Diagnosis as Planning Revisited

Shirin Sohrabi
Department of Computer Science
University of Toronto
Toronto, Canada

Jorge A. Baier
Depto. de Ciencia de la Computación
Pontificia Universidad Católica de Chile
Santiago, Chile

Sheila A. McIlraith
Department of Computer Science
University of Toronto
Toronto, Canada

Abstract

In discrete dynamical systems change results from actions. As such, given a set of observations, diagnoses often take the form of posited events that result in the observed behaviour. In this paper we revisit formal characterizations of diagnosis, and their relationship to planning. We do so from both a theoretical and a computational perspective. In particular, we extend the characterization of diagnosis to deal with the case of incomplete information, and rich preferences. We also explore the use of state-of-the-art planning technology for the automated generation of diagnoses. Examining several classes of diagnosis problems, we provide both proof of concept and benchmark experiments, the latter showing superior performance to a leading diagnosis engine. Our findings help support the hypothesis that planning technology holds great promise for efficient generation of diagnoses.

Introduction

System diagnosis is critical to the operation of many engineered systems. As the complexity of such systems increases, diagnosis often eludes human reasoning requiring the automation of diagnosis. A number of today's engineered systems are dynamical and are controlled by discrete event controllers. It is the problem of diagnosing discrete dynamical systems that is the subject of this work.

Given a theory of system behaviour and a set of observations, diagnosis is often cast as the task of identifying a (minimal) set of components whose malfunctioning is consistent with the observations (e.g., Reiter 1987; de Kleer et al. 1992). In dynamical settings where actions cause change, diagnosis can be cast in terms of the occurrence of actions and events. Hence, in order to explain *what happened* to the system it is often enough to find a sequence of actions or events that have occurred and that can account for the observed behaviour. While several researchers observed the relationship between actions and diagnoses (e.g., Cordier and Thiébaux 1994; McIlraith 1994), Sampath et al. (1995) were the first to present comprehensive results diagnosing discrete event systems by modeling them as finite state automata and characterizing diagnosis as a reachability analysis problem. Thielscher (1997), McIlraith (1998), and subsequently Iwan (2001) and Baral et al. (2000) cast the diagnosis problems in terms of an Artificial Intelligence (AI) theory of action and change. McIlraith and Iwan proposed generating diagnoses using deductive plan synthesis, whereas Baral et al. proposed the use of logic programming and later answer set programming (ASP). Research on diagnosis of discrete dynamical systems continued with the development of algorithms for generating trajectories that account for observations (e.g., Lamperti and Zanella 2003; Pencolé and Cordier 2005). However, generating such diagnoses was computationally expensive.

The problem of diagnosing discrete dynamical systems has been the topic of renewed interest with the work of Grastien, Rintanen and colleagues (e.g., Grastien et al. 2007a, Rintanen and Grastien 2007) who also observed that diagnosis is reducible to a path finding problem. However, more in keeping with Reiter's original characterization of diagnosis of static systems and to previous ASP approaches, they characterized diagnosis as a satisfiability problem (SAT), embedding the path finding aspect by appealing to an encoding of their dynamical domain in terms of Kautz and Selman's encoding (1999) for SAT-based planning. Building on their characterization, they implemented an approach to generating diagnoses using SAT solvers. Their experiments showed impressive performance relative to the state of the art.

In this paper, we revisit the characterization of diagnosis of discrete dynamical systems, and its relationship to planning from both a theoretical and a computational perspective. Our contributions are 3-fold: 1) We provide a formal characterization of diagnosis of discrete dynamical systems in terms of a theory of action and change. This characterization generalizes previous work, dealing both with incomplete information and with the important issue of incorporating commonsense and expert knowledge. 2) We establish a formal correspondence between our characterization and AI planning. 3) As a result of (2) we show how the generation of diagnoses can be realized by state-of-the-art planning techniques. Examining several classes of diagnosis problems, we provide both proof of concept and benchmark experiments, the latter showing superior performance to a leading dynamical diagnosis engine, based on SAT. Our findings help support the hypothesis that planning technology holds great promise for efficient generation of diagnoses.

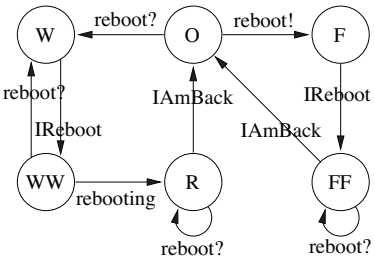

Figure 1: The automaton representation of one component

Preliminaries

A Running Example

Throughout this paper we consider a computer system diagnosis domain introduced and described more fully by Grastien et al. (2007a). This domain is sufficiently complex to be challenging to traditional approaches to dynamical diagnosis and serves as a source of comparison between the performance of our work and that of Grastien et al. The domain describes a system consisting of 20 identical components/computers in a 5×4 grid. Each component has four neighbours; corner and border components are neighbours to components on their opposite sides.

Figure 1 provides an automaton representation of one component. Each component starts initially in the state O. When a component fails, its state changes from O to F and sends a message reboot! to its neighbours. The neighbours receive the message reboot? and change their state from O to W, WW to W, R to R or FF to FF depending on their current state. The events IReboot and IAmBack correspond to a component sending an alarm that can be observed. Given these observations the objective is to monitor this system and determine what happened to the system.

Situation Calculus

We use the situation calculus and first-order logic to axiomatize our diagnosis domains. The situation calculus is a sorted logical language for specifying and reasoning about dynamical systems (Reiter 2001). The sorts of the language are *situation*, *action*, *fluent* and a catch-all *object* sort. Situations are sequences of actions that represent a history of the evolution of the world from an initial situation, which is denoted by S_0. The distinguished function $do(a, s)$ maps a situation and an action into a new situation, thus inducing a tree of situations rooted in S_0. Fluents are predicate or function symbols used to describe the properties that hold true in a particular situation. Thus, a fluent is a predicate or function with a situation argument, e.g., $F(x, s)$. Finally the atomic expression $Poss(a, s)$ is true if action a is possible in situation s.

Dynamical systems are modelled using a *basic action theory* (BAT). Reiter's BAT has the form

$$\Sigma = \Sigma_F \cup \Sigma_{ss} \cup \Sigma_{ap} \cup \Sigma_{una} \cup \Sigma_{S_0}.$$

Below we elaborate on the form of each of these sets, as we use them here. Note that we sometimes stipulate that a formula be *uniform in* s. This designates that the only situation term in the formula is s. Following standard notational convention for the situations calculus, free variables in formulae are assumed to be universally quantified from the outside unless otherwise noted.

- Σ_F is a set of foundational axioms. Foundational axioms axiomatize situations and the situation predecessor relation, $\sqsubset$. A situation s' precedes a situation s, i.e., $s' \sqsubset s$, if and only if s' is a proper prefix of s.
- Σ_{ss} is a set of successor state axioms (SSAs), one for each fluent F, of the form:

  $$F(x_1, \ldots, x_n, do(a, s)) \equiv \Phi_F(a, x_1, \ldots, x_n, s),$$

  where $\Phi_F(a, x_1, \ldots, x_n, s)$ is a formula uniform in s and whose free variables are among a, $x_1, \ldots, x_n$, and s.
- Σ_{ap} is a set of action precondition axioms. For each action function symbol A of the language, there is a precondition axiom of the form:

  $$Poss(A(x_1, \ldots, x_n), s) \equiv \Pi_A(x_1, \ldots, x_n, s),$$

  where $\Pi_A(x_1, \ldots, x_n, s)$ is a formula uniform in s and whose free variables are among $x_1, \ldots, x_n$ and s. $\Pi_A(x_1, \ldots, x_n, s)$ expresses all the conditions under which a can be performed in s.
- Σ_{una} is a set of unique names axioms for actions.
- Σ_{S_0} is a set of axioms describing the initial state of the world. As such formulae are uniform in S_0.

We henceforth refer to actions and events synonymously as actions. We use the notation $\mathbf{a}$ to abbreviate $[a_1, ..., a_n]$, and $do(\mathbf{a}, s)$ to abbreviate $do(a_n, do(a_{n-1}, ..., do(a_1, s)))$. We also use notation $s \sqsubseteq s'$ to abbreviate $s \sqsubset s' \vee s = s'$, and notation $s \sqsubset s' \sqsubset s''$ to abbreviate $s \sqsubset s' \wedge s' \sqsubset s''$. A situation s is *executable* if all actions in s have their preconditions satisfied in the situation where they are performed:

$$executable(s) \stackrel{\text{def}}{=} (\forall a, s').do(a, s') \sqsubseteq s \supset Poss(a, s').$$

Dynamical Diagnosis

In this section, we characterize dynamical diagnosis in the situation calculus. Our characterization is a generalization of those proposed by McIlraith (1998) and Iwan (2001). As both of those characterizations do, we encode the behaviour of the system to be diagnosed as a situation calculus BAT, encode observations as situation calculus formulae, and conjecture a sequence of actions to explain the observations, i.e., what went wrong with the system. Our characterization is differentiated in that our notion of diagnosis includes an assumption regarding the initial state, in the spirit of *assumption-based* diagnosis (McIlraith 1998, Def. 7). Further, in our definition, observations can be expressed as a conjunction of formulae relativized to multiple situations within the trajectory of the diagnosis rather than a formula that must hold in the final state. This generalizes both McIlraith's and Iwan's account of observations, and supports both the observation of the state of the system at various stages and the occurrence of events or actions. Baral et al. (2000) considered a more limited form of such

temporally extended observations, representing them as narrative to be explained. Finally, note that we do not exploit an explicit representation of time, which we see as a feature, though time can be represented in the situation calculus.

In addition to the definition of diagnosis, we formally define two classes of criteria that can be used to differentiate diagnoses. The first class of criteria is domain-independent; criteria are based on notions of *minimality*, in keeping with previous work in diagnosis. The second class of criteria supports the definition of domain-specific commonsensical *preferences* over diagnoses.

Characterizing Diagnoses

We begin with a definition of the system to be diagnosed.

Definition 1 (System)
A system DS is a tuple $(\Sigma, OBS[S_0, s])$ such that:

- *Σ is the system description, encoded as a consistent BAT.*
- *The set of action terms $\mathcal{A}$ in the language of Σ is the union of two disjoint sets $\mathcal{A}_{normal}$ and $\mathcal{A}_{faulty}$. These sets distinguish normal and faulty actions of the system.*
- *$OBS[S_0, s]$, is a sentence of the situation calculus whose only free variable is s and whose situation terms are restricted to s_i such that $S_0 \sqsubseteq s_i \sqsubseteq s$.*

We use the square bracket suffix $[S_0, s]$ in $OBS[S_0, s]$, in place of a single situation term, to denote a formula of restricted syntactic form. In particular OBS refers only to situations along the branch of the situation tree that starts in S_0 and ends in s. More precisely, the formula only mentions situation terms s_i such that $S_0 \sqsubseteq s_i \sqsubseteq s$.

Note that Definition 1 distinguishes a subset of the actions in our system description as faulty. This is exploited in the definition of minimal diagnosis to reflect a preference for diagnoses that minimize the occurrence of faulty actions. Such a distinction is often compelling for engineered hardware systems, but may not be compelling in general. As such, nothing in our general definition of diagnosis requires such a distinction – $\mathcal{A}_{faulty}$ may be the empty set. More generally, we make few assumptions here about the axiomatization of the behaviour of the system. Designation of a subset of components of interest ($COMPS$) and a subset of AB-fluents ($AB(c)$) is common practice in diagnosis (e.g., Reiter 1987) and is supported by our formalism, leading to other preference criteria over diagnoses.

Also observe that the sentence $OBS[S_0, s]$ describes observations on the state of the system and/or the occurrence of actions in the history of situation s. It has the property that it does not refer to situations other than those that occur in the branch associated with s. The following example illustrates a possible observation formula. In the first example, observations are totally ordered and include both an action occurrence and observations about the state of the system. The domain is a car diagnosis domain. Initially the car is at home and the gas tank full. The car is driven to work, it subsequently would not start and the radio is not working.

$$\exists s_1, s_2, s_3.\ athome(S_0) \wedge fulltank(S_0) \wedge s_2 = do(drive2work, s_1) \wedge \neg carstart(s_3) \wedge \neg radioworks(s) \wedge S_0 \sqsubset s_1 \sqsubset s_2 \wedge s_2 \sqsubset s_3 \sqsubset s.$$

Partially ordered observations can be modelled in a similar way by only specifying the required ordering constraints.

Returning to our example computer domain, recall that events IReboot and IAmBack send observable alarms. Here we encode these alarms as fluents $obsReboot$ and $obsBack$ respectively. The following is an observation formula that specifies a set of totally ordered observations. Note that c_{ij} refers to the component in grid location (i, j).

$$\exists s_1, s_2, s_3, s_4, s_5, s_6.\ obsReboot(c_{11}, s_1) \wedge obsReboot(c_{00}, s_2) \wedge obsReboot(c_{20}, s_3) \wedge obsReboot(c_{11}, s_4) \wedge obsReboot(c_{13}, s_5) \wedge obsBack(c_{11}, s_6) \wedge S_0 \sqsubset s_1 \sqsubset s_2 \wedge s_2 \sqsubset s_3 \sqsubset s_4 \wedge s_4 \sqsubset s_5 \sqsubset s_6 \wedge s_6 \sqsubset s.$$

The following is an observation formula that specifies a set of partially ordered observations for the computer domain.

$$\exists s_1, s_2, s_3, s_4, s_5, s_6.\ obsReboot(c_{22}, s1) \wedge obsReboot(c_{12}, s2) \wedge obsReboot(c_{32}, s3) \wedge obsReboot(c_{23}, s4) \wedge obsReboot(c_{21}, s5) \wedge obsBack(c_{32}, s6) \wedge S_0 \sqsubset s_1 \wedge S_0 \sqsubset s_2 \sqsubset s \wedge S_0 \sqsubset s_3 \sqsubset s \wedge S_0 \sqsubset s_4 \sqsubset s \wedge S_0 \sqsubset s_5 \sqsubset s \wedge s_1 \sqsubset s_6 \sqsubset s.$$

Now we are ready to provide a formal definition of dynamical system diagnosis. Intuitively, a diagnosis is composed of two elements: a set of consistent assumptions regarding the initial state, and an executable sequence of actions that entails the observation. The set of consistent assumptions regarding the initial state is encoded in a formula referred to as $H(S_0)$. In cases where we have complete information about the initial state, $H(S_0)$ is the empty set. However, in cases where we have incomplete information about the initial state, we may be forced to make certain assumptions about the initial state, either because we need to establish the preconditions for actions we want to conjecture (e.g., we may wish to assume that our gas tank is low in order to generate a diagnosis that includes the action that we ran out of gas), or we may just need to establish conditions that we did not observe but that affect our diagnoses. Consider a medical diagnosis example where a patient presents with difficulty breathing and a rash on the face. In the absence of information about whether the patient has allergies, it is sensible to generate two different diagnoses, one predicated on the assumption (in $H(S_0)$) that the patient has allergies, and another predicated on the assumption of no allergies. Finally, in the case where we have a domain with components that are deemed to be faulty or AB, it is often compelling to assume $\neg AB(c_i, S_0)$, for all components. I.e., all components are operating normally in the initial situation.

Definition 2 (Diagnosis)
Given a system $DS = (\Sigma, OBS[S_0, s])$, a diagnosis is a tuple $(H(S_0), \mathbf{a})$, where $\mathbf{a} = [a_1, ..., a_k]$ such that:

$$\Sigma \cup H(S_0) \models \exists s.\, s = do(\mathbf{a}, S_0) \wedge executable(s) \wedge OBS[S_0, s],$$

where $H(S_0)$ contains sentences of the form $[\neg]F(\mathbf{c}, S_0)$, F is a fluent symbol and $\mathbf{c}$ is a vector of constants. Furthermore, $\Sigma \cup H(S_0)$ is consistent.

Our definition does not distinguish between diagnoses of differing quality. In all cases, we likely prefer diagnoses that avoid conjecturing extraneous actions or lengthy convoluted action sequences that are not germane to accounting for system observations. Beyond this, there may be domain-specific information that we would like to bring to bear in determining preferred diagnoses. Definition 3 provides a general definition of *preferred diagnosis* in terms of a reflexive and transitive preference relation $\preceq$ between diagnoses. If D_1 and D_2 are diagnoses for system DS and $D_1 \preceq D_2$ we say that D_1 is *at least as preferred as* D_2. $D_1 \prec D_2$ is an abbreviation for $D_1 \preceq D_2$ and $D_2 \not\preceq D_1$.

Definition 3 (Preferred Diagnosis)
Given a system DS, D is a preferred diagnosis for DS if and only if D is a diagnosis for DS and there does not exists another diagnosis D' for DS such that $D' \prec D$.

The $\prec$ relation can be realized in many ways. One such measure is to *minimize abnormality* or *faultiness*. In the case of static diagnosis, this translates into Reiter's notion (1987) of minimizing the number of components that a diagnosis determines to be behaving abnormally. In a dynamical diagnosis setting, this translates into minimizing the number of abnormal action occurrences (e.g., (Cordier and Thiébaux 1994)). In the next section we proposed different preference criteria for diagnoses.

Characterizing Preferred Diagnoses

In this section we explore two classes of criteria for defining preferred diagnoses: domain-independent notions based on minimality, and domain-specific notions based on commonsensical or expert knowledge about the domain.

Minimality Preferences over Diagnoses

In many situations, the most important criterion that distinguishes two diagnoses is the number of faulty actions posited by the diagnoses. Below we define a preference relation for diagnoses that accounts for this notion.

Definition 4 (Minimal Fault Diagnosis Criterion)
Given a system DS and diagnoses $D = (H(S_0), \mathbf{a})$ and $D' = (H'(S_0), \mathbf{a}')$ for DS, D is preferred to D' with respect to the minimal fault diagnosis criterion – stated $D \prec_{minfault} D'$ – iff $|\mathbf{a} \circledast \mathcal{A}_{faulty}| < |\mathbf{a}' \circledast \mathcal{A}_{faulty}|$, where $\mathbf{a} \circledast B$ denotes the longest subsequence of $\mathbf{a}$ whose elements are in set B.

Note that in the case of incomplete information about the initial state and where additional assumptions are being made about the initial state to account for the observations, this criterion (and the other minimality criteria that follow) can be misleading. At the extreme, consider the case where $H(S_0)$ states that c_1 and c_2 are faulty in S_0 and $\mathbf{a}$ contains no faulty actions, compared to the case where $H'(S_0)$ is empty and $\mathbf{a}'$ posits the occurrence of two faulty actions resulting in c_1 and c_2 becoming faulty. The minimal fault diagnosis criterion would prefer the first diagnosis to the second, but it is not clear that this is desirable. Further (domain-specific) refinement of these criteria can address this issue.

Two other domain-independent notions of preference are *shorter* and *simpler*: the former preferring diagnoses of shorter length; the latter preferring diagnoses that contain a subset of the actions of the other, both in name and in number. Both are best applied in conjunction with other preference criteria as a tie-breaker.

Definition 5 (Shorter Diagnosis Criterion)
Given a system DS and diagnoses $D = (H(S_0), \mathbf{a})$ and $D' = (H'(S_0), \mathbf{a}')$ for DS, D is preferred to D' with respect to its length – stated $D \prec_{short} D'$ – iff $|\mathbf{a}| < |\mathbf{a}'|$

Definition 6 (Simpler Diagnosis Criterion)
Given a system DS and diagnoses $D = (H(S_0), \mathbf{a})$ and $D' = (H'(S_0), \mathbf{a}')$ for DS, D is preferred to D' with respect to its simplicity – stated $D \prec_{simple} D'$ – iff $\{a \mid a \text{ in } \mathbf{a}\} \subseteq \{a \mid a \text{ in } \mathbf{a}'\}$ and $|\mathbf{a}| < |\mathbf{a}'|$

Domain-Specific Preferences over Diagnoses

In addition to the above domain-independent notions of preference, it is often natural to make use of some domain-specific preferences over diagnoses. In cases where we have a probability distribution and can characterize a notion of most likely diagnosis, this is a good quality measure and provides for an effective characterization of preferred diagnoses. Approximations of Dynamic Bayes Nets and sampling techniques such as particle filtering have proven to be useful paradigms for addressing this class of problems (e.g., Ng et al. 2002). Unfortunately, in many real-world systems we do not have probability estimates and must rely on commonsense knowledge of what typically goes wrong with a system under different circumstances. Instead we have commonsense or expert knowledge that may be in a form similar to the following: 1) *"If the car does not start, and it is cold, I strongly suspect a battery malfunction."* 2) *"When photocopier YY-2233 fails, I strongly suspect the lamp is burnt out."* 3) *"If the person is male and over 40 and presents urinary problems, then I strongly suspect a prostate problem."* 4) *"If I puncture my tire, it will eventually be flat."*

The notion of incorporating expert and commonsensical knowledge in the form of fault models and default preferences over diagnoses is not new and has proven effective in the diagnosis of static systems (e.g., Junker 1991). Indeed, much of the success of expert systems for fault diagnosis can be attributed to this form of knowledge. However, to date, rich domain-specific preferences have not been incorporated into the diagnosis of dynamical systems. Here we propose both a means of specifying rich preferences for dynamical systems, and later a means of computing them. To do so we make use of preference languages proposed for

planning to encode preferences over diagnoses. The qualitative preference language $\mathcal{LPP}$ (Bienvenu et al. 2010; 2006) and the preference language extension to the Planning Domain Definition Language PDDL3 (Gerevini et al. 2009) both present viable options. In particular, both support the expression of temporally extended preferences which enable preferred orderings on the occurrence of events. We use $\mathcal{LPP}$ here since its semantics has been defined in the situation calculus.

Specifying Diagnostic Preferences $\mathcal{LPP}$ (Bienvenu et al. 2010; 2006), is a first-order language for specifying user preferences. It enables the specification of preferences over properties of state and action occurrences as well as temporally extended preferences over multiple states. Unlike many preference languages, $\mathcal{LPP}$ provides a total order on preferences. It is qualitative in nature, facilitating elicitation.

User preferences in $\mathcal{LPP}$ are specified via a so-called *General Preference Formula* (GPF), Φ. This formula is a composition of individual preference statements combined together using conditionals, conjunction, and disjunction. In order to construct a GPF, we begin with the basic building block, a *Trajectory Property Formula (TPF)* (formerly called a BDF) which describes properties of situations (i.e., trajectories from S_0). The definitions that follow are taken from Bienvenu et al.'s paper (2010).

Definition 7 (Trajectory Property Formula (TPF))
A trajectory property formula *is a sentence drawn from the smallest set $\mathcal{B}$ where:*

- *$\mathcal{F} \subset \mathcal{B}$, where $\mathcal{F}$ is a set of fluent predicates*
- *$\mathcal{R} \subset \mathcal{B}$, where $\mathcal{R}$ is a set of non-fluent predicates*
- *If $f \in \mathcal{F}$, then **final**$(f) \in \mathcal{B}$*
- *If $a \in \mathcal{A}$, then **occ**$(a) \in \mathcal{B}$*
- *If φ_1 and φ_2 are in $\mathcal{B}$, then so too are $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, $\exists x\, \varphi_1$, $\forall x\, \varphi_1$, **next**(φ_1), **always**(φ_1), **eventually**(φ_1), and **until**(φ_1, φ_2)*

Following convention, fluents are represented in situation-suppressed form in $\mathcal{LPP}$ (e.g., the fluent $noisy(c, s)$ is represented as $noisy(c)$). **final**(f) states that fluent f holds in the final situation, **occ**(a) states that action a occurs in the present situation, and **next**(φ_1), **always**(φ_1), **eventually**(φ_1), and **until**(φ_1, φ_2) are basic linear temporal logic (LTL) constructs. TPFs establish properties of situations. By combining TPFs using boolean and temporal connectives, we are able to express a wide variety of properties. Given a particular situation (diagnosis), a TPF is either satisfied or not satisfied by the situation relative to a BAT.

The following are examples of TPFs from an augmentation of the computer diagnosis domain introduced in Section 2. Recall that these are hard properties of situations that will be used softly to express preferences.

$$(\forall c).\textbf{always}(\neg\textbf{occ}(fault(c))) \quad \text{(P1)}$$
$$(\forall c).\textbf{always}(\textbf{occ}(dskFail(c)) \supset \textbf{eventually}(scanSlow(c))) \quad \text{(P2)}$$
$$(\forall c).\textbf{always}(\textbf{occ}(dskFail(c)) \supset noisy(c)) \quad \text{(P3)}$$
$$(\forall c).\textbf{always}(\textbf{occ}(CPUFail(c)) \supset noisy(c)) \quad \text{(P4)}$$
$$(\forall c).\textbf{always}(newSoft(c) \supset \textbf{occ}(softFail(c))) \quad \text{(P5)}$$

P1 states that faulty actions never occur. P2 states that if component c has a hard disk failure, scanning of the component will eventually be slow. Similarly, P3 (resp. P4) states that if the component has a hard disk (resp. CPU) failure then it is noisy. Both of these statements can be used to establish corroborative evidence for a conjectured failure, and as such, they can be used as building blocks in establishing preferred diagnoses. However, since they will be applied as preferences, the absence of the evidence ($noisy(c)$) does not cause a diagnosis to be completely eliminated from consideration. Finally, P5 states that if the software is newly installed, then a software fault will occur.

To define preference orderings over alternative properties of situations, we define *Atomic Preference Formulae*. Each alternative being ordered comprises two components: the property of the situation, specified by a TPF, and a *value* term which stipulates the relative strength of the preference.

Definition 8 (Atomic Preference Formula (APF))
Let $\mathcal{V}$ be a totally ordered, finite set with minimal element v_{min} and maximal element v_{max}. An atomic preference formula is a formula $\varphi_0[v_0] \gg ... \gg \varphi_n[v_n]$, where each φ_i is a TPF, each $v_i \in \mathcal{V}$, $v_i < v_j$ for $i < j$, and $v_0 = v_{min}$. When $n = 0$, atomic preference formulae correspond to TPFs.

$\mathcal{V}$ can be any ordered set of descriptors, e.g., *excellent, good, ok, poor, horrible* or alternatively it can be numeric, with v_{min} most desirable. For ease of explication, let $\mathcal{V} = [0, 1]$ and consider the APF

$$P3[0] \gg P4[0.7] \quad \text{(P6)}$$

which states that we strongly prefer a diagnosis in which a hard disk fault results in a noisy component over a diagnosis in which a CPU fault results in a noisy component. The *strong* preference is reflected in the different values associated with each of the two TPFs.

To allow the user to aggregate preferences, General Preference Formulae extend $\mathcal{LPP}$ to conditional, conjunctive, and disjunctive preferences.

Definition 9 (General Preference Formula (GPF))
A formula Φ is a general preference formula if one of the following holds:

- *Φ is an atomic preference formula*
- *Φ is $\gamma : \Psi$, where γ is a TPF and Ψ is a general preference formula [Conditional]*
- *Φ is one of*
  - *$\Psi_0 \,\&\, \Psi_1 \,\&\, ... \,\&\, \Psi_n$ [General Conjunction]*
  - *$\Psi_0 \mid \Psi_1 \mid ... \mid \Psi_n$ [General Disjunction]*

where $n \geq 1$ and each Ψ_i is a general preference formula.

For example, the following GPF presents one way to combine preferences for the computer domain:

$$P1 \,\&\, P5 \,\&\, P6 \quad \text{(P7)}$$

Note that in both (P6) and (P7) we used equation numbers within formulae. This was a shorthand to facilitate readability and understanding and is not part of the syntax of the language. Preference formulae must be specified in full within APFs and GPFs.

Semantics The semantics of $\mathcal{LPP}$ is defined in the situation calculus, as fully described in (Bienvenu et al. 2010). Here we give an intuitive overview. TPFs are interpreted as situation calculus formulae and are evaluated as either true or false with respect to a BAT and situation. To define the semantics of our preference formulae, we associate a qualitative value or *weight* with a situation term. The weight of GPF Φ with respect to situation s is written as $w_s(\Phi)$. This weight is a composition of its constituents. For TPFs, a situation s is assigned the value v_{min} if the TPF is satisfied in s, v_{max} otherwise. Recall that in our example above $v_{min} = 0$ and $v_{max} = 1$, though they could equally well have been qualitative measures such as [excellent,...,horrible]. Similarly, given an APF, and a situation s, s is assigned the weight of the best TPF that it satisfies within the defined APF. Finally GPF semantics follow the natural semantics of boolean connectives. As such conjunction yields the maximum of its constituent GPF weights and disjunction yields the minimum of its constituent GPF weights. For a full explanation of the situation calculus semantics, please see (Bienvenu et al. 2010).

The following definition shows us how to compare two situations with respect to a GPF.

Definition 10 (Preferred Situations)
A situation s_1 is preferred to situation s_2 with respect to a GPF Φ, written $spref(s_1, s_2, \Phi)$, iff $w_{s_1}(\Phi) < w_{s_2}(\Phi)$.

Now we are ready to define the notion of preferred diagnosis in the presence of $\mathcal{LPP}$ preferences. Returning to our definition of Preferred Diagnosis (Definition 3), the $\prec$ relation can be realized as follows.

Definition 11 (Domain-Specific Preference Criterion)
Given a system DS, diagnoses $D = (H(S_0), \mathbf{a})$ and $D' = (H'(S_0), \mathbf{a}')$ for DS, and a GPF Φ, D is preferred to D' with respect to the preference criterion Φ – stated $D \prec_{pref(\Phi)} D'$ – iff $spref(do(\mathbf{a}, S_0), do(\mathbf{a}', S_0), \Phi)$.

Diagnosis as Planning

In this section we establish a correspondence between our characterization of diagnosis of dynamical systems and planning. Our ultimate objective is to enable the exploitation of state-of-the-art planning techniques for the purpose of generating diagnoses. Note that the majority of these planners operate over propositional domains. As such, we restrict our attention to diagnosis problems whose system descriptions conform to certain syntactic restrictions.

The Planning Domain Definition Language (PDDL) (McDermott 1998) is the *de facto* standard language for describing planning domains for input to most state-of-the-art planners. PDDL has a state-transitional semantics (Fox and Long 2003) and a situation-calculus semantics (Claßen et al. 2007). We use it as the description language for our planning domains. The theorems that follow in this section establish correspondences between variants of diagnosis described in Section and different types of planning problems.

In order to establish a correspondence between diagnosis and planning, we must define the relationship between a diagnosis system DS and its corresponding planning problem, $\mathcal{P}$. The first step is to transform the system description, Σ, itself a situation calculus BAT, into a PDDL planning domain description, T. To do so, we appeal to the work by Röger et al. (2008), which provides a maximally expressive transformation from a situation calculus BAT to the ADL subset of PDDL. We henceforth refer to this transformation as *the Röger et al. transformation*. The transformation necessitates imposing restrictions on the form of the BAT, Σ. The restrictions we appeal to are as follows.

1. We restrict to finite domains of named objects, and thus assume that Σ now includes a singleton set of axioms Σ_{fin} with the axiom:

   $$x = O_1 \vee x = O_2 \vee \ldots \vee x = O_n,$$

   where $O_1, \ldots, O_n$ are all the 0-ary function symbols of the sort *object* in Σ. Further, we assume that Σ_{una} contains unique axioms of the form $\neg O_i = O_j$ for every $i, j \in \{1, \ldots, n\}$, when $i \neq j$.
2. The arity of functional terms is restricted to zero, excluding the distinguished function $do(a, s)$.
3. The initial database is complete, i.e., for all fluent terms there is either an axiom of the form

   $$\neg F(x_1, \ldots, x_n, S_0), \quad (1)$$

   or

   $$F(x_1, \ldots, x_n, S_0) \equiv (x_1 = c_1^1 \wedge \ldots \wedge x_n = c_n^1) \vee \ldots \vee (x_1 = c_1^m \wedge \ldots \wedge x_n = c_n^m). \quad (2)$$

   where c_j^i are 0-ary function symbols.
4. An analogous restriction for situation-independent predicates.

Many dynamical systems have finite domains, and thus in the rest of this section we assume that the dynamical system modelled by Σ conforms to this restriction. However, the restriction on the completeness of the initial state – necessary for translation to PDDL and for use by most planners – is significant. Indeed, many diagnosis systems have incomplete information about the initial state. Intuitively, in the presence of an incomplete initial state, we would like our planners to find a completion of the initial state supporting the diagnosis (plan) generated. This differs from conformant planning in which the plan must hold for *all* consistent completions.

With this intuition in mind, we define the *completion* of an initial state as follows.

Definition 12 (Completion of an Initial State)
Let Σ be a finite-domain BAT. A completion $\Sigma^+_{S_0}$ of Σ_{S_0} is a set that:

1. *Contains, for each fluent F in Σ, an axiom of the form of Eq. (1) or Eq. (2) and contains no other axioms.*
2. *Only mentions 0-ary functions in Σ_{fin}, and*
3. *Is such that $\Sigma^+_{S_0} \cup \Sigma_{una} \cup \Sigma_{fin} \models \Sigma_{S_0}$.*

A *completion of a finite-domain BAT* Σ is constructed by replacing Σ_{S_0} with one of its completions. It is easy to see that if Σ_{S_0} is consistent, then it has a positive, bounded number of completions. Thus a BAT Σ induces a family of completions $\Sigma_1, \ldots, \Sigma_k$ (of the BAT) that comply with the restrictions described by Röger et al. We apply the Röger et al. transformation to this family of system descriptions in order to create a family of planning problems, each corresponding to an assumed consistent completion of the initial state of the original diagnosis system description.

Definition 13 (Corresponding Planning Problems)
Given a diagnosis system, $DS = (\Sigma, OBS[S_0, s])$, where Σ is finite domain and whose domain functions are restricted to 0-ary functions, we define a family of completed system descriptions, $\Sigma_1, \ldots, \Sigma_k$ to be the set of completions of Σ. A family of corresponding planning problems, is a set, $\mathcal{CPP}$, each of whose elements is of the form $\mathcal{P} = \langle T, F, A, I, G\rangle$ such that for each completed system description, Σ_i:

- *T is the domain description extracted from the Röger et al. transformation of Σ_i.*
- *F are the planning predicates corresponding to the predicates in Σ_i.*
- *A are the actions as in Σ_i.*
- *I is the initial state extracted from the Röger et al. transformation of Σ_i.*
- *G is the temporally extended goal, corresponding to $OBS[S_0, s]$ in DS.*

In the case where S_0 is complete, there is a single unique corresponding planning problem for DS. Furthermore, given a planning problem $\mathcal{P}$, generated by this procedure, we denote the completion of the BAT system description that generates it by $Th(\mathcal{P})$.

Note that the observations, $OBS[S_0, s]$ play the role of temporally extended goal, G, in the context of planning.

Definition 14 (Plan)
Given a diagnosis system DS and a corresponding planning problem $\mathcal{P} = \langle T, F, A, I, G\rangle$ drawn from $\mathcal{CPP}$, a sequence of actions $\pi = a_1, a_2, \ldots, a_n$, for $a_i \in A$, is a plan for $\mathcal{P}$ if and only if G holds appropriately from the execution of π, starting in I.

Theorem 1 (Diagnosis as Planning)
Given a diagnosis system $DS = (\Sigma, OBS[S_0, s])$ and family of corresponding planning problems $\mathcal{CPP}$, if $(H(S_0), \mathbf{a})$ is a diagnosis of DS then for all $\mathcal{P} \in \mathcal{CPP}$ such that $H(S_0)$ is entailed by the (complete) initial database of $Th(\mathcal{P})$, $\mathbf{a}$ is a plan for $\mathcal{P}$. On the other hand, if there is a plan $\mathbf{a}$ for $\mathcal{P} \in \mathcal{CPP}$, then there exists an $H(S_0)$, which can be constructed straightforwardly from $\mathcal{P}$ and $\mathbf{a}$, such that $(H(S_0), \mathbf{a})$ is a diagnosis of DS.

Proof sketch. ($\Rightarrow$) The proof is based on two intermediate results. First, for each model $\mathcal{M}$ such that $\mathcal{M} \models \Sigma \cup H(S_0)$, there is a completion Σ^+ of Σ such that $\mathcal{M} \models \Sigma^+$. Second, all models of a complete BAT are isomorphic. From these two observations it is simple to prove that for any model $\mathcal{M} \models \Sigma \cup H(S_0)$, there is some Σ^+ such that $\Sigma^+ \models \exists s.\, s = do(\mathbf{a}, S_0) \wedge executable(s) \wedge OBS[S_0, s]$. From here we use Röger et al.'s result to prove that such an $\mathbf{a}$ is obtained by planning in one of the problems $\mathcal{P} \in \mathcal{CPP}$. Finally, we prove that $\mathbf{a}$ is a plan in *all* completions of Σ that entail $H(S_0)$. To do this we use a contradiction argument and assume that there is some completion Σ^+ such that $\Sigma^+ \not\models \exists s.\, s = do(\mathbf{a}, S_0) \wedge executable(s) \wedge OBS[S_0, s]$. However, since $\Sigma^+ \models H(S_0)$ all models of Σ^+ are also models of $\Sigma \cup H(S_0)$ (this follows from the definition of completion), which leads to a contradiction under the assumption that $(H(S_0), \mathbf{a})$ is a diagnosis.
($\Leftarrow$) Using Röger et al.'s result if $\mathbf{a}$ is a plan for some $\mathcal{P} \in \mathcal{CPP}$, then it is also a plan in the BAT whose (complete) initial state is described by $\mathcal{P}$. Then, we define $H(S_0)$ to contain $R(\mathbf{c}, S_0)$ if and only if $R(\mathbf{c})$ is in I, and $\neg R(\mathbf{c}, S_0)$ if and only if $R(\mathbf{c})$ is in $F \setminus I$. It follows straightforwardly that $(H(S_0), \mathbf{a})$ is a diagnosis for DS. ■

Corollary 1 *Given $DS = (\Sigma, OBS[S_0, s])$ with complete information about the initial state, and its unique corresponding planning problems $\mathcal{P}$, $(\{\}, \mathbf{a})$ is a diagnosis of DS if and only if $\mathbf{a}$ is a plan for $\mathcal{P}$.*

Theorem 2 establishes a correspondence between the computation of minimal diagnoses and cost optimal planning.

Theorem 2 (Min. Diagnosis as Cost Optimal Planning)
Let $DS = (\Sigma, OBS[S_0, s])$ be diagnosis system and let S be the number of possible states described by Σ. Furthermore, let $\mathcal{CPP}$ be a family of corresponding planning problems, where we assign a uniform cost $c \geq 1$ to each action $a \in \mathcal{A}_{faulty}$, and a cost of $\epsilon < 1/S$ to all other actions. If $(H(S_0), \mathbf{a})$ is a minimal fault diagnosis of DS then $\mathbf{a}$ is a plan for some $\mathcal{P} \in \mathcal{CPP}$ such that the initial database of $Th(\mathcal{P})$ entails $H(S_0)$, and for any $\mathbf{a}'$ that is a plan for a $\mathcal{P}' \in \mathcal{CPP}$, the cost of $\mathbf{a}$ is not greater than the cost of $\mathbf{a}'$. On the other hand, if $\mathbf{a}$ is the minimum cost plan for any $\mathcal{P} \in \mathcal{CPP}$, then there exists $H(S_0)$ such that $(H(S_0), \mathbf{a})$ is a minimal fault diagnosis of DS.

Proof sketch. Follows from the fact that no optimal plan for any planning problem in $\mathcal{CPP}$ can have more than S actions. The rest of the proof is analogous to that of Theorem 1. ■

An analogous corollary to Corollary 1 exists for Theorem 2. Another way of computing minimal diagnoses is to express a preference against the occurrence of faulty actions and to generate a preferred plan that is optimal. As noted previously, preferences are an excellent way of characterizing nonprobabilistic causal fault models and commonsense or expert knowledge concerning the diagnosis of a dynamical system. We can compute these preferred diagnoses by appealing to preference-based planning (e.g., Baier et al. 2009).

Definition 15 (Preference-Based Planning (PBP))
Given a preference formula ϕ, an associated binary preference relation $\prec$, and a planning problem, $\mathcal{P} = \langle T, F, A, I, G\rangle$, let π_1 and π_2 be plans for $\mathcal{P}$. Plan π_1 is

preferred to plan π_2 if and only if $\pi_1 \prec \pi_2$. A plan π_1 is optimal for $\mathcal{P}$ if no plan π_2 is such that $\pi_2 \prec \pi_1$.

Theorem 3 (Preferred Diagnosis as PBP)
Given a diagnosis system $DS = (\Sigma, OBS[S_0, s])$, a preference formula ϕ, and family of corresponding planning problems $\mathcal{CPP}$, if $(H(S_0), \mathbf{a})$ is an optimal diagnosis of DS with respect to the preference criterion (Def. 11) then for some $\mathcal{P} \in \mathcal{CPP}$ such that $H(S_0)$ is entailed by the initial database of $Th(\mathcal{P})$, $\mathbf{a}$ is a plan for $\mathcal{P}$ and $\mathbf{a}$ is optimal relative to every other plan for any $\mathcal{P}' \in \mathcal{CPP}$. On the other hand, if $\mathbf{a}$ is the optimal plan relative to other plans for any $\mathcal{P} \in \mathcal{CPP}$, then there is an $H(S_0)$ such that $(H(S_0), \mathbf{a})$ is the optimal diagnosis of DS with respect to the preference criterion.

Proof sketch. Analogous to that of Theorem 1. ■

From Theory to Practice

In the previous section we established a correspondence between the generation of diagnoses and the generation of plans. More precisely, we showed that the generation of a diagnosis for a discrete dynamical system could be achieved by generating a plan for a corresponding planning problem that had complete information about the initial state and a temporally extended goal representing the observations. In this section we examine this relationship from a computational perspective. In particular, we examine the hypothesis that state-of-the-art planners are an effective means of generating dynamical diagnoses.

Planning technology has advanced tremendously in the last few years. In particular, the community has produced very efficient classical planners that assume complete information about the initial state and a final state goal. We have also seen great advances in cost-optimizing planners and in planners that plan with temporally extended goals (TEGs) and preferences (TEPs). These latter planners also assume complete information about the initial state, but the technology is less mature and they do not yet achieve the same quality of performance as classical planners. Here we explore the use of these different planners with respect to the generation of diagnoses. Exploiting such planners to generate diagnoses presents at least three challenges.

Challenge 1: Observations In Section we proposed that diagnostic observations can be represented as TEGs. However, classical planners generally have no way to directly express TEGs or partially ordered observations. We explored two ways of processing observations in our experiments. Following Grastien et al. (2007b), we compiled observations away by exploiting an action called "*advance*", which, through its effects, forces observation to be processed in the correct order. Essentially, it makes an observation possible only after all the observations that precede it have been observed. Using this technique, we could investigate the use of arbitrary classical planners. We also investigated encoding observations as both TEGs and TEPs exploiting advances in heuristic search planning designed precisely for these specialized goals and preferences (e.g., Baier and McIlraith 2006).

Challenge 2: Preferred Diagnoses In Section we explicitly examined the generation of two forms of preferred diagnosis: minimal diagnosis, and preferred diagnosis based on domain-specific preferences. Not surprisingly, the encoding required to generate such preferred diagnoses via planning is planner specific. Following Theorems 2 and 3, we explore the use of cost-optimizing planners to compute minimal fault diagnoses and we explore the use of preference-based planners to compute domain-specific preferred diagnoses. In some cases, it is possible to encode action costs as preferences or to encode preferences into a planning problem with action costs alone (Keyder and Geffner 2009).

Challenge 3: Incomplete Initial State As noted earlier, generating diagnoses from an incomplete initial state does not correspond to *conformant planning* but rather to planning in *some* complete-initial-state problem drawn from a *family* of such problems. This task is not one that has been addressed by the planning community. Here, we reduce this task to planning in a *single* planning problem. Our solution is as follows. Given the family of planning problems $\mathcal{CPP}$ of Theorem 1, we define a single planning problem $\mathcal{P}_{all}$ in the following way. $\mathcal{P}_{all}$ has the same predicates, actions, and goal as any of the problems in $\mathcal{CPP}$. (Note they all share these elements.) Its initial state only contains the ground predicates that are known to be true in S_0 (i.e., that entailed by Σ_{S_0}) plus an extra predicate called $init$. We conjoin $\neg init$ to the precondition of each action in $\mathcal{P}_{all}$. For each $\mathcal{P} \in \mathcal{CPP}$, we define an action in $\mathcal{P}_{all}$ that takes $init$ as a precondition, and whose effect leads to the initial state of $\mathcal{P}$. Each action has $\neg init$ as an effect, and each of the original effects in $\mathcal{CPP}$.

In the rest of this section we describe our efforts to investigate the feasibility of using planning technology to generate diagnoses. We had two objectives. The first was to compare planning technology to results obtained by Grastien et al.'s SAT-solving approach (2007b), a leading diagnosis engine. The second objective was to illustrate the performance of different planners with respect to challenging aspects of the diagnosis task including incomplete information about the initial state, the generation of minimal fault diagnoses, the explicit encoding of observations as TEGs, and computing diagnoses in the presence of rich preferences. The results presented here provide a small but reasonably representative subset of the experiments we ran.

Experiments

Comparison to SAT approach We compared the performance of several classical planners to results achieved by Grastien et al. (2007b) using their MINISAT system. Our comparison was restricted to Grastien et al.'s computer domain with complete information about initial state and observations encoded using *advance*. Grastien et al. (2007b)'s work investigated the effectiveness of MINISAT to find a diagnosis with a predetermined number of faults, i, under the

	Total-Order				Partial-Order			
#	MINISAT	FF	LAMA	SatPlan	MINISAT	FF	LAMA	SatPlan
1	0.03	0.00	0.00	0.30	0.04	0.00	0.01	0.10
3	0.26	0.02	0.05	2.14	0.14	0.03	0.07	0.89
5	1.90	0.03	0.14	6.38	9.70	0.08	0.36	2.27
7	9.10	0.16	0.40	12.56	350.00	24.98	12.93	4.10
9	61.00	0.10	0.85	-	63.00	37.73	>600	11.68
11	3.90	0.25	1.22	-	>600	586.76	>600	25.23
13	8.70	0.20	1.67	-	>600	32.22	>600	35.96
15	18.70	0.17	4.99	-	>600	16.76	-	36.71
17	30.00	0.22	4.95	-	>600	128.36	-	96.15
19	66.00	0.22	8.10	-	>600	169.34	-	99.12

Figure 2: Runtime comparison (in seconds) between MINISAT, FF, LAMA, SatPlan, on total- and partial-order observations. The MINISAT results are taken directly from (Grastien et al. 2007b). Dash entries indicate out of memory.

	Total-Order				Partial-Order		
#	LAMA	$SGPlan_6$	Baseline	HPLAN-P	LAMA	Baseline	HPLAN-P
1	0.01	0.08	0.01	0.29	0.02	0.04	0.8
3	0.07	0.14	1.93	25.25	0.11	47.09	2.14
5	0.21	0.25	24.39	107.21	0.97	>600	579.85*
7	0.64	7.64	54.53	449.53	17.95	>600	30.37
9	1.62	6.1	73.1	>600	334.01	-	286.16*
11	2.88	8.97	48.88	>600	>600	-	>600
13	3.82	7.97	80.74	>600	-	-	>600
15	16.28	12.49	66.89	>600	-	-	>600
17	20.17	10.54	101.15	>600	-	-	>600
19	29.54	10.87	139.98	>600	-	-	>600

Figure 3: Computing minimal fault diagnosis with action costs and preferences. * indicates the plan found has one extra fault action over the minimum admitted by the problem. Dash entries indicate out of memory.

	Total-Order		Partial-Order	
#	TEGs	TEPs	TEGs	TEPs
1	0.33	0.22	0.19	0.2
2	2.26	1.49	0.74	0.71
3	164.76	137.26	10.82	7.37
4	292.74	218.99	99.31	69.67
5	405.66	328.13	>600	>600
6	>600	>600	>600	>600
7	>600	>600	>600	>600

Figure 4: Running HPLAN-P (modified to prune non-perfect partial plans) on problems in which observations are encoded as temporally extended goals (TEGs) and as temporally extended preferences (TEPs).

assumption of complete information about the initial state. To do so, they proposed to generate a SAT theory of bounded length k and solve it using MINISAT. If a satisfaction was not achieved, they would increment k and iterate until satisfaction was achieved. This procedure is guaranteed to find a minimal diagnosis. However, in the experiments reported in (Grastien et al. 2007b), which we use here for comparison, no incremental computation was used. Instead, MINISAT was given the number of faults, i, a priori and MINISAT solved a satisfiable formula with bound i without the need for iteration. As such, the results don't reflect the iteration required to determine a minimal fault diagnosis.

Based on a PDDL encoding provided by Grastien, we constructed our own lifted encoding of the computer domain in PDDL. We constructed two sets of 20 problems so that problem i has a minimal diagnosis with i faulty actions. The first set considers only totally ordered observations, and the second, only partially ordered. We evaluated our approach using state-of-the-art planners FF (Hoffmann and Nebel 2001), LAMA (Richter et al. 2008), and SatPlan'06 (Kautz et al. 2006). We report the results for odd-numbered problems in Figure 2. This table also includes the results reported in (Grastien et al. 2007b) using the SAT solver, MINISAT. We did not run MINISAT ourselves on their encoding. As such, different machines were used and the comparison is not ideal. Nevertheless, our results show that for all problems, at least one of the planners outperforms (Grastien et al. 2007b)'s MINISAT results, sometimes by an order of magnitude on total-order problems. In the case of partially ordered observations, two of our three planners were able to solve problems that MINISAT could not solve (i.e., problems 11-19).

It is important to note that the planners we used were not configured to find minimal fault diagnosis (only LAMA can run in such a mode). Nevertheless, in all total-order problems planners found minimal fault diagnoses. In the partial-order problems, FF and SatPlan found a solution with one extra fault on two of the problems (i.e., in problem 10, FF found a solution with 11 faults). The results demonstrate that state-of-the-art classical planners provide an effective means of computing dynamical diagnoses and underline the promise of our approach.

Minimal Fault Diagnoses In many situations, it is important to generate diagnoses that minimize the number of faults. Theorem 2 establishes a way to do this using cost optimal planning. As such, we experimented with the following planners that support action costs: LAMA, $SGPlan_6$ (Hsu and Wah 2008), and Baseline. (Baseline was the best-performing planner in the 2008 International Planning Competition, sequential optimization track. It performs uniform-cost search (A*, $h = 0$) using LAMA's search engine.) Note that planners are not generally guaranteed to find the shortest plan; however, in many cases, they are designed to find a short plan.

We also encoded the action costs as preferences using PDDL3 (for each component, there is a preference that indicates absence of faults), and used HPLAN-P (Baier et al. 2009) to plan with these PDDL3 preferences. Figure 3 shows a subset of the results we obtained. Note that with the exception of Baseline, the planners are not guaranteed to find a minimal plan, but in all problems we ran they found the minimal plan (diagnosis) with the exception of HPLAN-P on problem 5 and 9-partial which ran out of time and could not improve the quality of the generated plan. The results for $SGPlan_6$ on the partial-ordered cases are not shown since except for problem 1 (0.07) the rest ran out of time.

Observations as TEGs and TEPs In the results reported to this point, observations were compiled away using *advance*. Here we explore encoding them as TEGs and as TEPs in PDDL3, exploiting advances in heuristic search

planning (e.g., Baier and McIlraith 2006) to compute plans (diagnoses). In the case where observations were encoded as TEPs, we modified HPLAN-P so that as soon as a partial plan violated a preference it was pruned. This ensured that all observations were enforced. The result of running HPLAN-P for the first 7 problems is shown in Figure 4. HPLAN-P did not show the same high-quality performance as the classical planners with observations compiled away using *advance*, and indeed encoding observations as TEPs was slightly more effective than encoding them as TEGs. We attribute this performance to the lack of maturity of TEG and TEP planning and in particular to an idiosyncrasy with HPLAN-P's heuristics. We still fundamentally believe this to be a promising way of encoding observations that will be borne out as TEG planning technology matures.

Diagnoses under Incomplete Information To compute diagnoses with incomplete information we appealed to the procedure described at the outset of this section that exploits special actions to *complete* the incomplete initial state. We tested two variants of our procedure one that completed the state in one action, and another that completed the state in multiple sequential actions. We tested our approach on the computer domain, modified to make the initial state incomplete with respect to the state of the components.

Recall that a component can be in one of 6 different states. Let x be the number of components whose state is unknown in the initial state. The first approach added 6^x new actions to the domain, each such action simultaneously establishes the initial state of all x components. The second approach adds $6 * x$ new actions, where each action only establishes the initial state of a single component, instead requiring x of these actions to occur in sequence to complete the initial state. In Figure 5 we illustrate the scalability of the second approach as the number of unknown components, x, increases. We do not report results on the first approach since the size of the planning problem does not scale well with x.

Preferred Diagnoses To evaluate the use of preference-based planners to generate diagnoses with domain-specific preferences, we again extended our computer domain, this time so that components could be faulty as a result of power, hardware, software, or hard-disk failure. Our preferences look similar to the examples in Section , but were expressed in PDDL3. PDDL3 aggregates preferences via a metric function. For these problems we were interested in minimizing the PDDL3 metric function to achieve an optimal plan (diagnosis). We assigned a penalty of one to every preference formula not satisfied, so a metric value of n indicates that n preferences were violated.

We ran the PDDL3 planner HPLAN-P on this modified domain. Since HPLAN-P is incremental (finds better and better plans until timeout is reached), the reported result is on the last plan returned within the timeout of 600 seconds. Results in Figure 6 show that as we increase the number of preferences, in almost all problems either more time is spent or a worse-quality plan is found.

#	Type	$x=2$	$x=4$	$x=6$	$x=8$	$x=10$	$x=12$
1	1-total	0.00	0.00	0.00	0.00	0.30	8.15
2	1-partial	0.00	0.00	0.00	0.00	0.01	0.01
3	3-total	0.01	0.02	0.01	0.01	0.01	0.02
4	3-partial	0.04	0.06	0.27	1.31	7.93	174.08
5	5-total	0.11	0.17	0.80	7.53	79.55	338.69
6	5-partial	0.10	0.15	0.14	0.36	2.93	5.61
7	7-total	0.17	0.17	0.20	2.39	6.71	14.78
8	7-partial	26.66	28.23	46.23	122.01	>600	>600
9	9-total	0.14	0.23	0.83	1.92	4.55	17.79
10	9-partial	56.96	55.21	126.00	391.95	554.98	>600
11	11-total	0.25	0.36	0.95	1.58	3.97	17.32
12	11-partial	>600	>600	>600	>600	>600	>600
13	13-total	0.22	0.33	1.18	4.51	4.36	10.53
14	13-partial	56.85	98.16	100.69	>600	>600	>600
15	15-total	0.26	0.37	0.89	4.98	13.48	39.91
16	15-partial	17.85	17.76	17.89	287.02	>600	>600
17	17-total	0.23	0.25	0.27	0.31	0.37	0.35
18	17-partial	222.30	373.73	393.48	>600	>600	>600
19	19-total	0.25	1.10	1.58	1.80	4.49	18.46
20	19-partial	183.94	196.44	400.52	>600	>600	>600

Figure 5: Testing the performance of FF on problems in which we have incomplete information about the initial state. The unknown in question is the initial state of a component. We increase the complexity of the problem by increasing the number of unknowns.

		40-preferences		80-preferences		120-preferences	
#	Type	Time	Metric	Time	Metric	Time	Metric
1	1-total	1.03	0.00	3.17	0.00	1.39	0.00
2	1-partial	0.73*	0.00	7.31	0.00	6.51	0.00
3	2-total	8.9*	0.00	36.79	0.00	38.74	0.00
4	2-partial	9.60	0.00	52.39	0.00	9.83	0.00
5	3-total	81.04*	0.00	194.17	0.00	273.47	0.00
6	3-partial	18.00	0.00	45.88	0.00	82.00	0.00
7	4-total	164.42	0.00	565.33	1.00	537.77	0.00
8	4-partial	46.00*	0.00	91.09*	0.00	105.23*	0.00
9	5-total	352.35*	2.00	404.15*	2.00	382.94*	3.00
10	5-partial	>600	N/A	>600	N/A	>600	N/A
11	6-total	594.19*	2.00	331.71*	3.00	343.18*	3.00
12	6-partial	257.52	0.00	471.31	1.00	280.16	0.00

Figure 6: Running HPLAN-P on problems that deal with finding a preferred diagnosis. * indicates the plan found has one extra fault action over the minimum admitted by the problem.

Summary

In this paper, we revisit the characterization of diagnosis of discrete dynamical systems, and its relationship to planning from both a theoretical and a computational perspective. Acknowledging that many diagnosis problems suffer from incomplete information, our characterization of diagnosis specifies a diagnosis as a tuple: a sequence of actions that account for the observations, potentially predicated on some assumptions about the (incomplete) initial state. The incorporation of expert and commonsensical knowledge about the (faulty) behaviour of systems has been key to the diagnosis of static systems. This paper is the first to explore this issue for dynamical diagnosis. To this end, we propose a characterization of preferred diagnosis in terms of rich, temporally extended domain-specific preferences.

Automated planning is an active area of research and one where we are seeing rapid advances with respect to both the

efficiency of solvers and the classes of planning problems they can solve. A key contribution of this work is the establishment of a correspondence between dynamical diagnosis and planning. With this correspondence in hand, researchers can exploit advances in planning technology for the purposes of diagnosis generation. Our investigations show that current planning technology is competitive with, and in some cases outperforms by an order of magnitude, a leading diagnosis engine. We also confirm the feasibility of planning technology in computing minimal diagnoses, domain-specific preference-based diagnoses, and diagnoses in the face of incomplete information.

Acknowledgements

We thank Alban Grastien for providing us with an encoding of the computer problem, which we used in this paper for benchmarking. We also thank Christian Fritz for many interesting discussions relating to the use of preferences in diagnosis. Christian did some unpublished early work on this topic as part of a course project. Finally, we gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Ontario Ministry of Innovations Early Researcher Award (ERA).

References

J. Baier and S. McIlraith. Planning with first-order temporally extended goals using heuristic search. In *Proc. of the 21st National Conference on Artificial Intelligence (AAAI)*, 788–795, 2006.

J. A. Baier, F. Bacchus, and S. A. McIlraith. A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence*, 173(5-6):593–618, 2009.

C. Baral, S. McIlraith, and T. Son. Formulating diagnostic problem solving using an action language with narratives and sensing. In *Proc. of the 7th International Conference on Knowledge Representation and Reasoning (KR)*, 311–322, 2000.

M. Bienvenu, C. Fritz, and S. A. McIlraith. Planning with qualitative temporal preferences. In *Proc. of the 10th International Conference on Knowledge Representation and Reasoning (KR)*, 134–144, 2006.

M. Bienvenu, C. Fritz, and S. A. McIlraith. Specifying and computing preferred plans. *Artificial Intelligence*, 2010. To appear.

J. Claßen, Y. Hu, and G. Lakemeyer. A situation-calculus semantics for an expressive fragment of PDDL. In *Proc. of the 22nd National Conference on Artificial Intelligence (AAAI)*, 956–961, 2007.

M.-O. Cordier and S. Thiébaux. Event-based diagnosis of evolutive systems. In *Proc. of the 5th International Workshop on Principles of Diagnosis (DX)*, 64–69, 1994.

J. deKleer, A. K. Mackworth, and R. Reiter. Characterizing diagnoses and systems. *Artificial Intelligence*, 56(2-3):197–222, 1992.

M. Fox and D. Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.

A. Gerevini, P. Haslum, D. Long, A. Saetti, and Y. Dimopoulos. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6):619–668, 2009.

A. Grastien, Anbulagan, J. Rintanen, and E. Kelareva. Diagnosis of discrete-event systems using satisfiability algorithms. In *Proc. of the 22nd National Conference on Artificial Intelligence (AAAI)*, 305–310, 2007.

A. Grastien, Anbulagan, J. Rintanen, and E. Kelareva. Modeling and solving diagnosis of discrete-event systems via satisfiability. In *Proc. of the 18th International Workshop on Principles of Diagnosis (DX)*, 2007.

J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

C.-W. Hsu and B. Wah. The SGPlan planning system in IPC-6, 2008.

G. Iwan. History-based diagnosis templates in the framework of the situation calculus. In *Proc. of the Joint German/Austrian Conference on Artificial Intelligence (KR/ÖGAI)*, 244–259. 2001.

U. Junker. Prioritized defaults: Implementation by tms and application to diagnosis. In *IJCAI91*, 310–317, 1991.

H. A. Kautz and B. Selman. Unifying SAT-based and graph-based planning. In *Proc. of the 16th International Joint Conference on Artificial Intelligence (IJCAI)*, 318–325, 1999.

H. Kautz, B. Selman, and J. Hoffmann. Planning as satisfiability. Abstracts of the 5th international planning competition, 2006.

E. Keyder and H. Geffner. Soft Goals Can Be Compiled Away. *Journal of Artificial Intelligence Research*, 36:547–556, 2009.

G. Lamperti and M. Zanella. *Diagnosis of active systems*. Kluwer Academic Publishers, 2003.

D. V. McDermott. PDDL — The Planning Domain Definition Language. Technical Report TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

S. McIlraith. Towards a theory of diagnosis, testing and repair. In *Proc. of the 5th International Workshop on Principles of Diagnosis (DX)*, 185–192, 1994.

S. A. McIlraith. Explanatory diagnosis: Conjecturing actions to explain observations. In *Proc. of the 6th International Conference of Knowledge Representation and Reasoning (KR)*, 167–179, 1998.

B. Ng, L. Peshkin, and A. Pfeffer. Factored particles for scalable monitoring. In *Proc. of the 18th Conference in Uncertainty in Artificial Intelligence (UAI)*, 370–377, 2002.

Y. Pencolé and M.-O. Cordier. A formal framework for the decentralised diagnosis of large scale discrete event systems and its application to telecommunication networks. *Artificial Intelligence*, 164(1-2):121–170, 2005.

R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

R. Reiter. *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press, Cambridge, MA, 2001.

S. Richter, M. Helmert, and M. Westphal. Landmarks revisited. In *Proc. of the 23rd National Conference on Artificial Intelligence (AAAI)*, 975–982, Chicago, IL, 2008.

J. Rintanen and A. Grastien. Diagnosability testing with satisfiability algorithms. In *IJCAI*, 532–537, 2007.

G. Röger, M. Helmert, and B. Nebel. On the relative expressiveness of ADL and Golog: The last piece in the puzzle. In *Proc. of the 11th International Conference on Knowledge Representation and Reasoning (KR)*, 544–550, 2008.

M. Sampath, R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. Teneketzis. Diagnosability of discrete-event systems. *IEEE Transactions on Automatic Control*, 40(9):1555–1575, 1995.

M. Thielscher. A theory of dynamic diagnosis. Electronic transactions on artificial intelligence. *Electron. Trans. Artif. Intell.*, 1:73–104, 1997.