Representing Actions and State Constraints in Model-Based Diagnosis[*] [†]

Sheila A. McIlraith
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94301
mcilrait@parc.xerox.com

Knowledge Systems Laboratory
Stanford University
Stanford, CA 94305-9020
sam@ksl.stanford.edu

Abstract

In this paper we examine an important set of representation issues which have not been addressed by the model-based diagnosis community. In particular, we examine the problem of integrating a model-based diagnosis system description, SD, with a theory of action to parsimoniously represent the effect of actions on a system and the effects of system state on performing actions in the world. We employ the situation calculus, a first-order language, as our representation language. In the context of the situation calculus, SD presents an, often complex, set of state constraints. These state constraints implicitly define indirect effects of actions as well as indirectly imposing further preconditions on the performance of actions. As a consequence, SD presents further complications to addressing the frame, ramification and qualification problems. For the purposes of this paper, we examine a syntactically restricted SD, which commonly occurs in the axiomatization of model-based diagnosis domains. The contributions of this paper include: 1) a framework for integrating SD and a theory of action. 2) a procedure for compiling SD into a set of successor state axioms. These axioms capture the intended interpretation of SD, while providing a closed-form solution to the frame and ramification problems.

Introduction

Of recent years, a number of researchers have argued that diagnostic problem solving (DPS) is purposive in nature, that in some instances, identifying candidate diagnoses is only relevant to the extent that it enables an agent to act — to execute a test, to repair a system, to control it, or perhaps to invoke a contingency plan. From this viewpoint, we claim that a comprehensive account of DPS must involve reasoning about action and change (McIlraith 1997).

It is widely acknowledged that providing an accurate representation of the behaviour of an electro-mechanical device or physical system is one of the most challenging aspects of diagnostic problem solving (Hamscher, Console, & de Kleer 1992). Indeed, any form of model-based reasoning is only as good as the model it employs. In this paper, we examine the problem of integrating a model-based diagnosis system description, SD (de Kleer, Mackworth, & Reiter 1992) with a theory of action, to parsimoniously represent the effect of actions on a system and the effects of a system on performing actions in the world. We employ the situation calculus (McCarthy 1968) as our representation language for action. In the context of the situation calculus, SD presents an, often complex, set of state constraints. These state constraints implicitly define indirect effects of actions as well as indirectly imposing further preconditions on performing an action. Consequently, integrating SD and a theory of action requires us to address the frame problem – identifying and parsimoniously representing the situation invariants, the ramification problem – identifying the implicit effects of actions, and the qualification problem – identifying the conditions underwhich an action is possible.

We begin our paper with an overview of the situation calculus. Next, we describe a method for representing a DPS domain in the situation calculus through a straightforward transformation of SD, followed by the definition of action-related axioms. The axiomatization is illustrated via a power plant example. We adopt the view (e.g., (Reiter 1991)) that successor state axioms and action precondition axioms provide an attractive solution to the frame and ramification problems, and the qualification problem, respectively, because they are parsimonious, axiomatic and monotonic. Nevertheless, we also show that previous solutions to the frame and ramification problems are not sufficiently discriminating to capture the *intended interpretation* of our domain axiomatization. The subsection entitled "A Closed-Form Solution", describes our proposal for a closed-form solution to the frame and ramification problems for syntactically restricted state constraints, which occur commonly in DPS SD's. The solution comprises a simple syntactic manipulation which compiles our DPS axiomatization into a set of successor state axioms, capturing the intended interpretation of our domain. We subsequently augment this compilation with an existing solution to the qualification problem. In the final sections we provide a brief discussion of our representation, outlining its use in achieving various DPS tasks, and contrasting it to related work.

[*] This work was carried out while the author was a doctoral student at the University of Toronto, Canada.

Situation Calculus Overview

The situation calculus language we employ to 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 (Lin & Reiter 1994). 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, $do(tn\_on\_pmp, S_0)$ 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., $Poss(tn\_on\_pmp, S_0)$). 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 property that the pump is on in situation s could be represented by the fluent $on(Pmp, s)$.

The situation calculus language we employ in this paper is restricted to primitive, determinate actions. Our language does not include a representation of time, concurrency, or complex actions, but we believe the results presented herein can be extended to more expressive languages.

Axiomatizing a DPS Domain

The axiomatization of a system comprises both domain-independent and domain-specific axioms. The domain-independent axioms are to be the foundational axioms of the discrete situation calculus, Σ_{found} (Lin & Reiter 1994). They define the branching structure of our situation tree. The domain-specific axioms, must specify both the *behaviour of the static system*, and the *actions* that can affect the state of the system, as well as those actions required to achieve testing and repair.

We take as our starting point the extensive research on model-based diagnosis (MBD) (Hamscher, Console, & de Kleer 1992) and assume a system description, SD. Our task is to provide an axiomatization that integrates this SD with a domain action theory. Our domain action theory is described in terms of situation calculus effect axioms, unique names axioms, and necessary conditions for actions (e.g., (Reiter 1991)). In the rest of this section we describe a straightforward four step procedure to axiomatize a DPS domain. In the section that follows, we provide a procedure for automatically transforming these axioms into a final axiomatization which addresses the frame, ramification and qualification problems. For the purposes of this paper, we restrict our attention to systems that are inherently static in nature but whose behaviour can change as the result of an action performed by an agent[1].

Illustrative Example

The results in this paper are illustrated in terms of a simplified power plant feedwater system drawn from (Kramer *et al.* 1996). The system consists of three potentially malfunctioning components: a power supply (Pwr); a pump (Pmp); and a boiler (Blr). The power supply provides power to both the pump and the boiler. The pump fills the header with water, (wat_ent_hdr), which in turn provides water to the boiler, producing steam. Alternately, the header can be filled manually (mnl_fill). To make the example more interesting, we take liberty with the functioning of the actual system and assume that once water is entering the header, a siphon is created. Water will only stop entering the header when the siphon is stopped. The system also contains lights and an alarm.

Example: We commence with a system description, SD:

$$\neg AB(Pwr) \wedge \neg AB(Pmp) \wedge on(Pmp) \supset wat\_ent\_hdr \quad (1)$$

$$mnl\_fill \supset wat\_ent\_hdr \quad (2)$$

$$\ldots$$

Axiom (1) states that if the power and pump are normal and if the pump is on, then water will be entering the header.

Axiomatization Procedure

Step 1. Transform SD into a set of situation calculus state constraints, T_{SC} by indexing any predicate that can change as the result of an action with a situation term s.

Example: Axiom (1) above becomes:

$$\neg AB(Pwr, s) \wedge \neg AB(Pmp, s) \wedge on(Pmp, s) \supset wat\_ent\_hdr(s).$$

Step 2. Distinguish the state constraints, T_{SC} into:

- T_{ram}, the set of ramification constraints.
- T_{qual}, the set of qualification constraints.
- T_{domain}, those state constraints that are neither ramification nor qualification constraints.

While we can provide no provably correct method for automatically differentiating the axioms of T_{SC}, experience has provided the following intuitions.

- Axioms that are causal or definitional in nature belong in T_{ram}. In MBD terminology, these would include typical fault model axioms of SD as well as axioms of SD that describe the correct behaviour of a system (e.g., (de Kleer, Mackworth, & Reiter 1992)). Such axioms are often characterized syntactically by inclusion of an implication sign, e.g.,
$$A_1 \wedge A_2 \wedge \ldots \wedge A_k \supset A_j,$$
where each A_i is a literal with or without a situation term.
- The physical impossibility axioms of SD (Friedrich, Gottlob, & Nejdl 1990), which describe physically impossible states, should be included in T_{qual}. Physical impossibility axioms are often characterized syntactically as a negated conjunction of literals, e.g.,
$$\neg(A_1 \wedge A_2 \wedge \ldots \wedge A_k).$$

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

Example: The static behaviour of such a feedwater system can be represented by the following sets of axioms composing T_{SC}. T_{ram} is as follows:

$$\neg AB(Pwr,s) \wedge \neg AB(Pmp,s) \wedge on(Pmp,s) \supset wat\_ent\_hdr(s) \quad (3)$$

$$mnl\_fill(s) \supset wat\_ent\_hdr(s) \quad (4)$$

$$AB(Pwr,s) \supset lights\_out(s) \quad (5)$$

$$\neg AB(Pwr,s) \supset \neg lights\_out(s) \quad (6)$$

$$wat\_ent\_hdr(s) \wedge \neg AB(Pwr,s) \wedge \neg AB(Blr,s) \wedge on(Blr,s) \supset steam(s) \quad (7)$$

$$\neg\,(wat\_ent\_hdr(s) \wedge \neg AB(Pwr,s) \wedge \neg AB(Blr,s) \wedge on(Blr,s)) \supset \neg steam(s) \quad (8)$$

$$\neg wat\_ent\_hdr(s) \wedge on(Blr,s) \supset alarm(s) \quad (9)$$

$$AB(Blr,s) \supset alarm(s). \quad (10)$$

T_{qual} is as follows:

$$\neg(on(Pmp,s) \wedge mnl\_fill(s)). \quad (11)$$

T_{domain} is as follows:

$$Pwr \neq Pmp \neq Blr. \quad (12)$$

Step 3. Identify the actions that can affect the system or that are required for testing, repairing, and reacting. Axiomatize them as effect axioms,T_{ef}; necessary conditions for actions, T_{nec}; and unique names for actions T_{UNA}.

Step 3a. T_{ef}, the set of **positive and negative effect axioms**. These describe the changes in the truth values of fluents as a result of performing actions. For each fluent F,

$$Poss(a,s) \wedge \gamma_F^+(\vec{x},a,s) \supset F(\vec{x},do(a,s)) \quad (13)$$

$$Poss(a,s) \wedge \gamma_F^-(\vec{x},a,s) \supset \neg F(\vec{x},do(a,s)) \quad (14)$$

where $\gamma_F^+(\vec{x},a,s)$ and $\gamma_F^-(\vec{x},a,s)$ are simple formulas[2] whose free variables are among $\vec{x}, a, s$.

Example: The following axioms compose T_{ef}.

$$Poss(a,s) \wedge a = tn\_on\_pmp \supset on(Pmp,do(a,s)) \quad (15)$$

$$Poss(a,s) \wedge a = tn\_off\_pmp \supset \neg on(Pmp,do(a,s)) \quad (16)$$

$$Poss(a,s) \wedge a = tn\_on\_blr \supset on(Blr,do(a,s)) \quad (17)$$

$$Poss(a,s) \wedge a = tn\_off\_blr \supset \neg on(Blr,do(a,s)) \quad (18)$$

$$Poss(a,s) \wedge a = pwr\_fail \supset AB(Pwr,do(a,s)) \quad (19)$$

$$Poss(a,s) \wedge a = aux\_pwr \supset \neg AB(Pwr,do(a,s)) \quad (20)$$

$$Poss(a,s) \wedge a = pwr\_fix \supset \neg AB(Pwr,do(a,s)) \quad (21)$$

$$Poss(a,s) \wedge a = pmp\_burn\_out \supset AB(Pmp,do(a,s)) \quad (22)$$

$$Poss(a,s) \wedge a = pmp\_fix \supset \neg AB(Pmp,do(a,s)) \quad (23)$$

$$Poss(a,s) \wedge a = blr\_blow \supset AB(Blr,do(a,s)) \quad (24)$$

$$Poss(a,s) \wedge a = blr\_fix \supset \neg AB(Blr,do(a,s)) \quad (25)$$

$$Poss(a,s) \wedge a = tn\_on\_mnl\_fill \supset mnl\_fill(do(a,s)) \quad (26)$$

$$Poss(a,s) \wedge a = tn\_off\_mnl\_fill \supset \neg mnl\_fill(do(a,s)) \quad (27)$$

$$Poss(a,s) \wedge a = stp\_siphon \supset \neg wat\_ent\_hdr(do(a,s)) \quad (28)$$

$$Poss(a,s) \wedge a = tn\_on\_alarm \supset alarm(do(a,s)) \quad (29)$$

$$Poss(a,s) \wedge a = tn\_off\_alarm \supset \neg alarm(do(a,s)) \quad (30)$$

[2] A simple formula with respect to s is one in which only domain specific predicate symbols are mentioned (i.e., they do not mention $Poss$ or $<$), in which fluents do not include the function symbol do, in which there is no quantification over sort $situations$, and in which there is at most one free $situations$ variable.

Step 3b. T_{nec}, the set of axioms representing the **necessary conditions actions** to be performed. For each action prototype, A,

$$Poss(A(\vec{x}),s) \supset \pi_A^i \quad (31)$$

where π_A^i is a simple formula with respect to s, whose free variables are among $\vec{x}, s$.

Example: The following axioms compose *some* of T_{nec}.

$$Poss(tn\_on\_pmp,s) \quad (32)$$

$$\ldots$$

$$Poss(tn\_on\_mnl\_fill,s) \supset \neg alarm(s) \quad (33)$$

Axiom (33) states that if it is possible to turn on the manual filling then the alarm must be off.

Step 3c. T_{UNA}, a set of **unique names axioms for actions**. They state that identical actions have identical arguments, and every action name refers to a distinct action. For each different action prototype A and A',

$$A(x_1,\ldots,x_n) = A(y_1,\ldots,y_n) \supset x_1 = y_1 \wedge \ldots \wedge x_n = y_n \quad (34)$$

$$A(x_1,\ldots,x_n) \neq A'(x_1,\ldots,x_m) \quad (35)$$

Example: The following axioms compose *some* of T_{UNA}.

$$tn\_on\_pmp \neq tn\_off\_pmp \neq \ldots \neq tn\_off\_alarm \quad (36)$$

Step 4. Provide what is known of the initial state, T_{S_0}.

Example: The following axioms might compose T_{S_0}.

$$\neg AB(Pwr,S_0) \wedge \neg mnl\_fill(S_0) \wedge \neg AB(Pmp,S_0) \quad (37)$$

$$\neg wat\_ent\_hdr(S_0) \wedge \neg on(Blr,S_0) \quad (38)$$

$$\neg on(Pmp,S_0) \wedge \neg AB(Blr,S_0) \quad (39)$$

The Frame and Ramification Problems

In the previous section, we axiomatized a DPS domain. The resultant theory comprises the following sets of axioms:

$$T_{SC} \cup T_{ef} \cup T_{nec} \cup T_{UNA} \cup T_{S_0}. \quad (40)$$

The job of the axiomatizer is done, but unfortunately, these axioms do not provide a solution to the frame, ramification and qualification problems. In this section, we propose a solution to the frame and ramification problems for a typical class of DPS theories. The qualification problem is discussed in a subsequent section.

(Lin & Reiter 1994) gave a semantic definition for a solution to the frame and ramification problems using circumscription and minimal model semantics. This solution has its limitations. Sometimes there is no minimal model. In other cases, there are multiple minimal models, some of which do not reflect the intended interpretation of the ramification and effect axioms. Most importantly, there is no guaranteed procedure to produce a closed-form solution.

Our contribution is to provide an automatic procedure for generating a closed-form solution to the frame and ramification problems for a class of state constraints that is common to DPS domains. This solution is distinguished because it captures the *intended* interpretation of T_{SC} with respect to the theory of actions.

The Problem

We illustrate our problem with a subset of the feedwater system example. Consider the ramification constraints, (3) and (4) above. The effect axioms, necessary conditions for actions and initial conditions are as defined in the previous section. Assume for the sake of simplicity that $Poss(a, s)$, i.e., that all actions are possible in all situations.
Assume the action tn_on_pmp is performed in S_0, resulting in the situation $S_1 = do(tn\_on\_pmp, S_0)$. From effect axiom (15), we infer that $on(Pmp, S_1)$. What do our ramification constraints tell us about the indirect effect of this action? Under Lin and Reiter's minimization policy to maximize persistence, three minimal models[3] are apparent.

$\mathcal{M}_1 : \{\neg AB(Pwr, S_1), \neg AB(Pmp, S_1), wat\_ent\_hdr(S_1)\}$
$\mathcal{M}_2 : \{AB(Pwr, S_1), \neg AB(Pmp, S_1), \neg wat\_ent\_hdr(S_1)\}$
$\mathcal{M}_3 : \{\neg AB(Pwr, S_1), AB(Pmp, S_1), \neg wat\_ent\_hdr(S_1)\}$

Clearly, the intended model is $\mathcal{M}_1$. Turning on the pump results in water entering the header. It does not result in an abnormal power supply, or an abnormal pump. We intuitively know that this is the intended model, because we have a basic understanding of machinery. More importantly, the axiomatizer has communicated the intended interpretation through the syntactic form of the ramification constraints.

In the context of reasoning about action and change, state constraints serve two purposes. On the one hand, they define consistent states of our system, and the world. In this role, state constraints have traditionally been used to generate model-based diagnoses. In the context of a theory of action and change, state constraints have an additional role. They also serve as ramification and qualification constraints, indirectly constraining the effects of actions and further constraining the preconditions for actions.

When employing the ramification constraints to infer the indirect effects of actions, the implication connective is interpreted as causal or *definitional*, in the logic programming sense. Following (Levi 1994), we say that a fluent is **defined** in an axiom or set of axioms if it appears on the right-hand side of an implication connective in that axiom or set of axioms. Thus, it follows that an effect axiom for fluent F also serves to define fluent F.

If we assume that a fluent only changes value according to the effect axioms and the ramification constraints that *define* it, then the ramification constraints above only provide information about changes in the truth value of fluent $wat\_ent\_hdr(s)$. With this assumption, we can conclude that the consequence of performing tn_on_pmp in S_0 is captured by model $\mathcal{M}_1$.

In the section to follow, we use the intuition above to generate successor state axioms that reflect the intended interpretation of the ramification constraints and effect axioms, for a syntactically restricted class of theories.

A Closed-Form Solution

In this section we provide a closed-form solution to the frame and ramification problems for axiomatizations whose syntactic representation of ramification constraints and effect axioms, collectively form a *solitary stratified theory*.

We combine the notion of solitary theory (Lifschitz 1985) and stratified logic program (e.g., (Levi 1994)) to define the notion of a solitary stratified theory. Note that unlike stratified logic programs, we use a strictly $<$ relation to distinguish the strata of our theories. Intuitively, a solitary stratified theory is a stratified logic program that allows negation in the consequent. If such a theory were represented as a dependency graph, the graph would have no cycles. The stratification of a solitary stratified theory need not be unique and we could write a procedure to determine a stratification automatically.

Definition 1 (Solitary Stratified Theory)
Suppose T is a theory in the language of the situation calculus with domain fluents, $\mathcal{L}$. Then T is a solitary stratified theory with ***stratification*** *$(T_1, T_2, \ldots, T_n)$, and* ***partition*** *$(\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_n)$ if,*

- *for $i = 1, \ldots, n$, $\mathcal{L}_i$ is the set of fluents F_i that are defined in stratum T_i, and $\mathcal{L}_1 \cup \mathcal{L}_2 \cup \ldots \cup \mathcal{L}_n = \mathcal{L}$, and*
- *T is the union $T_1 \cup T_2 \cup \ldots T_n$ of sets of axioms T_i where for each stratum, T_i is solitary with respect to $\mathcal{L}_i$; i.e., each T_i can be written as the union $(\mathcal{D}_i \leq \neg\mathcal{L}_i) \cup (\mathcal{E}_i \leq \mathcal{L}_i)$,*
  1. *$\mathcal{L}_i$ is the set of fluents, F_i such that $[\neg]F_i$ is defined in T_i;*
  2. *$\mathcal{D}_i \leq \neg\mathcal{L}_i$, is a set of formulae of the form $(D_i \supset \neg F_i)$, – at most one for each fluent $F_i \in \mathcal{L}_i$. Each D_i is a formula containing no fluents drawn from $\mathcal{L}_i \cup \ldots \cup \mathcal{L}_n$.*
  3. *$\mathcal{E}_i \leq \mathcal{L}_i$, is a set of formulae of the form $(E_i \supset F_i)$, – at most one for each fluent $F_i \in \mathcal{L}_i$. Each E_i is a formula containing no fluents drawn from $\mathcal{L}_i \cup \ldots \cup \mathcal{L}_n$.*

Example: In our feedwater example, $T = T_{ram} \cup T_{ef}$ is a solitary stratified theory with stratification (T_1, T_2, T_3).

- T_1 comprises Effect Axioms (15) – (27),
- T_2 comprises Ramification Constraints (3) – (8), and Effect Axiom (28).
- T_3 comprises Ramification Constraints (9) – (10), and Effect Axioms (29) and (30).

In what follows, we define a seven step syntactic manipulation procedure which results in a closed-form solution to the frame and ramification problems for solitary stratified theory $T = T_{ef} \cup T_{ram}$. The solution is predicated on an appeal to a completeness assumption which enables us to generate explanation closure axioms.

Transformation Procedure

Let $T = T_{ram} \cup T_{ef}$ be a solitary stratified theory, with stratification $(T_1, T_2, \ldots, T_n)$.

Step 1. For every fluent F_i defined in an effect axioms of T_i, generate general positive and negative effect axioms, in the form of axioms (13) and (14) above.

Step 2. For every fluent F_i defined in a ramification constraint of T_i, generate **general positive and negative ramification axioms**, relativized to situation $(do(a, s))$.

[3] We only list the relevant portion of the models here.

$$v^+_{F_i}(do(a,s)) \supset F_i(do(a,s))^4 \quad (41)$$

$$v^-_{F_i}(do(a,s)) \supset \neg F_i(do(a,s)) \quad (42)$$

$v^+_{F_i}(do(a,s))$ and $v^-_{F_i}(do(a,s))$ are formulae whose free variables are among a, s, and any state or action arguments.

Step 3. Combine the above sets of axioms, to define **extended positive and negative effect axioms**, at most one for every fluent F_i.

$$Poss(a,s) \wedge (\gamma^+_{F_i}(a,s) \vee v^+_{F_i}(do(a,s))) \supset F_i(do(a,s)) \quad (43)$$

$$Poss(a,s) \wedge (\gamma^-_{F_i}(a,s) \vee v^-_{F_i}(do(a,s))) \supset \neg F_i(do(a,s)) \quad (44)$$

Example: Extended positive and negative effect axioms for the fluent $(on(Pmp, do(a,s))$, defined in T_1.

$$Poss(a,s) \wedge a = tn\_on\_pmp \supset on(Pmp, do(a,s)) \quad (45)$$

$$Poss(a,s) \wedge a = tn\_off\_pmp \supset \neg on(Pmp, do(a,s)) \quad (46)$$

For the fluent $wat\_ent\_hdr(do(a,s))$, defined in T_2.

$$(\neg AB(Pwr, do(a,s)) \wedge \neg AB(Pmp, do(a,s)) \wedge on(Pmp, do(a,s))) \vee mnl\_fill(do(a,s)) \supset wat\_ent\_hdr(do(a,s)) \quad (47)$$

$$Poss(a,s) \wedge a = stp\_siphon \supset \neg wat\_ent\_hdr(do(a,s)) \quad (48)$$

Step 4. Make the following **completeness assumption** regarding the effects and the ramifications.
All the conditions underwhich an action a can lead, directly or indirectly, to fluent F becoming true or false in the successor state are characterized in the extended positive and negative effect axioms for fluent F.

Step 5. From the completeness assumption, generate **explanation closure axioms**.

We argue that if action a is possible in s and if the truth value of fluent F_i changes from $true$ to $false$ upon doing action a in situation s, then either $\gamma^-_{F_i}(a,s)$ is $true$ or $v^-_{F_i}(do(a,s))$ is $true$. An analogous argument can be made when the truth value of fluent F changes from $false$ to $true$ upon doing action a in situation s. This assumption is captured in the following positive and negative explanation closure axioms. For every fluent F_i,

$$Poss(a,s) \wedge F_i(s) \wedge \neg F_i(do(a,s)) \supset \gamma^-_{F_i}(a,s) \vee v^-_{F_i}(do(a,s))$$

$$Poss(a,s) \wedge \neg F_i(s) \wedge F_i(do(a,s)) \supset \gamma^+_{F_i}(a,s) \vee v^+_{F_i}(do(a,s))$$

Step 6. From the extended positive and negative effect axioms and the explanation closure axioms, define **intermediate successor state axioms** for each fluent F_i.

We distinguish them as *intermediate* because, in the next step, we simplify them through a further syntactic transformation. For every fluent F_i,

$$Poss(a,s) \supset [F_i(do(a,s)) \equiv \Phi^*_{F_i}] \quad (49)$$

$$\Phi^*_{F_i} \equiv \gamma^+_{F_i}(a,s) \vee v^+_{F_i}(do(a,s)) \vee (F(s) \wedge \neg(\gamma^-_{F_i}(a,s) \vee v^-_{F_i}(do(a,s))))$$

The set of intermediate successor state axioms, $T_{ISS} = \bigcup_{i=1,\ldots,n} T_{ISS_i}$, where T_{ISS_i} is the set of axioms for every fluent $F_i \in \mathcal{L}_i$.

Example: Intermediate successor state axioms for fluents $on(Pmp, do(a,s))$ and $wat\_ent\_hdr(do(a,s))$ follow.

$$Poss(a,s) \supset [on(Pmp, do(a,s)) \equiv a = tn\_on\_pmp \vee (on(Pmp,s) \wedge a \neq tn\_off\_pmp)] \quad (50)$$

$$Poss(a,s) \supset [wat\_ent\_hdr(do(a,s)) \equiv mnl\_fill(do(a,s)) \vee (\neg AB(Pwr, do(a,s)) \wedge \neg AB(Pmp, do(a,s)) \wedge on(Pmp, do(a,s))) \vee wat\_ent\_hdr(s) \wedge a \neq stp\_siphon] \quad (51)$$

Step 7. By regressing[5] the intermediate successor state axioms, generate (final) **successor state axioms**. These axioms are simple formulae containing no reference to fluents indexed by the situation $do(a,s)$. For every fluent F_i,

$$Poss(a,s) \supset [F_i(do(a,s)) \equiv \Phi_{F_i}] \quad (52)$$

where Φ_{F_i} is the following simple formula.

$$\Phi_{F_i} \equiv \gamma^+_{F_i}(a,s) \vee \mathcal{R}^{i-1}_{SS}[v^+_{F_i}(do(a,s))] \vee (F(s) \wedge \neg(\gamma^-_{F_i}(a,s) \vee \mathcal{R}^{i-1}_{SS}[v^-_{F_i}(do(a,s))]))$$

where $\mathcal{R}^{i-1}_{SS}[\phi]$ is the regression of formula ϕ under successor state axioms $T_{SS_1}, \ldots, T_{SS_{i-1}}$.

The set of successor state axioms, $T_{SS} = \bigcup_{i=1,\ldots,n} T_{SS_i}$, where T_{SS_i} is the set of axioms for every fluent $F_i \in \mathcal{L}_i$.

Example: Transformation of intermediate successor state axiom (51) into its corresponding successor state axiom.

$$\begin{aligned} Poss(a,s) \supset [&wat\_ent\_hdr(do(a,s)) \equiv \\ &a = tn\_on\_mnl\_fill \\ &\vee (mnl\_fill(s) \wedge a \neq tn\_off\_mnl\_fill) \\ &\vee [(a \neq pwr\_fail \\ &\quad \wedge (\neg AB(Pwr,s) \vee a = aux\_pwr \\ &\quad\quad \vee a = pwr\_fix)) \\ &\quad \wedge (a \neq pmp\_burn\_out \\ &\quad\quad \wedge (\neg AB(Pmp,s) \vee a = pmp\_fix)) \\ &\quad \wedge (a = tn\_on\_pmp \\ &\quad\quad \vee (on(Pmp,s) \wedge a \neq tn\_off\_pmp))] \\ &\vee (wat\_ent\_hdr(s) \wedge a \neq stp\_siphon)] \end{aligned} \quad (53)$$

Our successor state axioms provide a closed-form solution to the frame and ramification problems. Since we have

[4] Henceforth, action and state arguments, $\vec{x}$ will not be explicitly represented in canonical formulae.

[5] Regression (e.g., (Waldinger 1977)) is a recursive rewriting procedure used here to reduce the nesting of the do function in situation terms. If F is a fluent with successor state axiom $Poss(a,s) \supset F(\vec{x}, do(a,s)) \equiv \Phi_F$ in T_{SS} then $\mathcal{R}_{SS}[F(t_1,\ldots,t_n, do(\alpha,\sigma))] = \Phi_F \,|^{x_1,\ldots,x_n,a,s}_{t_1,\ldots,t_n,\alpha,\sigma}$.

compiled T_{ef} and T_{ram} into T_{SS}, we can replace T_{ef} and T_{ram} by T_{SS} and $T^{S_0}_{ram}$ in (40). $T^{S_0}_{ram}$ is the set of ramification constraints, relativized to S_0. Note that our closed-form solution to the frame and ramification problem loosely appeals to a completeness assumption in order to generate explanation closure axioms. In (McIlraith 1997), we provide an independent semantic justification via prioritized circumscription. From those results we show that our solution is predicated on the following consistency condition. In particular, that

$$T_{UNA} \cup T_{nec} \models (\forall a, s).Poss(a, s) \supset \neg[(\gamma^+_{F_i}(a, s) \vee \mathcal{R}_{SS}[v^+_{F_i}(do(a, s))]) \wedge (\gamma^-_{F_i}(a, s) \vee \mathcal{R}_{SS}[v^-_{F_i}(do(a, s))])]. \quad (54)$$

This condition ensures that either an action is impossible, or if it is possible, that it is never the case that the direct effects or ramifications of an action(γ's and v's, respectively) can make a fluent both false and true in the same situation.

Qualification Problem

Our theory now provides a solution to the frame and ramification problems. It remains to address the qualification problem. As previously observed the qualification constraints in T_{qual} can further restrict those situations s in which an action a is $Poss$-ible. We propose to use the solution proposed by (Lin & Reiter 1994). It transforms the necessary conditions for actions, T_{nec} and the qualification constraints, T_{qual} into a set of action precondition axioms T_{AP}, one for each action prototype A of the domain. Following their results, we add one more step to our transformation procedure.

Step 8. Define one **action precondition axiom** for each action prototype A as follows.

$$Poss(A(\vec{x}), s) \equiv \Pi_A \wedge \bigwedge_{C \in T_{qual}} \Pi_C \quad (55)$$

$$\Pi_C \equiv \mathcal{R}_{SS}[C(do(A(\vec{x}), s))] \quad (56)$$

$\Pi_A \equiv \pi^1_A \vee \ldots \vee \pi^n_A$ for each π^i_A of (31) in T_{nec}. $\mathcal{R}_{SS}$ is the regression operator under the successor state axioms, T_{SS}.

Example: Consider (11) of T_{qual}, and (33) and (32) of T_{nec}, the action precondition axioms for $tn\_on\_mnl\_fill$ and tn_on_pmp are:

$$Poss(tn\_on\_mnl\_fill, s) \equiv \neg alarm(s) \wedge \neg on(Pmp, s) \quad (57)$$

$$Poss(tn\_on\_pmp, s) \equiv \neg mnl\_fill(s) \quad (58)$$

The action precondition axioms provide a closed-form solution to the qualification problem. Since we have compiled T_{nec} and T_{qual} into T_{AP}, we can replace T_{nec} and T_{qual} by T_{AP} and $T^{S_0}_{qual}$ in our theory. Lin and Reiter's solution also requires a domain closure axiom for actions, T_{DCA}.

Discussion

The results of the previous sections yield the following theory which integrates SD and a theory of action,

$$T_{UNA} \cup T_{DCA} \cup T_{SS} \cup T_{AP} \cup T_{S_0} \cup T^{S_0}_{SC} \cup T_{domain}.$$

This 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 signs by implication connectives. The Lloyd-Topor transformation (Lloyd 1987) must then be applied to convert this theory into Prolog clausal form.

The state constraints that play the role of ramification constraints with respect to our theory of actions are compiled into successor state axioms, one for every fluent in our theory. When state constraints are absent, as in the case of Reiter's solution to the frame problem (Reiter 1991), successor state axioms provide a parsimonious representation for frame and effect axioms. In the presence of ramification constraints, the successor state axioms can, under certain conditions, grow exceedingly long. This presents the problem of trying to find the best trade-off between precompilation and runtime computation; a problem that many AI researchers face, and one that is often best addressed with respect to the specific domain. Fortunately, in our case we have an ideal compromise in those cases where T_{SS} proves to be unwieldy, that is to employ the *intermediate* successor state axioms as our representation. The axioms in T_{ISS} capture the intended interpretation of our domain but are only partially compiled, and thus don't risk the length concerns associated with the axioms in T_{SS}. Further, T_{ISS} preserves the compositionality of our representation, which is a hallmark of model-based representations.

The purpose of this paper was to address the knowledge representation issues associated with integrating a DPS system description and a theory of action. In (McIlraith 1997) we used this representation to characterize the tasks of diagnosis, testing, and repair. Integrating a theory of action with SD provides for a broad definition of diagnosis. The traditional notions of consistency-based and abductive diagnosis map seemlessly into our representation framework, with the distinction that diagnoses are now relativized to a situation. Further we can employ actions both as observations to project what will be wrong with a system, and as diagnoses to explain what has happened to result in some observed behaviour. Computationally, many aspects of diagnosis, achieving tests, and achieving repairs are simply instances of the planning problem, and can be achieved through some combination of logical consequence finding, database progression, regression, theorem proving, and abductive planning techniques. That said, computing diagnoses *without* a representation of action is already computationally taxing. The real challenge is to exploit the benefit of our rich declarative representation and to approach DPS differently. We believe that part of the answer to this challenge lies in the purposive nature of DPS, and in exploiting our representation to generate and/or verify high-level control procedures that can in turn provide timely runtime response to our DPS problems.

Contributions and Related Work

This paper provides research contributions in two rather distinct areas: model-based diagnosis/qualitative reasoning, and knowledge representation/nonmonotonic reasoning. For the MBD community, this paper addresses an important representation issue, namely how to integrate SD with a theory of action. For the knowledge representation community, this paper contributes a semantically justified, closed-form solution to the frame, ramification and qualification problems, for a commonly occurring class of theories.

The author knows of no work in the diagnosis community that addresses the problem of integrating SD and a theory of action, save some preliminary work in (Forbus 1989), examining the problem of integrating actions and qualitative process theory. Forbus' *action-augmented envisionment*, ($\mathcal{AE}$) captures computationally some aspects of the intuition found in this paper, while neglecting to address a number of fundamental knowledge representation issues. It is interesting to note that the ATMS, the computational machinery that underlies the qualitative process engine, can provide a runtime mechanism for compiling and caching a relevant subset of the closed-form representation proposed here. While we do not discuss continuous systems in this paper, the work presented here provides a formal foundation for integrating continuous systems with discrete systems whose specification includes state constraints.

Within the knowledge representation community, related work is more abundant. The intuition behind our solution to the frame and ramification problems – the notion of interpreting our ramification constraints as definitional in nature, was influenced by research on the semantics of normal logic programs and deductive databases (e.g., (Przymusinski 1989)), and is related to preliminary work on this problem by Pinto (Pinto 1994). Indeed the spirit of this solution – the notion of imposing a *directional* interpretation on our implication connective in our ramification constraints, is akin to the intuition behind proposed solutions to the ramification problem that advocate minimizing an explicitly represented notion of causality (e.g., (Lin 1995), (McCain & Turner 1995), (Thielscher 1995), (Giunchiglia 1996)). Indeed the author suspects that for the syntactically restricted case studied here, all our different proposed solutions may produce the same results, just as many of the independent solutions to the frame problem prove to be identical under certain conditions. What distinguishes this work in particular is that it provides an axiomatic closed-form solution; it retains the dual role played by our state constraints; and finally it provides a solution (sometimes) to the general problem of integrating a theory of action with an existing set of state constraints.

Acknowledgements

I would like to thank Fangzhen Lin, Ray Reiter, Yves Lespérance, Hector Levesque and Allan Jepson (with particular thanks to Fangzhen and Ray) for their insightful comments on the work presented in this paper. I would also like to thank the diligent reviewers for providing a thorough and helpful review of this paper.

References

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

Forbus, K. 1989. Introducing Actions into Qualitative Simulation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1273–1278.

Friedrich, G.; Gottlob, G.; and Nejdl, W. 1990. Physical impossibility instead of fault models. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 331–336.

Giunchiglia, E. 1996. Determining ramifications in the situation calculus. In *Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning*, 76–86.

Hamscher, W.; Console, L.; and de Kleer, J. 1992. *Readings in Model-based Diagnosis*. Morgan Kaufmann.

Kramer, B.; Mylopolous, J.; Benjamin, M.; Chou, Q.; Elder, D.; and Opala, J. 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*, 172–186.

Levi, G. 1994. *Advances in Logic Programming Theory*. Oxford Science Publications.

Lifschitz, V. 1985. Computing Circumscription. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 121–127.

Lin, F., and Reiter, R. 1994. State constraints revisited. *Journal of Logic and Computation* 4(5):655–678. Special Issue on Action and Processes.

Lin, F. 1995. Embracing causality in specifying the indirect effects of actions. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1985–1991.

Lloyd, J. 1987. *Foundations of Logic Programming*. Springer Verlag, second edition.

McCain, N., and Turner, H. 1995. A causal theory of ramifications and qualifications. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1978–1984.

McCarthy, J. 1968. Programs with common sense. In Minsky, M., ed., *Semantic Information Processing*. The MIT Press. chapter 7, 403–418.

McIlraith, S. 1997. *Towards a Formal Account of Diagnostic Problem Solving*. Ph.D. Dissertation, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.

Pinto, J. 1994. *Temporal Reasoning in the Situation Calculus*. Ph.D. Dissertation, Department of Computer Science, University of Toronto, Toronto, Ontario, Canada.

Przymusinski, T. 1989. On the declarative and procedural semantics of logic programs. *Journal of Automated Reasoning* 5:167–205.

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.

Thielscher, M. 1995. Computing ramifications by postprocessing. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1994–2000.

Waldinger, R. 1977. Achieving several goals simultaneously. In Elcock, E., and Michie, D., eds., *Machine Intelligence 8*. Edinburgh, Scotland: Ellis Horwood. 94–136.