Explanatory Diagnosis: Conjecturing Actions to Explain Observations[*]

Sheila A. McIlraith

Knowledge Systems Laboratory
Department of Computer Science
Stanford University
Stanford, CA 94305

Abstract. In this paper we present contributions towards a logical theory of diagnosis for systems that can be affected by the actions of agents. Specifically, we examine the task of conjecturing diagnoses to explain *what happened* to a system, given a theory of system behaviour and some observed (aberrant) behaviour. We characterize what happened by introducing the notion of explanatory diagnosis in the language of the situation calculus. Explanatory diagnoses conjecture sequences of actions to account for a change in system behaviour. As such, we show that determining an explanatory diagnosis is analogous to classical AI planning with state constraints and incomplete knowledge. The representation scheme we employ provides an axiomatic solution to the frame, ramification and qualification problems for a syntactically restricted class of state constraints. Exploiting this representation, we show that determining an explanatory diagnosis can be achieved by regression followed by theorem proving in the database describing what is known of the initial state of our system. Further, we show that by exploiting features inherent to diagnosis problems, we can simplify the diagnosis task.

Foreword

It is a great pleasure to contribute a paper to this book in honour of Ray Reiter. Ray was my Ph.D. supervisor at the University of Toronto. Inspired by Ray's work on diagnosis from first principles, my early work proposed an augmentation of his characterization of diagnosis to include a formal account of testing and test generation. The work on testing convinced me that a comprehensive account of diagnostic problem solving must involve reasoning about action and change. Again, Ray's work on the situation calculus had a tremendous influence. The problem of intergrating actions into diagnostic problem solving posed many interesting challenges, which led to a solution to the ramification problem (sometimes), and to several new characterizations of diagnosis, one of which is described in this paper.

So, Happy Birthday Ray! Here's to the pleasures of life. There are many more theorems to prove, mountains to climb, and butterflies to chase. You've had a long and impressive research career. The field, and the many researchers you've touched with your work, owe you a debt of gratitude.

[*] A version of this paper originally appeared in [23]

1 Introduction

Given a theory of system behaviour and some observed aberrant behaviour, the traditional objective of diagnosis is to conjecture *what is wrong* with the system, (e.g., which components of the device are behaving abnormally, what diseases the patient is suffering from, etc.). Each candidate diagnosis consists of a subset of distinguished literals that are conjectured to be true or false in order to account for the observation in some way. Different criteria have been proposed for determining the space of such candidate diagnoses. Within formal accounts of diagnosis, two widely accepted definitions of diagnosis are consistency-based diagnosis (e.g., [8,26]), and abductive explanation (e.g., [4,8,20,25]). Such research has historically focussed on static systems. Recently, some researchers have advocated extending diagnostic problem solving (DPS) to enable reasoning about actions, under the argument that DPS is purposive in nature and that systems operate within and are affected by agents[1].

In this paper we focus upon one aspect of diagnosing such dynamic systems. In particular, given a theory of system behaviour and some observation of (aberrant) behaviour, our concern is with the task of conjecturing diagnoses to explain *what happened* to the system (i.e., what actions or events occurred to result in the observed behaviour). Knowing or conjecturing what happened is interesting in its own right, but it can also help to further constrain the space of possible states of the system. In so doing, conjecturing what happened facilitates conjecturing of what is wrong with a system, as well as predicting other relevant system behaviour. Compared to our traditional notion of *what is wrong* diagnoses, knowing *what happened* can more accurately capture the root cause of system malfunction rather than its manifestations, thus providing for the identification of future preventative as well as prescriptive actions.

In the spirit of previous foundational work in model-based diagnosis (MBD) (e.g., Reiter [26], Console and Torasso [4], de Kleer et al. [8]), this paper presents a logical characterization of the diagnosis task. We take as our starting point the existing MBD research on characterizing diagnoses for static systems *without* a representation of actions (e.g., [8,4,26]). Next, we exploit a situation calculus representation scheme previously proposed by the author [21] that enables the integration of a representation of action with the representation of the behaviour of a static system. With this representation in hand, we provide a logical characterization for the task of determining *what happened* to a system. The characterization is presented in the guise of *explanatory diagnosis.*

The distinguishing features of our characterization are afforded in great part by the richness of our representation scheme which provides a comprehensive and semantically justified representation of action and change. In

[1] An agent could be another system, a robot, a human, or nature.

particular, our representation provides an axiomatic closed-form solution to the frame and ramification problems for a syntactically restricted class of state constraints that is common to diagnosis. This representation captures the direct and indirect effects of actions in a compiled representation, which is critical to our ability to generate explanatory diagnoses efficiently. Further, our representation provides a closed-form solution to the qualification problem, thus identifying the conditions underwhich an action is possible. It is interesting to note that when we are dealing with incomplete knowledge of our initial state, conjecturing an action or sequence of actions also requires conjecturing that its preconditions are satisfied, which in many instances serves to further constrain our search.

As we show in the sections to follow, our characterization establishes a direct link between explanatory diagnosis and planning, deductive plan synthesis, and abductive planning. As a consequence of a completeness assumption embedded in our representation, we can exploit goal-directed reasoning in the form of regression [34] in order to generate diagnoses. This completeness assumption also provides for an easy mapping of our situation calculus representation to Prolog. While explanatory diagnoses can be mathematically characterized in an analogous fashion to plans, an important distinction of explanatory diagnoses is that they can be refined to exploit features of our diagnosis problem that have no meaning in the context of planning. Indeed, we use diagnosis-specific attributes to define variants of explanatory diagnosis to deal with the challenges of incomplete knowledge and large search spaces.

2 Representation Scheme

2.1 Situation Calculus Language

The situation calculus language in which we axiomatize our domains is a sorted first-order language with equality. The sorts are of type $\mathcal{A}$ for primitive *actions*, $\mathcal{S}$ for *situations*, and $\mathcal{D}$ for everything else, including domain objects [17]. We represent each action as a (possibly parameterized) first-class object within the language. Situations are simply sequences of actions. The evolution of the world can be viewed as a tree rooted at the distinguished initial situation S_0. The branches of the tree are determined by the possible future situations that could arise from the realization of particular sequences of actions. As such, each situation along the tree is simply a history of the sequence of actions performed to reach it. The function symbol *do* maps an action term and a situation term into a new situation term. For example, $do(turn\_on\_pump, S_0)$ is the situation resulting from performing the action of turning on the pump in situation S_0. The distinguished predicate $Poss(a, s)$ denotes that an action a is possible to perform in situation s (e.g., $Poss(turn\_on\_pump, S_0)$). Thus, $Poss$ determines the subset of the situation tree consisting of situations that are possible in the world. Finally, those properties or relations whose truth value can change from situation to situation are referred to as *fluents*. For example, the fluent $on(Pump, s)$ expresses that the pump is on in situation s.

The dialect of the situation calculus that we use in this paper is restricted to primitive, determinate actions. Our language does not include functional fluents, nor does it include a representation of time, concurrency, or complex actions, but we believe the results presented herein can be extended to more expressive dialects of the situation calculus (e.g., [24,29]).

2.2 Domain Representation

In this section we overview the representation scheme we use to characterize the system we will be diagnosing. The scheme, proposed in [21], integrates a situation calculus theory of action with a MBD system description, SD [8]. The resulting representation of a system comprises both domain-independent and domain-specific axioms. The domain-independent axioms are the foundational axioms of the discrete situation calculus, Σ_{found} [17]. They are analogous to the axioms of Peano arithmetic, modified to define the branching structure of our situation tree, rather than the number line. The domain-specific axioms, T specify both the *behaviour of the static system*, and the *actions*[2] that can affect the state of the system, as well as those actions required to achieve testing and repair. Together they define our situation calculus representation $\Sigma = \Sigma_{found} \wedge T$.

The domain-specific axioms composing T result from application of a procedure proposed in [21] that compiles a typical MBD system description, SD and a set of axioms relating to the preconditions and effects of actions into a representation that provides a closed-form solution to the frame, ramification and qualification problems for a syntactically restricted class of domain axiomatizations. The resulting domain axiomatization $T = T_{SC}^{S_0} \wedge T_{domain} \wedge T_{SS} \wedge T_{AP} \wedge T_{UNA} \wedge T_{DCA} \wedge T_{S_0}$ is described below. The representation is predicated on an explicit causal ordering of fluents and a completeness assumption. The assumption states that all the conditions underwhich an action a can lead, directly or indirectly, to fluent F becoming true or false in the successor state are captured in the axiomatization of our system.

We illustrate the representation scheme in terms of a small portion of a power plant feedwater system [22] derived from the APACS project [14]. This simplified example models the filling of a vessel either by the operation of an electrically powered ($Power$) pump ($Pump$), by manual filling, or by a siphon that was started by the pump or by manual filling. For notational convenience, all formulae are understood to be universally quantified with respect to their free variables, unless explicitly indicated otherwise. For a more thorough description of this representation scheme, and for a more extensive example please see [21,22].

[2] Actions can be performed by agents: a human, another system, or nature.

Every domain axiomatization, T comprises the following sets of axioms.

$$T_{SC}^{S_0} \wedge T_{domain} \wedge T_{SS} \wedge T_{AP} \wedge T_{UNA} \wedge T_{DCA} \wedge T_{S_0}$$

The set of state constraints relativized to situation S_0, $T_{SC}^{S_0}$ capture what is implicitly true about the initial database. They can be acquired from a typical MBD system description, SD as described in [21]. Note that these state constraints hold for all situations s. We represent them explicitly for the initial situation S_0, and they are compiled into the successor state axioms, T_{SS} for all subsequent situations. In our simple example, $T_{SC}^{S_0}$ is as follows.

$$\neg AB(Power, S_0) \wedge \neg AB(Pump, S_0) \wedge on(Pump, S_0) \supset filling(S_0) \quad (1)$$
$$manual\_fill(S_0) \supset filling(S_0) \quad (2)$$
$$\neg(on(Pump, S_0) \wedge manual\_fill(S_0)) \quad (3)$$

Axiom (1) states that if the pump and power are operating normally and the pump is *on* in S_0, then $filling$ will be true in S_0. The set of domain constraints, T_{domain} is as follows.

$$Power \neq Pump \quad (4)$$

The set of successor state axioms, T_{SS} is composed of axioms of the following general form, one for each fluent F.

$$Poss(a, s) \supset [F(do(a, s)) \equiv \Phi_F] \quad (5)$$

where Φ_F is a simple formula[3] of a particular syntactic form. Intuitively, a successor state axiom says the following:

$$\begin{aligned} &Poss(a,s) \supset [fluent(do(a,s)) \equiv \\ &\quad \textit{an action made it true} \\ &\quad \vee \ \textit{a state constraint made it true} \\ &\quad \vee \ \textit{it was already true} \\ &\quad\quad \wedge \ \textit{neither an action nor a state constraint} \\ &\quad\quad\quad\quad \textit{made it false}]. \end{aligned}$$

The following axioms compose T_{SS} for our example.

$$\begin{aligned} Poss(a,s) \supset [&on(Pump, do(a,s)) \equiv a = turn\_on\_pump \\ &\vee (on(Pump, s) \wedge a \neq turn\_off\_pump)] \end{aligned} \quad (6)$$

$$\begin{aligned} Poss(a,s) \supset [&AB(Power, do(a,s)) \equiv a = power\_failure \\ &\vee (AB(Power, s) \wedge a \neq aux\_power \wedge a \neq power\_fix)] \end{aligned} \quad (7)$$

[3] A *simple formula* only mentions domain-specific predicate symbols, fluents do not include the function symbol *do*, there is no quantification over sort *situation*, and there is at most one free *situation* variable.

$$\begin{aligned}Poss(a,s) \supset [&AB(Pump, do(a,s)) \equiv a = pump\_burn\_out \\ &\vee (AB(Pump,s) \wedge a \neq pump\_fix)]\end{aligned} \tag{8}$$

$$\begin{aligned}Poss(a,s) \supset [&manual\_fill(do(a,s)) \equiv a = turn\_on\_manual\_fill \\ &\vee (manual\_fill(s) \wedge \ a \neq turn\_off\_manual\_fill)]\end{aligned} \tag{9}$$

$$\begin{aligned}Poss(a,s) \supset [&filling(do(a,s)) \equiv a = turn\_on\_manual\_fill \\ &\vee \ (manual\_fill(s) \wedge a \neq turn\_off\_manual\_fill) \\ &\vee \ [(a \neq power\_failure \\ &\qquad\qquad \wedge \ (\neg AB(Power,s) \vee a = aux\_power \\ &\qquad\qquad\qquad \vee \ a = power\_fix)) \\ &\quad \wedge \ (a \neq pump\_burn\_out \\ &\qquad\qquad \wedge \ (\neg AB(Pump,s) \vee a = pump\_fix)) \\ &\quad \wedge \ (a = turn\_on\_pump \\ &\qquad\qquad \vee \ (on(Pump,s) \wedge a \neq turn\_off\_pump))] \\ &\vee \ (filling(s) \wedge a \neq stop\_siphon)]\end{aligned} \tag{10}$$

Axiom (6) states that if action a is possible in situation s, then the pump is on in the situation resulting from performing action a in situation s (i.e., $on(Pump, do(a,s))$) if and only if the action a is $turn\_on\_pump$, or the pump was already on in s and a is not $turn\_off\_pump$. Note that the successor state axioms presented here, and in particular axiom (10), are fully compiled. They can be expressed more compactly as intermediate successor state axioms [22].

The set of action precondition axioms, T_{AP} is composed of axioms of the following general form, one for each action prototype A in the domain.

$$Poss(A(\boldsymbol{x}), s) \equiv \Pi_A \tag{11}$$

where Π_A is a simple formula with respect to situation variable s, capturing the necessary and sufficient conditions for prototype action $A(\boldsymbol{x})$ to be executable in situation s.

$$Poss(stop\_siphon, s) \equiv (\neg manual\_fill(s) \wedge \ \neg on(Pump,s)) \tag{12}$$

$$Poss(pump\_fix, s) \equiv \neg on(Pump,s) \tag{13}$$

$$Poss(pump\_burn\_out, s) \equiv on(Pump,s) \tag{14}$$

$$Poss(turn\_on\_manual\_fill, s) \equiv \neg on(Pump,s) \tag{15}$$

$$Poss(turn\_on\_pump, s) \equiv \neg manual\_fill(s) \tag{16}$$

$$Poss(turn\_off\_pump, s) \equiv on(Pump,s) \tag{17}$$

$$Poss(turn\_off\_manual\_fill, s) \equiv manual\_fill(s) \tag{18}$$

$$Poss(power\_failure, s) \equiv \text{true} \tag{19}$$

$$Poss(power\_fix, s) \equiv \text{true} \tag{20}$$

$$Poss(aux\_power, s) \equiv \text{true} \tag{21}$$

Finally, we provide a possible set of initial conditions for our system, T_{S_0}. These constitute the explicit aspect of the initial database. Note that in general we do not have complete knowledge of the initial state of our system. This makes the task of diagnosis all the more challenging. In this example, we do not know initially whether the pump and power are operating normally. We also do not know whether the vessel was filling in the initial state.

$$on(Pump, S_0) \wedge \neg manual\_fill(S_0) \tag{22}$$

T_{UNA} and T_{DCA} are the unique names axioms for actions and the domain closure axiom for actions, respectively.

This concludes the description of our representation scheme. Before advancing to issues of diagnosis, we note [30] that our proposed situation calculus representation can be viewed as an executable specification because it is easily realized in Prolog by exploiting Prolog's completion semantics and simply replacing the equivalence connectives characteristic of axioms in T_{SS} and T_{AP} by implication connectives. The Lloyd-Topor transformation [19] must then be applied to convert this theory into Prolog clausal form. Later in this paper, we will advocate using Waldinger's notion of regression to rewrite axioms of our representation and simplify computation. This type of regression rewriting is precisely achieved by Prolog's backwards chaining mechanism.

3 Preliminaries

With our representation in hand, we turn our attention to the task of diagnosis. In this section we introduce the framework for performing diagnosis relative to our representation. For our purposes we adopt the ontological and notational convention of the MBD literature and view the systems we are diagnosing as comprising a number of interacting *components*, $COMPS$. These components have the property of being either abnormal or normal in a situation. We express this property in our situation calculus language using the fluent AB. For example, $AB(Pump, s)$ denotes that the pump component is abnormal in situation s. Note that the use of AB is not mandatory to the contributions of this paper. Once again, following the convention in the MBD literature, we define our diagnoses relative to the domain-independent concept of a *system* [8], adapted to our situation calculus framework.

Definition 1 (System).
A system is a quadruple (Σ, $HIST$, $COMPS$, OBS) where:

- Σ, the background theory, is a set of situation calculus sentences describing the behaviour of our system and the actions that can affect it.
- $HIST$, the history, is a sequence of ground actions $[a_1, \ldots, a_k]$ that were performed starting in S_0.
- $COMPS$, the components, is a finite set of constants.
- OBS_F, the observation, is a simple formula composed of fluents whose only free variable is the situation variable s, and which are otherwise ground.

Example 1.
In our power plant example above, Σ is our axiomatization $\Sigma_{found} \wedge T$ and $COMPS = \{Pump, Power\}$. The observation, OBS_F could be $filling(s)$, for example. $HIST$ could be empty, i.e., [ ], or perhaps $[turn\_on\_pump]$.

4 Explanatory Diagnosis

In this section we introduce and formally characterize the notion of an explanatory diagnosis which conjectures *what happened* to result in some observed (aberrant) behaviour. Given a system, (Σ, $HIST$, $COMPS$, OBS_F), the objective of explanatory diagnosis is to conjecture a sequence of actions, $[\alpha_1, \ldots, \alpha_n]$ such that our observation is true in the situation resulting from performing that sequence of actions in $do(HIST, S_0)$.

Definition 2 (Explanatory Diagnosis).
An explanatory diagnosis for system (Σ, $HIST$, $COMPS$, OBS_F) is a sequence of actions $E = [\alpha_1, \ldots, \alpha_n]$ such that,

$$\Sigma \models Poss(HIST \cdot E, S_0)^3 \wedge OBS_F(do(HIST \cdot E, S_0)).$$

Thus, E is an explanatory diagnosis when the observation is true in the situation resulting from performing the sequence of actions E in situation $do(HIST, S_0)$, and further that the preconditions for each action of the action sequence $HIST \cdot E$ are true in the appropriate situations, commencing at S_0.

The problem of determining explanatory diagnoses is an instance of temporal explanation or postdiction (e.g., [31]), and is related to the classical AI planning problem. In particular, identifying the sequence of actions composing an explanatory diagnosis, E is analogous to the plan synthesis problem, and thus is realizable using deduction on the situation calculus axioms. According to [12], a plan to achieve a goal $G(s)$ is obtained as a side effect of

[3] **Notation:**
$HIST \cdot E$ is an abbreviation for $[a_1, \ldots, a_k, \alpha_1, \ldots, \alpha_n]$.
$do([a_1, \ldots, a_m], s)$ is an abbreviation for
$do(a_m,(do(a_{m-1},(do(a_{m-2},(\ldots,(do(a_1, s))))))))$.
Finally, $Poss([a_1, \ldots, a_n], s)$ is an abbreviation for
$Poss(a_1, s) \wedge Poss(a_2, do(a_1, s)) \wedge \ldots \wedge Poss(a_n, do([a_1, \ldots, a_{n-1}], s))$.

proving $Axioms \models \exists s.G(s)$. The binding for the situation variable s represent the sequence of actions. In our case, $Axioms \models \exists s.G(s)$ is analogous to $\Sigma \models \exists s.OBS_F(s)$. As such, our representation enables us to generate explanatory diagnoses deductively, just as we could deductively generate a plan in the situation calculus. Note that the task of generating explanatory diagnoses is analogous to plan synthesis in the presence of state constraints – a challenging problem. Our representation scheme eliminates the additional challenges presented by state constraints by providing a domain axiomatization that a priori solves the frame, ramification and qualification problems.

Example 2.
Given the power plant example system $(\Sigma, [\,], \{Power, Pump\}, \neg filling(s))$, the sequence of actions $[power\_failure]$ constitutes *one* example of an explanatory diagnoses for the system. Another explanatory diagnosis for our system is $[turn\_off\_pump]$.

Observe that for certain problems there can be an infinite number of sequences of actions that constitute explanatory diagnoses. For example, the following sequences of actions also constitute valid explanatory diagnoses for our example system:
$[power\_failure, power\_fix, power\_failure]$,
$[power\_failure, aux\_power, power\_failure]$,
$[turn\_off\_pump, power\_failure, turn\_on\_pump]$,
and so on.

Definition 2 is not sufficiently discriminating to eliminate these, clearly suboptimal explanatory diagnoses. We must define a preference criterion. Probability measures, even simple order of magnitude probabilities have provided an effective preference criterion for many applications of MBD [7]. Likewise, in the case of determining explanatory diagnoses in the context of the situation calculus, probabilities will serve us well in identifying preferred explanatory diagnoses. Unfortunately, probability measures are not always available and the correct treatment of probabilities in our situation calculus framework is only now being developed. In this paper, we limit our discussion to what we refer to as a chronologically simple preference criterion.

In our chronologically simple preference criterion, we prefer diagnoses that are relativized to situations reached without performing any extraneous actions. Note that this preference criterion is syntactic in nature, relying on the notion of a primitive action as a unit measure.

Definition 3 (Simpler).
Given a sequence of actions $HIST = [\alpha_1, \ldots, \alpha_n]$, define $ACTS(HIST)$ to be the set $\{\alpha_1, \ldots, \alpha_n\}$, and $LEN(HIST)$ to be the the length of the sequence of actions composing $HIST$.

Thus, given $HISTA = [a_1, \ldots, a_n]$ and $HISTB = [b_1, \ldots, b_n]$, situation $S_A = do(HISTA, S_0)$ is simpler than situation $S_B = do(HISTB, S_0)$ iff $ACTS(HISTA) \subseteq ACTS(HISTB)$ and $LEN(HISTA) < LEN(HISTB)$.

Definition 4 (Chronologically Simple Diagnosis).
E is a chronologically simple explanatory diagnosis for system (Σ, $HIST$, $COMPS$, OBS_F) iff E is an explanatory diagnosis for the system, and there is no explanatory diagnosis $E^{'}$ such that situation $S^{'} = do(HIST \cdot E^{'}, S_0)$ is simpler than situation $S = do(HIST \cdot E, S_0)$.

We might further distinguish this criterion to prefer chronologically simple explanatory diagnoses comprised solely of actions performed by nature.

Finally, observe that the characterization of explanatory diagnosis just presented assumes that E and OBS_F occur *after* $HIST$. While this assumption is not critical to characterizing explanatory diagnoses, it acts as a form of preference, facilitating computation of E.

5 Exploiting Regression

In the previous section, we provided a characterization of explanatory diagnosis. We observed that computing an explanatory diagnosis for system (Σ, $HIST$, $COMPS$, OBS_F) is analogous to generating a plan to achieve a goal $OBS_F(s)$ starting with axioms Σ and situation $do(HIST, S_0)$. At first glance, the general problem of computing explanatory diagnoses does not look very promising for at least three reasons: the second-order induction axiom in Σ_{found}, the potential incompleteness of the initial database, and the potentially large size of the situation search space. In this section, we show how diagnoses can be computed by exploiting regression [34]. We are able to exploit regression as a direct result of the embedded completeness assumption inherent in our representation scheme. In this context, regression is a recursive rewriting procedure that we use to reduce the nesting of the *do* function in situation terms, or to eliminate the *Poss* predicate. We show that generating explanatory diagnoses reduces to regression followed by entailment with respect to the initial database. Computationally, the merit of regression is that it searches backwards through the situation space from the observation rather than searching forward from the initial database. Under the assumption that the observation consists of fewer literals than the initial database, regression will make for more efficient search. Observe that Prolog's backwards chaining mechanism achieves the substitution performed by regression.

Following directly in the spirit of previous work by Reiter, [27,28] and more recently [30], on the exploitation of regression for planning and query answering, we first define two regression operators, $\mathcal{R}^*$ and $\mathcal{R}_{Poss}$.

Definition 5 (Regression Operator $\mathcal{R}^*$).
Given a set of successor state axioms, T_{SS} composed of axioms of the form $Poss(a, s) \supset [F(do(a, s)) \equiv \Phi_F]$, $\mathcal{R}^*[\Psi]$, the repeated regression of formula Ψ with respect to successor state axioms T_{SS} is the formula that is obtained from Ψ by repeatedly replacing each fluent $F(do(a, s))$ in Ψ by Φ_F, until the resulting formula makes no mention of the function symbol *do*.

For example,

$$\begin{aligned}
&\mathcal{R}^*[on(Pump, do(turn\_on\_manual\_fill, do(turn\_on\_pump, S_0))]\\
&\quad = \mathcal{R}^*[(turn\_on\_manual\_fill = turn\_on\_pump)\\
&\qquad\quad \vee\ (on(Pump, do(turn\_on\_pump, S_0))\\
&\qquad\qquad \wedge\ (turn\_on\_manual\_fill \neq turn\_off\_pump))]\\
&\quad = \mathcal{R}^*[\text{false} \vee (on, (Pump, do(turn\_on\_pump, S_0)) \wedge \text{true})]\\
&\quad = \mathcal{R}^*[(turn\_on\_pump = turn\_on\_pump)\\
&\qquad\quad \vee\ (on(Pump, S_0)) \wedge (turn\_on\_pump \neq turn\_off\_pump))]\\
&\quad = \text{true}
\end{aligned}$$

We can similarly define a *Poss* regression operator over the set of action precondition axioms, T_{AP}. This regression operation rewrites each occurrence of the literal $Poss(a, s)$ by Π_A as defined in the action precondition axioms.

Definition 6 (Regression Operator $\mathcal{R}_{Poss}$).
Given a set of action precondition axioms, T_{AP} composed of axioms of the form $Poss(A(\boldsymbol{x}), s) \equiv \Pi_A$, $\mathcal{R}_{Poss}[W]$ is the formula obtained by replacing each occurrence of predicate $Poss(A(\boldsymbol{x}), s)$ by Π_A. All other literals of W remain the same.

Reiter proved soundness and completeness results for regression applied to a theory with no state constraints [28, Theorem 1, Theorem 2]. In the theorem below, we prove soundness and completeness results for our representation scheme which includes state constraints. The theory Σ_{init} mentioned in the theorem below is a subset of Σ containing only information about the initial situation, and no information about successor situations. It also excludes the induction axiom of Σ_{found}.

Theorem 1 (Soundness and Completeness).
Given

- Σ_{init}, *a subset of the situation calculus theory* Σ, *such that*
  $\Sigma_{init} = \Sigma_{UNS} \wedge T_{S_0} \wedge T_{SC}^{S_0} \wedge T_{domain} \wedge T_{UNA}$,
  where Σ_{UNS} *is a subset of* Σ_{found} *containing the set of unique names axioms for situations.*
- *a sequence of ground actions,* s_HIST *such that*
  $\Sigma_{init} \wedge \mathcal{R}^*\ [\mathcal{R}_{Poss}\ [Poss(s\_HIST,\ S_0)]]$ *is satisfiable.*
- $Q(s)$, *a simple formula whose only free variable is situation variable* s.

Further, suppose $S = do(s\_HIST, S_0)$, *then*

- $\Sigma \models Q(do(s\_HIST, S_0))$ *iff* $\Sigma_{init} \models \mathcal{R}^*[Q(do(s\_HIST, S_0))]$,
- $\Sigma \models Poss(s\_HIST, S_0)$ *iff* $\Sigma_{init} \models \mathcal{R}^*[\mathcal{R}_{Poss}[Poss(s\_HIST, S_0)]]$,
- $\Sigma \wedge Poss(s\_HIST, S_0) \wedge Q(do(s\_HIST, S_0))$ *is satisfiable iff*
  $\Sigma_{init} \wedge \mathcal{R}^*[\mathcal{R}_{Poss}[Poss(s\_HIST, S_0)]] \wedge \mathcal{R}^*[Q(do(s\_HIST, S_0))]$
  is satisfiable.

Thus, assuming situation s is a possible situation and exploiting regression, $Q(s)$ holds at situation s iff its regression is entailed in the initial database. The beauty of Theorem 1 is that it enables us to generate explanatory diagnoses via regression followed by theorem proving in the initial database, without the need for the second-order induction axiom in Σ_{found}. From these results, we can characterize explanatory diagnosis with respect to regression.

Corollary 1 (Explanatory Diagnosis with Regression).
The sequence of actions $E = [\alpha_1, \ldots, \alpha_k]$ is an explanatory diagnosis for system (Σ, $HIST$, $COMPS$, OBS_F) iff

$$\Sigma_{init} \models \mathcal{R}^*[\mathcal{R}_{Poss}[Poss(HIST \cdot E, S_0)]] \wedge \ \mathcal{R}^*[OBS_F(do(HIST \cdot E, S_0))].$$

6 Exploiting the Task

In the previous section we showed that we could exploit regression to simplify the computation of explanatory diagnoses. A remaining source of difficulty in generating explanatory diagnoses is that our search space may be large and our initial database may be incomplete. An incomplete initial database both underconstrains our search problem, and precludes us from using certain planning machinery, such as STRIPS [10], that assumes a complete or near-complete initial database [18]. In this section we show how to exploit features of diagnosis problems to further assist in the generation of explanatory diagnoses. In particular, we propose to 1) make assumptions regarding our domain, 2) relax our criteria for explanatory diagnoses, and 3) verify rather than generate diagnoses. An additional means of simplifying our computational task is to use likelihoods of actions and action sequences to focus search for explanatory diagnoses. Detailed discussion of this option is beyond the scope of this paper.

6.1 Assumption-Based Diagnoses

In diagnostic problem solving it is common to make further assumptions that are consistent with what we know of the world. For example, we may assume that in the absence of information to the contrary, components are operating normally, or certain properties hold of the world. To support such assumption-based reasoning, we define the notion of an assumption-based explanatory diagnosis.

Definition 7 (Assumption-Based Explanatory Diagnosis).
Given an assumption $H(S)$ relativized to ground situation S such that

- $S_0 \leq {}^4S \leq do(HIST \cdot E, S_0)$,
- $\Sigma \wedge H(S)$ is satisfiable, and
- $\Sigma \wedge H(S) \models Poss(HIST, S_0)$.

An assumption-based explanatory diagnosis for system (Σ, $HIST, COMPS$, OBS_F) under assumption $H(S)$ is a sequence of actions $E = [\alpha_1, \ldots, \alpha_k]$ such that,

$$\Sigma \wedge H(S) \models Poss(HIST \cdot E, S_0) \wedge OBS_F(do(HIST \cdot E, S_0)).$$

In Example 2 of the previous section, we did not have complete information about the initial state of our system. It could actually have been the case that observation $\neg filling$ was true in S_0, i.e., $\neg filling(S_0)$, but since it was not entailed by Σ, the empty action sequence was not proposed as a valid explanatory diagnosis, and we were forced to conjecture a sequence of actions to account for our observation. If we assume $\neg filling(S_0)$, then the empty sequence of actions is indeed an assumption-based explanatory diagnosis.

In generating explanatory diagnoses, we may want to make a priori assumptions about the world, conjoin these assumptions to our theory and then try to compute our explanatory diagnoses. For example, we may wish to assume that all components are operating normally in S_0. This would be achieved by making $H(S)$ in our definition equal to $\bigwedge_{c \in COMPS} \neg AB(c, S_0)$ (i.e., $\neg AB(Pump, S_0) \wedge \neg AB(Power, S_0)$). Similarly, we may wish to assume that the observation, OBS_F is true in $do(HIST, S_0)$. In our example above, this would mean assuming $H(S) = \neg filling(S_0)$. In other instances, we might want $H(S)$ to equal a *what is wrong* diagnosis that we are currently entertaining, relativized to a previous situation. For example, we might want to assume that the pump was abnormal in the initial situation, i.e. $H(S) = AB(Pump, S_0)$.

In still other instances, we may not want to fix our assumptions a priori but rather make the minimum number of assumptions necessary to generate a chronologically simple explanatory diagnosis. Such assumptions might be limited to a distinguished set of literals which the domain axiomatizer considers to be legitimately assumable (e.g., AB fluents). There might also be a partial ordering, e.g. a likelihood ordering, on such assumables fluents.

The distinction regarding when we make our assumptions affects the machinery by which we compute assumption-based explanatory diagnoses. If assumptions are made prior to computation we simply conjoin the regression of the assumption to the initial database and use regression and theorem proving as we would for generating normal explanatory diagnoses. When assumptions are interleaved with computation, generating an assumption-based explanatory diagnosis requires abduction.

In a theorem prover, abduction is generally implemented as proof-tree completion, i.e., by resolving dead-ends of proof trees with abducible literals.

[4] **Notation:** The transitive binary relation $<$ defined in Σ_{found} further limits our situation tree by restricting the actions that are applied to a situation to those whose preconditions are satisfied in the situation. Intuitively, if $s < s'$, then s and s' are on the same branch of the tree with s closer to S_0 than s'. Further, s' can be obtained from s by applying a sequence of actions whose preconditions are satisfied by the truth of the $Poss$ predicate.

To generate an explanatory diagnosis, an abductive theorem prover would attempt to prove $OBS_F(s)$. If an attempted proof failed because it dead-ended on a literal or literals that were assumable, then these would be abduced and the proof continued. We have not yet implemented such an abduction engine for our situation calculus system. It is interesting to note that in the context of planning, [32] has used abduction to implement a partial order planner for the event calculus. In contrast to the deductive approach of the situation calculus, this planner abduces actions and the relative order of certain actions.

6.2 Potential Diagnoses

In addition to making assumptions to help complete our theory, we can also facilitate computation by relaxing the criteria for defining an explanatory diagnosis. To this end, we observe that the requirement in Definitions 2 and 7 that $\Sigma \models Poss(HIST \cdot E, S_0)$ may be too stringent in the case of an incomplete initial database. That is, it may not be reasonable to require that we know that an action is possible in a situation that is incompletely specified. We may prefer to consider explanatory diagnoses, where the theory allows us to consistently assume that the preconditions for $HIST$ or for $HIST \cdot E$ hold, but not necessarily that they are entailed by our theory. To this end, we propose the following alteration to our definition of explanatory diagnosis, Definition 2. A comparable refinement can be made to our definition of assumption-based explanatory diagnosis, Definition 7.

Definition 8 (Potential Explanatory Diagnosis).
A potential explanatory diagnosis for system $(\Sigma, HIST, COMPS, OBS_F)$ is a sequence of actions $E = [\alpha_1, \ldots, \alpha_k]$ such that,

$$\Sigma \wedge Poss(HIST \cdot E, S_0) \text{ is satisfiable, and}$$
$$\Sigma \wedge Poss(HIST \cdot E, S_0) \models OBS_F(do(HIST \cdot E, S_0)).$$

Note that $HIST$ is a sequence of actions that we know to have been performed starting in situation S_0. Thus, the preconditions for each of the actions in $HIST$ are true in the corresponding situations. This provides us with further information concerning the truth values of fluents at various situations, helping to constrain our search.

6.3 Verifying Likely Diagnoses

To further address the problem of generating explanatory diagnoses, we propose exploiting domain information and maintaining a library of most likely (assumption-based) explanatory diagnoses, indexed by observations and/or situation histories. With candidate diagnoses in hand, the problem of computing explanatory diagnoses reduces to a verification problem, rather than a generation problem. Given a system (Σ, $HIST$, $COMPS$, OBS_F), and a candidate diagnosis E, such that $S = do(HIST \cdot E, S_0)$, we are interested in verifying that E is indeed a diagnosis of the system. Verifying a candidate diagnosis is simply a query evaluation problem. It can be accomplished by regression and theorem proving in the initial database, as per Theorem 1 above.

Example 3.
Given the system (Σ, [ ], $\{Power, Pump\}$, $\neg filling(s)$), and the candidate diagnosis E=[$power\_failure$], E can be verified to be an explanatory diagnosis with respect to the system by evaluating the query

$$\mathcal{R}^*[\mathcal{R}_{Poss}[Poss(do(power\_failure, S_0))]] \\ \wedge\ \mathcal{R}^*[\neg filling(do(power\_failure, S_0))]$$

with respect to the initial database, Σ_{init}.

7 Related Work

This work has been influenced by formal characterizations of diagnosis for systems without an explicit representation of actions (e.g., [8,26,4]) and by Reiter's work on the frame problem and the problem of temporal projection [28] and [30]. Aside from previous work by the author (e.g., [22,21]), research to date has not explicitly addressed the problem of integrating a rich representation of actions into diagnostic reasoning. As such, there is little related work that exploits a comprehensive representation of action.

A recent notable exception is Thielscher's work on dynamic diagnosis [33]. Thielscher's basic representation is similar in spirit to ours, though he does not use the situation calculus and he does not exploit an axiomatic solution to the frame, ramification and qualification problems. Like us, he adopts the basic MBD ontology, employs an action theory to represent actions, and represents certain state constraints causally to capture the indirect effects of actions as dictated by the system description of the device. Where he differs is in the actual task he is performing, and the assumptions he makes. To use terminology discussed here, he is computing *what is wrong* diagnoses. Given an action history and some observed aberrant behaviour, Thielscher conjectures that certain components must be abnormal to account for the observed behavior. He is not conjecturing actions. To simplify computation, he assumes that all components are initially normal, and he uses a priori likelihood of failure to select most likely candidate diagnoses. While his paper examines *what is wrong* diagnoses, rather than *what happened* diagnoses, clearly the two are intimately related. [22] discusses the inter-relationship between these two types of diagnosis in the context of the situation calculus framework described here.

The research on temporal diagnosis and diagnosis of dynamic systems originating in the diagnosis research community (e.g., [2,3,13,11,15,9]) and in particular [5] is also loosely related. Brusoni et al. [2] recently provided a characterization of temporal abductive diagnosis together with algorithms for computing these diagnoses under certain restrictions. Building on earlier work by some of the authors [3], they decouple atemporal and temporal diagnoses, using SD to represent the behaviour of the atemporal components and transition graphs to represent the temporal components. The later work uses temporal constraints to represent the temporal components. Also related is the work on event-based diagnosis by Cordier and Thiébaux [5]. Their work is similar in motivation to our work on explanatory diagnosis, viewing the diagnosis task as the determination of the event-history of a system between successive observations. While this work is related, the representation of action is impoverished. It does not provide a comprehensive representation of the preconditions for and the effects of actions, nor does it address the frame, ramification and qualification problems. Their transition system is sufficiently expressive for their application but the lack of a compact representation proves problematic.

In the area of reasoning about action, research on temporal explanation and postdiction has an interesting relationship to this work (e.g., [6,1]). Of particular note is Shanahan's research [31]. While Shanahan also proposes the situation calculus as a representation language for axiomatizing his domain, he does so without an axiomatic solution to the frame and ramification problems. As such these problems must be addressed coincidentally with generating explanatory diagnoses. In contrast, our characterization of explanatory diagnosis, with its axiomatic solution to the frame and ramification problems, enables simpler characterization and computation of temporal explanation.

8 Summary

The results in this paper provide contributions to model-based diagnosis and to reasoning about actions. Our concern in this paper was, given a system that affects and can be affected by the actions of agents, and given some observed (aberrant) behaviour, how do we capture the notion of *what happened*, i.e., how do we go about conjecturing a sequence of actions that account for the behaviour we have observed. As we discussed at the beginning of this paper, not only is conjecturing candidate explanatory diagnoses interesting in its own right, but it facilitates determining the current state of the system.

We addressed this problem by providing a mathematical characterization of the notion of explanatory diagnosis in the context of a rich situation calculus representation, proposed in [21]. Our characterization made apparent the direct relationship of explanatory diagnosis to the planning task, and in particular to Green's notion of deductive plan synthesis. However, generating explanatory diagnoses is actually akin to planning in the face of state constraints and a potentially incomplete initial database. Our representation scheme addressed the challenges presented by state constraints by providing an axiomatic a priori solution to the frame, ramification and qualification problems. This enabled us to extend Reiter's results on the soundness and completeness of regression and to show that we can generate explanatory diagnoses by regression followed by theorem proving in the initial database. A remaining difficulty was that our initial database is often incomplete. Exploiting features of diagnosis problems, we proposed the notions of assumption-based and potential explanatory diagnosis, to allow for the conjectured sequences of actions that constitute a diagnosis to be predicated on some other assumptions we choose to make about the world. Finally, we proposed exploiting a library of precomputed likely diagnoses, indexed by context and observations. This enabled us to verify, rather than generate explanatory diagnoses.

In our dissertation work [22], we have also addressed the complementary problem of conjecturing *what is wrong* diagnoses. In future work we examine the application of Golog procedures [16] to diagnostic problem solving.

Acknowledgements I would like to thank Ray Reiter, Yves Lespérance, Hector Levesque and Allan Jepson for helpful comments on the work presented in this paper. I would also like to thank Oskar Dressler, for engaging me in an interesting discussion regarding this work that helped me look at the material in a different light. Finally, thanks to the KR'98 reviewers for their conscientious reviews.

References

1. Baker, A. (1991) Nonmonotonic Reasoning in the Framework of the Situation Calculus. *Artificial Intelligence* **49**:5–23.
2. Brusoni, V., Console, L., Terenziani, P., and Theseider Dupré, D. (1995) An Efficient Algorithm for Computing Temporal Abductive Diagnoses. In *Proceedings of the Sixth International Workshop on Principles of Diagnosis*, 41–48.
3. Console, L., Portinale, L., Theseider Dupré, D. and Torasso, P. (1994) Diagnosing Time-Varying Misbehavior: An Approach Based on Model Decomposition. *Annals of Mathematics and Artificial Intelligence* **11**(1–4):381–398.
4. Console, L. and Torasso, P. (1991) A Spectrum of Logical Definitions of Model-Based Diagnosis. *Computational Intelligence* **7**(3):133–141.
5. Cordier, M. and Thiébaux, S. (1994) Event-Based Diagnosis for Evolutive Systems. In *Proceedings of the Fifth International Workshop on Principles of Diagnosis*, 64–69.
6. Crawford, J. and Etherington, D. (1992) Formalizing Reasoning About Change: A Qualitative Reasoning Approach. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, 577–582.
7. de Kleer, J. (1991) Focusing on Probable Diagnoses. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, 842–848.
8. de Kleer, J., Mackworth, A. and Reiter, R. (1992) Characterizing Diagnoses and Systems. *Artificial Intelligence* **56**(2–3):197–222.

9. Dressler, O. (1994) Model-based diagnosis on board: Magellan-MT inside. In *Proceedings of the Fifth International Workshop on Principles of Diagnosis*, 87–92.
10. Fikes, R. and Nilsson, N. (1971) STRIPS: A New Approach to Theorem Proving in Problem Solving. *Artificial Intelligence* **2**:189–208.
11. Friedrich, G. and Lackinger, F. 1991. Diagnosing Temporal Misbehaviour. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, 1116–1122.
12. Green, C. C. (1969) Theorem Proving by Resolution as a Basis for Question-Answering systems. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 4*. New York: American Elsevier. 183–205.
13. Hamscher, W. (1991) Modeling Digital Circuits for Troubleshooting. *Artificial Intelligence* **51**(1–3):223–271.
14. Kramer, B. and Mylopolous, J. et al. (1996) Developing an Expert System Technology for Industrial Process Control: An Experience Report. In *Proceedings of the Conference of the Canadian Society for Computational Studies of Intelligence (CSCSI'96)*, 172–186.
15. Lackinger, F. and Nejdl, W. (1991) Integrating Model-Based Monitoring and Diagnosis of Complex Dynamic Systems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence (IJCAI-91)*, 1123–1128.
16. Levesque, H., Reiter, R., Lespérance, Y., Lin, F. and Scherl, R. (1997) Golog: A Logic Programming Language for Dynamic Domains. In *Journal of Logic Programming, Special Issue on Actions* 31(1–3):59–83.
17. Lin, F. and Reiter, R. (1994) State Constraints Revisited. *Journal of Logic and Computation* 4(5):655–678. Special Issue on Action and Processes.
18. Lin, F. and Reiter, R. (1995) How to Progress a Database II: The STRIPS Connection. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, 2001–2007.
19. Lloyd, J. (1987) *Foundations of Logic Programming*. Springer Verlag, second edition.
20. McIlraith, S. (1994) Further Contributions to Characterizing Diagnosis. *Annals of Mathematics and Artificial Intelligence* **11**(1–4):137–167.
21. McIlraith, S. (1997) Representing Actions and State Constraints in Model-Based Diagnosis. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, 43–49.
22. McIlraith, S. (1997) *Towards a Formal Account of Diagnostic Problem Solving*. Ph.D. Dissertation, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.
23. McIlraith, S. (1998) Explanatory Diagnosis: Conjecturing Actions to Explain Observations. In Cohn, A. G., Schubert, L., Shapiro, S. S., eds., *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, 167–177. Morgan Kaufmann.
24. Pinto, J. (1994) *Temporal Reasoning in the Situation Calculus*. Ph.D. Dissertation, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.
25. Poole, D. (1988) Representing Knowledge for Logic-Based Diagnosis. In *Proceedings of the Fifth Generation Computer Systems Conference (FGCS-88)*, 1282–1290.
26. Reiter, R. (1987) A Theory of Diagnosis from First Principles. *Artificial Intelligence* **32**:57–95.

27. Reiter, R. (1991) *The Frame Problem in the Situation Calculus: A Simple Solution (sometimes) and a completeness result for goal regression.* Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of J. McCarthy. San Diego, CA: Academic Press. 359–380.
28. Reiter, R. (1992) The Projection Problem in the Situation Calculus: a Soundness and Completeness Result, with an Application to Database Updates. In *Proceedings First International Conference on AI Planning Systems*, 198–203.
29. Reiter, R. (1996) Natural Actions, Concurrency and Continuous Time in the Situation Calculus. In Aiello, L.; Doyle, J.; and Shapiro, S., eds., *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, 2–13. Cambridge, Massachusetts, USA.: Morgan Kaufmann.
30. Reiter, R. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems.* In preparation. Draft available at http://www.cs.toronto.edu/~cogrobo/.
31. Shanahan, M. (1993) Explanation in the Situation Calculus. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, 160–165.
32. Shanahan, M. (1997) Event Calculus Planning Revisited. In *Proceedings 1997 European Conference on Planning (ECP 97), Springer-Verlag Lecture Notes in Artificial Intelligence (no. 1348)*, 390—402.
33. Thielscher, M. (1997) A Theory of Dynamic Diagnosis. In *Linköping Electronic Articles in Computer and Information Science* **2**(11). Research article received for discussion October, 1997. Revised March, 1998. http://www.ep.liu.se/ea/cis/1997/011/.
34. Waldinger, R. (1977) Achieving Several Goals Simultaneously. In Elcock, E., and Michie, D., eds., *Machine Intelligence 8.* Edinburgh, Scotland: Ellis Horwood. 94–136.