Determining Ramifications in the Situation Calculus

Enrico Giunchiglia
Mechanized Reasoning Group
DIST - Università di Genova
Via Opera Pia 13, 16145 Genova, Italy
Email: enrico@dist.unige.it

Abstract

We focus on the ramification problem in the setting of the situation calculus. Our analysis is restricted to theories (or domains) consisting of effect propositions and (ramification) constraints. For these domains, we first identify the class of models we are interested in characterizing. In the case of finite domains, we provide a set of second order formulas defining the models we are interested in, and show how sometimes such formulas can be reduced to a first order theory. When a domain (in the situation calculus) "corresponds" to a domain description in the high level action description language $\mathcal{AR}$, we show that these two formalizations yield the same conclusions.

1 INTRODUCTION

The "frame problem" —how to represent succinctly what remains unchanged as a result of performing an action— has long been a challenge for many researchers in Artificial Intelligence. Several solutions have been proposed (see e.g. [Pednault, 1989; Baker, 1991; Reiter, 1991; Gelfond and Lifschitz, 1992; Kartha and Lifschitz, 1995]), some of these proposals have been formally compared [Kartha, 1993], and there is the feeling that at least for domains without ramification constraints the frame problem is solved.

In this paper we address the frame problem in presence of ramification constraints. We also assume that actions may not be executable, e.g. because their effects violate some constraint. Our analysis is restricted to domains consisting of effect propositions and ramification constraints.

The path we follow is the following. We first identify the "intended models" we want to characterize (Section 2). We thus introduce the notion of model which is "minimally abnormal" and in which "actions are maximally executable". Second, (Section 3), we try to provide a syntactic characterization for these classes of models. In the case of finite domains, we provide two second order formulas each defining one of the classes of models we are interested in. As a third step, we see under which circumstances we can reduce the second order formulas to first order equivalent ones (Section 4). Finally (Section 5), we show that the class of intended models of a domain in the situation calculus entails exactly the same conclusions entailed by the corresponding domain description in a high level action language like $\mathcal{A}$ [Gelfond and Lifschitz, 1992], $\mathcal{AR}_0$ [Kartha and Lifschitz, 1994] or $\mathcal{AR}$ [Giunchiglia *et al.*, 1996].

The novelties and motivations of each of the above four steps are the following. Definitions similar to our definition of "minimally abnormal model" have already been proposed in the literature [Lin and Shoham, 1991; Lin and Reiter, 1994]. One difference is that Lin, Reiter and Shoham restrict their attention to languages with propositional fluents. On the other hand, nonpropositional fluents allow for more natural formulations of many scenarios and their treatment is not as straightforward as it could be expected (see e.g. [Giunchiglia *et al.*, 1995]). The definition of model with maximally executable actions (as far as we know) has never been proposed in the literature. Singling out these two characterizations is important for the goal of building "provably correct theories of actions" in the spirit of [Lin and Shoham, 1991].

The closest work to our second order characterization of the class of minimally abnormal models seems [Kartha and Lifschitz, 1995]. Both approaches are based on the "theory update" view of [Winslett, 1988]. One difference is that the set of domains we consider is wider. In particular, we allow for nonpropositional fluents and arbitrary formulas (and not just atomic) in the post-condition part of effect propositions. Another difference is that in [Kartha and Lifschitz, 1995], a Nested Abnormality Theory (NAT) [Lifschitz, 1995] is used, while we propose a direct encoding in second order logic. Though the former seems more elegant, it requires at least one more second order quantifier (which have to be eliminated if we want to find a first order reduction). Another approach for such a characterization, is Baker's [Baker, 1991] (but see also [Kartha, 1994; Kartha, 1996]). The biggest drawback that we see in Baker's approach is the presence of higher order circumscription for the existence of situations. For these reasons, the approaches of [Kartha and Lifschitz, 1995] and [Baker, 1991] do not seem to allow for first order reductions as straightforward as ours. The second order characterization of the class of models in which actions are maximally executable (as far as we know) has never been proposed for the set of domains we consider.

The reduction to first order seems to be of particular interest from the perspective of implementing these theories. Recently, there seems to be a growing interest in implementing nonmonotonic reasoning systems (see e.g. [IJCAI, 1995; ECAI, 1996]). As far as circumscription is involved, a lot of efforts are put in mechanizing systems for the elimination of second order quantifiers (see for example [Gabbay and Ohlbach, 1992; Kartha and Lifschitz, 1995; Doherty *et al.*,]). Having a direct encoding in first order logic eliminates the need for this complex step. Furthermore, the set of domains for which the reduction is possible is fairly big. For example, it includes the domains for which Lin and Reiter [Reiter, 1991; Lin and Reiter, 1994] propose a simple solution for the characterization of a class of minimally abnormal models. For such a characterization, Lin and Reiter impose certain consistency conditions and a second order formula ensuring a "tree structure" for the universe of situations. Our solution does need neither of them. Furthermore the reduction to first order is possible for any domain without terms denoting objects, e.g. the domain corresponding to Example 3 of [Kartha and Lifschitz, 1995]. That example involves a "ternary state constraint" (in the sense of Section 3.1 of [Pinto, 1995]). Thus for that domain it is not possible to compute the successor state axiom following the methods developed in [Pinto, 1995].

Finally, the last step is interesting in two ways. First, it supports our definition of "intended model". When for a domain D there is a corresponding domain description $\mathcal{D}$ in $\mathcal{AR}$, the consequences of $\mathcal{D}$ are exactly those entailed by the class of intended models of D. Second, we give a sound and complete translation of *any* domain description in $\mathcal{AR}$ into first order logic. This translation is also applicable to domain descriptions which do not have a counter-part in the domains we consider in Section 3. For example, it is applicable to domain descriptions in which some of the fluents are not inertial (or not in the "frame" [Lifschitz, 1990]) or can nondeterministically change their values.

2 A SEMANTIC SOLUTION

In this paper, all free variables in a formula are considered implicitly universally quantified from the outside. We also assume that bound variables are renamed in such a way that any two quantifiers one in the scope of the other bind distinct variables. By $\phi(z)$ we indicate a formula whose free variables are in z.

We are using a many sorted language with sorts for situations (*situation*), actions (*action*), objects (*object*) and values (*value*). We denote the variables for situations, actions, values and tuples of objects with $s, s', s'', s_1, \ldots, a, a_1, \ldots, v, w, v_1, w_1, \ldots$ and $y, x, x_1, x_2, \ldots$ respectively. As usual, we assume that the sorts be pairwise disjoint and that for each sort S, an equality symbol $=_S$ belongs to the language. We usually drop the S and simply write $=$ for $=_S$.

The only symbols (other than equality) involving the sort *situation* are:

- a set of fluent symbols, denoted with $F, F_1, F_2, \ldots$ Formally, a *fluent symbol* F is a function symbol F: $object^n \times situation \rightarrow value$,
- the binary function do: $action \times situation \rightarrow situation$,
- the constants $\perp$ and S_0, for the undefined and initial situation respectively.

The domains we consider are characterized by a set of

1. *ramification constraints*, (or *constraints*) of the form
$$\Upsilon(s) \tag{1}$$
where s is the only situation term occurring in Υ, and
2. *effect propositions*, of the form
$$do(a, s) = s' \wedge s' \neq \perp \supset \forall x(\Phi(x, a, s) \supset \Psi(x, a, s')) \tag{2}$$
where s [resp. s'] is the only situation term occurring in Φ [resp. Ψ].

Formally, a *domain* consists of the formula

$$S_0 \neq \perp, \tag{3}$$

effect propositions and constraints. Equation (3) is required for ruling out from the class of models those structures whose situation universe is a singleton (and thus in which all the ground situation terms are equal to the undefined situation).

A domain D is *propositional* if the constraint

$$\neg v = \mathsf{T} \equiv v = \mathsf{F} \tag{4}$$

belongs to D. When considering propositional domains, we abbreviate any atomic formula

$$F(\chi, \xi) = \mathsf{T}.$$

with $F(\chi, \xi)$. In writing domains, we usually omit (3) and, in the cases of propositional domains, we omit also (4). With these conventions, the "murder mystery" [Baker, 1991] can be formalized by the propositional domain:

$$\begin{array}{l} do(a,s) = s' \wedge s' \neq \perp \supset (a = Load \supset Loaded(s')), \\ do(a,s) = s' \wedge s' \neq \perp \supset \\ \qquad (a = Shoot \wedge Loaded(s) \supset \neg Alive(s')), \\ do(a,s) = s' \wedge s' \neq \perp \supset (a = Shoot \supset \neg Loaded(s')), \\ Walking(s) \supset Alive(s), \\ Load \neq Wait, Load \neq Shoot, Wait \neq Shoot. \end{array} \tag{5}$$

In the above domain, the first three formulas are effect propositions. All the other formulas are constraints.

Given (5), we are interested in characterizing the models satisfying, e.g.

$$\neg Alive(do(Shoot, do(Wait, do(Load, S_0)))) \tag{6}$$

(i.e. the person is not alive after we load the gun, wait and then shoot), and

$$do(Shoot, do(Wait, do(Load, S_0))) \neq \perp \tag{7}$$

(i.e. it is possible to load the gun, wait and finally shoot).

The intended models of a domain are those which are "minimally abnormal" and in which "actions are maximally executable", in the sense made precise by the following two definitions.

In the following, we indicate the interpretation of a symbol K in a structure M with $M[\![K]\!]$. We also extend the signature of the language by adding to it, for every element k in a universe of M, a new object constant k^* of the appropriate sort. The interpretation of the new symbols is defined by $M[\![k^*]\!] = k$.

Definition 1 *A model M of a domain D is minimally abnormal if there does not exist a model M' of D such that*

1. *M and M' have the same universes.*
2. *M and M' differ only in the interpretation of fluent symbols.*
3. *There is a ground situation term ξ and a ground action term α such that*
   - *For any ground fluent $F(\chi, \xi)$,*
     $$M'[\![F(\chi, \xi)]\!] = M[\![F(\chi, \xi)]\!].$$
   - $M'[\![do(\alpha, \xi)]\!] = M[\![do(\alpha, \xi)]\!] \neq M[\![\perp]\!]$.
   - *For any ground fluent $F(\chi, do(\alpha, \xi))$, either*
     $$M'[\![F(\chi, do(\alpha, \xi))]\!] = M[\![F(\chi, do(\alpha, \xi))]\!],$$
     or
     $$M'[\![F(\chi, do(\alpha, \xi))]\!] = M[\![F(\chi, \xi)]\!].$$
   - *There exists a ground fluent $F(\chi, do(\alpha, \xi))$ such that*
     $$M'[\![F(\chi, do(\alpha, \xi))]\!] \neq M[\![F(\chi, do(\alpha, \xi))]\!].$$

However, the class of minimally abnormal model of domain (5) does entail neither (6) nor (7). In fact, such a class includes also models in which (7) is not satisfied. As a consequence, in these models, the effects of shooting are canceled.

Definition 2 *Let M be a model of a domain D. We say that actions are maximally executable in M if M satisfies*

$$s = \perp \supset do(a, s) = \perp, \tag{8}$$

and there does not exist a model M' of D satisfying (8) such that

1. *M and M' have the same universes for actions, objects and values. The universe for situations in M is a subset of the universe for situations in M'.*
2. *M and M' differ only in the interpretation of the fluent symbols and the function do.*
3. *There is a ground situation term ξ and a ground action term α such that*
   - *For any ground fluent $F(\chi, \xi)$,*
     $$M'[\![F(\chi, \xi)]\!] = M[\![F(\chi, \xi)]\!].$$
   - $M'[\![do(\alpha, \xi)]\!] \neq M[\![do(\alpha, \xi)]\!] = M[\![\perp]\!]$.

We say that a model M of a domain D is an *intended model* if M is minimally abnormal and actions are maximally executable in M. Of course, each intended model of domain (5) satisfies both (6) and (7).

3 A SECOND ORDER SOLUTION

We assume to have also function variables for fluent symbols. For each fluent symbol F, we denote with F' a fluent variable with the same domain as F. We also assume that $F_1, \ldots, F_n$ are all the fluent symbols in the language and that the domain D consists of a finite number of propositions. By $\bigwedge_D \Upsilon(s)$ [$\bigwedge_D \forall x(\Phi(x,a,s) \supset \Psi(x,a,s'))$, resp.] we mean the conjunction of the constraints [of the right hand side of the effect propositions, resp.] in D. $\phi[t_1/z_1, \ldots, t_m/z_m]$ is the expression obtained via simultaneous substitution of t_i for z_i ($1 \leq i \leq m$) in ϕ.

We begin with a second order characterization of the class of minimally abnormal models.

$$\begin{array}{l} do(a,s) = s' \wedge s' \neq \bot \supset \ \neg \exists F'_1 \ldots F'_n(\\ \quad \bigwedge_D \Upsilon(s)[s'/s, F'_1/F_1, \ldots, F'_n/F_n] \hfill \wedge \\ \quad \bigwedge_D \forall x(\Phi(x,a,s) \supset \Psi(x,a,s')[F'_1/F_1, \ldots, F'_n/F_n]) \wedge \\ \quad \bigwedge_{1 \leq i \leq n} (\forall x_i (F'_i(x_i, s') = F_i(x_i, s') \vee \\ \qquad\qquad F'_i(x_i, s') = F_i(x_i, s))) \hfill \wedge \\ \quad \bigvee_{1 \leq i \leq n} (\exists x_i F'_i(x_i, s') \neq F_i(x_i, s'))). \end{array} \tag{9}$$

Theorem 1 *Let M be a model of a domain D. M is minimally abnormal iff M satisfies (9).*

Intuitively, (9) imposes to consider only those models M in which, for each action α and situation ξ such that

$$M \models do(\alpha, \xi) \neq \bot,$$

it is not possible to find fluent symbols $F'_1, \ldots, F'_n$

- satisfying the conditions imposed by the constraints and the effect propositions (second and third lines of (9)); and
- for each tuple of objects χ_i,

  $$\begin{array}{l} M \models F'_i(\chi_i, do(\alpha,\xi)) = F_i(\chi_i, do(\alpha,\xi)) \vee \\ \qquad F'_i(\chi_i, do(\alpha,\xi)) = F_i(\chi_i, \xi), \end{array}$$

  with $1 \leq i \leq n$ (third and fourth line of (9)); and
- for at least one tuple of objects χ, either

  $$M \models F'_1(\chi, do(\alpha,\xi)) \neq F_1(\chi, do(\alpha,\xi)),$$

  or

  $$M \models F'_2(\chi, do(\alpha,\xi)) \neq F_2(\chi, do(\alpha,\xi)),$$

  ...
  or

  $$M \models F'_n(\chi, do(\alpha,\xi)) \neq F_n(\chi, do(\alpha,\xi))$$

  (last line of (9)).

The class of models in which actions are maximally executable is characterized by the following formula.

$$\begin{array}{l} do(a,s) = s' \supset (s' = \bot \equiv (s = \bot \vee \neg \exists F'_1 \ldots F'_n(\\ \quad \bigwedge_D \Upsilon(s)[s'/s, F'_1/F_1, \ldots, F'_n/F_n] \wedge \\ \quad \bigwedge_D \forall x(\Phi(x,a,s) \supset \Psi(x,a,s')[F'_1/F_1, \ldots, F'_n/F_n])))). \end{array} \tag{10}$$

Theorem 2 *Let M be a model of a domain D. Actions are maximally executable in M iff M satisfies (10).*

It can also be proved that a model M of a domain D satisfies (10) if and only if M satisfies (8) and

$$\begin{array}{l} do(a,s) = s' \wedge s' = \bot \supset (s = \bot \vee \neg \exists F'_1 \ldots F'_n(\\ \quad \bigwedge_D \Upsilon(s)[s'/s, F'_1/F_1, \ldots, F'_n/F_n] \wedge \\ \quad \bigwedge_D \forall x(\Phi(x,a,s) \supset \Psi(x,a,s')[F'_1/F_1, \ldots, F'_n/F_n]))). \end{array}$$

4 REDUCTION TO FIRST ORDER

In general, it is not possible to reduce (9) to an equivalent first order formula. The following example is from [Lin and Reiter, 1994].

Example 1 *Let P and R be ternary fluent symbols and A an action constant. Consider the propositional domain consisting of the effect proposition*

$$do(a,s) = s' \wedge s' \neq \bot \supset (a = A \wedge P(x,y,s) \supset R(x,y,s')),$$

and of the constraints

$$\begin{array}{l} R(x,x,s), \\ R(x,y,s) \supset R(y,x,s), \\ R(x,y,s) \wedge R(y,z,s) \supset R(x,z,s). \end{array}$$

It is not possible to characterize the set of minimally abnormal models of D with a first order theory. Otherwise, such a theory, together with the first order formula

$$do(A, S_0) \neq \bot \wedge \neg R(x,y,S_0)$$

would entail the transitive closure of $P(x,y,S_0)$ (which is not first order definable).

However, there are some special cases in which both (9) and (10) have first order equivalent formulas.

We say that a formula is *solitary in $F(x,s')$*[1] if it has the form

$$\exists y \forall x \phi(y,x,a,s,s')$$

where

[1] This terminology is suggested by a vague analogy with the corresponding concept in [Lifschitz, 1985].

- y and x are tuples (eventually empty) of object variables, and
- for each occurrence in $\phi(y,x,a,s,s')$ of a fluent $G(\chi,\xi)$, if ξ is s' then $G(\chi,\xi)$ is $F(x,s')$.

For example, the first, second and third propositions in (5) are solitary in $Loaded(s')$, $Alive(s')$ and $Loaded(s')$ respectively.

We say that a formula is *separable in* s' if it is a combination of conjunctions and disjunctions of solitary formulas in some fluent $F(x,s')$.

The formula $(Walking(s') = \mathsf{F} \vee Alive(s') = \mathsf{T})$ is separable in s'.

Theorem 3 *Let D be a domain such that for each constraint (1) and effect proposition (2), $\Upsilon(s)$ and $\forall x(\Phi(x,a,s) \supset \Psi(x,a,s'))$ are separable in s and s' respectively. Then both (9) and (10) have a first order equivalent formula.*

Consider e.g. (9). In the hypotheses of Theorem 3, we can rewrite

$$\begin{array}{ll}\exists F'_1 \ldots F'_n(& \\ \quad \bigwedge_D \Upsilon(s)[s'/s, F'_1/F_1, \ldots, F'_n/F_n] & \wedge \\ \quad \bigwedge_D \forall x(\Phi(x,a,s) \supset \Psi(x,a,s')[F'_1/F_1, \ldots, F'_n/F_n]) \wedge & \\ \quad \bigwedge_{1\leq i\leq n}(\forall x_i(F'_i(x_i,s') = F_i(x_i,s') \vee & \\ \qquad F'_i(x_i,s') = F_i(x_i,s))) & \wedge \\ \quad \bigvee_{1\leq i\leq n}(\exists x_i F'_i(x_i,s') \neq F_i(x_i,s'))). & \end{array}$$

in a form

$$\exists F'_1 \ldots F'_n \exists y \bigvee_i \bigwedge_{1\leq j\leq n} \forall x_j \phi_{ij}(y,x_j,a,s,s'), \qquad (11)$$

by first order manipulation. Furthermore, we can assume that each occurrence in $\phi_{ij}(y,x_j,a,s,s')$ of a fluent with a fluent variable has the form $F'_j(x_j,s')$.

Then, (11) can be rewritten as

$$\exists y \bigvee_i \bigwedge_{1\leq j\leq n} \exists F'_j \forall x_j s''(s'' = s' \supset \phi_{ij}(y,x_j,a,s,s')[s''/s'])$$

which can be reduced to a first order formula using the equivalence (see e.g. [Enderton, 1972])

$$\models \forall x s \exists v \phi(x,v,s) \equiv \exists F \forall x s \phi(x,v,s)[F(x,s)/v].$$

There are two important classes of domains meeting the conditions of Theorem 3 and thus for which we can find first order formulas equivalent to (9) and (10).

To the first class belong the propositional domains in which the constraints are without situation terms and the effect propositions have one of the following two forms

$$do(a,s) = s' \wedge s' \neq \bot \supset \forall x(P(x,a,s) \supset F(x,s')),$$
$$do(a,s) = s' \wedge s' \neq \bot \supset \forall x(Q(a,x,s) \supset \neg F(x,s')).$$

For example, the propositional domain (5) without the constraint $Walking(s) \supset Alive(s)$, belongs to this set. These are essentially the domains for which Reiter [Reiter, 1991] proposes a method for the generation of "successor state axioms".

To the second class belong the domains in which each proposition does not contain object terms. For example, the domain (5) belongs to this set. For such domains, (9) is equivalent to

$$\begin{array}{ll}do(a,s) = s' \wedge s' \neq \bot \supset\ \neg\exists v_1 \ldots v_n(& \\ \quad \bigwedge_D \Upsilon(s)[v_1/F_1(s), \ldots, v_n/F_n(s)] & \wedge \\ \quad \bigwedge_D(\Phi(a,s) \supset \Psi(a,s')[v_1/F_1(s'), \ldots, v_n/F_n(s')]) & \wedge \\ \quad \bigwedge_{1\leq i\leq n}(v_i = F_i(s') \vee v_i = F_i(s)) & \wedge \\ \quad \bigvee_{1\leq i\leq n}(v_i \neq F_i(s'))), & \end{array} \qquad (12)$$

while (13) is equivalent to:

$$\begin{array}{l}do(a,s) = s' \supset (s' = \bot \equiv (s = \bot \vee \neg\exists v_1 \ldots v_n(\\ \quad \bigwedge_D \Upsilon(s)[v_1/F_1(s), \ldots, v_n/F_n(s)] \wedge \\ \quad \bigwedge_D(\Phi(a,s) \supset \Psi(a,s')[v_1/F_1(s'), \ldots, v_n/F_n(s')])))).\end{array} \qquad (13)$$

For example, in the case of domain (5), (12) and (13) become[2]

$$\begin{array}{l}do(a,s) = s' \wedge s' \neq \bot \supset \\ \qquad Res(a, Loaded(s'), Alive(s'), Alive(s'), s)\end{array}$$

and

$$do(a,s) = \bot \equiv s = \bot \vee \neg\exists v_1 v_2 v_3 Res^0(a, v_1, v_2, v_3, s)$$

respectively.

The predicate $Res^0(a,v_1,v_2,v_3,s)$ is defined by the formula

$$\begin{array}{ll}(a = Load \supset v_1 = \mathsf{T}) & \wedge \\ (a = Shoot \wedge Loaded(s) = \mathsf{T} \supset v_2 = \mathsf{F}) & \wedge \\ (a = Shoot \supset v_1 = \mathsf{F}) & \wedge \\ (v_3 = \mathsf{T} \supset v_2 = \mathsf{T}), & \end{array}$$

while $Res(a,v_1,v_2,v_3,s)$ is defined by:

$$\begin{array}{ll}\neg\exists w_1 w_2 w_3(Res^0(a,w_1,w_2,w_3,s) & \wedge \\ \qquad (w_1 = v_1 \vee w_1 = Loaded(s)) & \wedge \\ \qquad (w_2 = v_2 \vee w_2 = Alive(s)) & \wedge \\ \qquad (w_3 = v_3 \vee w_3 = Walking(s)) & \wedge \\ \qquad (w_1 \neq v_1 \vee w_2 \neq v_2 \vee w_3 \neq v_3)). & \end{array}$$

[2]Notice that we do not need to consider the constraints in the last line of (5) since without occurrences of fluents.

The following example is Example 3 in [Kartha and Lifschitz, 1995].

Example 2 *Consider a table divided into three locations F, G and H. A block can be in exactly one of these three locations. There is an action A, which when performed when the block is in location F moves it out of that location. Hence, after the action is performed, the block is in location G or H, but we do not know which.*

This scenario is formalized by the following propositional domain:

$$do(a,s) = s' \wedge s' \neq \perp \supset (a = A \wedge F(s) \supset \neg F(s')),$$
$$F(s) \vee G(s) \vee H(s),$$
$$\neg F(s) \vee \neg G(s),$$
$$\neg F(s) \vee \neg H(s),$$
$$\neg G(s) \vee \neg H(s).$$

This domain admits a first order reduction a-priori computable. Furthermore, because of the ternary constraint, the methods developed in [Pinto, 1995] for the generation of successor state axioms are not applicable.

5 RELATIONS WITH $\mathcal{AR}$

$\mathcal{AR}$ [Giunchiglia *et al.*, 1996] is an "(high level) action description language" in the spirit of $\mathcal{A}$ from [Gelfond and Lifschitz, 1992]. Preliminary reports on $\mathcal{AR}$ appeared in [Kartha and Lifschitz, 1994] and [Giunchiglia *et al.*, 1995]. Differently from these two previous reports, [Giunchiglia *et al.*, 1996] incorporates the idea from [Lifschitz, 1996] of distinguishing between the "action description" and "action query" components of an action language. Intuitively, an action description language is for describing the scenario, i.e. its possible states and the effects of executing actions in states. An action query language is for asking queries about the scenario. A typical query is whether a certain condition is true after executing a given sequence of actions. The sum of an action description and an action query language defines an action language.

In this section we first review the syntax (Section 5.1) and semantics (Section 5.2) of $\mathcal{AR}$. In Section 5.3 we introduce a query language that can be used, together with $\mathcal{AR}$, to address temporal projection problems. These first three subsections are directly derived from [Giunchiglia *et al.*, 1996]. In Section 5.4, we provide a correct and complete translation of any domain description $\mathcal{D}$ of $\mathcal{AR}$ in a domain D_F in the situation calculus. Finally, in Section 5.6, we show that the models of D_F are exactly the intended models of the straightforward translation of $\mathcal{D}$ in the situation calculus.

5.1 SYNTAX OF $\mathcal{AR}$

$\mathcal{AR}$ is a family of "action description languages". An $\mathcal{AR}$ language is characterized by

- a nonempty set of fluent symbols (or fluent names),
- a function, associating with every fluent name F a nonempty set Rng_F of symbols that is called the *range* of F,
- a subset of fluent names, which are said to be *inertial*,
- a nonempty set of symbols, that are called *action names*, or *actions*.

A *value* is a symbol that belongs to the range of a fluent.

An *atomic formula* has the form (F **is** V), where F is a fluent symbol, and $V \in Rng_F$. A *formula* is a propositional combination of atomic formulas. In the following, *True* stands for some fixed tautology and *False* for $\neg$*True*. We also say that a fluent F is *propositional* if $Rng_F = \{\mathsf{F}, \mathsf{T}\}$. For a propositional fluent F, we abbreviate the atomic formula

$$F \textbf{ is } \mathsf{T}$$

by F.

There are three types of *propositions* in an $\mathcal{AR}$ language: determinate effect propositions, indeterminate effect propositions, and constraints. A *determinate effect proposition* is an expression of the form

$$A \textbf{ causes } C \textbf{ if } P, \tag{14}$$

where A is an action, and C and P are formulas. An *indeterminate effect proposition* is an expression of the form

$$A \textbf{ possibly changes } F \textbf{ if } P, \tag{15}$$

where A is an action, F an inertial fluent symbol and P a formula. In effect propositions (14), (15), the part

$$\textbf{if } P$$

is dropped if P is *True*.

Finally, if C is a formula then a *constraint* has the form

$$\textbf{always } C. \tag{16}$$

An *action description* is a set of propositions.

For example, the "murder mystery" can be formalized in $\mathcal{AR}$ as follows (all the fluents are propositional and inertial):

$$\begin{array}{l} \mathit{Load}\ \textbf{causes}\ \mathit{Loaded}, \\ \mathit{Shoot}\ \textbf{causes}\ \neg\mathit{Alive}\ \textbf{if}\ \mathit{Loaded}, \\ \mathit{Shoot}\ \textbf{causes}\ \neg\mathit{Loaded}, \\ \textbf{always}\ \ \mathit{Walking} \supset \mathit{Alive}. \end{array} \tag{17}$$

5.2 SEMANTICS OF $\mathcal{AR}$

The semantics of an action description is defined in terms of a transition system. A transition system is characterized by a set of states, and a transition function mapping a state σ and an action A into a set of states —the states resulting from the execution of A in σ.

A *valuation* is a function that associates to each fluent symbol F an element of Rng_F. A valuation σ is extended to atomic formulas as follows

$$\sigma(F\ \textbf{is}\ V) = \begin{cases} \mathsf{T}, \text{ if } \sigma(F) = V, \\ \mathsf{F}, \text{ otherwise.} \end{cases}$$

and to arbitrary formulas according to the truth tables of propositional logic.

Consider an action description $\mathcal{A}$. A valuation σ is said to be a *state* if, for every constraint (16) in $\mathcal{A}$, $\sigma(C) = \mathsf{T}$. The action description (17) has six states, corresponding to the six possible valuations satisfying the constraint $\mathit{Walking} \supset \mathit{Alive}$.

The transition function Res associated to each action description is defined in two steps.

First, $\mathit{Res}^0(A,\sigma)$ is defined to be the set of states σ' such that, for each determinate effect proposition (14) in $\mathcal{A}$, $\sigma'(C) = \mathsf{T}$ whenever $\sigma(P) = \mathsf{T}$. Then, $\mathit{Res}(A,\sigma)$ is defined to be the set of states σ' such that for any state σ'' in $\mathit{Res}^0(A,\sigma)$, $\mathit{New}_A(\sigma,\sigma'')$ is not a proper subset of $\mathit{New}_A(\sigma,\sigma')$. $\mathit{New}_A(\sigma,\sigma')$ is the set of formulas

$$F\ \textbf{is}\ \sigma'(F)$$

such that

- F is inertial and $\sigma'(F) \neq \sigma(F)$, or
- for some indeterminate effect proposition (15) in $\mathcal{A}$, $\sigma(P) = \mathsf{T}$.

5.3 TEMPORAL PROJECTION IN $\mathcal{AR}$

The specification of a temporal projection problem includes an action description and a set of "initial conditions" that have to be satisfied by the initial situation. An initial condition is specified by a proposition of the form

$$\textbf{initially}\ C \tag{18}$$

where C is a formula in the syntax of $\mathcal{AR}$. We will use the term *domain description* to mean a set of propositions consisting of an action description union a set of initial conditions. (Thus an action description is also a domain description.)

Given a domain description, a temporal projection problem consists in the check of whether a certain condition is true after executing a given sequence of actions starting from the initial situation. To specify such a condition we introduce *value propositions* as expressions of the form

$$C\ \textbf{after}\ A_1;\ldots;A_n \tag{19}$$

where C is a formula and $A_1,\ldots,A_n$ $(n > 0)$ are action names.

A *history* for an action description $\mathcal{A}$ is a finite sequence

$$\sigma_0, A_1, \sigma_1, \ldots, A_n, \sigma_n \tag{20}$$

$(n \geq 0)$ such that σ_0 is a state and

$$\sigma_i \in \mathit{Res}(A_i, \sigma_{i-1}) \qquad (1 \leq i \leq n).$$

A history (20) *satisfies* an initial condition (18) if σ_0 satisfies C. Given an action description $\mathcal{A}$ and a set $\mathcal{I}$ of initial conditions, we say that (19) is a *consequence* of the domain description $\mathcal{A}\cup\mathcal{I}$ if, for any history (20) satisfying all the initial conditions in $\mathcal{I}$, $\sigma_n(C) = \mathsf{T}$.

For example, the domain description (17) entails

$$\neg\mathit{Alive}\ \textbf{after}\ \mathit{Load};\mathit{Wait};\mathit{Shoot}.$$

If we add the initial condition

$$\textbf{initially}\ \mathit{Loaded} \tag{21}$$

then, (17) together with (21) entails also

$$\neg\mathit{Alive}\ \textbf{after}\ \ \mathit{Wait};\mathit{Shoot}.$$

5.4 TRANSLATION OF $\mathcal{AR}$ INTO FIRST ORDER LOGIC

We restrict our attention to *finite* domain descriptions, that is to domain descriptions with finitely many fluent symbols, values, actions and propositions. We also assume that $F_1,\ldots,F_m$ $[F_{m+1},\ldots,F_n$, resp.$]$ are the inertial [not inertial, resp.] fluent symbols, listed according to a fixed enumeration.

We will translate each finite domain description $\mathcal{D}$ into a many sorted first order theory D_F in the language L_F.

5.4.1 Definition of the language L_F

As in Section 2, the language L_F will be many sorted. We have sorts for situations (*situation*), actions (*action*) and values (*value*). We adopt the same conventions used in Section 2.

For the constants, we assume that all the values of $\mathcal{D}$, all actions of $\mathcal{D}$, and the situation constants S_0 and $\perp$ are declared as constants of the appropriate sort. The function symbols are:

- for each fluent symbol F_i, a unary function F_i: *situation* $\rightarrow$ *value*,
- the function *do*: *action* $\times$ *situation* $\rightarrow$ *situation*.

Finally, the predicate constants are:

- $State_F$ whose arguments are an n-tuple of values,
- Res^0_F whose arguments are an action, an n-tuple of values, and a situation,
- Res_F whose arguments are an action, an n-tuple of values, and a situation.

Each predicate is meant to capture the corresponding concept of $\mathcal{AR}$ semantics.

5.4.2 Unique Name Assumptions

For each two distinct action names A_i and A_j,

$$A_i \neq A_j. \tag{22}$$

For each fluent symbol F_i and for each pair V_j, V_k of distinct values in $Rng(F_i)$,

$$V_j \neq V_k. \tag{23}$$

Finally,

$$S_0 \neq \perp. \tag{24}$$

5.4.3 Definition of $State_F$, Res^0_F and Res_F

For any formula C of $\mathcal{D}$

- with C_v we indicate the formula of L_F obtained replacing each fluent symbol F_i with the variable v_i in C,
- for any situation term ξ, with C_ξ we indicate the formula of L_F obtained replacing each fluent symbol F_i with the term $F_i(\xi)$ in C.

The next three equations define the predicates $State_F$, Res^0_F and Res_F, respectively. $State_F(v_1, \ldots, v_n)$ is defined to be equivalent to:

$$\bigwedge_{1\leq i\leq n} (\bigvee_{V\in Rng(F_i)} v_i = V) \wedge \bigwedge_{\textbf{always } C\in\mathcal{D}} C_v. \tag{25}$$

Proposition 1 *Let M be a structure of L_F satisfying* (23) *and* (25). *A set $\{\langle F_i, V_i\rangle \mid 1 \leq i \leq n\}$ is state of $\mathcal{D}$ iff $M \models State_F(V_1, \ldots, V_n)$.*

Proposition 1 trivially follows from the fact that in a model equality is interpreted as identity.

$Res^0_F(a, v_1, \ldots, v_n, s)$ is defined equivalent to:

$$\begin{array}{c} State_F(v_1, \ldots, v_n)\wedge \\ \bigwedge_{A \textbf{ causes } C \textbf{ if } P\in\mathcal{D}}((a = A \wedge P_s) \supset C_v). \end{array} \tag{26}$$

Given a structure M of L_F, we say that a ground situation term ξ *situates* a valuation σ of $\mathcal{D}$ if

$$M[\![F_i(\xi)]\!] = M[\![\sigma(F_i)]\!] \qquad (1 \leq i \leq n).$$

If ξ is a ground situation term situating σ in M then, for any formula P of $\mathcal{D}$, $\sigma(P) = \mathsf{T}$ iff $M \models P_\xi$, (this can be proved by induction on the structure of the formula).

Proposition 2 *Let A be an action and σ a valuation of $\mathcal{D}$. Let M be a structure of L_F satisfying* (22)–(26) *and ξ a ground situation term situating σ in M. $\sigma' \in Res^0(A, \sigma)$ iff $M \models Res^0_F(A, \sigma'(F_1), \ldots, \sigma'(F_n), \xi)$.*

The thesis follows from Proposition 1 and the uniqueness of names both for actions and values within each range associated to a fluent.

$Res_F(a, v_1, \ldots, v_n, s)$ is defined equivalent to:

$$\begin{array}{ll} Res^0_F(a, v_1, \ldots, v_n, s)\wedge & \\ \neg\exists w_1 \ldots w_n (Res^0_F(a, w_1, \ldots, w_n, s) & \wedge \\ \quad \bigwedge_{1\leq i\leq m}(w_i = v_i \vee w_i = F_i(s)) & \wedge \\ \quad \bigvee_{1\leq i\leq m}(w_i \neq v_i) & \wedge \\ \quad \bigwedge_{A \textbf{ possibly changes } F_i \textbf{ if } P\in\mathcal{D}} ((a = A \wedge P_s) \supset v_i = w_i)). & \end{array} \tag{27}$$

Proposition 3 *Let A be an action and σ a valuation of $\mathcal{D}$. Let M be a structure of L_F satisfying* (22)–(27) *and ξ a ground situation term situating σ in M. $\sigma' \in Res(A, \sigma)$ iff $M \models Res_F(A, \sigma'(F_1), \ldots, \sigma'(F_n), \xi)$.*

The proof is based on the following fact about $\mathcal{AR}$. $\sigma' \in Res(A, \sigma)$ iff $\sigma' \in Res^0(A, \sigma)$ and there is not a valuation $\sigma'' \in Res^0(A, \sigma)$ such that:

- $\sigma''(F) = \sigma'(F)$ for each fluent symbol F such that for some indeterminate effect proposition (15) in $\mathcal{D}$, $\sigma(P) = \mathsf{T}$;

- for each inertial fluent symbol F, either $\sigma''(F) = \sigma'(F)$ or $\sigma''(F) = \sigma(F)$; and

- there exists at least one inertial fluent symbol F such that $\sigma''(F) \neq \sigma'(F)$.

5.4.4 Model conditions

Each initial condition has to be satisfied:

$$\bigwedge_{\textbf{initially } C \in \mathcal{D}} C_{S_0}. \tag{28}$$

Also the constraints have to be satisfied:

$$State_F(F_1(s), \ldots, F_n(s)). \tag{29}$$

If an action A is executable in a state σ, the resulting states shall belong to $Res(A, \sigma)$:

$$\begin{aligned} &do(a, s) \neq \perp \supset \\ &\quad Res_F(a, F_1(do(a, s)), \ldots, F_n(do(a, s)), s). \end{aligned} \tag{30}$$

Normally, actions are executable:

$$\begin{aligned} &do(a, s) = \perp \equiv s = \perp \vee \\ &\quad \neg \exists v_1 \ldots v_n Res^0_F(a, v_1, \ldots, v_n, s). \end{aligned} \tag{31}$$

5.5 CORRECTNESS AND COMPLETENESS OF THE TRANSLATION

Let $\mathcal{D}$ be a finite domain description. The *translation of $\mathcal{D}$ in first order logic* is the first order theory D_F consisting of (22)–(31).

For any string of actions $A_1 \ldots A_n$ ($n \geq 0$) we abbreviate the term $do(A_n, do(A_{n-1}, \ldots, do(A_1, S_0) \ldots))$ with $do(A_1 \ldots A_n, S_0)$.

Theorem 4 *Let D_F be the translation of a finite domain description $\mathcal{D}$ in first order logic. A value proposition C* **after** *$\overline{A}$ is a consequence of $\mathcal{D}$ iff D_F entails $do(\overline{A}, S_0) \neq \perp \supset C_{do(\overline{A}, S_0)}$.*

Notice that, domains

- having at least one model in the action language $\mathcal{A}$ [Gelfond and Lifschitz, 1992], or

- without "releases" propositions in the action language $\mathcal{AR}_0$ [Kartha and Lifschitz, 1994], or

- without dependent fluents in the action language $\mathcal{ARD}$ [Giunchiglia and Lifschitz, 1995],

can be regarded as $\mathcal{AR}$ domains. Hence, the above translation and results can be applied also to them.

5.6 MODELS OF THE TRANSLATION

Consider a finite domain description $\mathcal{D}$. Let D_F the translation of $\mathcal{D}$ in first order logic. Define D to be the domain (in the language L_F of D_F, see Section 5.4.1) consisting of:

- the formulas (22)–(24); and

- the effect proposition

  $$do(A, s) = s' \wedge s' \neq \perp \supset (P_s \supset C_{s'})$$

  for each proposition (14) in $\mathcal{D}$; and

- the constraint

  $$C_s$$

  for each proposition (16) in $\mathcal{D}$, and

  $$\bigvee_{V \in Rng(F_i)} F_i(s) = V$$

  for each fluent symbol F_i.

Theorem 5 *Let $\mathcal{D}$ be a finite domain description such that*

- *all the fluent symbols are inertial, and*

- *$\mathcal{D}$ consists only of determinate effect propositions and constraints.*

A structure M of L_F is a model of D_F iff M is an intended model of D.

To check the above theorem, it is enough to

1. unfold the definitions of $State_F$, Res^0_F and Res_F in (29), (30) and (31), and

2. compare the result with D union (12) and (13) (we remind that these two last formulas characterize the set of intended models for domains without object terms): the two theories are logically equivalent.

As a trivial consequence of Theorem 4 and Theorem 5 we have the following Corollary.

Corollary 1 *In the hypotheses of Theorem 5, a value proposition C* **after** $\overline{A}$ *is a consequence of $\mathcal{D}$ iff the class of intended models of D entails $do(\overline{A}, S_0) \neq \bot \supset C_{do(\overline{A},S_0)}$.*

Acknowledgments

This work has been initiated while the author was joining the "Logical Foundations of Artificial Intelligence Group" in the Computer Science Department of the University of Texas at Austin. Thanks to Neelakantan Kartha and Norm McCain for useful discussions on subjects related to this paper. Thanks to Vladimir Lifschitz and Hudson Turner for very useful comments on a early version of this paper.

This work has been supported by the Italian National Research Council (CNR), under the projects "Sistemi Autonomi Ragionanti e Intelligenti (SARI)" and "Analisi Sperimentale delle proprietà computazionali delle logiche per la rappresentazione della conoscenza".

References

[Baker, 1991] Andrew Baker. Nonmonotonic reasoning in the framework of situation calculus. *Artificial Intelligence*, 49:5–23, 1991.

[Doherty *et al.*,] Patrick Doherty, Witold Lukaszewicz, and Andrzey Szalas. Computing circumscription revisited: A reduction algorithm. In *Proc. IJCAI-95*, pages 1502–1508.

[ECAI, 1996] ECAI. Workshop on Integrating Nonmonotonicity into Automated Reasoning Systems, 1996.

[Enderton, 1972] H.B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

[Gabbay and Ohlbach, 1992] Dov Gabbay and Hans Ohlbach. Quantifier elimination in second-order predicate logic. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *Proc. of the Third Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 425–435, 1992.

[Gelfond and Lifschitz, 1992] Michael Gelfond and Vladimir Lifschitz. Representing actions in extended logic programming. In Krzysztof Apt, editor, *Proc. Joint Int'l Conf. and Symp. on Logic Programming*, pages 559–573, 1992.

[Giunchiglia and Lifschitz, 1995] Enrico Giunchiglia and Vladimir Lifschitz. Dependent fluents. In *Proc. IJCAI-95*, 1995.

[Giunchiglia *et al.*, 1995] Enrico Giunchiglia, G. Neelakantan Kartha, and Vladimir Lifschitz. Actions with indirect effects (extended abstract). In *Working Notes of the AAAI Spring Symposium 95 on Extending Theories of Actions*, 1995.

[Giunchiglia *et al.*, 1996] Enrico Giunchiglia, G. Neelakantan Kartha, and Vladimir Lifschitz. Actions with indirect effects. Manuscript. Extended and revised version of [Giunchiglia *et al.*, 1995]., 1996.

[IJCAI, 1995] IJCAI. Workshop on Applications and Implementations of Nonmonotonic Reasoning Systems, 1995.

[Kartha and Lifschitz, 1994] G. Neelakantan Kartha and Vladimir Lifschitz. Actions with indirect effects (preliminary report). In *Proc. of the Fourth Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 341–350, 1994.

[Kartha and Lifschitz, 1995] G. Neelakantan Kartha and Vladimir Lifschitz. A simple formalization of actions using circumscription. In *Proc. IJCAI-95*, 1995.

[Kartha, 1993] G. Neelakantan Kartha. Soundness and completeness theorems for three formalizations of action. In *Proc. of IJCAI-93*, pages 724–729, 1993.

[Kartha, 1994] G. Neelakantan Kartha. Two counterexamples related to Baker's approach to the frame problem. *Artificial Intelligence*, 69:379–391, 1994.

[Kartha, 1996] G. Neelakantan Kartha. On the range of applicability of Baker's approach to the frame problem. In *Proc. of AAAI-96*, 1996.

[Lifschitz, 1985] Vladimir Lifschitz. Computing circumscription. In *Proc. of IJCAI-85*, pages 121–127, 1985.

[Lifschitz, 1990] Vladimir Lifschitz. Frames in the space of situations. *Artificial Intelligence*, 46:365–376, 1990.

[Lifschitz, 1995] Vladimir Lifschitz. Nested abnormality theories. *Artificial Intelligence*, 74:351–365, 1995.

[Lifschitz, 1996] Vladimir Lifschitz. Two components of an action language. In *Working Papers of* *the Third Symposium on Logical Formalizations of Commonsense Reasoning*, 1996.

[Lin and Reiter, 1994] Fangzhen Lin and Raymond Reiter. State constraints revisited. *Journal of Logic and Computation*, 4:655–678, 1994.

[Lin and Shoham, 1991] Fangzhen Lin and Yoav Shoham. Provably correct theories of action (preliminary report). In *Proc. AAAI-91*, pages 349–354, 1991.

[Pednault, 1989] Edwin Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In Ronald Brachman, Hector Levesque, and Raymond Reiter, editors, *Proc. of the First Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 324–332, 1989.

[Pinto, 1995] Andres Javier Pinto. *Temporal Reasoning in the Situation Calculus.* PhD thesis, 1995. PhD thesis, University of Toronto.

[Reiter, 1991] Raymond Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, 1991.

[Winslett, 1988] Marianne Winslett. Reasoning about action using a possible model approach. In *Proc. AAAI-88*, pages 89–93, 1988.