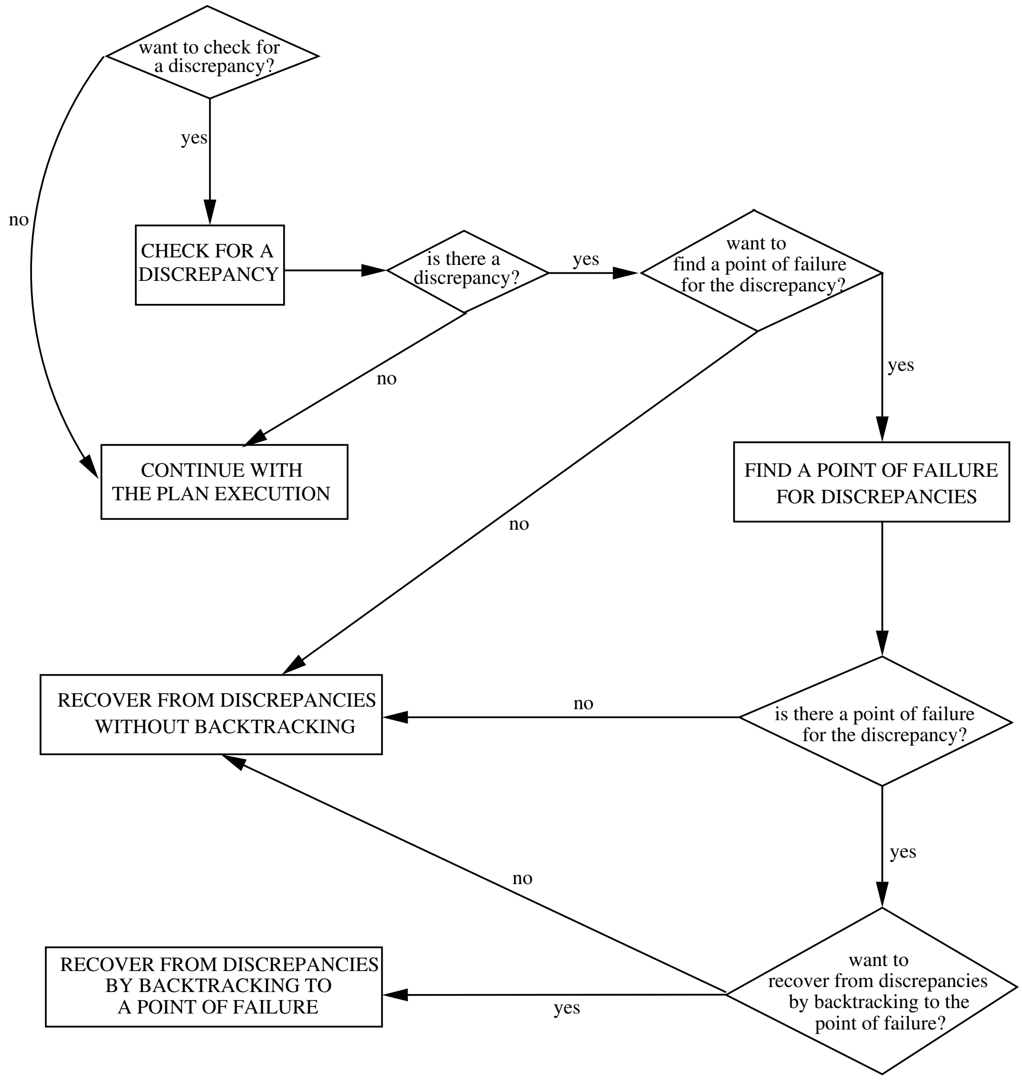


Figure 1: Steps 1–3 of the monitoring framework described in Section 3.

while she is moving it from the table onto block x, and the block ends up on block r. If this effect of the action of moving p onto x is not intended at time stamp 4 with respect to the given set of trajectories, a discrepancy is observed between the actual state and the intended one. When such a discrepancy is detected, the agent can look for an explanation, i.e., a point of failure. In this case, the point of failure for the observed discrepancy can be characterized by the state S_3 and the time stamp 3. If the agent executes this plan often, then she may want to check for a discrepancy at time stamp 4 every time she executes P. Therefore, she may determine time stamp 4 as a checkpoint. On the other hand, since such a discrepancy may prevent the execution of the plan later on, the agent may want to recover from this discrepancy. For that, the agent can undo the plan execution until the diagnosed point of failure (to reach S_3) by moving block p onto the table, and then continue with the plan execution.

As described in the example above, at Step 2, the diagnosis of the detected discrepancies found by the agent can be helpful in two ways. First, it provides an explanation for some weaknesses of the plan being executed, and such an explanation can be used to determine some checkpoints specifying when to check for discrepancies. These checkpoints can be useful, especially when this plan is executed many times. Second, it can be used for plan recovery, at Step 3. For instance, if the diagnosis of discrepancies provides a possible point of failure for the discrepancies then, at Step 3, the monitoring agent can backtrack to this diagnosed point of failure, and execute the plan from that point on, as described in [5].

In a more general framework, at Step 2, the agent may check whether the detected discrepancies are "relevant" to the successful execution of the rest of the plan, like in [11, 21, 22], considering some other possible trajectories as well. Also, the agent does not have to find a diagnosis of discrepancies every time there is a detected discrepancy. When to look for a diagnosis can be handled by a decision support model. On the other hand, Step 3 of the framework can be refined further. Depending on the diagnosis of discrepancies, the length of the plan executed so far, the length of the remaining plan, and possibly some other criteria, the agent, with the help of a decision support model, can pick a plan recovery method among many, including replanning, backtracking, and patch planning. Such extensions and refinements of the framework above are possible. However, in this paper, we will not study them. In the following, we will concentrate on how to detect discrepancies and find diagnoses of these discrepancies, and we will give formal specifications of that.

4 Discrepancies

Suppose that we are given a planning problem whose domain is represented in the action representation framework of Section 2, whose initial conditions and goal conditions are described by formulas $\mathit{init}(\mathcal{F})$ and $\mathit{goal}(\mathcal{F})$ respectively.

Let $\mathcal{T}$ be a set of trajectories of the form

$$S_0, A_0, S_1, \ldots, S_{n-1}, A_{n-1}, S_n$$

described by a formula

$$\bigwedge_{t=0}^{n-1} tr(\mathcal{F}_t, \mathcal{A}_t, \mathcal{F}_{t+1}) \wedge \phi(\mathcal{F}_0, \mathcal{A}_0, \ldots, \mathcal{A}_{n-1}, \mathcal{F}_n) \tag{5}$$

such that each S_i corresponds to an interpretation of $\mathcal{F}_i$, and each A_i corresponds to an interpretation of $\mathcal{A}_i$. Here, intuitively, the formula $\phi(\mathcal{F}_0, \mathcal{A}_0, \ldots, \mathcal{A}_{n-1}, \mathcal{F}_n)$ specifies some "intended" or "preferred" conditions on trajectories of length n, and, thus, $\mathcal{T}$ describes some "intended" or "preferred" trajectories.

In the following, formula (5) will be denoted by

$$traj^{\mathcal{T}}(\mathcal{F}_0, \mathcal{A}_0, \ldots, \mathcal{A}_{n-1}, \mathcal{F}_n).$$

Let $P = \langle A_0, \ldots, A_{n-1} \rangle$ be a plan of length n for the given planning problem. Let S_i be the state reached at stage i from an initial state after action occurrences $A_0, \ldots, A_{i-1}$. We say that there is a *discrepancy* between S_i and $\mathcal{T}$ relative to P, if there is no trajectory

$$S'_0, A_0, S'_1, \ldots, S'_{n-1}, A_{n-1}, S'_n$$

in $\mathcal{T}$ such that S'_n is a goal state and $S_i = S'_i$. In our framework, this can be expressed as follows. Let us denote by $discrepancy_i^{P,\mathcal{T}}(\mathcal{F}_i)$ the formula

$$\forall \mathcal{F}'_0, \ldots, \mathcal{F}'_n\, traj^{\mathcal{T}}(\mathcal{F}'_0, A_0, \ldots, A_{n-1}, \mathcal{F}'_n) \wedge goal(\mathcal{F}'_n) \supset \neg(\mathcal{F}'_i \equiv \mathcal{F}_i).$$

There is a discrepancy between S_i and $\mathcal{T}$ relative to P if $discrepancy_i^{P,\mathcal{T}}(S_i)$ holds. Note that in general not all trajectories in $\mathcal{T}$ are feasible for P.

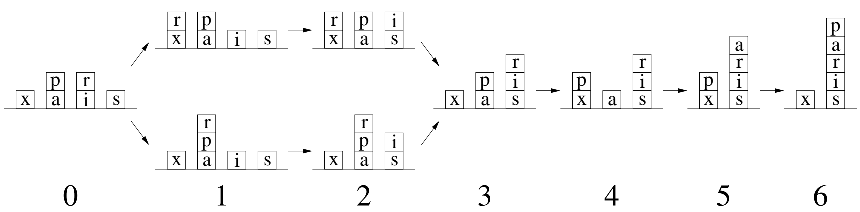
Example 4 (Example 3 continued) Consider the planning problem $\mathcal{P}$. Assume that the agent prefers to put a block onto a block instead of the table; so she considers the set $\mathcal{T}$ of trajectories

$$S_0, A_0, S_1, \ldots, S_5, A_{n-1}, S_6$$

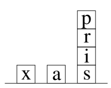
such that $move_i(B, table) \notin A_i$ for any block B. The set $\mathcal{T}$ can be described by the formula

$$traj^{\mathcal{T}}(\mathcal{F}_0, \mathcal{A}_0, \ldots, \mathcal{A}_5, \mathcal{F}_6) = \bigwedge_{t=0}^{5} tr(\mathcal{F}_t, \mathcal{A}_t, \mathcal{F}_{t+1}) \wedge \bigwedge_{i=0,\ldots,5,\ B=a,p,x,i,r,s} \neg move_i(B, table)$$

and it can be presented by the graph



Assume that, during the execution of P, at time stamp 4, the following state S_4 is observed:



Since this state is different from the state at time stamp 4 of any trajectory in $\mathcal{T}$, there is a discrepancy between S_4 and $\mathcal{T}$ relative to P.□

5 Diagnosis of Discrepancies

We identify the point of failure for an observed discrepancy between S_i and $\mathcal{T}$ relative to P by finding the latest time stamp after which every "evolution" of S_i "deviates" from every trajectory in $\mathcal{T}$.[4] One such possible latest time stamp is the one *at* which an evolution of S_i "matches" a trajectory in $\mathcal{T}$; another possible latest time stamp is the one *until* which an evolution of S_i "matches" a trajectory in $\mathcal{T}$.

A point of failure k provides, for instance, the execution of a sequence $\langle A_k, \ldots, A_{i-1}\rangle$ of actions at time stamp k as an explanation for a discrepancy observed at time stamp i ($i > k$). Therefore, for a discrepancy observed at an initial state, at time stamp 0, there is no point of failure.

In the following, first we define an evolution of a state from an initial state relative to a plan, and then precisely describe the two kinds of diagnoses of discrepancies mentioned above.

An *evolution of a state* S_i reached at time stamp i from an initial state S_0 after action occurrences $A_0, \ldots, A_{i-1}$ of a plan P is a trajectory

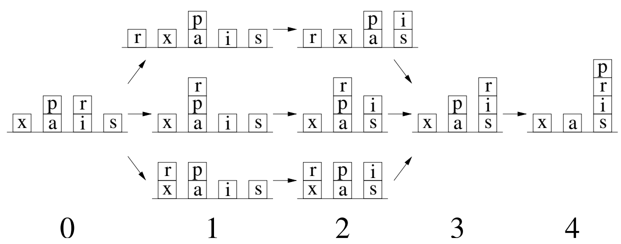
$$S_0, A_0, S_1, \ldots, S_{i-1}, A_{i-1}, S_i$$

obtained by finding a model of the formula

$$\mathit{init}(\mathcal{F}_0) \wedge \bigwedge_{t=0}^{i-1} \mathit{tr}(\mathcal{F}_t, A_t, \mathcal{F}_{t+1}),$$

where every S_i is the state that corresponds to an interpretation of $\mathcal{F}_i$.We will denote this formula by $\mathit{traj}^P(\mathcal{F}_0, \ldots, \mathcal{F}_i)$.

Example 5 (Example 4 continued) The evolutions of the state S_4 are:



$\square$

5.1 State-oriented diagnosis

In the first approach, a point of failure for a discrepancy between S_i and $\mathcal{T}$ relative to P is characterized by a state S_k and a time stamp k such that the following holds:

[4]The reason we pick the latest time stamp as the point of failure is the intuition that, by Occam's Razor, backtracking in the process of recovering from discrepancies can be done with least effort for it.

(S1) *(at-point of deviation)* Some evolution of S_i "matches" a goal-establishing trajectory in $\mathcal{T}$ "at" time stamp k ($0 \leq k < i \leq n$) in state S_k and deviates from the trajectory at time stamp $k+1$.

(S2) (*at-maximality*) No evolution of S_i matches a goal-establishing trajectory in $\mathcal{T}$ at a time stamp greater than k.

Towards a formalization, we say that an evolution of S_i *matches* a goal-establishing trajectory in $\mathcal{T}$ *at* time stamp k ($0 \leq k \leq i \leq n$) if the sentence $matchState^{P,\mathcal{T}}_{k,i}(S_i)$, defined by

$$\exists \mathcal{F}_0, \ldots, \mathcal{F}_{i-1}, \mathcal{F}'_0, \ldots, \mathcal{F}'_n \; matchAt^{P,\mathcal{T}}_{k,i}(\mathcal{F}_0, \ldots, \mathcal{F}_{i-1}, S_i, \mathcal{F}'_0, \ldots, \mathcal{F}'_n), \tag{6}$$

holds, where $matchAt^{P,\mathcal{T}}_{k,i}(\mathcal{F}_0, \ldots, \mathcal{F}_i, \mathcal{F}'_0, \ldots, \mathcal{F}'_n)$ stands for the formula

$$\begin{array}{l} \mathit{traj}^P(\mathcal{F}_0, \ldots, \mathcal{F}_k, \mathcal{F}_{k+1}, \ldots, \mathcal{F}_i) \wedge \\ \mathit{traj}^{\mathcal{T}}(\mathcal{F}'_0, A_0, \ldots, \mathcal{F}'_k, A_k, \mathcal{F}'_{k+1}, \ldots, A_{n-1}, \mathcal{F}'_n) \; \wedge \mathit{goal}(\mathcal{F}'_n) \wedge \\ \mathcal{F}'_k \equiv \mathcal{F}_k. \end{array}$$

In the formula above, the first conjunct describes an evolution of a state reached at time stamp i (i.e., a state described by an interpretation of $\mathcal{F}_i$) while executing a plan P. The second and the third conjuncts describe an intended trajectory in $\mathcal{T}$ that achieves the goal. The fourth conjunct expresses that, at time stamp k, the corresponding states of the evolution and the intended trajectory, given by the interpretations of $\mathcal{F}_k$ and $\mathcal{F}'_k$, respectively, are identical.

Condition (S1) is expressed by $deviateState^{P,\mathcal{T}}_{k,i}(S_i, S_k)$ defined as the sentence

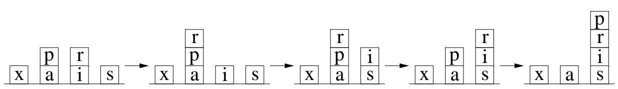
$$\begin{array}{l} \exists \mathcal{F}_0, \ldots, \mathcal{F}_{k-1}, \mathcal{F}_{k+1}, \ldots, \mathcal{F}_{i-1}, \mathcal{F}'_0, \ldots, \mathcal{F}'_n \\ \quad matchAt^{P,\mathcal{T}}_{k,i}(\mathcal{F}_0, \ldots, \mathcal{F}_{k-1}, S_k, \mathcal{F}_{k+1}, \ldots, \mathcal{F}_{i-1}, S_i, \mathcal{F}'_0, \ldots, \mathcal{F}'_n) \wedge \\ \quad \neg(\mathcal{F}'_{k+1} \equiv \mathcal{F}_{k+1}) \end{array} \tag{7}$$

and condition (S2) is expressed by the sentence

$$\bigwedge_{j=k+1}^{i} \neg matchState^{P,\mathcal{T}}_{j,i}(S_i). \tag{8}$$

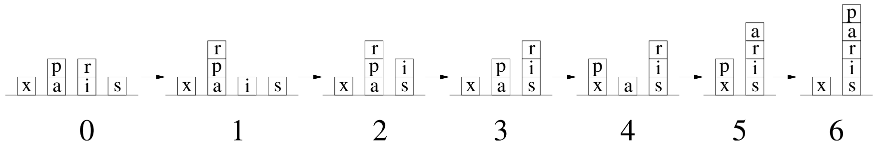
A *state-oriented point of failure* (or a *state-oriented diagnosis*) for a discrepancy between S_i and $\mathcal{T}$ relative to P at time i is then a pair (S_k, k) of a state S_k and a time stamp k ($0 \leq k < i \leq n$) such that the sentence $(7) \wedge (8)$, denoted by $diagnosisState^{P,\mathcal{T}}_{k,i}(S_i, S_k)$, holds.

Example 6 In the setting of Example 5, consider the following evolution of S_4

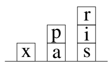


0 1 2 3 4

and the following intended trajectory in $\mathcal{T}$



Let S_3 be the state:



Since the states of the evolution and the intended trajectory at time stamp 3 are identical to S_3, and since the states of the evolution and the intended trajectory at time stamp 4 are different, $deviateState^{P,\mathcal{T}}_{3,4}(S_4, S_3)$ holds.

Also, since there is no evolution of S_4, and no intended trajectory in $\mathcal{T}$ such that their states at time stamp 4 are identical, $matchState^{P,\mathcal{T}}_{4,4}(S_4)$ does not hold. Therefore, $diagnosisState^{P,\mathcal{T}}_{3,4}(S_4, S_3)$ holds, and $(S_3, 3)$ is a state-oriented point of failure.

Moreover, since $matchState^{P,\mathcal{T}}_{3,4}(S_4)$ holds, $\bigwedge_{j=k+1}^{4} \neg matchState^{P,\mathcal{T}}_{j,4}(S_4)$ does not hold for $k = 1, 2$. Then, $diagnosisState^{P,\mathcal{T}}_{k,4}(S_k, S_4)$ does not hold for $k = 1, 2$. Therefore, $(S_3, 3)$ is the only state-oriented point of failure.□

Example 7 Consider a variation of Example 1 where the action $dryWithTowel$ may not make the puppy dry sometimes. Let $\mathcal{T}$ be the singleton consisting of the trajectory (3) presented in Example 2. Suppose that the state reached at time stamp 1, from the state $S_0 = \{wet\}$, by executing the plan

$$P = \langle dryWithTowel, putIntoWater\rangle,$$

is $\{wet\}$, which may have evolved as

$$\{wet\}, \{dryWithTowel\}, \{wet\}. \tag{9}$$

Then, $(S_0, 0)$ is the only state-oriented point of failure. Indeed, the initial states of the evolution (9) and the intended trajectory (3) are identical whereas their next states (at time stamp 1) are different.□

5.2 History-oriented diagnosis

In the second approach, a *point of failure* for a discrepancy between S_i and $\mathcal{T}$ relative to P is characterized by a state S_k and a time stamp k which satisfy the following alternative conditions:

(H1) (*until-point of deviation*) Some evolution of S_i "matches" a goal-establishing trajectory in $\mathcal{T}$ "until" time stamp k in state S_k and deviates from the trajectory at time stamp $k + 1$.

(H2) (*until-maximality*) No evolution of S_i matches a goal-establishing trajectory in $\mathcal{T}$ until time stamp $k + 1$.

Note that, with this approach, unlike the state-oriented one, the monitoring agent is concerned about that the agent "obeys" a goal-establishing trajectory in $\mathcal{T}$ while executing the plan.[5] Here $\mathcal{T}$ may contain some trajectories satisfying the initial conditions for the given planning problem. If these initial conditions are known to the monitoring agent, she can use them to narrow the set of goal-establishing trajectories in $\mathcal{T}$ she considers.

We say that an evolution of S_i *matches* a goal-establishing trajectory in $\mathcal{T}$ *until* time stamp k ($0 \leq k \leq i \leq n$), if the sentence $matchHistory_{k,i}^{P,\mathcal{T}}(S_i)$, defined as

$$\exists \mathcal{F}_0, \ldots, \mathcal{F}_{i-1}, \mathcal{F}'_{k+1}, \ldots, \mathcal{F}'_n \; matchUntil_{k,i}^{P,\mathcal{T}}(\mathcal{F}_0, \ldots, \mathcal{F}_{i-1}, S_i, \mathcal{F}'_{k+1}, \ldots, \mathcal{F}'_n) \tag{10}$$

holds, where $matchUntil_{k,i}^{P,\mathcal{T}}(\mathcal{F}_0, \ldots, \mathcal{F}_i, \mathcal{F}'_{k+1}, \ldots, \mathcal{F}'_n)$ stands for the formula

$$\begin{array}{l} traj^P(\mathcal{F}_0, \ldots, \mathcal{F}_k, \mathcal{F}_{k+1}, \ldots, \mathcal{F}_i) \wedge \\ traj^{\mathcal{T}}(\mathcal{F}_0, A_0, \ldots, \mathcal{F}_k, A_k, \mathcal{F}'_{k+1}, \ldots, A_{n-1}, \mathcal{F}'_n) \wedge goal(\mathcal{F}'_n). \end{array}$$

Condition (H1) is now expressed by $deviateHistory_{k,i}^{P,\mathcal{T}}(S_i, S_k)$, defined as the sentence

$$\begin{array}{l} \exists \mathcal{F}_0, \ldots, \mathcal{F}_{k-1}, \mathcal{F}_{k+1}, \ldots, \mathcal{F}_{i-1}, \mathcal{F}'_{k+1}, \ldots, \mathcal{F}'_n \\ \quad matchUntil_{k,i}^{P,\mathcal{T}}(\mathcal{F}_0, \ldots, \mathcal{F}_{k-1}, S_k, \mathcal{F}_{k+1}, \ldots, \mathcal{F}_{i-1}, S_i, \mathcal{F}'_{k+1}, \ldots, \mathcal{F}'_n) \wedge \\ \quad \neg(\mathcal{F}'_{k+1} \equiv \mathcal{F}_{k+1}) \end{array} \tag{11}$$

and condition (H2) is expressed by the sentence

$$\neg matchHistory_{k+1,i}^{P,\mathcal{T}}(S_i). \tag{12}$$

A *history-oriented point of failure* (or a *history-oriented diagnosis*) for a discrepancy between S_i and $\mathcal{T}$ relative to P at time i is then a pair (S_k, k) of a state S_k and a time stamp k ($0 \leq k < i \leq n$) such that the sentence $(11) \wedge (12)$, denoted by $diagnosisHistory_{k,i}^{P,\mathcal{T}}(S_i, S_k)$, holds.

Example 8 In the setting of Example 5, consider the evolution of S_4, the intended trajectory in $\mathcal{T}$, and the state S_3 shown in Example 6. Since the states of the evolution and the intended trajectory until time stamp 3 are identical, and the states of the evolution and the intended trajectory at time stamp 4 are different, $deviateHistory_{3,4}^{P,\mathcal{T}}(S_4, S_3)$ holds. Since there is no evolution of S_4, and no intended trajectory in $\mathcal{T}$ such that their states until time stamp 4 are identical, $matchHistory_{4,4}^{P,\mathcal{T}}(S_4)$ does not hold. Therefore, $diagnosisHistory_{3,4}^{P,\mathcal{T}}(S_4, S_3)$ holds, and $(S_3, 3)$ is a history-oriented point of failure.

Moreover, since $matchHistory_{k,4}^{P,\mathcal{T}}(S_4)$ holds for $k = 1, 2, 3$, $\neg matchHistory_{k+1,4}^{P,\mathcal{T}}(S_4)$ does not hold for $k = 1, 2$. Then, $diagnosisHistory_{k,4}^{P,\mathcal{T}}(S_4, S_k)$ does not hold for $k = 1, 2$. Therefore, $(S_3, 3)$ is the only history-oriented point of failure.□

[5] To guarantee that the execution exactly follows one of the intended trajectories until the point of failure, the monitoring agent must check for a discrepancy at every time stamp, since otherwise not one particular evolution of S_i may be considered.

5.3 Semantic properties of diagnoses

In this section, we discuss several properties of the notions of diagnosis defined earlier. In particular, we first verify that both notions comply with the principle of parsimony, then we examine the relationship between state- and history-oriented diagnosis, identifying a condition under which they coincide. Finally, we discuss issues concerning the existence of diagnoses.

5.3.1 Principle of Parsimony

As discussed in Sections 1 and 5, a desirable feature of diagnoses of discrepancies is, by applying Occam's Razor, that they identify points of failure according to which recovering from the detected discrepancies is possible with the least amount of effort. The idea is to undo the plan execution until a point of failure and then to reuse the plan from that point on. Therefore, ideally, these points of failure should be at a minimal distance from the state of the observed discrepancy, or, equivalently, at a maximal distance from the initial state. The following proposition shows that this is indeed the case according to our definitions.

Proposition 1 *For any discrepancy, if there is a state-oriented (resp. history-oriented) point of failure* (S_k, k)*, then there does not exist any state-oriented (resp. history-oriented) point of failure* (S_j, j) *such that* $j > k$*.*

We point out that if there is a state-oriented (resp. history-oriented) point of failure (S_k, k), also no state-oriented (resp. history-oriented) deviation exists for any time stamp $j > k$. This can be seen in the proof of Proposition 1.

5.3.2 Equivalence of State-oriented and History-oriented Diagnosis

Under some conditions, state-oriented diagnosis and history-oriented diagnosis coincide, as seen in Examples 6 and 8. The following proposition precisely describes one such condition.

Proposition 2 *Let* P *be a plan* $\langle A_0, \ldots, A_{n-1} \rangle$*, and let* $\mathcal{T}$ *be a set of trajectories such that, for every trajectory* $S_0, A_0, \ldots, A_{n-1}, S_n$ *in* $\mathcal{T}$*, init*(S_0) *holds. Let* S_k *be a state, and let* k *be a time stamp. Then, for any discrepancy between a state and* $\mathcal{T}$ *relative to* P*,* (S_k, k) *is a state-oriented point of failure iff* (S_k, k) *is a history-oriented point of failure.*

If the trajectories $S_0, A_0, \ldots, A_{n-1}, S_n$ in $\mathcal{T}$ do not start at an initial state, state-oriented diagnosis and history-oriented diagnosis are incomparable, in general, as seen in the following example.

Example 9 Let $\mathcal{F} = \{X\}$, and $\mathcal{A} = \{A\}$. Let *state*$(\mathcal{F})$ be any tautology, and let

$$act(\mathcal{F}, \mathcal{A}, \mathcal{F}') = ((\neg X \wedge A) \supset \neg X') \vee ((X \wedge \neg A) \supset X').$$

Let $\mathcal{P}$ be a planning problem with the initial state $\{X\}$, and with the goal state $\{X\}$. Suppose that a solution to this problem is the plan $P = \langle \{A\}, \emptyset \rangle$ of length 2.

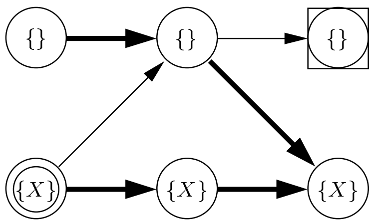


Figure 2: The trajectories in $\mathcal{T}$ (shown by the paths from 0 to 2 with thick edges) and the evolution of the state S_2 (shown by the path from 0 to 2 with thin edges), as described in Example 9.

Assume that the set $\mathcal{T}$ of intended trajectories is described by the formula

$$tr(\mathcal{F}_0, \{A_0\}, \mathcal{F}_1) \wedge tr(\mathcal{F}_1, \{\}, \{X_2\}) \wedge (\mathcal{F}_0 \equiv \mathcal{F}_1).$$

There are two trajectories obtained from the models of this formula:

$$T_1 = \{X\}, \{A\}, \{X\}, \{\}, \{X\}$$

and

$$T_2 = \{\}, \{A\}, \{\}, \{\}, \{X\}.$$

Since T_2 does not originate at the initial state, the condition of Proposition 2 is not satisfied.

The trajectories T_1 and T_2 are represented by paths from 0 to 2 with thick edges in Figure 2, where the initial state is double circled. For instance, the trajectory T_2 is represented by the path $\{\}, \{\}, \{X\}$ in this figure.

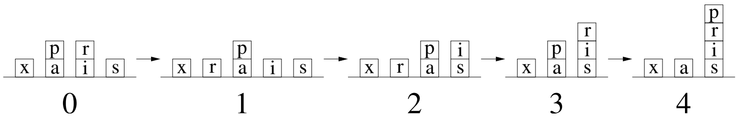
Assume that, while the agent is executing the plan P, at time stamp 2, the state $S_2 = \{\}$ is observed. This observed state is boxed in Figure 2. Note that there is a discrepancy between S_2 and $\mathcal{T}$ relative to P at time stamp 2. Evidently, there is only one evolution of S_2, which is $\{X\}, \{A\}, \{\}, \{\}, \{\}$, represented by the path $\{X\}, \{\}, \{\}$ in Figure 2.

The evolution of S_2 matches the goal-establishing trajectory T_1 at time stamp 0, and deviates from T_1 at time stamp 1. On the other hand, the evolution of S_2 matches the goal-establishing trajectory T_2 at time stamp 1, and deviates from T_2 at time stamp 2. Therefore, due to the maximality condition (S2), the state-oriented point of failure for the discrepancy detected at time stamp 2 is $(\{\}, 1)$, whereas the history-oriented point of failure for this discrepancy is $(\{X\}, 0)$.$\Box$

An important consequence of Proposition 2 is that, if the goal-establishing trajectories in $\mathcal{T}$ are feasible for the plan, one can freely choose between algorithms computing one of the two notions of diagnosis, discussed in Section 6, possibly yielding optimization.

Note that all evolutions of a state can be reconstructed using the action domain description and initial state description. If the agent obtains (partial) history information (e.g. by performing and recording periodic observations) then she can use it to restrict the set of evolutions of the observed state. Then, the two notions of diagnosis may become incomparable, as seen in the following example.

Example 10 Consider the set of evolutions of the observed state S_4 in Example 5, and assume that additional state information has been recorded at time stamp 1, restricting this set of evolutions to



In this scenario, with respect to the set $\mathcal{T}$ of trajectories presented in Example 4, the state-oriented diagnosis remains the same, i.e., $(S_3, 3)$ is the state-oriented point of failure for the discrepancy. However, for history-oriented diagnosis, the initial state is now the only point of explanation, since no evolution matches until time stamp 1.□

5.3.3 Existence of Diagnoses

It is not always the case that, when a discrepancy is detected, a diagnosis can be found for it. For instance, if a discrepancy is observed at time stamp 0 then no diagnosis exists for it, as discussed at the beginning of Section 5. On the other hand, for a discrepancy detected at a state S_i at time stamp i ($i > 0$), if no evolution of the observed state S_i matches with any of the goal-based intended trajectories at an earlier time stamp, there is no state-oriented diagnosis for the discrepancy, and if no evolution of the observed state S_i matches with any prefix of the goal-based intended trajectories, there is no history-oriented diagnosis for the discrepancy.

Concerning our monitoring framework, as described in Section 3, since in such a situation a recovery by backtracking is never feasible (even backtracking to the initial state does not recover), one has to resort to other means of recovery, such as replanning.

Let us first consider the relationship between state- and history-oriented diagnosis with respect to existence of a diagnosis. It turns out that the existence of a history-oriented diagnosis implies the existence of a state-oriented diagnosis (but not vice versa).

Proposition 3 *For any discrepancy, if there is a history-oriented point of failure* (S_k, k) *then there is a state-oriented point of failure* $(S_{k'}, k')$ *such that* $k' \geq k$.

The other direction is not always possible, as seen in the following example.

Example 11 Consider the action domain description, the planning problem $\mathcal{P}$, the plan P, and the observed state S_2 described in Example 9. Suppose that the set $\mathcal{T}$ consists of the trajectory T_2, i.e.,

$$\{\}, \{A\}, \{\}, \{\}, \{X\}$$

of Example 9. Then there is a discrepancy between S_2 and $\mathcal{T}$ relative to P at time stamp 2. Note that the evolution of S_2, i.e.,

$$\{X\}, \{A\}, \{\}, \{\}, \{\},$$

matches the goal-establishing trajectory T_2 at time stamp 1. This can be seen in Figure 2 as well. Therefore, $(\{\}, 1)$ is the state-oriented point of failure for the discrepancy detected at time stamp 2. However, there is no history-oriented point of failure for this discrepancy.□

A more general observation is that the absence of a diagnosis implies that none of the initial states of all possible evolutions of the observed state is in a goal-establishing trajectory in $\mathcal{T}$. So, conversely, if the goal-establishing trajectories in $\mathcal{T}$ contain all initial states, a diagnosis must necessarily exist.

Proposition 4 *Let P be a plan $\langle A_0, \ldots, A_{n-1}\rangle$ and let $\mathcal{T}$ be a set of trajectories such that for any state S_0, for which init(S_0) holds, $\mathcal{T}$ contains a trajectory $S_0, A_0, \ldots, A_{n-1}, S_n$, such that goal$(S_n)$ holds. Then, for any discrepancy, a point of failure exists.*

An interesting class of action domains are *deterministic* domains. In these, for any state and executed action set, there is a unique resulting state.

Definition 1 *An action domain is* deterministic *iff* $\forall \mathcal{F}, \mathcal{A}, \mathcal{F}', \mathcal{F}''(tr(\mathcal{F}, \mathcal{A}, \mathcal{F}') \wedge tr(\mathcal{F}, \mathcal{A}, \mathcal{F}'') \supset \mathcal{F}' \equiv \mathcal{F}'')$.

An important consequence of such a setting is that deviations cannot occur after the execution of actions. Since a deviation is a necessary condition for any diagnosis, no diagnosis can exist in such a setting.

Proposition 5 *If the action domain is deterministic then no diagnosis exists for a discrepancy.*

This means that if it is known that an action domain is deterministic, then the monitoring system can skip the test for diagnoses, as it will always be negative, and resort to a recovery method different from backtracking.

6 Computational Aspects

In this section, first we address the complexity of the three main computational problems in execution monitoring: detecting a discrepancy, verifying a diagnosis for a discrepancy, and finding a diagnosis for a discrepancy. After that, we introduce an algorithm to generate diagnoses for a discrepancy in a generic setting.

6.1 Computational Complexity

In the following, we assume that a background $\mathcal{B}$ of execution monitoring is formed by

- an action domain description, specified by a set $\mathcal{F}$ of fluents, a set $\mathcal{A}$ of actions, and formulas *state*$(\mathcal{F})$ and *act*$(\mathcal{F}, \mathcal{A}, \mathcal{F}')$,
- a planning problem $\mathcal{P}$ with the initial states specified by a formula *init*$(\mathcal{F})$, and with the goal conditions specified by a formula *goal*$(\mathcal{F})$,
- a plan $P = \langle A_0, \ldots, A_{n-1}\rangle$ of length n for the planning problem $\mathcal{P}$, and

- a set $\mathcal{T}$ of intended trajectories specified by a formula $traj^{\mathcal{T}}(\mathcal{F}_0, \mathcal{A}_0, \ldots, \mathcal{A}_{n-1}, \mathcal{F}_n)$

where each formula above is an arbitrary propositional formula under classical semantics, like in Example 1.

The generic setting above accommodates nondeterminism and incomplete information (e.g. there may be several initial states). Dealing with nondeterminism and incomplete information in action execution and planning is computationally complex in general, and leads to intractability even in plain settings [2, 7, 19, 25]; intuitively, nondeterminism causes branchings in execution of a plan, and thus enlarges the search space progressively. Due to this generic setting, one may expect that the computational tasks we are interested in are intractable in general, and at least NP-hard respectively coNP-hard. However, as shown by the results below, the complexity of the main problems in discrepancy detection and diagnosis is not much above the well-known complexity of propositional logic (NP for satisfiability and coNP for consequence, respectively) underlying the basic framework, and is in fact only "mildly" harder than NP and coNP.

We start with the problem of discrepancy detection. Here, we have to decide whether a state S_i observed at time stamp i is compatible with some trajectory in $\mathcal{T}$. The following theorem shows that this problem is intractable in general.

Theorem 1 *Given a background $\mathcal{B}$, a reached state S_i, and a time stamp i ($i \leq n$), deciding whether a discrepancy exists between S_i and $\mathcal{T}$ relative to P at time stamp i is coNP-complete.*

The problem of recognizing a diagnosis for a discrepancy is slightly harder. Intuitively, to recognize that a state S_k and a stage k characterize a point of failure for a discrepancy, we need to verify two conditions: (i) some goal-establishing trajectory in $\mathcal{T}$ and an evolution of the observed state S_i pass through S_k at stage k, and (ii) no goal-establishing trajectory in $\mathcal{T}$ and no evolution of the observed state S_i pass through a state at a stage later than k. The verification of (i) amounts to a satisfiability problem, and the verification of (ii) amounts to an independent *unsatisfiability* problem. Problems like the verification of both (i) and (ii) are captured by the complexity class D^p = $\{L \times L' \mid L \in \mathrm{NP},\ L' \in \mathrm{coNP}\}$, which can be viewed as the "conjunction" of NP and coNP.

Theorem 2 *Given a background $\mathcal{B}$, a state S_k, a reached state S_i, and time stamps k and i ($k < i \leq n$), such that there is a discrepancy between S_i and $\mathcal{T}$ relative to P at time stamp i, deciding whether (S_k, k) is a point of failure for the discrepancy is D^p-complete.*

The last computational problem we consider here is the computation of a point of failure (S_k, k) for a discrepancy. A naive way of solving this problem is simply to guess a state S_k and a time stamp k, and then to check whether (S_k, k) is a point of failure for the discrepancy. Since the second part of this method is coNP, due to Theorem 2, this naive way of problem solving is at the second level of the Polynomial Hierarchy.

In another way, some point of failure (S_k, k) for a discrepancy can be computed in (deterministic) polynomial time with the help of an NP oracle, using a technique similar to the one used for computing an optimal tour in the Traveling Salesman Problem (TSP) in [17]: First compute the stage k, and then construct S_k fluent by fluent, by making suitable calls to the NP oracle. A more refined analysis yields the following result.

Theorem 3 *Given a background $\mathcal{B}$, a reached state S_i, and a time stamp i ($i \leq n$), such that there is a discrepancy between S_i and $\mathcal{T}$ relative to P at time stamp i, computing some point of failure (S_k, k) for the discrepancy is* $\mathrm{FNP}//\mathrm{OptP}[O(\log n)]$*-complete.*

Here $\mathrm{FNP}//\mathrm{OptP}[O(\log n)]$ (for short, $\mathrm{FNP}//\log$) is a complexity class from [4]. Intuitively, $\mathrm{FNP}//\log$ contains all problems such that a solution for an instance I can be nondeterministically computed by a transducer in polynomial time, if the result $opt(I)$ of an NP optimization problem on I is known. Here, $opt(I)$ is an integer having $O(\log |I|)$ bits; an NP *optimization problem* is the problem of computing the maximum value of any solution for an instance I, given that deciding $opt(I) \geq k$ is in NP, and recognizing solutions is polynomial.

For example, computing the largest set S of pairwise connected nodes in a given graph G (i.e., a maximum clique) is a problem in $\mathrm{FNP}//\log$ (observe that different maximum cliques may exist). Indeed, computing the *size* of a maximum clique in G is an NP-optimization problem with $O(\log |G|)$ output bits, since testing whether a set S is a clique is easy (just check whether G has an edge between each pair of nodes in S), and deciding whether $opt(G) \geq k$ is in NP (guess a clique of size $\geq k$). Note, however, that this problem is not known to be $\mathrm{FNP}//\log$-complete.

We conclude this subsection with some remarks concerning variants and special cases of the problems above.

(1) The complexity results above hold in a general setting where there may be several initial states (in particular, when there is incomplete information about some fluents), and where actions can have nondeterministic effects. These results remain the same if there is a single initial state (i.e., there is no incompleteness). For deterministic actions, even in the presence of several initial states, diagnosis of a discrepancy is trivial due to Proposition 5.

(2) In a background $\mathcal{B}$ for execution monitoring, instead of *state*$(\mathcal{F})$, *act*$(\mathcal{F}, \mathcal{A}, \mathcal{F}')$, and *init*$(\mathcal{F})$, one can consider formulas $evols_i^P(\mathcal{F}_0, \ldots, \mathcal{F}_i)$ ($i \in \{0, \ldots, n\}$) that describe evolutions of states reached at time stamp i by executing P. Alternatively, one can consider a single multi-sorted formula $evols^P(I, \mathcal{F}_0, \mathcal{F}_1, \ldots, \mathcal{F}_n)$ where I is an integer variable, describing the evolutions. In such background settings, the computational problems we are interested in do not become harder, and we obtain the same complexity results.

(3) Similarly, the complexity results remain the same if the formulas in $\mathcal{B}$ contain hidden variables, i.e., existentially quantified propositional variables that are projected off. For example, in the formula $\phi(X, Y) = \exists Z(X \supset Z) \wedge (Z \supset Y)$, variable Z is a hidden variable; $\phi(X, Y)$ is logically equivalent to the quantifier-free formula $X \supset Y$. Hidden variables are useful as auxiliary variables in problem representations. On the other hand, using polynomially many such variables, the transition-based semantics of systems such as CCALC and DLV$^{\mathcal{K}}$ can be emulated by classical semantics. In fact, this applies to any action representation framework with a transition-based semantics, in which the constituents $state(\mathcal{F})$, $act(\mathcal{F}, \mathcal{A}, \mathcal{F}')$ can be decided in non-deterministic polynomial time. In such frameworks, the results above provide upper bounds on the complexity of the computational problems we consider. On the other hand, if the action representation framework above can be polynomially expressed in a particular representation framework, like that of CCALC and DLV$^{\mathcal{K}}$, then the results above provide lower bounds as well. Putting things together, the complexity results above hold if the background is described for CCALC and DLV$^{\mathcal{K}}$.

6.2 Computation

Algorithms for computing diagnoses have already been encountered in the discussion of the computational complexity in the previous subsection. However, these algorithms do not take into account possible functionalities of the existing systems for action representation and reasoning. For this reason, we consider here the computation of diagnoses at a generic level, which may be refined for the existing systems.

We assume that the description of the action domain as well as the set of trajectories in $\mathcal{T}$ (5) is fixed. Furthermore, we assume that a function *match_T*, which returns the set of all trajectories in $\mathcal{T}$ satisfying some given conditions, is available. The input of *match_T* are

- a sequence of actions $\langle A_0, \ldots, A_{n-1}\rangle$ of length n;
- a list $[M_0, M_1, \ldots, M_m]$ ($0 \leq m \leq n$) of sets of fluents, such that each M_i corresponds to an interpretation of $\mathcal{F}_i$; and of $\mathcal{A}_i$; and
- a formula $\phi(\mathcal{F})$ describing some conditions on $\mathcal{F}$.

The output of *match_T* is a set of prefixes

$$S_0, A_0, S_1, \ldots, S_{m-1}, A_{m-1}, S_m \tag{13}$$

of all trajectories

$$S_0, A_0, S_1, \ldots, S_{n-1}, A_{n-1}, S_n$$

in $\mathcal{T}$ such that the following hold:

(a) Prefix (13) matches M_0, A_0, …, A_{m-1}, M_m, i.e., $S_i = M_i$. A wild card symbol “?” may replace any M_i, matching any S_i;

(b) $\phi(S_n)$ holds.

For example, consider the action domain description presented in Example 1. Suppose that trajectory (3), i.e.,

$$\{wet\}, \{dryWithTowel\}, \{\}, \{putIntoWater\}, \{inWater, wet\},$$

belongs to $\mathcal{T}$, and suppose that $P = \langle\{dryWithTowel\}, \{putIntoWater\}\rangle$ and $\phi(\mathcal{F})$ = $inWater \wedge wet$. Then, one of the prefixes returned by *match_T*$(P, [\{wet\}, ?], \phi)$ is

$$\{wet\}, \{dryWithTowel\}, \{\}. \tag{14}$$

Indeed, prefix (14) matches trajectory (3), and $\phi(\mathcal{F})$ holds at state $\{inWater, wet\}$. On the other hand, prefix (14) is not a part of the output of *match_T*$([\{wet\}], \phi(\mathcal{F}))$ because the prefix (14) does not match trajectory (3).

We assume that a similar function

$$\textit{match\_Plan}(P, [M_0, M_1, \ldots, M_m])$$

is available, which returns, for the given plan $P{=}\langle A_0, \ldots, A_{n-1}\rangle$, the set of all evolutions $S_0, A_0, \ldots, A_{m-1}, S_m$ of states S_m according to P that match M_0, A_0, M_1, ..., A_m, M_m.

For example, consider the plan $P{=}\langle\{\mathit{dryWithTowel}\}, \{\mathit{putIntoWater}\}\rangle$ for the planning problem with the initial state $\{\mathit{wet}\}$ and the goal condition $\mathit{wet}\wedge\mathit{inWater}$. Then, *match_Plan*$(P, [\{\mathit{wet}\}, ?])$ returns the single evolution (14), whereas *match_Plan*$(P, [\{\mathit{wet}\}, ?, ?])$ returns the two evolutions: (3) and $\{\mathit{wet}\}, \{\mathit{dryWithTowel}\}, \{\}, \{\mathit{putIntoWater}\}, \{\}$.

The functions *match_T* and *match_Plan* can be easily implemented on top of the systems CCALC and DLV$^{\mathcal{K}}$, by pushing the values of fluents and actions into a planning problem, and then by returning all trajectories.

The problem of discrepancy detection between a state S_i and $\mathcal{T}$ relative to P at stage i is then accomplished by testing whether

$$\mathit{match\_T}(P, [?, \ldots, ?, S_i], \mathit{goal}(\mathcal{F})) = \emptyset.$$

Exploiting the functions *match_T* and *match_Plan*, the state-oriented diagnoses for a discrepancy between an observed state S_i and $\mathcal{T}$ relative to P at a time stamp i can be computed by the generic algorithm STATE_DIAGNOSIS presented in Figure 3. This algorithm computes all state-oriented point of failures iteratively, by finding, for each evolution of S_i, the maximum time stamp k and a set of states S_k such that the evolution deviates from every goal-establishing trajectory in $\mathcal{T}$ at state S_k at stage k. For computing history-oriented diagnoses for a discrepancy, we can use the algorithm HISTORY_DIAGNOSIS presented in Figure 4. This algorithm is obtained from STATE_DIAGNOSIS by plugging the states S_0, ..., S_{j-1}, extracted from the trajectories returned by *match_Plan*, in the call to *match_T*.

These basic algorithms can be refined and improved in different ways. For instance, caching techniques can be used to reduce the number of calls to *match_T* and *match_Plan*. Such modifications of the algorithms are not discussed here.

7 Related Work

Other frameworks that describe execution monitoring in a logic-based framework for reasoning about actions, in which a planning problem can be formulated and solved, are [11, 21, 22] and [9]. In the former three, execution monitoring is described in the situation calculus as in [18], which makes them applicable to Golog programs [14]. In [9], the authors describe execution monitoring in the fluent calculus [23] for FLUX programs [24]. Our framework is different from these logic-based frameworks with respect to discrepancy detection and diagnosis as follows.

First of all, we assume that the current state is observed by the agent, but we do not discuss how the description of this state is obtained. In [11], the authors assume that the agent knows which exogenous actions the environment has executed, so the current situation can be identified easily. In [21, 22] and in [9], the authors introduce sensing actions so that the monitoring agent can find the truth values of some fluents at the current situation.

In our framework, the monitoring agent detects a discrepancy when the current state and the corresponding states of the given trajectories are different. In [11, 21, 22], the agent detects a

Algorithm STATE_DIAGNOSIS(S_i, i)

Input: Observed state S_i at stage i satisfying $discrepancy_i^{P,\mathcal{T}}(S_i)$,
with respect to plan $P{=}\langle A_0, \ldots, A_{n-1}\rangle$ and trajectories $\mathcal{T}$.
Output: All state-oriented diagnoses (S_k, k) for the discrepancy

PoF_States := $\emptyset$; k := 0;
for each $S_0, A_0, \ldots, A_{i-1}, S_i \in$ *match_Plan*$(P, [?, \ldots, ?, S_i])$ **do**
begin j := $i - 1$
  while $j \geq k$ **do begin**
    if *match_T*$(P, [?, \ldots, ?, S_j], goal(\mathcal{F})) \neq \emptyset$ **then**
      if $j > k$ **then**
      begin k := j; PoF_States := $\{S_k\}$ **end**
      else PoF_States := PoF_States $\cup\{S_k\}$
    j := $j - 1$
  end
end;
output $\{(S, k) \mid S \in$ PoF_States $\}$.

Figure 3: A generic algorithm to compute all state-oriented diagnoses.

Algorithm HISTORY_DIAGNOSIS(S_i, i)

Input: Observed state S_i at stage i satisfying $discrepancy_i^{P,\mathcal{T}}(S_i)$,
with respect to plan $P{=}\langle A_0, \ldots, A_{n-1}\rangle$ and trajectories $\mathcal{T}$.
Output: All history-oriented diagnoses (S_k, k) for the discrepancy

PoF_States := $\emptyset$; k := 0;
for each $S_0, A_0, \ldots, A_{i-1}, S_i \in$ *match_Plan*$(P, [?, \ldots, ?, S_i])$ **do**
begin j := $i - 1$
  while $j \geq k$ **do begin**
    if *match_T*$(P, [S_0, S_1, \ldots, S_{j-1}, S_j], goal(\mathcal{F})) \neq \emptyset$ **then**
      if $j > k$ **then**
      begin k := j; PoF_States := $\{S_k\}$ **end**
      else PoF_States := PoF_States $\cup\{S_k\}$
    j := $j - 1$
  end
end;
output $\{(S, k) \mid S \in$ PoF_States $\}$.

Figure 4: A generic algorithm to compute all history-oriented diagnoses.

discrepancy when the remaining plan (or, in general, program) is not successful, considering all possible trajectories. In [9], the agent detects a discrepancy when the action is not executable or when the effects of an action are not as intended, considering all possible transitions, i.e., trajectories of length 1. Though a precise description of a discrepancy is missing in this paper.

In both [9] and our framework, the detected discrepancies may not be relevant to the successful execution of the rest of the plan: the effects of actions may not be as intended but the execution of the rest of the plan may lead to a goal state.

Our notion of diagnosis can provide explanations for detected discrepancies. In [11, 21, 22], the problem of finding explanations for detected discrepancies is not considered. In [9], explanations for action failures are specified by introducing an abnormality predicate for each action. For instance, if the action of picking milk at the grocery store fails because the agent can not pay for the milk, then one possible explanation for this failure is that the agent might have picked a more expensive milk. To take into account this abnormality, the authors introduce an abnormality predicate for the action of picking milk, and modify the state update axioms accordingly. Note that, later on, if the user decides to consider some other possible abnormalities, he needs to modify the domain description again. In our framework, the user can specify such abnormalities by a formula, whose models describe the intended or preferred trajectories, and thus does not have to modify the domain description for various abnormalities.

Another difference of our work from [11, 21, 22, 9] is that we have analyzed, in our framework, the computational complexity of detecting discrepancies and finding diagnosis of discrepancies.

Another line of work that is related to diagnosis of discrepancies is the work on diagnostic reasoning in the sense of [3] and [1], where action descriptions can be represented by transition diagrams. The kind of diagnosis discussed in [3] and [1] is different from ours in that they detect faulty components of a system by observing its behavior. For instance, if the light is not on when the lamp is turned on, it may be because the bulb is broken. To diagnose such faulty components, the authors introduce an abnormality predicate for each component, and describe the effects of actions by default rules. For instance, instead of saying that light is on after the lamp is turned on, they would say that light is on after the lamp is turned on if the bulb is not broken. To obtain an explanation, they also introduce an exogenous action for each abnormality of a component, like, for instance, the action of breaking the bulb. Like in [9], if the user decides to consider some other possible abnormalities of components, he needs to modify the domain description.

Note that another advantage of describing an intended or preferred behavior of an agent with a set of trajectories, instead of abnormality predicates, is that it can be formulated separately from the action domain description. For instance, we can formalize the action domain in an action language such as $\mathcal{C}+$ [13] or $\mathcal{K}$ [6], and specify the intended or preferred behavior of an agent with a logic program. This kind of modular representation allows us to talk about intended or preferred behavior of an agent beyond single transitions if the action domain is described in a language, like the action languages [10], where we can not easily describe the effects of a sequence of several actions, like moving several blocks consecutively onto the table, without introducing new auxiliary atoms. Also, this kind of modularity gives the user the flexibility of not having to determine all possible abnormalities while describing the action domain, thus leading to more elaboration tolerant representations [16].

On the other hand, some abnormalities can not be identified from the given set of trajectories. For instance, if the light is not on, we can find out, with respect to a given set of trajectories, that there is something wrong with the execution of the action of switching the lamp on; however, we can not pinpoint that it is because the bulb is broken if this abnormality is not specified in the action domain description. Fortunately, our modular representation framework allows us to use abnormality predicates in the action domain description to express such abnormalities, and, at the same time, use a set of trajectories to express the intended or preferred behavior of the agent.

8 Conclusion

We have investigated two problems related to monitoring of a plan execution: detection of discrepancies, and finding explanations for detected discrepancies. We have described some solutions to these problems in a logic-based framework, different from those suggested in the literature. According to our solution to the former problem, the monitoring agent checks the compatibility of the plan execution with a given set of intended trajectories, considering the current state information without a log of the past states. For the latter problem, the monitoring agent finds points of failure by comparing the possible evolutions of the observed state with the given trajectories. Such a comparison can be made in two ways leading to state- and history-oriented explanations of discrepancies that single out points of failures. These points of failures can describe some weaknesses of the plan being executed, and can be considered as checkpoints specifying when to check for discrepancies. This can be useful, in particular, if the plan is executed many times. The points of failures can also be used for recovering from the detected discrepancies. For instance, the monitoring agent can roll back the plan execution until the point of failure from which the remainder of the plan can be (re)executed, and thus can avoid an expensive (re)planning step. This kind of plan recovery with backtracking can be useful, in particular, when the remaining plan is very long. We have shown that state- and history-oriented explanations of discrepancies coincide in a common setting, and have investigated semantical properties of diagnoses. Finally, we have analyzed their complexity and we have addressed the computation of diagnoses on top of existing action and planning frameworks. An implementation of our approach for the DLV$^{\mathcal{K}}$ action and planning framework, as discussed in Section 6, is currently underway.

Several issues remain for further research. One issue is the extension of the results in this paper to action descriptions with partially observable states. Another interesting extension would be to monitor execution of non-linear plans, such as conditional plans, with respect to a set of "intended" or "preferred" trajectories. The main difference to the setting here is that alternative sequences of actions must be respected that according to the plan, might be executed in the future or have been executed in the past (unless the action history is known). A different direction is the usage and integration of these results for plan recovery in a more general monitoring framework, as discussed in Section 3. This is part of our future work.

Acknowledgments We thank Matthias Fichtner and Mikhail Soutchanski for useful discussions and other colleagues for helpful comments. This work was supported by FWF (Austrian Science

Funds) under project P16536-N04.

A Proofs of Theorems

A.1 Proof of Proposition 1

Proposition 1 *For any discrepancy, if there is a state-oriented (resp. history-oriented) point of failure (S_k, k), then there does not exist any state-oriented (resp. history-oriented) point of failure (S_j, j) such that $j > k$.*

Proof. Suppose that there is a state-oriented point of failure (S_k, k) for a discrepancy between a state S_i and a set $\mathcal{T}$ of trajectories relative to a plan P at a time stamp i, i.e., $diagnosisState^{P,\mathcal{T}}_{k,i}(S_i, S_k)$ holds. Then, by definition, for every j $(0 \leq k < j \leq i)$, $\neg matchState^{P,\mathcal{T}}_{j,i}(S_i)$ holds. That is, for every j $(0 \leq k < j \leq i)$, there do not exist any states $S_0, \ldots, S_{i-1}, S'_0, \ldots, S'_n$ such that $matchAt^{P,\mathcal{T}}_{j,i}(S_0, \ldots, S_{i-1}, S_i, S'_0, \ldots, S'_n)$ holds. Then, by definition, there does not exist a time stamp j $(0 \leq k < j < i)$ such that, for some state S_j, $deviateState^{P,\mathcal{T}}_{j,i}(S_i, S_j)$ holds. Therefore, by the definition of a state-oriented point of failure, there does not exist a time stamp j $(0 \leq k < j < i)$ such that, for some state S_j, $diagnosisState^{P,\mathcal{T}}_{j,i}(S_i, S_j)$ holds. The uniqueness of a history-oriented point of failure can be shown similarly. □

A.2 Proof of Proposition 2

Proposition 2 *Let P be a plan $\langle A_0, \ldots, A_{n-1}\rangle$, and let $\mathcal{T}$ be a set of trajectories such that, for every trajectory $S_0, A_0, \ldots, A_{n-1}, S_n$ in $\mathcal{T}$, init(S_0) holds. Let S_k be a state, and let k be a time stamp. Then, for any discrepancy between a state and $\mathcal{T}$ relative to P, (S_k, k) is a state-oriented point of failure iff (S_k, k) is a history-oriented point of failure.*

We will use the following lemmas for the proof of Proposition 2.

Lemma 1 *Let P be a plan $\langle A_0, \ldots, A_{n-1}\rangle$, and $\mathcal{T}$ be a set of trajectories. Let i and k $(0 \leq k < i \leq n)$ be time stamps, and let S_i be a state. Then the following hold:*

(a) $matchHistory^{P,\mathcal{T}}_{k,i}(S_i) \supset matchState^{P,\mathcal{T}}_{k,i}(S_i)$;

(b) for every state S_k, $deviateHistory^{P,\mathcal{T}}_{k,i}(S_i, S_k) \supset deviateState^{P,\mathcal{T}}_{k,i}(S_i, S_k)$.

Lemma 2 *Let P be a plan $\langle A_0, \ldots, A_{n-1}\rangle$, and let $\mathcal{T}$ be a set of trajectories such that, for every trajectory $S_0, A_0, \ldots, A_{n-1}, S_n$ in $\mathcal{T}$, init(S_0) holds. Let i and k $(0 \leq k < i \leq n)$ be time stamps, and let S_i be a state. Then the following hold:*

(a) $matchState^{P,\mathcal{T}}_{k,i}(S_i) \equiv matchHistory^{P,\mathcal{T}}_{k,i}(S_i)$;

(b) for every state S_k, $deviateState^{P,\mathcal{T}}_{k,i}(S_i, S_k) \equiv deviateHistory^{P,\mathcal{T}}_{k,i}(S_i, S_k)$.

Proof of Proposition 2. Let P be a plan $\langle A_0, \ldots, A_{n-1}\rangle$, and let $\mathcal{T}$ be a set of trajectories such that, for every trajectory $S_0, A_0, \ldots, A_{n-1}, S_n$ in $\mathcal{T}$, *init*(S_0) holds. Suppose that a discrepancy is detected between a state S_i and $\mathcal{T}$ relative to P at a time stamp i ($i \leq n$). Let k be a time stamp such that $0 \leq k < i \leq n$, and S_k be a state. We want to show that

$$diagnosisState_{k,i}^{P,\mathcal{T}}(S_i, S_k) \equiv diagnosisHistory_{k,i}^{P,\mathcal{T}}(S_i, S_k).$$

Assume that $diagnosisState_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds. That is, by definition, $deviateState_{k,i}^{P,\mathcal{T}}(S_i, S_k)$, and, for all j ($k < j \leq i$), $\neg matchState_{j,i}^{P,\mathcal{T}}(S_i)$ hold. By Lemma 2(b), $deviateHistory_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds, and by Lemma 2(a), $\neg matchHistory_{k+1,i}^{P,\mathcal{T}}(S_i)$ holds. Therefore, $diagnosisHistory_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds.

Assume that $diagnosisHistory_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds. That is, by definition, $deviateHistory_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ and $\neg matchHistory_{k+1,i}^{P,\mathcal{T}}(S_i)$ hold. By the latter, it follows that for all j ($k < j \leq i$), $\neg matchHistory_{j,i}^{P,\mathcal{T}}(S_i)$ holds. By Lemma 2(b), $deviateState_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds, and by Lemma 2(a), for all j ($k < j \leq i$), $\neg matchState_{j,i}^{P,\mathcal{T}}(S_i)$ holds. Therefore, $diagnosisState_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds. $\square$

Proof of Lemma 1. Let P be a plan $\langle A_0, \ldots, A_{n-1}\rangle$, and $\mathcal{T}$ be a set of trajectories. Let i and k ($0 \leq k < i \leq n$) be time stamps, and let S_i be a state.

(a) Suppose that $matchHistory_{k,i}^{P,\mathcal{T}}(S_i)$ holds. Then, by definition, for some states $S_0, \ldots, S_{i-1}$, $S'_{k+1}, \ldots, S'_n$,

$$matchUntil_{k,i}^{P,\mathcal{T}}(S_0, \ldots, S_{i-1}, S_i, S'_{k+1}, \ldots, S'_n) \tag{15}$$

holds. That is, by definition, there is an evolution

$$S_0, A_0, S_1, \ldots, S_k, A_k, S_{k+1}, \ldots, S_{i-1}, A_{i-1}, S_i$$

of the state S_i, and a goal-based trajectory

$$S_0, A_0, S_1, \ldots, S_k, A_k, S'_{k+1}, \ldots, S'_{n-1}, A_{n-1}, S'_n$$

in $\mathcal{T}$ such that *goal*(S'_n). Then, by definition,

$$matchAt_{k,i}^{P,\mathcal{T}}(S_0, \ldots, S_{i-1}, S_i, S_0, \ldots, S_k, S'_{k+1}, \ldots, S'_n) \tag{16}$$

holds as well. Therefore, if $matchHistory_{k,i}^{P,\mathcal{T}}(S_i)$ holds, then $matchState_{k,i}^{P,\mathcal{T}}(S_i)$ holds.

(b) Let S_k be a state. Suppose that $deviateHistory_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds. Then, by definition, for some states $S_0, \ldots, S_{i-1}$, $S'_{k+1}, \ldots, S'_n$, (15) holds, and $S'_{k+1} \neq S_{k+1}$. Then, by (b), (16) holds as well. Since (16) and $S'_{k+1} \neq S_{k+1}$, $deviateState_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds. $\square$

Proof of Lemma 2. Let P be a plan $\langle A_0, \ldots, A_{n-1}\rangle$, and let $\mathcal{T}$ be a set of trajectories such that, for every trajectory $S_0, A_0, \ldots, A_{n-1}, S_n$ in $\mathcal{T}$, *init*(S_0) holds. Let i and k ($0 \leq k < i \leq n$) be time stamps, and let S_i be a state.

(a) Suppose that $matchState^{P,\mathcal{T}}_{k,i}(S_i)$. Then, by definition, for some states $S_0, \ldots, S_{i-1}, S'_0, \ldots, S'_n$,

$$matchAt^{P,\mathcal{T}}_{k,i}(S_0, \ldots, S_{i-1}, S_i, S'_0, \ldots, S'_n) \tag{17}$$

holds. By definition, there exists an evolution

$$S_0, A_0, S_1, \ldots, S_{k-1}, A_{k-1}, S_k, A_k, S_{k+1}, \ldots, S_{i-1}, A_{i-1}, S_i$$

of S_i from an initial state (i.e., *init*(S_0)) and a goal-establishing trajectory

$$S'_0, A_0, S'_1, \ldots, S'_{k-1}, A_{k-1}, S'_k, A_k, S'_{k+1}, \ldots, S'_{n-1}, A_{n-1}, S'_n \tag{18}$$

in $\mathcal{T}$ (i.e., *goal*(S'_n)) where $S_k{=}S'_k$. Since, by assumption, *init*(S'_0),

$$S'_0, A_0, S'_1, \ldots, S'_{k-1}, A_{k-1}, S'_k, A_k, S_{k+1}, \ldots, S_{i-1}, A_{i-1}, S_i \tag{19}$$

is also an evolution of S_i from an initial state. Then, since the evolution (19) and the goal-establishing trajectory (18) in $\mathcal{T}$ match until time stamp k,

$$matchUntil^{P,\mathcal{T}}_{k,i}(S'_0, \ldots, S'_{k-1}, S'_k, S_{k+1}, \ldots, S_{i-1}, S_i, S'_{k+1}, \ldots, S'_n) \tag{20}$$

holds. Therefore, $matchHistory^{P,\mathcal{T}}_{k,i}(S_i)$ holds. Then, by Lemma 1(a),

$$matchState^{P,\mathcal{T}}_{k,i}(S_i) \equiv matchHistory^{P,\mathcal{T}}_{k,i}(S_i).$$

(b) Let S_k be a state. Suppose that $deviateState^{P,\mathcal{T}}_{k,i}(S_i, S_k)$ holds. Then, by definition, for some states $S_0, \ldots, S_{i-1}, S'_0, \ldots, S'_n$, (17) holds. By (a), (20) holds as well. Thus $deviateHistory^{P,\mathcal{T}}_{k,i}(S_i)$ holds. By Lemma 1(b),

$$deviateState^{P,\mathcal{T}}_{k,i}(S_i, S_k) \equiv deviateHistory^{P,\mathcal{T}}_{k,i}(S_i, S_k).$$

□

A.3 Proof of Proposition 3

Proposition 3 *For any discrepancy, if there is a history-oriented point of failure* (S_k, k) *then there is a state-oriented point of failure* $(S_{k'}, k')$ *such that* $k' \geq k$.

Proof. Suppose that there is a history-oriented point of failure (S_k, k) for a discrepancy between a state S_i and a set $\mathcal{T}$ of trajectories relative to a plan P at a time stamp i. That is, $diagnosisHistory^{P,\mathcal{T}}_{k,i}(S_i, S_k)$ holds. By definition, $deviateHistory^{P,\mathcal{T}}_{k,i}(S_i, S_k)$ and $\neg matchHistory^{P,\mathcal{T}}_{k+1,i}(S_i)$ hold. Then, by Lemma 1(b), $deviateState^{P,\mathcal{T}}_{k,i}(S_i, S_k)$ holds, and, by Lemma 1(a), $\neg matchState^{P,\mathcal{T}}_{k+1,i}(S_i)$ holds. Consider two cases. *Case 1.* If, for all j ($k < j \leq i$), $\neg matchState^{P,\mathcal{T}}_{j+1,i}(S_i)$, then $(S_k, k) = (S_{k'}, k')$ is the state-oriented point of failure for the discrepancy. *Case 2.* Otherwise, there is some j' ($k < j' < i$) and some state $S_{j'}$ such that $matchState^{P,\mathcal{T}}_{j',i}(S_i)$ and, due to the existence of a discrepancy at time stamp i, $\neg matchState^{P,\mathcal{T}}_{j'+1,i}(S_i)$ hold. Take the maximum of such j'. Then, $(S_{j'}, j') = (S_{k'}, k')$ is the state-oriented point of failure for the discrepancy. □

A.4 Proof of Proposition 4

Proposition 4 *Let P be a plan $\langle A_0, \ldots, A_{n-1}\rangle$ and let $\mathcal{T}$ be a set of trajectories such that for any state S_0, for which $init(S_0)$ holds, $\mathcal{T}$ contains a trajectory $S_0, A_0, \ldots, A_{n-1}, S_n$, such that $goal(S_n)$ holds. Then, for any discrepancy, a point of failure exists.*

Proof. Suppose that a discrepancy is detected between a state S_i and $\mathcal{T}$ relative to P at a time stamp i ($i \leq n$), i.e. $discrepancy_i^{P,\mathcal{T}}(S_i)$ holds.

First observe that $i > 0$ must hold. Otherwise an initial state S_0 would exist, such that $\forall \mathcal{F}'_0, \ldots, \mathcal{F}'_n traj^{\mathcal{T}}(\mathcal{F}'_0, A_0, \ldots, A_{n-1}, \mathcal{F}'_n) \wedge goal(\mathcal{F}'_n) \supset \neg(\mathcal{F}'_i \equiv S_i)$, i.e. no goal-establishing trajectory starting in S_0 would exist in $\mathcal{T}$, contrary to our assumption.

Consider history-oriented diagnosis. Since at least one evolution $S_0, \ldots, S_i$ exists for S_i (for which $init(S_0)$ necessarily holds), then by assumption also some trajectory $S_0, A_0, S'_1, \ldots, A_{n-1}, S'_n$ in $\mathcal{T}$ exists, and hence $matchUntil_{0,i}^{P,\mathcal{T}}(S_0, \ldots, S_i, S'_1, \ldots, S'_n)$ holds. Then, for some k ($0 \leq k < i$), $deviateHistory_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ must hold, as otherwise $matchUntil_{j,i}^{P,\mathcal{T}}(S_0, \ldots, S_i, S'_{j+1}, \ldots, S'_n)$ would hold for all j ($0 \leq j \leq i$), contradicting $discrepancy_i^{P,\mathcal{T}}(S_i)$.

Without loss of generality, assume that this k is maximal for the evolution $S_0, \ldots, S_i$ among all evolutions. Then, also $\neg matchHistory_{k+1,i}^{P,\mathcal{T}}(S_i)$ holds, and hence (S_k, k) is a history-oriented point of failure. By Proposition 3, also some state-oriented point of failure exists.

A.5 Proof of Proposition 5

Proposition 5 *If the action domain is deterministic then no diagnosis exists for a discrepancy.*

We will use the following lemma for the proof of Proposition 5.

Lemma 3 *If the action domain is deterministic, then, for any two trajectories, $S_0, A_0, \ldots, A_{n-1}, S_n$ and $S'_0, A_0, \ldots, A_{m-1}, S'_m$, ($0 \leq n \leq m$) and, for any time stamp i ($i \leq n$), the following holds. If $S_i = S'_i$, then, for all j ($i < j \leq n$), $S_j = S'_j$.*

Proof of Proposition 5. Suppose that the given action domain is deterministic. That is,

$$\forall \mathcal{F}, \mathcal{A}, \mathcal{F}', \mathcal{F}''(tr(\mathcal{F}, \mathcal{A}, \mathcal{F}') \wedge tr(\mathcal{F}, \mathcal{A}, \mathcal{F}'') \supset \mathcal{F}' \equiv \mathcal{F}''.$$

Suppose that there is a discrepancy between a state S_i and a set $\mathcal{T}$ of trajectories relative to a plan $P = \langle A_0, \ldots, A_{n-1}\rangle$ at a time stamp i. That is, for every goal-establishing trajectory

$$S'_0, A_0, \ldots, A_{n-1}, S'_n \tag{21}$$

in $\mathcal{T}$, $S_i \neq S'_i$. Since every action is deterministic, there is a unique evolution

$$S_0, A_0, \ldots, A_{i-1}, A_i$$

of state S_i from an initial state, if one exists. Then, due to determinism and due to Lemma 3, the evolution of the observed state S_i does not intersect with any goal-establishing trajectory (21) in $\mathcal{T}$

at any state before time stamp i, i.e., there is no trajectory (21) in $\mathcal{T}$ such that, for some time stamp k ($k < i$), $S_k = S'_k$. Indeed, otherwise, there would not be a discrepancy detected at time stamp i, since, for all j ($k \leq j \leq i$), S_j would be identical to S'_j. Then, by definition, there is no time stamp k ($k < i$) such that $matchState^{P,\mathcal{T}}_{k,i}(S_i)$ holds. By definition, no state-oriented diagnosis exists. Due to Proposition 3, no history-oriented diagnosis exists. □

Proof of Lemma 3. Assume that the action domain is deterministic. Assume also that, for any two trajectories, $S_0, A_0, \ldots, A_{n-1}, S_n$ and $S'_0, A_0, \ldots, A_{m-1}, S'_m$, ($0 \leq n \leq m$) and, for any time stamp i ($i \leq n$), $S_i = S'_i$. Then, we can show that, for all j ($i < j \leq n$), $S_j = S'_j$, by induction on j.

A.6 Proof of Theorem 1

Theorem 1 *Given a background $\mathcal{B}$, a reached state S_i, and a time stamp i ($i \leq n$), deciding whether a discrepancy exists between S_i and $\mathcal{T}$ relative to P at time stamp i is* coNP*-complete.*

Proof. For a background $\mathcal{B}$, a state S_i, and a time stamp i ($i \leq n$), we want to show that the problem of deciding whether $discrepancy^{P,\mathcal{T}}_i(S_i)$ holds is coNP-complete.

Membership. By definition, $discrepancy^{P,\mathcal{T}}_i(S_i)$ is false if there exist states $S'_0, S'_1, \ldots, S'_n$ such that (i) $S_i = S'_i$, and (ii) $traj^{\mathcal{T}}(S'_0, A_0, \ldots, A_{n-1}, S'_n) \wedge goal(S'_n)$ is true. Such states S'_i can be guessed in polynomial time, and (i) and (ii) can be verified in polynomial time. Hence, deciding that $discrepancy^{P,\mathcal{T}}_i(S_i)$ is false is in NP, which means that deciding whether $discrepancy^{P,\mathcal{T}}_i(S_i)$ holds is in coNP.

Hardness. Given a propositional formula $\phi(\mathcal{X})$ on atoms $\mathcal{X} = \{X^1, \ldots, X^m\}$, deciding whether $\phi(\mathcal{X})$ is unsatisfiable is a well-known coNP-complete problem. To show coNP-hardness, we need a background $\mathcal{B}$, a state S_i, and a time stamp i obtainable from $\phi(\mathcal{X})$ and $\mathcal{X}$ in polynomial time such that $discrepancy^{P,\mathcal{T}}_i(S_i)$ holds iff $\phi(\mathcal{X})$ is unsatisfiable.

Such a background $\mathcal{B}$, a state S_i, and a time stamp i can be obtained from $\phi(\mathcal{X})$ and $\mathcal{X}$ in polynomial time as follows. Let $\mathcal{F} = \{Y, X^1, \ldots, X^m\}$ and let $\mathcal{A} = \{D\}$. Take $state(\mathcal{F}) = act(\mathcal{F}, \mathcal{A}, \mathcal{F}') = init(\mathcal{F}) = goal(\mathcal{F}) = \top$, take $P = \langle\{D\}\rangle$, and take

$$traj^{\mathcal{T}}(\mathcal{F}_0, \mathcal{A}_0, \mathcal{F}_1) = Y_1 \supset \phi(\mathcal{X}_0).$$

Notice that in this domain, D is a dummy action which does not have any effects. The formula $traj^{\mathcal{T}}(\mathcal{F}_0, \mathcal{A}_0, \mathcal{F}_1)$ describes transitions according to which a state S_1 is reachable from a state S_0 iff Y is false in S_1 or the formula $\phi(\mathcal{X})$ evaluates to true in S_0.

Let $i = 1$ and $S_1 = \{Y\}$. Now we need to show that, with respect to $\mathcal{B}$ defined above, S_i, and i, $discrepancy^{P,\mathcal{T}}_1(S_1)$ holds iff $\phi(\mathcal{X})$ is unsatisfiable. For the if-direction, suppose that $\phi(\mathcal{X})$ is unsatisfiable. Then, by definition, for every trajectory $S'_0, \{D\}, S'_1$ in $\mathcal{T}$, it holds that Y is false in S'_1 and thus $S'_1 \neq S_1$. Hence, $discrepancy^{P,\mathcal{T}}_1(S_1)$ holds. For the only-if direction, suppose that $\phi(\mathcal{X})$ is satisfiable. Then some state S_0 exists such that $\phi(\mathcal{X})$ evaluates to true in it. Hence, $S_0, \{D\}, S_1$ is a trajectory in $\mathcal{T}$. Consequently, $\mathit{discrepancy}_1^{P,\mathcal{T}}(S_1)$ does not hold. Therefore, deciding whether $\mathit{discrepancy}_i^{P,\mathcal{T}}(S_i)$ holds is coNP-hard. $\Box$

A.7 Proof of Theorem 2

Theorem 2 *Given a background $\mathcal{B}$, a state S_k, a reached state S_i, and time stamps k and i ($k < i \leq n$), such that there is a discrepancy between S_i and $\mathcal{T}$ relative to P at time stamp i, deciding whether (S_k, k) is a point of failure for the discrepancy is* D^p*-complete.*

Proof. For a background $\mathcal{B}$, states S_k and S_i, and time stamps k and i ($k < i \leq n$), we want to show that

(a) the problem of deciding whether $\mathit{diagnosisState}_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds is D^p-complete, and

(b) the problem of deciding whether $\mathit{diagnosisHistory}_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds is D^p-complete.

Membership. (a) By definition, $\mathit{diagnosisState}_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds iff (S1) $\mathit{deviateState}_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds, and (S2) for all j ($k < j \leq i$), $\neg \mathit{matchState}_{j,i}^{P,\mathcal{T}}(S_i)$ holds. Deciding whether (S1) holds is in NP, since proper values for the existential variables in the definition (7) of $\mathit{deviateState}_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ can be guessed, and then the formula can be evaluated in polynomial time. Deciding whether (8) does *not* hold is in NP, since a proper j and values for the existential variables in the definition (6) of $\mathit{matchState}_{j,i}^{P,\mathcal{T}}(S_i)$ can be guessed, and then the formula can be evaluated in polynomial time. Hence, deciding that (S2) holds is in coNP. Since (S1) and (S2) can be decided independently of each other, it follows that deciding whether $\mathit{diagnosisState}_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds is in D^p.

(b) By definition, $\mathit{diagnosisHistory}_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds iff (H1) $\mathit{deviateHistory}_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds, and (H2) $\neg \mathit{matchHistory}_{k+1,i}^{P,\mathcal{T}}(S_i)$ holds. Similarly, deciding whether (H1) holds is in NP, and whether (H2) holds is in coNP. Hence, also deciding whether $\mathit{diagnosisHistory}_{k,i}^{P,\mathcal{T}}(S_i, S_k)$, holds is in D^p.

Hardness. Given propositional formulas $\phi(\mathcal{X})$ and $\psi(\mathcal{X})$ on atoms $X = \{X^1, \ldots, X^m\}$, deciding whether $\phi(\mathcal{X})$ is satisfiable and $\psi(\mathcal{X})$ is unsatisfiable is a D^p-complete problem. To show D^p-hardness, we need a background $\mathcal{B}$, states S_i and S_k, and time stamps i and k ($k < i \leq n$) obtainable from $\phi(\mathcal{X})$, $\psi(\mathcal{X})$, and $\mathcal{X}$ in polynomial time such that

(a) $\phi(\mathcal{X})$ is satisfiable and $\psi(\mathcal{X})$ is unsatisfiable iff $\mathit{diagnosisState}_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds, and

(b) $\phi(\mathcal{X})$ is satisfiable and $\psi(\mathcal{X})$ is unsatisfiable iff $\mathit{diagnosisHistory}_{k,i}^{P,\mathcal{T}}(S_i, S_k)$ holds.

We construct such a background $\mathcal{B}$, states S_i and S_k, and stages k and i, in polynomial time, as follows. Let $\mathcal{F} = \{X^1, \ldots, X^m, Y\}$, and let $\mathcal{A} = \{D\}$. Take *state*$(\mathcal{F}) =$ *init*$(\mathcal{F}) = \top$, and define the other formulas in $\mathcal{B}$ as follows:

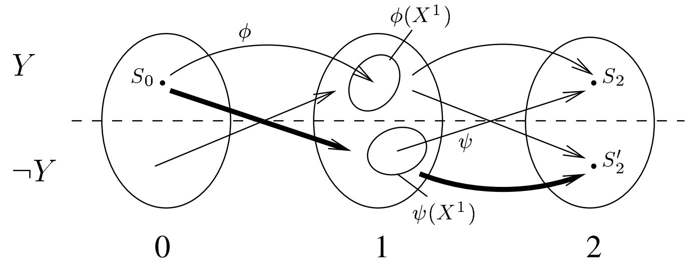


Figure 5: The evolutions of states reached at time step 2 by executing the plan $P = \langle D, \emptyset\rangle$ (shown by the paths from 0 to 2) and the trajectory in $\mathcal{T}$ (shown by the path from 0 to 2 with thick edges), as described in Proof of Theorem 2.

- $act(\mathcal{F}, \mathcal{A}, \mathcal{F}') \;=\; (\neg Y \wedge D \supset Y') \wedge (Y \wedge D \wedge Y' \supset \phi(\mathcal{X}')) \wedge (\neg Y \wedge \neg D \wedge Y' \supset \psi(\mathcal{X}))$
- $goal(\mathcal{F}) \;=\; \neg Y$
- $traj^{\mathcal{T}}(\mathcal{F}_0, \mathcal{A}_0, \mathcal{F}_1, \mathcal{A}_1, \mathcal{F}_2) \;=\; tr(\mathcal{F}_0, \mathcal{A}_0, \mathcal{F}_1) \wedge tr(\mathcal{F}_1, \mathcal{A}_1, \mathcal{F}_2) \wedge Y_0 \wedge \neg Y_1 \wedge \neg Y_2.$

Take $P = \langle D, \emptyset\rangle$, $i = 2$, $k = 0$, $S_2 = \{Y\}$, and $S_0 = \{Y\}$.

The possible evolutions of a state reached at time step 2, with respect to P, are represented by paths from 0 to 2 in Figure 5. The path with thick edges shows the goal-establishing trajectory in $\mathcal{T}$. Informally, from an initial state in which Y is false, all states in which Y is true and only those can be reached by executing D; from an initial state in which Y is true, all states can be reached except those in which Y is true and $\phi(\mathcal{X})$ does not hold. From a state S_1 at stage 1, all states can be reached by doing nothing, except those in which Y is true, if Y is false in S_1 and $\psi(\mathcal{X})$ does not hold in S_1. All trajectories in $\mathcal{T}$ are goal-establishing; they start in those states in which Y is true, and have Y false at stage 1.

(a) We claim that $\phi(\mathcal{X})$ is satisfiable and $\psi(\mathcal{X})$ is unsatisfiable iff $diagnosisState^{P,\mathcal{T}}_{0,2}(S_2, S_0)$ holds, i.e., (S1) $deviateState^{P,\mathcal{T}}_{0,2}(S_2, S_0)$ holds and (S2) $\neg matchState^{P,\mathcal{T}}_{1,2}(S_2) \wedge \neg matchState^{P,\mathcal{T}}_{2,2}(S_2)$ holds.

For the only-if direction, suppose that $\phi(\mathcal{X})$ is satisfiable and $\psi(\mathcal{X})$ is unsatisfiable. Then there is an evolution

$$S_0, \{D\}, S_1, \emptyset, S_2 \tag{22}$$

of S_2, according to P, such that Y is true at S_1. The evolution (22) of S_2 matches every goal-establishing trajectory

$$S_0, \{D\}, S_1', \emptyset, S_2' \tag{23}$$

in $\mathcal{T}$ at time stamp 0, i.e., $matchAt^{P,\mathcal{T}}_{0,2}(S_0, S_1, S_2, S_0, S_1', S_2')$ is true. Since Y is not true at state S_1', $S_1 \neq S_1'$, (S1) holds. Furthermore, since $\psi(\mathcal{X})$ is unsatisfiable, in every evolution (22) of S_2 according to P, Y is true S_1. Since in each trajectory (23) in $\mathcal{T}$, Y is false in S_1', $matchAt^{P,\mathcal{T}}_{1,2}(S_0, S_1, S_2, S_0, S_1', S_2')$ is false, hence, (S2) holds. Therefore, $diagnosisState^{P,\mathcal{T}}_{0,2}(S_2, S_0)$ holds.

For the if-direction, suppose that $diagnosisState_{0,2}^{P,\mathcal{T}}(S_2, S_0)$ holds, i.e., (S1) and (S2) hold, but either $\phi(\mathcal{X})$ is unsatisfiable or $\psi(\mathcal{X})$ is satisfiable. Since (S1) holds, there exists an evolution (22) of S_2, according to P, such that $S_1 \models Y \wedge \phi(\mathcal{X})$. Consequently, by assumption, $\psi(\mathcal{X})$ must be satisfiable. This implies that there exists an evolution (22) of S_2, according to P, such that Y is false in S_1. Since the goal-establishing trajectory

$$S_0, \{D\}, S_1, \emptyset, S_2'$$

is in $\mathcal{T}$, the sentence $matchAt_{1,2}^{P,\mathcal{T}}(S_2)$ is true, and hence (S2) fails, which contradicts our assumption. This proves the claim, and hence D^p-hardness of recognizing a state-oriented diagnosis.

(b) According to the background $\mathcal{B}$ constructed above, the trajectories in $\mathcal{T}$ are of the form (22) such that *init*(S_0) holds. Then, by Proposition 2, state-oriented diagnosis and history-oriented diagnosis coincide. This implies D^p-hardness of recognizing a history-oriented diagnosis. □

A.8 Proof of Theorem 3

Theorem 3 *Given a background $\mathcal{B}$, a reached state S_i, and a time stamp i ($i \leq n$), such that there is a discrepancy between S_i and $\mathcal{T}$ relative to P at time stamp i, computing some point of failure (S_k, k) for the discrepancy is* $\mathrm{FNP}//\mathrm{OptP}[O(\log n)]$*-complete.*

Proof. For a background $\mathcal{B}$, a state S_i, and a time stamp i ($i \leq n$), such that there is a discrepancy between S_i and $\mathcal{T}$ relative to P at time stamp i, we want to show that

(a) computing a state-oriented point of failure (S_k, k) for the discrepancy is $\mathrm{FNP}//\mathrm{OptP}[O(\log n)]$-complete, and

(b) computing a history-oriented point of failure (S_k, k) for the discrepancy is $\mathrm{FNP}//\mathrm{OptP}[O(\log n)]$-complete.

Membership. (a) The computation of k in a state-oriented diagnosis (S_k, k) can be viewed as the optimization problem of finding the largest stage k such that the sentence

$$\exists \mathcal{F}_k\, deviateState_{k,i}^{P,\mathcal{T}}(S_i, \mathcal{F}_k) \tag{24}$$

holds. Given a stage k, deciding whether (24) holds is in NP. Here, k has $O(\log |I|)$ many bits, where I is the problem input $(\mathcal{B}, S_i, i)$. When k is known, the state S_k in a diagnosis (S_k, k) can be obtained in polynomial time by nondeterministically generating proper values $S_0, \ldots, S_{i-1}$, $S_0', \ldots, S_n'$ for the existentially quantified variables in (24) so that the formula evaluates to true. Hence, computing a state-oriented diagnosis is in $\mathrm{FNP}//\log$.

(b) Membership of computing a history-oriented diagnosis in $\mathrm{FNP}//\log$ is shown similarly.

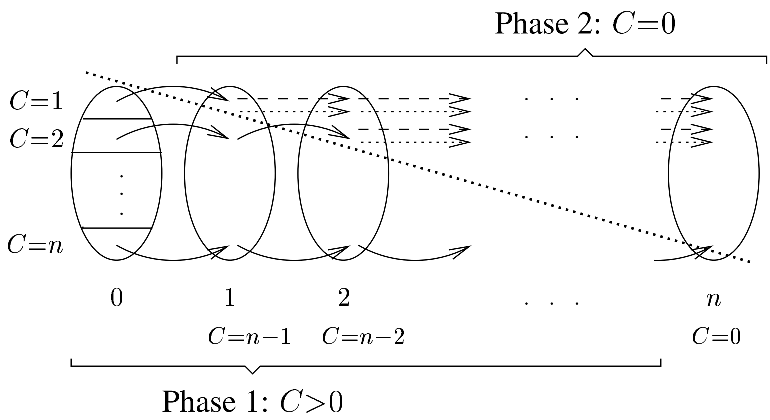


Figure 6: The evolutions of states reached at time stamp n by executing the plan $P = \langle\emptyset, \ldots, \emptyset\rangle$ (shown by paths from 0 to n), where the dashed and the dotted edges symbolize the commitment of B to false and true respectively, as described in Proof of Theorem 3.

Hardness. We show the FNP//log-hardness by a polynomial-time reduction from $\mathcal{X}$-MAXIMAL MODEL: Given a Boolean formula $\phi(\mathcal{Y})$ on atoms $\mathcal{Y} = \{Y^1, \ldots, Y^m\}$ and a subset $\mathcal{X} \subseteq \{Y^1, \ldots, Y^m\}$, compute the $\mathcal{X}$-part of a model M of $\phi(\mathcal{Y})$ such that $M \cap \mathcal{X}$ is maximal, i.e., no model M' of $\phi(\mathcal{Y})$ exists such that $M' \cap \mathcal{X} \supset M \cap \mathcal{X}$, where a model M is identified with the set of atoms that are mapped to true. Completeness of this problem for FNP//log is shown in [4].

We will reduce $\mathcal{X}$-MAXIMAL MODEL to computing some diagnosis in polynomial time in two parts, according to [4]. In Part 1, we will show that, for any instance $\phi(\mathcal{Y})$ of $\mathcal{X}$-MAXIMAL MODEL, an instance $f(\phi(\mathcal{Y})) = (\mathcal{B}, S_i, i)$ of our problem input is constructible in polynomial time, such that $f(\phi(\mathcal{Y}))$ has some diagnosis. In Part 2, we will show that, from every diagnosis (S_k, k) for $f(\phi(\mathcal{Y}))$ and $\phi(\mathcal{Y})$, some $\mathcal{X}$-maximal model M of $\phi(\mathcal{Y})$ can be constructed in time polynomial in the size of (S_k, k) and $\phi(\mathcal{Y})$ (provided $\phi(\mathcal{Y})$ is satisfiable). Without loss of generality, we assume that $\phi(\mathcal{Y})$ is satisfiable, and that in each model of $\phi(\mathcal{Y})$ some atom in $\mathcal{X}$ is true.

(b) We prove that computing some history-oriented diagnosis for a discrepancy is FNP//log hard in two parts as follows.

Part 1. Informally, we construct $(\mathcal{B}, S_i, i)$ as follows. Suppose that every evolution of a state S_n reached at time stamp n by executing the plan P starts at an initial state that corresponds to a model M of $\phi(\mathcal{Y})$. Consider two phases of such an evolution. In Phase 1, a counter C, which is set at the initial state to the number of atoms from $\mathcal{X}$ which are true in M, is decreased step by step until it reaches 0, and the model M is propagated by inertia. When C gets 0, Phase 2 begins. At the beginning of Phase 2, a special fluent B is committed to be either true or false. Throughout Phase 2, i.e., till the end of the evolution, C is stuck at 0, and the value of B is propagated by inertia. The observation is made at stage n (i.e., $i = n$). In the observed state S_i, B is false, while in the trajectories in $\mathcal{T}$, B is true at all stages.

With the construction above, for a history-oriented point of failure (S_k, k) between S_i and $\mathcal{T}$

relative to P at stage i, we consider an evolution of which Phase 1 is as long as possible, i.e., that C is as large as possible, and of which Phase 2 has B mapped to true. That is, a model M corresponding to an initial state has a maximum number C of atoms of $\mathcal{X}$ set to true. Then, from (S_k, k), such a model M can be easily extracted.

More formally, let $n = |\mathcal{X}| > 0$ and let

$$\mathcal{F} = \{Y^1, \ldots, Y^m\} \cup \{C^{i,j} : i \in \{0, \ldots, n\}, j \in \{0, \ldots, i\}\} \cup \{B\}.$$

Intuitively, $C^{i,j}$ expresses in the initial state that exactly j atoms among $X_1, \ldots, X_i$ of $\mathcal{X}$ are true, such that $C^{n,j}$ is true iff the initial value of the counter C is j. Let $\mathcal{A}$ be the set consisting of the dummy action symbol D.

We define the background $\mathcal{B}$ as follows:

- *state*$(\mathcal{F}) = \top$.
- *init*$(\mathcal{F}) = \phi(\mathcal{Y}) \wedge set\_counter\_C$, where

$$\begin{array}{rcl} set\_counter\_C & = & C^{0,0} \wedge \bigwedge_{i=1}^{n} (C^{i,0} \equiv C^{i-1,0} \wedge \neg X_i) \wedge \\ & & \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{i} C^{i,j} \equiv ((C^{i-1,j-1} \wedge X^i) \vee (C^{i-1,j} \wedge \neg X^i)). \end{array}$$

  Here formula $set\_counter\_C$ defines inductively the value of $C^{i,j}$ and thus C.

- *act*$(\mathcal{F}, \mathcal{A}, \mathcal{F}')$ = $dec\_C \wedge inertia$, where

$$\begin{array}{rcl} dec\_C & = & \neg C^{n,0} \supset (\neg C^{n,n'} \wedge \bigwedge_{j=0}^{n-1} (C^{n,j'} \equiv C^{n,j+1})) \\ inertia & = & \Big(\neg C^{n,0} \supset \bigwedge_{i=1}^{m} (Y^{i'} \equiv Y^i)\Big) \wedge \Big(C^{n,0} \supset (C^{n,0'} \wedge (B' \equiv B))\Big). \end{array}$$

  Formula dec_C decrements counter C in Phase 1 (when $C^{n,0}$ is false, i.e., $C > 0$) by shifting the values of all $C^{n,j}$. Formula $inertia$ carries on the model M in Phase 1, and propagates the value of $C^{n,0}$ and B in Phase 2 (when $C = 0$).

- *goal*$(\mathcal{F}) = B$.
- *traj*$^{\mathcal{T}}(\mathcal{F}_0, \mathcal{A}_0, \ldots, \mathcal{F}_n) = \bigwedge_{t=0}^{n-1}$ *tr*$(\mathcal{F}_t, A_t, \mathcal{F}_{t+1}) \wedge$ *init*$(\mathcal{F}_0) \wedge \bigwedge_{t=0}^{n} B_t$.

  That is, $\mathcal{T}$ contains all trajectories from initial states in which B is always true.

- P=$\langle\emptyset, \ldots, \emptyset\rangle$.

We set $S_i{=}S_n = \{C^{n,0}\}$ and $i{=}n$.

With the construction of $\mathcal{B}$, S_i and i above, there is a discrepancy between S_n and $\mathcal{T}$ relative to P at time stamp n, because B is false at S_n, and B is true at every state of every trajectory in $\mathcal{T}$.

Now we want to show that there is a point of failure for this discrepancy. First, note that there is an evolution

$$S_0, \emptyset, S_1, \emptyset, \ldots, \emptyset, S_n \tag{25}$$

of $S_n = \{C^{n,0}\}$ according to P. Indeed, from any model M of $\phi(\mathcal{Y})$, we can obtain a corresponding state S_0 such that $init(S_0)$ holds, by assigning B value true and all fluents $C^{i,j}$ the intended values; in particular, $C^{n,j}$ is true (i.e., counter $C = j$) iff $j = |M \cap \mathcal{X}|$. According to dec_C and $inertia$, the value of C is decreased in S_1, S_2, etc. while the value of Y^i stays the same as in S^1 until $C = 0$ (i.e., $C^{n,0}$ is true) at some state S_l where $l = |M \cap \mathcal{X}|$ and $l > 0$; the values of $C^{i,j}$, $i < n$, are set to false in these states $S_1, \ldots, S_l$; B is set true in $S_1, \ldots, S_{l-1}$ and false in S_l. In the states $S_{l+1}, \ldots, S_n$, $C^{n,0}$ is true while all other fluents (including B) are set to false.

Next, note that the trajectory

$$S'_0, \emptyset, S'_1, \emptyset, \ldots, \emptyset, S'_n, \tag{26}$$

where $S'_i{=}S_i$ for $i \in \{0, \ldots, l{-}1\}$ and $S'_i = S_i \cup \{B_i\}$ for $i \in \{l, \ldots, n\}$ (i.e., B is made true in all states S_i where it is false) is a goal-establishing trajectory in $\mathcal{T}$.

Then, due to the evolution (25) of S_n, and the presence of the goal-establishing trajectory (26) in $\mathcal{T}$, it is easy to see that the sentence $matchUntil^{P,\mathcal{T}}_{l-1,n}(S_0, \ldots, S_n,\ S'_l, \ldots, S'_l)$ is true. Thus $deviateHistory^{P,\mathcal{T}}_{l-1,n}(S_{l-1}, S_n)$ holds. This implies that there is some history-oriented diagnosis (S_k, k) for the discrepancy detected at time stamp n such that $k \geq l$.

Part 2. We claim that, for any history-oriented diagnosis (S_k, k) for S_n, the model

$$M = \{Y^j \mid Y^j \text{ is true in } S_k, 1 \leq j \leq m\},$$

has a maximal $\mathcal{X}$-part among the models of $\phi(\mathcal{Y})$.

To prove this claim, note that, since the evolution (25) and the goal-establishing trajectories (26) are constructed for an arbitrary model M of $\phi(\mathcal{Y})$, $k \geq l^*$ must hold, where

$$l^* = \max_{M \models \phi(\mathcal{Y})} |M \cap \mathcal{X}| - 1.$$

We first show that (i) $C^{n,0}$ is false in S_k and (ii) $k \leq l^*$, which means $k = l^*$.

The sentence $deviateHistory^{P,\mathcal{T}}_{k,i}(S_i, S_k)$ holds witnessed by some evolution (25) of S_n according to P and a trajectory (26) in $\mathcal{T}$, such that $S_j = S'_j$, for $j \in \{0, \ldots, k\}$. Suppose that $C^{n,0}$ is true in S_k. Since B must be true in S_k, it follows from $inertia$ that B is also true in S_n, which is a contradiction. Therefore, (i) holds. For (ii), suppose that $k > l^*$. Then, $S_{l^*} = S'_{l^*}$ holds. Since $init(S_0)$ holds, it follows from dec_C and $inertia$ that $C^{n,0}$ is true in S_k, which contradicts (i). Thus, $k = l^*$ holds.

Since $C^{n,0}$ is false in S_k and, by the assumption about $\phi(\mathcal{Y})$, false in S_0, it follows from dec_C and $inertia$ that Y^j has in S_k the same value as it has in S_0, for $j \in \{1, \ldots, m\}$. Hence M is a model of $\phi(\mathcal{Y})$ such that $|M \cap \mathcal{X}| - 1 = l^*$; equivalently,

$$|M \cap \mathcal{X}| = \max_{M \models \phi(\mathcal{Y})} |M \cap \mathcal{X}|.$$

Clearly, such an M has a maximal $\mathcal{X}$-part among the models of $\phi(\mathcal{Y})$. Since M can be easily extracted from (S_k, k), this completes Part 2 of the reduction.

We thus have proven $\mathrm{FNP}//\log$ hardness of computing some history-oriented diagnosis (S_k, k).

(a) Since in the reduction above, $init(S_0)$ holds for all initial states S_0 of trajectories in $\mathcal{T}$, Proposition 2 implies $\mathrm{FNP}//\log$ hardness of computing some state-oriented diagnosis (S_k, k). $\Box$

References

[1] M. Balduccini and M. Gelfond. Diagnostic reasoning with a-prolog. *Journal of Theory and Practice of Logic Programming*, 3(4–5):425–461, 2003.

[2] C. Baral, V. Kreinovich, and R. Trejo. Computational Complexity of Planning and Approximate Planning in the Presence of Incompleteness. *Artificial Intelligence*, 122(1-2):241–267, 2000.

[3] C. Baral, S. A. McIlraith, and T. C. Son. Formulating diagnostic problem solving using an action language with narratives and sensing. In *Principles of Knowledge Representation and Reasoning*, pages 311–322, 2000.

[4] Z.-Z. Chen and S. Toda. The Complexity of Selecting Maximal Solutions. *Information and Computation*, 119:231–239, 1995.

[5] T. Eiter, E. Erdem, and W. Faber. Plan Reversals for Recovery in Execution Monitoring. In J. P. Delgrande and T. Schaub, editors, *Proceedings 10th International Workshop on Nonmonotonic Reasoning (NMR-2004), Action and Causality Track*, pages 147–154, June 2004. ISBN 92-990021-0-X.

[6] T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. A logic programming approach to knowledge-state planning, II: The DLV$^{\mathcal{K}}$ system. *Artificial Intelligence*, 144(1–2):157–211, 2002.

[7] T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. A logic programming approach to knowledge-state planning: Semantics and complexity. *ACM TOCL*, 5(2):206–263, 2004.

[8] P. Ferraris and E. Giunchiglia. Planning as Satisfiability in Nondeterministic Domains. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI'00), July 30 – August 3, 2000, Austin, Texas USA*, pages 748–753. AAAI Press / The MIT Press, 2000.

[9] M. Fichtner, A. Großmann, and M. Thielscher. Intelligent execution monitoring in dynamic environments. *Fundamenta Informaticae*, 57(2–4):371–392, 2003.

[10] M. Gelfond and V. Lifschitz. Action languages. *Electronic Transactions on AI*, 3:195–210, 1998.

[11] G. D. Giacomo, R. Reiter, and M. Soutchanski. Execution monitoring of high-level robot programs. In *Principles of Knowledge Representation and Reasoning*, pages 453–465, 1998.

[12] E. Giunchiglia. Planning as Satisfiability with Expressive Action Languages: Concurrency, Constraints and Nondeterminism. In A. G. Cohn, F. Giunchiglia, and B. Selman, editors, *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2000), April 12-15, Breckenridge, Colorado, USA*, pages 657–666. Morgan Kaufmann, 2000.

[13] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, and H. Turner. Nonmonotonic Causal Theories. *Artificial Intelligence*, 153(1-2):49–104, 2004.

[14] H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. B. Scherl. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming*, 31(1-3):59–83, 1997.

[15] N. McCain and H. Turner. Satisfiability Planning with Causal Theories. In A. G. Cohn, L. Schubert, and S. C. Shapiro, editors, *Proceedings Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 212–223. Morgan Kaufmann Publishers, 1998.

[16] J. McCarthy. Elaboration tolerance. In progress, 1999.

[17] C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[18] R. Reiter. *Knowledge in action: Logical Foundations for specifying and implementing dynamical systems*. MIT Press, 2001.

[19] J. Rintanen. Constructing Conditional Plans by a Theorem-Prover. *Journal of Artificial Intelligence Research*, 10:323–352, 1999.

[20] T. C. Son and E. Pontelli. Planning with preferences using logic programming. In I. Niemelä and V. Lifschitz, editors, *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2004)*, number 2923 in LNCS, pages 247–260. Springer, 2004.

[21] M. Soutchanski. Execution monitoring of high-level temporal programs. In *Proc. of IJCAI Workshop on Robot Action Planning*, 1999.

[22] M. Soutchanski. High-level robot programming and program execution. In *Proc. of ICAPS Workshop on Plan Execution*, 2003.

[23] M. Thielscher. The concurrent, continuous Fluent Calculus. *Studia Logica*, 67(3):315–331, 2001.

[24] M. Thielscher. FLUX: A logic programming method for reasoning agents. *Theory and Practice of Logic Programming*, -(-), 2004. To appear.

[25] H. Turner. Polynomial-length planning spans the polynomial hierarchy. In *Proc. of Eighth European Conf. on Logics in Artificial Intelligence (JELIA'02)*, pages 111–124, 2002.