Representing Action: Indeterminacy and Ramifications

Enrico Giunchiglia
DIST — University of Genoa
16145 Genoa, Italy

G. Neelakantan Kartha
i2 Technologies
1603 LBJ Freeway, Suite 780
Dallas, Texas 75234, USA

Vladimir Lifschitz
University of Texas at Austin
Austin, Texas 78712, USA

Abstract

We define and study a high-level language for describing actions, more expressive than the action language $\mathcal{A}$ introduced by Gelfond and Lifschitz. The new language, $\mathcal{AR}$, allows us to describe actions with indirect effects (ramifications), nondeterministic actions, and actions that may be impossible to execute. It has symbols for nonpropositional fluents and for the fluents that are exempt from the commonsense law of inertia. Temporal projection problems specified using the language $\mathcal{AR}$ can be represented as nested abnormality theories based on the situation calculus.

1 Introduction

Mary jumped into the lake, then got out of the water and put her hat on.

Common sense allows us to answer some questions about the outcome of this series of events.

- *Is Mary in the lake?* Of course not. She just got out of it.

- *Does she have the hat on?* Of course she does. She just put it on.
- *Is she wet?* Of course she is. She just got out of the lake.

These are examples of reasoning about actions and their effects. Formalizing this form of commonsense reasoning has long been considered one of the central problems of Artificial Intelligence. Existing approaches differ by their temporal ontologies (linear or branching time, time points or intervals, situations, events or histories), by the logic used (classical logic, its nonmonotonic extensions, logic programming), and by other details of the formalization (which objects are reified, how circumscription is applied, etc.).

Properties of actions can be conveniently described in specialized "action languages," such as the language $\mathcal{A}$ from [7]. Consider, for instance, the background knowledge about the effects of actions that is involved in the examples of commonsense reasoning above. Mary has performed three actions: jumping into the lake (J), getting out of the lake (G) and putting her hat on (P). We are interested in the effect of each of these actions on whether or not Mary is in the lake (L). In the language $\mathcal{A}$, the effects of the actions J and G on L can be described by the propositions

$$J \textbf{ causes } L,$$
$$G \textbf{ causes } \neg L.$$

There is no need to specify that P has no effect on L, because the semantics of $\mathcal{A}$ incorporates the "commonsense law of inertia": when an action is performed, things are assumed by default to remain the same.

The language $\mathcal{A}$ has been used as a tool for investigating the relationship between several other techniques for describing actions. For instance, in [11], the methods for formalizing actions in classical logic developed in [25] and [26], as well as the use of circumscription in [1], are described as translations from $\mathcal{A}$, and each of the translations is found to be sound and complete relative to the semantics of $\mathcal{A}$. These theorems show that, for the action domains that can be represented in $\mathcal{A}$, the three formalization methods produce equivalent results.

In this paper, we introduce a new action language, $\mathcal{AR}$, which is in several ways more expressive than $\mathcal{A}$.

1. In $\mathcal{AR}$, actions may have indirect effects, or *ramifications*. Consider, for instance, the effect that jumping into the lake has on Mary being wet

(W). In $\mathcal{A}$, it can be described by the proposition

$$J \textbf{ causes } W.$$

We would prefer, however, to treat the effect of J on W as a ramification of its effect on L. *Any* action that causes L to become true will make W true also: if Mary walks into the lake, or crawls into the lake, or is thrown into the lake by her boyfriend, she will get wet all the same. This relationship between L and W can be described in the new action language by writing

$$\textbf{always } L \supset W.$$

2. Actions described in $\mathcal{AR}$ can be *nondeterministic*. If Mary jumps into the lake while having her hat on (H) then maybe she will lose her hat, or maybe not. In $\mathcal{AR}$, this indeterminacy can be described by the proposition

$$J \textbf{ possibly changes } H \textbf{ if } H.$$

3. In $\mathcal{AR}$, an action can be *impossible to execute* in some states. For instance, Mary cannot get out of the lake if she is already on the shore. We will write this as

$$\textbf{impossible } G \textbf{ if } \neg L.$$

In fact, this will be treated as an abbreviation for the proposition

$$G \textbf{ causes } \mathit{False} \textbf{ if } \neg L.$$

4. The language $\mathcal{AR}$ has symbols for *nonpropositional fluents*. Generally, a "fluent" [24] is something that depends on the state of the world. For instance, L (being in the lake), W (being wet) and H (having the hat on) are fluents, because each can be false or true depending on the particular situation. The possible values of these fluents are the truth values of propositional logic, F and T. In a more elaborate formalization, we might wish to introduce the nonpropositional fluent symbol *Location*, representing Mary's current location. $\mathcal{AR}$ allows us to write

$$\mathit{Location} \textbf{ is } \mathit{Lake}$$

instead of L, and to use other location symbols, such as *Shore*, *Home* or *Library*, in place of *Lake*.

5. In $\mathcal{AR}$, a fluent can be classified as *noninertial*, which makes it exempt from the commonsense law of inertia. For instance, the sun may be now

behind the clouds (C), but there is no guarantee that this will be still the case a minute later, no matter what Mary will be doing during this time. The fluent C is noninertial. Noninertial fluents are needed also for expressing explicit definitions (Section 4.3).

The next two sections of the paper describe the syntax and semantics of $\mathcal{AR}$ and relate this language to the action language $\mathcal{A}$ from [7]. Our work on the "debugging" of the semantics of $\mathcal{AR}$ has involved the verification of several properties of the language that can be naturally expected to hold for action languages; these properties are discussed in Section 4. In Section 5, we define the syntax and semantics of "initial conditions" and of "value propositions." This allows us to express formally temporal projection problems involving actions described in $\mathcal{AR}$. In Section 6 we show how temporal projection problems of this kind can be expressed in terms of the version of circumscription [22] called nested abnormality theories [18]. Reductions of this kind are of special interest in connection with recent advances in the automation of circumscriptive reasoning [6], [3]. The relation of this paper to earlier work on action is discussed in Section 7. Proofs are relegated to the appendix.

Preliminary reports on this work are published as [14] and [8]. $\mathcal{AR}$ differs from the language $\mathcal{AR}_0$ described in the first of these papers in two ways. First, $\mathcal{AR}_0$ did not include symbols for nonpropositional fluents (see the end of Section 4.4). Second, instead of **possibly changes**, it used the construct **releases**, whose semantics turned out to be less satisfactory (see Section 4.1). Furthermore, this paper differs from both preliminary publications in that "value propositions" are not treated here as part of $\mathcal{AR}$, and their use is restricted to the conceptually simpler case of temporal projection problems.

2 Syntax

To be precise, $\mathcal{AR}$ is not a single language, but rather a family of languages. A particular language in this group is characterized by

- a nonempty set of symbols that are called *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 that are called *inertial*,

- a nonempty set of symbols that are called *action names*.

2.1 Formulas, Propositions and Action Descriptions

An *atomic formula* is an expression of the form

$$(F \textbf{ is } V)$$

where F is a fluent name, and $V \in Rng_F$. A *formula* is a propositional combination of atomic formulas.

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

$$\textbf{always } C \tag{1}$$

where C is a formula. A *determinate effect proposition* is an expression of the form

$$A \textbf{ causes } C \textbf{ if } P \tag{2}$$

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

$$A \textbf{ possibly changes } F \textbf{ if } P \tag{3}$$

where A is an action name, F an inertial fluent name, and P a formula ("precondition").

An *action description* is a set of propositions.

2.2 Notational Conventions

In formulas, we will omit some parentheses, as customary in classical logic. We will denote some fixed tautological formula by *True*, and $\neg$*True* by *False*. A determinate effect proposition (2) will be written as

$$A \textbf{ causes } C$$

if P is *True*, and as

$$\textbf{impossible } A \textbf{ if } P$$

if C is *False*. An indeterminate effect proposition (3) will be written as

$$A \textbf{ possibly changes } F$$

if its precondition P is *True*. For any action name A and formula C,

$$A \textbf{ initiates } C$$

stands for the pair of propositions

$$\begin{array}{l} A \textbf{ causes } C, \\ \textbf{impossible } A \textbf{ if } C. \end{array}$$

A fluent name F is *propositional* if

$$Rng_F = \{\mathsf{F}, \mathsf{T}\}.$$

If F is a propositional fluent name, we will abbreviate the atomic formula

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

by F, and the atomic formula

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

by $\overline{F}$.

Using these notational conventions, we can formalize the example from the introduction as follows:

$$\begin{array}{l} \textbf{always } L \supset W, \\ J \textbf{ initiates } L, \\ J \textbf{ possibly changes } H \textbf{ if } H, \\ G \textbf{ initiates } \neg L, \\ P \textbf{ initiates } H. \end{array} \tag{4}$$

Here L, W, H are inertial propositional fluent names, and J, G, P are action names.

3 Semantics

The meaning of an action description D is represented by the corresponding *transition function*, Res_D. This function maps an action name and a state (defined below) to a set of states. Intuitively, $Res_D(A, \sigma)$ is the set of outcomes that may result from executing A in state σ.

3.1 States

A *valuation* is a function that is defined on the set of fluent names and maps each fluent name F to an element of its range. A valuation σ can be extended to atomic formulas in a standard way:

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

It can be further extended to arbitrary formulas according to the truth tables of propositional logic. A valuation σ *satisfies* a formula C, or a constraint (1), if $\sigma(C) = \mathsf{T}$.

A *state* for an action description D is a valuation that satisfes all constraints in D.

For instance, the states for description (4) are the truth-valued functions on $\{L, W, H\}$ that satisfy $L \supset W$. There are 6 such functions:

$$LWH,\ \overline{L}WH,\ \overline{L}\,\overline{W}H,\ LW\overline{H},\ \overline{L}W\overline{H},\ \overline{L}\,\overline{W}\,\overline{H}.$$

(We represent each state by the set of atomic formulas that are satisfied in that state.)

3.2 The Transition Function

For any action name A and state σ, let $Res^0_D(A, \sigma)$ stand for the set of states σ' such that, for each determinate effect proposition (2) in D, σ' satisfies C whenever σ satisfies P. The set $Res_D(A, \sigma)$ will be defined as the subset of $Res^0_D(A, \sigma)$ whose elements are "close" to σ.

In order to make this precise, the following notation is needed. For any action name A and any states σ, σ', by $New^A_D(\sigma, \sigma')$ we denote the set of formulas

$$F \textbf{ is } \sigma'(F) \tag{5}$$

such that

- F is inertial and $\sigma'(F) \neq \sigma(F)$, or
- for some indeterminate effect proposition (3) in D, σ satisfies P.

The condition $\sigma'(F) \neq \sigma(F)$ in the definition of this function expresses that (5) is a "new fact" that becomes true if the execution of A in state σ results in state σ'. The set $New^A_D(\sigma, \sigma')$ includes such "new facts" for all inertial fluent names F. (If F is noninertial then it is not expected to keep its old value after performing an action, so that the change in its value is disregarded.) On the other hand, if some indeterminate effect proposition allows F to change, we treat its value in state σ' as "new" even if it happens to coincide with the value of F in state σ.

Now the transition function Res_D is defined as follows: $Res_D(A, \sigma)$ is the set of states $\sigma' \in Res_D^0(A, \sigma)$ for which $New_D^A(\sigma, \sigma')$ is minimal relative to set inclusion—in other words, for which there is no $\sigma'' \in Res_D^0(A, \sigma)$ such that $New_D^A(\sigma, \sigma'')$ is a proper subset of $New_D^A(\sigma, \sigma')$. This minimality condition is the feature of the semantics of $\mathcal{AR}$ that represents the commonsense law of inertia.

We will sometimes drop the subscript D in the symbols New_D^A, Res_D^0 and Res_D, when the action description D is understood from the context.

3.3 Example

As an illustration, consider the transition function for example (4). The action name J is described by the propositions

$$\begin{array}{l} J \ \mathbf{causes}\ L, \\ J \ \mathbf{causes}\ \mathit{False}\ \mathbf{if}\ L, \\ J \ \mathbf{possibly\ changes}\ H \ \mathbf{if}\ H. \end{array}$$

For any states σ and σ', $\sigma' \in Res^0(J, \sigma)$ iff $\sigma'(L) = \mathsf{T}$ and $\sigma(L) = \mathsf{F}$. Consequently,

$$Res^0(J, \sigma) = \begin{cases} \{LWH, LW\overline{H}\}, & \text{if } \sigma(L) = \mathsf{F}, \\ \emptyset, & \text{otherwise.} \end{cases}$$

Furthermore,

$$Res(J, \sigma) = \begin{cases} \{LW\overline{H}\}, & \text{if } \sigma(L) = \sigma(H) = \mathsf{F}, \\ \{LWH, LW\overline{H}\}, & \text{if } \sigma(L) = \mathsf{F} \text{ and } \sigma(H) = \mathsf{T}, \\ \emptyset, & \text{otherwise.} \end{cases}$$

As an example, let us verify these assertions for σ equal to $\overline{L}\,\overline{W}\,\overline{H}$ and to $\overline{L}\,\overline{W}H$. In the first case, note that

$$\begin{array}{l} New^A(\overline{L}\,\overline{W}\,\overline{H}, LW\overline{H}) = \{L \ \mathbf{is}\ \mathsf{T},\ W \ \mathbf{is}\ \mathsf{T}\}, \\ New^A(\overline{L}\,\overline{W}\,\overline{H}, LWH) = \{L \ \mathbf{is}\ \mathsf{T},\ W \ \mathbf{is}\ \mathsf{T},\ H \ \mathbf{is}\ \mathsf{T}\}. \end{array}$$

Consequently, the element LWH of $Res^0(J, \overline{L}\,\overline{W}\,\overline{H})$ does not satisfy the minimality condition and is not included in $Res(J, \overline{L}\,\overline{W}\,\overline{H})$. In the second case,

$$\begin{array}{l} New^A(\overline{L}\,\overline{W}H, LWH) = \{L \textbf{ is } \mathsf{T},\ W \textbf{ is } \mathsf{T},\ H \textbf{ is } \mathsf{T}\}, \\ New^A(\overline{L}\,\overline{W}H, LW\overline{H}) = \{L \textbf{ is } \mathsf{T},\ W \textbf{ is } \mathsf{T},\ H \textbf{ is } \mathsf{F}\}, \end{array}$$

and the minimality condition does not require removing any of the elements of $Res^0(J, \overline{L}\,\overline{W}H)$.

The values of Res for the other two action names can be computed in a similar way. The following notation helps express the results of computation concisely. For any valuation σ, any fluent name F and any V in the range of F, the expression $\sigma[F/V]$ stands for the valuation that takes the value V on F and otherwise agrees with σ:

$$\sigma[F/V](F') = \begin{cases} V, & \text{if } F' = F, \\ \sigma(F'), & \text{otherwise.} \end{cases}$$

For instance,

$$(LWH)[L/\mathsf{F}] = (\overline{L}WH).$$

In this notation,

$$Res(G, \sigma) = \begin{cases} \{\sigma[L/\mathsf{F}]\}, & \text{if } \sigma(L) = \mathsf{T}, \\ \emptyset, & \text{otherwise,} \end{cases}$$

$$Res(P, \sigma) = \begin{cases} \{\sigma[H/\mathsf{T}]\}, & \text{if } \sigma(H) = \mathsf{F}, \\ \emptyset, & \text{otherwise.} \end{cases}$$

The transition function Res is graphically represented by Figure 1.

Let us go back now to the scenario described in the introduction: Mary jumped into the lake (J), then got out of the water (G) and put her hat on (P). The examples of commonsense conclusions given there can be interpreted as assertions about paths in the transition diagram. Consider any path of length 3 whose arcs are labeled J, G, P, and let σ be the end node of this path. Mary is not in the lake: $\sigma(L) = \mathsf{F}$. Mary has her hat on: $\sigma(H) = \mathsf{T}$. Mary is wet: $\sigma(W) = \mathsf{T}$.

3.4 Simple Descriptions

There is a special case when the definition of the transition function can be simplified.

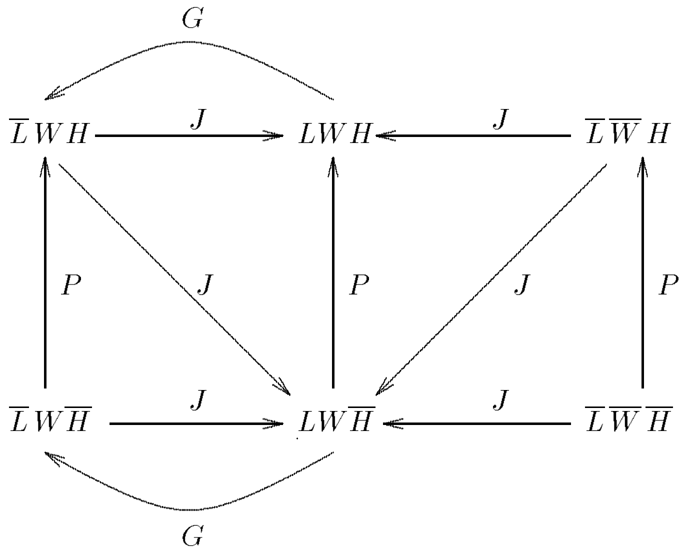

Figure 1: Transition Diagram for the Lake Example

A formula is *simple* if it is a conjunction of atomic formulas. Note that the empty conjunction *True* is included in simple formulas. A simple formula is *contradictory* if it includes a pair of atomic parts F **is** V_1, F **is** V_2 with $V_1 \neq V_2$; otherwise, it is *noncontradictory*. A determinate effect proposition (2) is *simple* if both C and P are noncontradictory simple formulas.

An action description D is *simple* if

- each fluent name of D is inertial,
- each proposition in D is a simple determinate effect proposition,
- for any propositions

  $$A \textbf{ causes } C_1 \textbf{ if } P_1,$$
  $$A \textbf{ causes } C_2 \textbf{ if } P_2$$

  in D such that $P_1 \wedge P_2$ is noncontradictory, $C_1 \wedge C_2$ is noncontradictory also.

Here is an example. Each of m blocks $B_1, \ldots, B_m$ can be located on any of n tables $T_1, \ldots, T_n$. The fluent names are *Location*(B_i) $(1 \leq i \leq m)$; all of them are inertial and have the range consisting of the symbols T_k

$(1 \leq k \leq n)$. The action names are $\mathit{MoveOnto}(B_i, T_k)$ (move block B_i onto table T_k) and $\mathit{MoveNextTo}(B_i, B_j)$ (move block B_i onto the table that has B_j on it). The action description consists of the propositions

$$\mathit{MoveOnto}(b,t) \textbf{ causes } (\mathit{Location}(b) \textbf{ is } t),$$
$$\mathit{MoveNextTo}(b,b') \textbf{ causes } (\mathit{Location}(b) \textbf{ is } t) \textbf{ if } (\mathit{Location}(b') \textbf{ is } t)$$

for all $b, b' \in \{B_1, \ldots, B_m\}$ and $t \in \{T_1, \ldots, T_n\}$. This description is simple.

The following theorem shows that, for a simple action description D, every value of Res_D is a singleton—there is exactly one way to execute any action in any state. Moreover, the only element of $\mathit{Res}_D(A, \sigma)$ can be described without an explicit reference to the minimality condition.

Let D be a simple action description, A an action name, and σ a state. By $\mathit{Res}^*_D(A, \sigma)$ we denote the state σ' defined as follows: For any fluent name F, if D includes a determinate effect proposition (2) such that

(i) σ satisfies P, and

(ii) C has a conjunctive term of the form F **is** V

then $\sigma'(F) = V$; if not then $\sigma'(F) = \sigma(F)$. It is clear that this definition characterizes the state σ' uniquely. In the state $\mathit{Res}^*_D(A, \sigma)$, fluents have the same values as in σ except when a different value is "required" by a proposition in D.

Theorem 1. *Let D be a simple action description. For any action name A and any state σ,*

$$\mathit{Res}_D(A, \sigma) = \{\mathit{Res}^*_D(A, \sigma)\}.$$

An "effect proposition" of the action language $\mathcal{A}$ [7] is essentially a simple proposition (2) which does not include nonpropositional fluent names and in which C is atomic. The description of the semantics of such propositions in $\mathcal{A}$ is similar to the definition of $\mathit{Res}^*_D(A, \sigma)$ given above. In this sense, Theorem 1 shows that $\mathcal{A}$ is a "sublanguage" of $\mathcal{AR}$.

4 The Mathematics of Action Descriptions

The mathematical properties of $\mathcal{AR}$ established in this section are not surprising—one would expect them to hold for any action language of this kind. The verification of these properties was, however, an important element of our work on the design of the language, because it has allowed us to catch a few nontrivial oversights in earlier versions.

4.1 Replacement

Mary has a little lamb, and the lamb always follows Mary wherever she goes. In $\mathcal{AR}$, this can be expressed by the constraints

$$\textbf{always } Location(Mary) \textbf{ is } X \supset Location(Lamb) \textbf{ is } X$$

for all symbols X in the range (assumed to be common) of the fluent names $Location(Mary)$ and $Location(Lamb)$.

In an action description that includes these constraints, the two fluent names are interchangeable: replacing $Location(Mary)$, in one or many occurrences, by $Location(Lamb)$ would not affect the meaning of the description.

To state this replacement principle in a general form, we need the following definition. Fluent names F, F' are *equivalent* with respect to a set S of constraints if

- they are either both inertial or both noninertial,
- they have a common range,
- for every valuation σ that satisfies all constraints in S, $\sigma(F) = \sigma(F')$.

Theorem 2. *Let S be a set of constraints, D an action description, and F, F' fluent names equivalent with respect to S. If an action description D' is obtained from D by replacing some occurrences of F with F' then*

$$Res_{S \cup D} = Res_{S \cup D'}.$$

As mentioned in the introduction, the earlier version of $\mathcal{AR}$ described in [14] treats indeterminate effect propositions differently, in a less satisfactory way. A problem with that version was that the assertion of Theorem 2 does not hold for it. An example communicated to us by Fangzhen Lin (October 26, 1993) demonstrated the importance of verifying this replacement property.

Besides the replacement of a fluent name, we can consider the replacement of a formula. We say that formulas B and B' are *equivalent* with respect to a set S of constraints if every valuation satisfying all constraints in S satisfies $B \equiv B'$.

Theorem 3. *Let S be a set of constraints, D an action description, and B, B' formulas equivalent with respect to S. If an action description D' is obtained from D by replacing some occurrences of B with B' then*

$$Res_{S \cup D} = Res_{S \cup D'}.$$

Consider, for instance, the expression $\neg L$ in (4), which stands for the formula $\neg(L \textbf{ is } \mathsf{T})$. Theorem 3, with $S = \emptyset$, shows that the meaning of (4) will not change if we replace this formula by $\overline{L}$, which stands for $(L \textbf{ is } \mathsf{F})$.

4.2 Constraints

Constraints play a double role in the semantics of $\mathcal{AR}$. First, they tell us which valuations are counted as states. Second, they are taken into account in calculating the indirect effects of actions. In this second function, a constraint can be replaced by a set of determinate effect propositions:

Theorem 4. *Let S be a set of constraints, and let S' be the set of determinate effect propositions A* **causes** *C for all constraints* **always** *C in S and all action names A. For any action description D, any action name A and any state σ of $S \cup D$,*

$$Res_{S \cup D}(A, \sigma) = Res_{S' \cup D}(A, \sigma).$$

For instance, the constraint

$$\textbf{always } L \supset W$$

in (4) can be replaced by the effect propositions

$$\begin{gathered} J \textbf{ causes } L \supset W, \\ G \textbf{ causes } L \supset W, \\ P \textbf{ causes } L \supset W. \end{gathered}$$

The transition diagram for the description $S \cup D$ is a part of the transition diagram for the new description $S' \cup D$: the former can be obtained from the latter by removing the nodes that violate the constraints in S and the arcs that begin in these nodes.

4.3 Explicit Definitions

Mary is ready to go back home when she is not wet and has her hat on. We would like to introduce an abbreviation for this propositional combination of fluents, $\neg W \wedge H$. To this end, extend description (4) by adding the propositional noninertial fluent name R to its language and by including in it the "explicit definition" of R:

$$\textbf{always } R \equiv (\neg W \wedge H).$$

We chose to declare the new name R noninertial because we expect the effect of any action on R to be determined by its effects on W and H and by the definition of R, not by inertia.

The theorem stated below shows that adding an explicit definition to an action description leaves its semantics essentially unchanged.

A constraint in an action description D is an *explicit definition* of a propositional fluent name F if it has the form

$$\textbf{always } F \equiv C \tag{6}$$

and F

- is noninertial,
- does not occur in C,
- does not occur in any other proposition in D.

An action description D that includes an explicit definition of a propositional fluent name F is "equivalent" to the action description D' obtained from D by deleting this definition and deleting F from the set of fluent names, but not in the strong sense that they define the same transition function, as in Theorem 2. Indeed, the fluent names and the states of D' are not the same as the fluent names and the states of D. We can only claim that there exists an isomorphism between D and D', in the following sense.

Let D and D' be action descriptions with the same action names. An *isomorphism* between D and D' is a one-to-one function f from the set of states of D onto the set of states of D' such that, for every action name A and every state σ of D, f maps $Res_D(A, \sigma)$ onto $Res_{D'}(A, f(\sigma))$.

Theorem 5. *Let D be an action description containing an explicit definition of a propositional fluent name F, and let D' be obtained from D by deleting this definition and by deleting F from the set of fluent names. For any state σ of D, let σ_F be the restriction of σ to the set of fluent names that are different from F. The function $\sigma \mapsto \sigma_F$ is an isomorphism between D and D'.*

It is essential for the validity of this theorem that F is required to be noninertial. Without noninertial fluent names, we would not be able to express explicit definitions in the language.

4.4 Nonpropositional Fluents

The next result shows that a nonpropositional fluent name with a finite range can be eliminated in favor of propositional fluent names. A fluent with n possible values can be replaced by n propositional fluents; constraints included in the new description will require that exactly one of these fluents be true in any state.

Let D be an action description, and let $\mathbf{F}$ be a set of fluent names in the language of D such that every fluent name in $\mathbf{F}$ has a finite range. The action description $D_{\mathbf{F}}$ is defined as follows.

The language of $D_{\mathbf{F}}$ is obtained from the language of D by replacing every F in $\mathbf{F}$ by new propositional fluent names F^V for all $V \in Rng_F$. The name F^V is declared inertial if F is inertial in the language of D. For any formula C in the language of D, $C_{\mathbf{F}}$ stands for the result of replacing all its atomic parts

$$F \textbf{ is } V$$

such that $F \in \mathbf{F}$ by

$$F^V \textbf{ is } \mathsf{T}.$$

The propositions of $D_{\mathbf{F}}$ are

- the constraints

  $$\textbf{always } C_{\mathbf{F}}$$

  for all constraints (1) in D,

- the determinate effect propositions

  $$A \textbf{ causes } C_{\mathbf{F}} \textbf{ if } P_{\mathbf{F}}$$

  for all determinate effect propositions (2) in D,

- the indeterminate effect propositions

  $$A \textbf{ possibly changes } F \textbf{ if } P_{\mathbf{F}}$$

  for all indeterminate effect propositions (3) in D with $F \notin \mathbf{F}$,

- the indeterminate effect propositions

  $$A \textbf{ possibly changes } F^V \textbf{ if } P_{\mathbf{F}}$$

  for all indeterminate effect propositions (3) in D with $F \in \mathbf{F}$ and all V in the range of F.

- the constraints
$$\textbf{always } \neg(F^{V_1} \wedge F^{V_2})$$
and
$$\textbf{always } \bigvee_{V \in Rng_F} F^V$$
for all fluent names $F \in \mathbf{F}$ and all pairs of distinct values V_1, V_2 in the range of F.

The theorem below shows that $D_{\mathbf{F}}$ is isomorphic to D. For every state σ of D, define the state $\sigma_{\mathbf{F}}$ of $D_{\mathbf{F}}$ as follows: for any $F \in \mathbf{F}$ and any $V \in Rng_F$,

$$\sigma_{\mathbf{F}}(F^V) = \begin{cases} \mathsf{T}, & \text{if } \sigma(F) = V, \\ \mathsf{F}, & \text{otherwise;} \end{cases}$$

for any $F \notin \mathbf{F}$,

$$\sigma_{\mathbf{F}}(F) = \sigma(F).$$

Theorem 6. *Let D be an action description, and let $\mathbf{F}$ be a set of fluent names in the language of D such that every fluent name in $\mathbf{F}$ has a finite range. The function $\sigma \mapsto \sigma_{\mathbf{F}}$ is an isomorphism between D and $D_{\mathbf{F}}$.*

In this sense, nonpropositional fluents with finite ranges are redundant. Any action description that involves only fluents with finite ranges can be effectively reduced to a description that involves propositional fluents only.

As an example, consider a disk divided into n sectors ($n > 1$) and an action which rotates the disk by $1/n$ of a full turn. One way to describe the system is to use the fluent name *Orientation* whose range consists of n symbols $\overline{0}, \ldots, \overline{n-1}$. The effect of the action *Turn* can be described by the proposition

$$\textit{Turn} \textbf{ causes } (\textit{Orientation} \textbf{ is } \overline{(i+1) \bmod n}\,) \textbf{ if } (\textit{Orientation} \textbf{ is } \overline{i}\,)$$

for $i = 1, \ldots, n-1$. Alternatively, we can introduce n propositional fluent names $\textit{Orientation}^i$ $(0 \leq i < n)$ and the propositions

$$\begin{array}{l} \textit{Turn} \textbf{ causes } \textit{Orientation}^{(i+1) \bmod n} \textbf{ if } \textit{Orientation}^i, \\ \textbf{always } \neg(\textit{Orientation}^i \wedge \textit{Orientation}^j), \\ \textbf{always } \textit{Orientation}^0 \vee \cdots \vee \textit{Orientation}^{n-1} \end{array}$$

$(0 \leq i < j < n)$.

4.5 Indeterminate Effect Propositions

If an indeterminate effect proposition

$$A \textbf{ possibly changes } F \textbf{ if } P$$

in an action description is replaced by a determinate effect proposition of the form

$$A \textbf{ causes } (F \textbf{ is } V) \textbf{ if } P$$

then the sets $Res(A, \sigma)$ will typically become smaller. Consider, for instance, the proposition

$$J \textbf{ possibly changes } H \textbf{ if } H \tag{7}$$

from the lake example. In Section 3.3 we saw that, for every state σ with $\sigma(H) = \mathsf{T}$,

$$Res(J, \sigma) = \{LWH, LW\overline{H}\}.$$

If we replace (7) by

$$J \textbf{ causes } (H \textbf{ is } \mathsf{F}) \textbf{ if } H$$

then the lake example will become "more determinate": the set $Res(J, \sigma)$ of the possible outcomes of the action J will lose the first of its two elements.

The theorem below shows that a state is a possible outcome of an action A in the original description if and only if it is a possible outcome of A in at least one of the "more determinate" descriptions corresponding to different choices of V in the range of F. Moreover, this process can be applied to many indeterminate effect propositions at once—for instance, to all indeterminate effect propositions in the given description.

Let S be a set of indeterminate effect propositions. A *choice function* for S is a function c such that

- the domain of c is the set of the fluent names F in all propositions (3) in S,
- for every F in the domain of c, $c(F) \in Rng_F$.

For any choice function c, S^c stands for the set of the determinate effect propositions

$$A \textbf{ causes } (F \textbf{ is } c(F)) \textbf{ if } P$$

for all propositions (3) in S.

Theorem 7. *For any action description D, any set S of indeterminate effect propositions, any action name A and any state σ of D, $Res_{D \cup S}(A, \sigma)$ is the union of the sets $Res_{D \cup S^c}(A, \sigma)$ over all choice functions c for S.*

5 Temporal Projection

As observed at the end of Section 3.3, some commonsense conclusions about the effects of actions described in $\mathcal{AR}$ can be viewed as assertions about paths in the corresponding transition diagram. In this section we introduce a "query language" that can be used, in conjunction with $\mathcal{AR}$, to describe properties of such paths.

Let us go back to the lake example. Even if nothing is known about the current state of affairs, we can predict that, should Mary jump into the lake, get out of the water and then put her hat on, she will be not in the lake but still wet, and she will have her hat on. This is an example of "temporal projection," that is, predicting the future on the basis of what is known about the effects of actions. Symbolically, this conclusion will be expressed in the query language defined below as follows:

$$\neg L \wedge W \wedge H \textbf{ after } J; G; P. \tag{8}$$

A temporal projection problem may include, in addition to an action description, some assumptions about the initial state of the world. In the lake example, if we know that

$$\textbf{initially } L \wedge H$$

—Mary is in the lake and has her hat on right now—then we can predict that, should she return to the shore, her hat will be still on her head:

$$H \textbf{ after } G. \tag{9}$$

As in Section 2, consider a set of fluent names with their ranges, some of them designated as inertial, and a set of action names. An *initial condition* is an expression of the form

$$\textbf{initially } C \tag{10}$$

where C is a formula. A *value proposition* is an expression of the form

$$C \textbf{ after } A_1; \ldots; A_n \tag{11}$$

where C is a formula and $A_1, \ldots, A_n$ $(n > 0)$ are action names. Value propositions are conditions on the values that fluents would have should a certain sequence of actions be executed.

A *domain description* is the union of an action description and a set of initial conditions. We will define when a value proposition is a "consequence" of a given domain description.

First, we need two auxiliary definitions. A *history* for an action description D is a path in the corresponding transition diagram, that is, a finite sequence

$$\sigma_0, A_1, \sigma_1, \ldots, A_n, \sigma_n \tag{12}$$

($n \geq 0$) such that $\sigma_0, \sigma_1, \ldots, \sigma_n$ are states, $A_1, \ldots, A_n$ are action names, and

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

A history (12) *satisfies* an initial condition (10) if σ_0 satisfies C.

Consider now an action description D, a set I of initial conditions, and a value proposition (11). We say that (11) is a *consequence* of the domain description $D \cup I$ if, for any history for D of the form $\sigma_0, A_1, \sigma_1, \ldots, A_n, \sigma_n$ that satisfies all initial conditions in I, σ_n satisfies C.

For example, (8) is a consequence of action description (4), and (9) is a consequence of the domain description obtained from (4) by adding the initial condition

$$\textbf{initially } L \wedge H. \tag{13}$$

Note that by asserting that (11) is a consequence of a certain domain description we do not claim that the actions $A_1, \ldots, A_n$ can be executed. On the contrary, in the case when this string of actions is not executable, the consequence relation trivially holds. For instance, if instead of (13) we add to (4) the initial condition

$$\textbf{initially } H$$

or even

$$\textbf{initially } \neg L \wedge H,$$

value proposition (9) will still be a consequence.

The consequence relation defined above is nonmonotonic, because some consequences of $D \cup I$ can be lost when a proposition is added to D. Adding an initial condition, however, can only make the set of consequences bigger. This fact is a restricted monotonicity property in the sense of [16].

6 Temporal Projection Problems as Abnormality Theories

In this section, we assume that the underlying language has finitely many fluent names, each with a finite range, and finitely many action names. We restrict attention to finite action descriptions in this language. An arbitrary temporal projection problem associated with such a description is encoded here, in a simple and modular fashion, as a problem of reasoning in a nested abnormality theory [18] based on the situation calculus.

The abnormality theory corresponding to a given finite action description D and a set I of initial conditions will be denoted by $NAT(D, I)$.

6.1 The Language of $NAT(D, I)$

The language of the theory is many-sorted, with the following sorts:

1. *Actions.* The universe of actions will be in a one-to-one correspondence with the set of action names of D.

2. *Values.* The universe of values will be in a one-to-one correspondence with the union of the ranges of the fluent names of D.

3. *Situations.* Intuitively, a situation "is the complete state of the universe at an instant of time" [24]. The universe of situations will include also an auxiliary object which stands for "undefined." It will help us represent actions that are not always executable.

4. *Aspects.* As in [22], aspects will be used to distinguish between different kinds of abnormality.

The variables of the first three sorts will be denoted by $a, a_1, a_2, \ldots$, $v, v_1, v_2, \ldots$ and $s, s_1, s_2, \ldots$.

The language includes the following object constants:

- every action name of D is an action constant,

- every element of the union of the ranges of the fluent names of D is a value constant,

- S_0 (for the initial situation) and $\perp$ ("undefined") are situation constants,

- for every inertial fluent name F of D, $\widetilde{F}$ is an aspect constant.

Finally, the language includes the following function and predicate constants:

- *Result* represents a function that maps an action and a situation to a situation,
- every fluent name F of D represents a function that maps a situation to a value,
- for every fluent name F of D, F^R represents the function that is explicitly defined by the formula
$$F^R(a, s) = F(Result(a, s)), \tag{14}$$
- *Poss* represents the predicate that is explicitly defined by the formula
$$Poss(a, s) \equiv Result(a, s) \neq \bot. \tag{15}$$

Note that formulas in the sense of $\mathcal{AR}$ (Section 2.1) are not among the formulas of the first-order language with these nonlogical constants. To avoid confusion, we will refer here to the formulas in the sense of $\mathcal{AR}$ as "domain formulas." For any domain formula C and any situation term t, by $C(t)$ we denote the formula obtained from C as the result of replacing each atomic part F **is** V by $F(t) = V$. For instance,

$$(\neg(Orientation \textbf{ is } \overline{2}))(S_0)$$

stands for

$$Orientation(S_0) \neq \overline{2}.$$

For any domain formula C, and action term t_1 and any situation term t_2, by $C_R(t_1, t_2)$ we denote the formula obtained from C as the result of replacing each atomic part F **is** V by $F^R(t_1, t_2) = V$.

6.2 The Axioms of $NAT(D, I)$

The commonsense law of intertia will be expressed in $NAT(D, I)$ by the formulas

$$Poss(a, s) \wedge v = F^R(a, s) \wedge \neg Ab(\widetilde{F}, v, a, s) \supset v = F(s)$$

for all inertial fluent names F ("normally, if v is the value of F after executing an action then v equals the value that F had previously"). We will denote the list of these formulas by LI.

The formulas expressing the effect propositions from D are defined as follows. If Q is a determinate effect proposition (2) then $\widehat{Q}$ is the formula

$$P(s) \wedge Poss(A, s) \supset C_R(A, s).$$

For instance, the translation of

$$\mathit{Turn}\ \textbf{causes}\ (\mathit{Orientation}\ \textbf{is}\ \overline{6})\ \textbf{if}\ (\mathit{Orientation}\ \textbf{is}\ \overline{5})$$

is

$$\mathit{Orientation}(s) = \overline{5} \wedge Poss(\mathit{Turn}, s) \supset \mathit{Orientation}_R(\mathit{Turn}, s) = \overline{6}.$$

If Q is an indeterminate effect proposition (3) then $\widehat{Q}$ is the formula

$$P(s) \supset Ab(\widetilde{F}, F^R(A, s), A, s).$$

Let $F_1, \ldots, F_l$ be all fluent names in the language of D. In the following list of axioms of $NAT(D, I)$, D_d stands for the set of all determinate propositions in D, D_i stands for the set of all indeterminate propositions in D, and D_c stands for the set of domain formulas that includes

- the formulas C for all constraints **always** C in D, and
- the formulas
$$\bigvee_{V \in Rng_{F_i}} F_i\ \textbf{is}\ V$$
for all i $(1 \leq i \leq l)$.

Group 1. Unique names axioms: $c_1 \neq c_2$ for all pairs c_1, c_2 of distinct object constants of the same sort. In particular, this group includes the axiom $S_0 \neq \bot$.

Group 2. Domain closure axioms:

$$a = A_1 \vee \cdots \vee a = A_m,$$
$$v = V_1 \vee \cdots \vee v = V_n,$$

where $A_1, \ldots, A_m$ are all action constants, and $V_1, \ldots, V_n$ are all value constants in the language.

Group 3. Translations of the constraints:

$$C(s) \qquad (C \in D_c).$$

Group 4. Explicit definitions: definitions (14) of F^R for all fluent names F, and definition (15) of *Poss*.

Group 5. Characterization of the effects of actions:

$$\begin{array}{l}
\{F_1^R, \ldots, F_l^R : \\
\quad LI, \\
\quad \widehat{Q} \qquad (Q \in D_i), \\
\quad \{F_1^R, \ldots, F_l^R, \max Poss : \\
\quad\quad \neg Poss(a, \bot), \\
\quad\quad \widehat{Q} \qquad (Q \in D_d), \\
\quad\quad C_R(a, s) \qquad (C \in D_c) \\
\quad \} \\
\}.
\end{array} \tag{16}$$

Group 6. Translations of the initial conditions:

$$C(S_0)$$

for every proposition **initially** C in I.

Group 5 is, of course, the main part of the theory. The inner block tells us that an action can be executed unless this is prohibited by the determinate effect propositions and constraints of D. The outer circumscription encodes the idea of inertia. The nesting of blocks reflects our intention to decide first which actions can be executed, and then what the effects of these actions are.

6.3 The Soundness and Completeness Theorem

To express value propositions in the language of the situation calculus, we need the following notation: for any actions $A_1, \ldots, A_n$,

$$[A_1, \ldots, A_n]$$

stands for the term

$$Result(A_n, Result(A_{n-1}, \ldots, Result(A_1, S_0) \cdots)).$$

If Q is a value proposition (11) then $\widehat{Q}$ stands for

$$[A_1, \ldots, A_n] \neq \bot \supset C([A_1, \ldots, A_n]).$$

The following theorem expresses the soundness and completeness of the translation described above.

Theorem 8. *For any finite action description D, any set I of initial conditions and any value proposition Q, $\widehat{Q}$ is a consequence of the nested abnormality theory $NAT(D, I)$ iff Q is a consequence of the domain description $D \cup I$.*

7 Related Work

Early attempts to describe properties of actions in classical logic have led to the discovery of the frame problem—the problem of specifying which facts do *not* change when an action is performed. The assumption that "if a person has a telephone, he still has it after looking up a number in the telephone book" [24] is an example. Methods have been developed for expressing such "frame axioms" in a systematic and compact way [29], [26], [5].

An alternative approach to the frame problem is to formalize, once and for all, the commonsense law of inertia, which, in combination with any set of domain-specific effect axioms, would lead to exactly the same conclusions as the appropriate set of frame axioms. This idea was among the first examples of default reasoning that motivated the development of nonmonotonic logics, including circumscription. However, the first circumscriptive solution to the frame problem [22] turned out to be unsatisfactory [10]. Among the formalizations proposed in response to this criticism, [1] was particularly influential, because it could handle actions with indirect effects; see [12] on its limitations. A survey of nonmonotonic solutions to the frame problem can be found in [28]. It can be argued that the difference between the two kinds of theories of action—"classical" and "nonmonotonic"—is not as significant as commonly thought ([18], Section 5.3).

In research on action, it turned out to be difficult to discuss the possibilities and limitations of the available methods in a precise and general way. For a long time, the tradition was to explain every new approach with reference to a few standard examples, such as the blocks world or the shooting scenario from [10]. Competing approaches used to be evaluated and compared mostly in terms of their ability to handle these examples and their enhancements. Such analysis does not say much about the range of applicability of each method. More recently, several researchers attempted to overcome this problem and to discuss representing action in a methodical and theoretically sound way. Three approaches to the systematic study of actions that are being pursued today most actively are associated with the ideas of a causal theory [20], of a dynamical system [27], and of an action language [7].

Some extensions of the original action language $\mathcal{A}$ proposed in the literature include features that are not incorporated in the language $\mathcal{AR}$ studied in this paper: parameters [4], concurrency [2], dependent fluents [9] and static causal laws [21], [30], [19]. Combining these proposals in one action language is a topic for future work.

The circumscriptive approach to action presented in Section 6 uses nested abnormality theories to combine the "theory update" view of [31] with the syntax of the situation calculus. Some computational experiments based on a similar use of circumscription are described in [15].

Acknowledgments

We are grateful to Michael Gelfond, Fangzhen Lin, Norman McCain and Hudson Turner for comments on a draft of this paper. This work was partially supported by National Science Foundation under grant IRI-9306751.

Appendix. Proofs

A.1 Proof of Theorem 1 (Section 3.4)

Theorem 1. *Let D be a simple action description. For any action name A and any state σ,*

$$Res_D(A,\sigma) = \{Res^*_D(A,\sigma)\}.$$

Consider an action name A and a state σ. Recall that σ' is defined as follows: For any fluent name F, if D includes a determinate effect proposition (2) such that

(i) σ satisfies P, and

(ii) C has a conjunctive term of the form F **is** V

then $\sigma'(F) = V$; if not then $\sigma'(F) = \sigma(F)$.

It is clear that $\sigma' \in Res^0(A, \sigma)$. We will show that, for any valuation σ'' in $Res^0(A, \sigma)$ that is different from σ', $New^A(\sigma, \sigma')$ is a proper subset of $New^A(\sigma, \sigma'')$.

Let Δ be the set of fluent names F such that D includes a determinate effect proposition (2) with properties (i) and (ii). Since $\sigma'' \in Res^0(A, \sigma)$,

$$\sigma'(F) = \begin{cases} \sigma''(F), & \text{if } F \in \Delta, \\ \sigma(F), & \text{otherwise.} \end{cases}$$

Consequently,

$$\begin{array}{rcl} New^A(\sigma, \sigma') & = & \{F \textbf{ is } \sigma'(F) \ : \ \sigma'(F) \neq \sigma(F)\} \\ & = & \{F \textbf{ is } \sigma'(F) \ : \ F \in \Delta,\ \sigma'(F) \neq \sigma(F)\} \\ & = & \{F \textbf{ is } \sigma''(F) \ : \ F \in \Delta,\ \sigma''(F) \neq \sigma(F)\} \\ & \subseteq & New^A(\sigma, \sigma''). \end{array}$$

To see that the two sets cannot be equal, consider any fluent name F such that $\sigma'(F) \neq \sigma''(F)$. For this F, $\sigma'(F) = \sigma(F)$, because otherwise F **is** $\sigma'(F)$ would be in $New^A(\sigma, \sigma')$ and hence in $New^A(\sigma, \sigma'')$, contrary to the choice of F. It follows that $\sigma''(F) \neq \sigma(F)$, so that F **is** $\sigma''(F)$ is in $New^A(\sigma, \sigma'')$, but not in $New^A(\sigma, \sigma')$.

A.2 Proof of Theorem 2 (Section 4.1)

Theorem 2. *Let S be a set of constraints, D an action description, and F, F' fluent names equivalent with respect to S. If an action description D' is obtained from D by replacing some occurrences of F with F' then*

$$Res_{S \cup D} = Res_{S \cup D'}.$$

Consider two different fluent names F, F' that are equivalent with respect to S. If C' is obtained from a formula C by replacing some occurrences of F with F' then, for every valuation σ satisfying S, $\sigma(C) = \sigma(C')$. It follows that $S \cup D$ and $S \cup D'$ have the same states, and, furthermore,

$$Res^0_{S \cup D} = Res^0_{S \cup D'}.$$

Take an action name A and states σ, σ', σ'' of $S \cup D$. We need to verify that

$$New^A_{S \cup D}(\sigma, \sigma') \subseteq New^A_{S \cup D}(\sigma, \sigma'') \tag{17}$$

iff

$$New^A_{S\cup D'}(\sigma,\sigma') \subseteq New^A_{S\cup D'}(\sigma,\sigma''). \tag{18}$$

Let W be the set of the fluent names F^* such that D includes an indeterminate effect proposition

$$A \textbf{ possibly changes } F^* \textbf{ if } P \tag{19}$$

for which σ satisfies P. For any state σ^*, $New^A_{S\cup D}(\sigma,\sigma^*)$ can be characterized as the set of all atomic formulas F^* **is** V such that

(i) F^* is inertial and $V = \sigma^*(F^*) \neq \sigma(F^*)$, or

(ii) $F^* \in W$ and $V = \sigma^*(F^*)$.

Similarly, let W' be the set of the fluent names F^* such that D' includes a proposition (19) for which σ satisfies P; then $New^A_{S\cup D'}(\sigma,\sigma^*)$ is the set of all atomic formulas F^* **is** V that satisfy (i) or

(ii′) $F^* \in W'$ and $V = \sigma^*(F^*)$.

Assume (17). To prove (18), take any formula F^* **is** V in $New^A_{S\cup D'}(\sigma,\sigma')$. *Case (i):* F^* is inertial and $V = \sigma'(F^*) \neq \sigma(F^*)$. Then F^* **is** V belongs to the left-hand side of (17), and consequently to the right-hand side also, so that $V = \sigma''(F^*)$. It follows that F^* **is** V belongs to $New^A_{S\cup D'}(\sigma,\sigma'')$. *Case (ii):* $F^* \in W'$ and $V = \sigma'(F^*)$. If $F^* \in W$ then, as in Case (i), we conclude by (17) that $V = \sigma''(F^*)$, which again implies that F^* **is** V belongs to $New^A_{S\cup D'}(\sigma,\sigma'')$. Otherwise $F^* \in W' \backslash W$. Recall that D' is obtained from D by replacing some occurrences of F with F', so that

$$W \cup \{F'\} = W' \cup \{F\}, \tag{20}$$

and consequently $F^* = F'$ and $V = \sigma'(F')$. Moreover, equality (20) implies that $F' \in W'$ and $F \in W$. From the last condition we see that F **is** $\sigma'(F)$ belongs to the left-hand side of (17). Consequently, it belongs to the right-hand side also, so that $\sigma'(F) = \sigma''(F)$. Since both σ' and σ'' satisfy S, $\sigma'(F) = \sigma'(F')$ and $\sigma''(F) = \sigma''(F')$. Thus

$$V = \sigma'(F') = \sigma'(F) = \sigma''(F) = \sigma''(F').$$

Since $F' \in W'$, it follows that the formula F^* **is** V is in $New^A_{S\cup D'}(\sigma,\sigma'')$.

The proof in the other direction, from (18) to (17), is similar.

A.3 Proof of Theorem 3 (Section 4.1)

Theorem 3. *Let S be a set of constraints, D an action description, and B, B' formulas equivalent with respect to S. If an action description D' is obtained from D by replacing some occurrences of B with B' then*

$$Res_{S\cup D} = Res_{S\cup D'}.$$

Consider formulas B, B' equivalent with respect to S. If C' is the formula obtained from a formula C by replacing some occurrences of B with B' then, for every valuation σ satisfying S, $\sigma(C) = \sigma(C')$. It follows that $S \cup D$ and $S \cup D'$ have the same states, and, furthermore,

$$Res^0_{S\cup D} = Res^0_{S\cup D'},$$

$$New^A_{S\cup D} = New^A_{S\cup D'},$$

$$Res_{S\cup D} = Res_{S\cup D'}.$$

A.4 Proof of Theorem 4 (Section 4.2)

Theorem 4. *Let S be a set of constraints, and let S' be the set of determinate effect propositions A* **causes** *C for all constraints* **always** *C in S and all action names A. For any action description D, any action name A and any state σ of $S \cup D$,*

$$Res_{S\cup D}(A, \sigma) = Res_{S'\cup D}(A, \sigma).$$

Let S and S' be as in the statement of the theorem, and let σ be a state of $S \cup D$. A valuation σ' belongs to $Res^0_{S'\cup D}(A, \sigma)$ iff it belongs to $Res^0_D(A, \sigma)$ and satisfies all constraints in S, that is to say, iff it belongs to $Res^0_{S\cup D}(A, \sigma)$. Consequently,

$$Res^0_{S\cup D}(A, \sigma) = Res^0_{S'\cup D}(A, \sigma).$$

Since $S \cup D$ and $S' \cup D$ have the same indeterminate effect propositions,

$$New^A_{S\cup D}(\sigma, \sigma') = New^A_{S'\cup D}(\sigma, \sigma').$$

These two formulas imply the assertion of the theorem.

A.5 Proof of Theorem 5 (Section 4.3)

Theorem 5. *Let D be an action description containing an explicit definition of a propositional fluent name F, and let D' be obtained from D by deleting this definition and by deleting F from the set of fluent names. For any state σ of D, let σ_F be the restriction of σ to the set of fluent names that are different from F. The function $\sigma \mapsto \sigma_F$ is an isomorphism between D and D'.*

It is clear that $\sigma \mapsto \sigma_F$ is a one-to-one function from the set of states of D onto the set of states of D'. We need to show that this function maps $Res_D(A,\sigma)$ onto $Res_{D'}(A,\sigma_F)$.

Observe first that it maps $Res^0_D(A,\sigma)$ onto $Res^0_{D'}(A,\sigma_F)$. Indeed, if $\sigma' \in Res^0_D(A,\sigma)$ then $\sigma'_F \in Res^0_{D'}(A,\sigma_F)$. On the other hand, if a valuation σ^* belongs to $Res^0_{D'}(A,\sigma_F)$ then $\sigma^* = \sigma'_F$ for the valuation σ' defined by

$$\sigma'(F') = \begin{cases} \sigma(C), & \text{if } F' = F, \\ \sigma^*(F'), & \text{otherwise,} \end{cases}$$

where C is the right-hand side of the equivalence (6) defining F. Clearly $\sigma' \in Res^0_D(A,\sigma)$.

Now it remains to note that, for any states σ, σ' of D, since F is noninertial and does not occur in indeterminate effect propositions,

$$New^A_D(\sigma,\sigma') = New^A_{D'}(\sigma_F,\sigma'_F).$$

A.6 Proof of Theorem 6 (Section 4.4)

Theorem 6. *Let D be an action description, and let $\mathbf{F}$ be a set of fluent names in the language of D such that every fluent name in $\mathbf{F}$ has a finite range. The function $\sigma \mapsto \sigma_{\mathbf{F}}$ is an isomorphism between D and $D_{\mathbf{F}}$.*

It is clear that $\sigma \mapsto \sigma_{\mathbf{F}}$ is a one-to-one function from the set of states of D into the set of valuations of $D_{\mathbf{F}}$. Furthermore, for every formula C in the language of D, $\sigma(C) = \sigma_{\mathbf{F}}(C_{\mathbf{F}})$; consequently, the range of this function is the set of states of $D_{\mathbf{F}}$. To complete the proof, we need to show that, for every action name A and every state σ of D, this function maps $Res_D(A,\sigma)$ onto $Res_{D_{\mathbf{F}}}(A,\sigma_{\mathbf{F}})$. It is clear that the image of $Res^0_D(A,\sigma)$ is $Res^0_{D_{\mathbf{F}}}(A,\sigma_{\mathbf{F}})$. Thus, we only need to verify that for any states σ', σ'' of D, the inclusion

$$New^A_D(\sigma,\sigma') \subseteq New^A_D(\sigma,\sigma'') \tag{21}$$

is equivalent to

$$New^A_{D_{\mathbf{F}}}(\sigma_{\mathbf{F}}, \sigma'_{\mathbf{F}}) \subseteq New^A_{D_{\mathbf{F}}}(\sigma_{\mathbf{F}}, \sigma''_{\mathbf{F}}). \tag{22}$$

Assume (21), and take any formula in $New^A_{D_{\mathbf{F}}}(\sigma_{\mathbf{F}}, \sigma'_{\mathbf{F}})$. If this formula is F **is** V for some $F \notin \mathbf{F}$ then it belongs to the left-hand side of (21), and consequently to the right-hand side too, which implies that it belongs to the right-hand side of (22). Otherwise, this formula has the form F^V **is** $\sigma'_{\mathbf{F}}(F^V)$ for some $F \in \mathbf{F}$. *Case 1:* F^V is inertial and $\sigma'_{\mathbf{F}}(F^V) \neq \sigma_{\mathbf{F}}(F^V)$. Then F is inertial and $\sigma'(F) \neq \sigma(F)$, so that F **is** $\sigma'(F)$ belongs to the the left-hand side of (21). Consequently, it belongs to the right-hand side also, which implies that $\sigma''(F) = \sigma'(F)$. Then $\sigma''_{\mathbf{F}}(F^V) = \sigma'_{\mathbf{F}}(F^V)$, so that F^V **is** $\sigma'_{\mathbf{F}}(F^V)$ belongs to the right-hand side of (22). *Case 2:* for some indeterminate effect proposition (3) in D, $\sigma_{\mathbf{F}}$ satisfies $P_{\mathbf{F}}$. Then σ satisfies P, and we can again use (21) to conclude that $\sigma''(F) = \sigma'(F)$ and that F^V **is** $\sigma'_{\mathbf{F}}(F^V)$ belongs to the right-hand side of (22).

Now assume (22), and take any formula F **is** $\sigma'(F)$ in $New^A_D(\sigma, \sigma')$. If $F \notin \mathbf{F}$ then it belongs to the left-hand side of (22), and consequently to the right-hand side too, which implies that it belongs to the right-hand side of (21). Assume that $F \in \mathbf{F}$. *Case 1:* F is inertial and $\sigma'(F) \neq \sigma(F)$. Then $F^{\sigma'(F)}$ is inertial, $\sigma'_{\mathbf{F}}(F^{\sigma'(F)}) = \mathsf{T}$ and $\sigma_{\mathbf{F}}(F^{\sigma'(F)}) = \mathsf{F}$. It follows that $F^{\sigma'(F)}$ **is** T belongs to the the left-hand side of (22). Consequently, it belongs to the right-hand side also, which implies that

$$\sigma''_{\mathbf{F}}(F^{\sigma'(F)}) = \sigma'_{\mathbf{F}}(F^{\sigma'(F)}) = \mathsf{T}.$$

Hence $\sigma''(F) = \sigma'(F)$, so that F **is** $\sigma'(F)$ belongs to the right-hand side of (21). *Case 2:* for some indeterminate effect proposition (3) in D, σ satisfies P. Then $\sigma_{\mathbf{F}}$ satisfies $P_{\mathbf{F}}$, and we can again use (22) to conclude that $\sigma''_{\mathbf{F}}(F^{\sigma'(F)}) = \sigma'_{\mathbf{F}}(F^{\sigma'(F)}) = \mathsf{T}$ and $\sigma''(F) = \sigma'(F)$, and that F **is** $\sigma'(F)$ belongs to the right-hand side of (21).

A.7 Proof of Theorem 7 (Section 4.5)

Theorem 7. *For any action description D, any set S of indeterminate effect propositions, any action name A and any state σ of D, $Res_{D\cup S}(A, \sigma)$ is the union of the sets $Res_{D\cup S^c}(A, \sigma)$ over all choice functions c for S.*

For any valuation σ, define S^σ to be S^c where c is the choice function obtained by restricting σ to the domain of c. We will derive the theorem from the following lemma.

Lemma. *For any state σ' of D, $\sigma' \in Res_{D\cup S}(A,\sigma)$ iff $\sigma' \in Res_{D\cup S^{\sigma'}}(A,\sigma)$.*

The fact that $Res_{D\cup S}(A,\sigma)$ is contained in the union of all sets of the form $Res_{D\cup S^c}(A,\sigma)$ immediately follows from the lemma. To prove the opposite inclusion, consider any choice function c such that $\sigma' \in Res_{D\cup S^c}(A,\sigma)$; we will show that

$$Res_{D\cup S^c}(A,\sigma) = Res_{D\cup S^{\sigma'}}(A,\sigma). \tag{23}$$

Consider a proposition

$$A \textbf{ causes } (F \textbf{ is } c(F)) \textbf{ if } P \tag{24}$$

in S^c such that σ satisfies P, and the corresponding proposition

$$A \textbf{ causes } (F \textbf{ is } \sigma'(F)) \textbf{ if } P \tag{25}$$

in $S^{\sigma'}$. Since $\sigma' \in Res^0_{D\cup S^c}(A,\sigma)$, $\sigma'(F) = c(F)$, so that (25) equals (24). Consequently,

$$Res^0_{D\cup S^c}(A,\sigma) = Res^0_{D\cup S^{\sigma'}}(A,\sigma).$$

Since $D\cup S^c$ and $D\cup S^{\sigma'}$ have the same indeterminate effect propositions, (23) immediately follows.

Proof of the Lemma. The proof is based on the following observations. Let Γ be the set of formulas F **is** $\sigma'(F)$ such that (i) S contains at least one proposition (3) for which σ satisfies P, (ii) D does not contain a proposition (3) for which σ satisfies P, and (iii) $\sigma(F) = \sigma'(F)$. (Intuitively, Γ is the set of all formulas that belong to $New^A_{D\cup S}(\sigma,\sigma')$ only because of some proposition in S.) For every σ^* in $Res^0_{D\cup S^{\sigma'}}(A,\sigma)$, if F **is** $\sigma'(F)$ belongs to Γ then $\sigma'(F) = \sigma^*(F)$. It follows that, for all $\sigma^* \in Res^0_{D\cup S^{\sigma'}}(A,\sigma)$,

$$New^A_{D\cup S}(\sigma,\sigma^*) = New^A_{D\cup S^{\sigma'}}(\sigma,\sigma^*) \cup \Gamma. \tag{26}$$

Moreover, the two sets in the right-hand side are disjoint:

$$New^A_{D\cup S^{\sigma'}}(\sigma,\sigma^*) \cap \Gamma = \emptyset. \tag{27}$$

Assume that $\sigma' \in Res_{D\cup S}(A,\sigma)$. Then σ' belongs to $Res^0_{D\cup S^{\sigma'}}(A,\sigma)$, because, for every determinate effect proposition (2) which is in $D\cup S^{\sigma'}$ but not in $D\cup S$, the formula C has the form F **is** $\sigma'(F)$. We need to show that there is no $\sigma'' \in Res^0_{D\cup S^{\sigma'}}(A,\sigma)$ such that

$$New^A_{D\cup S^{\sigma'}}(\sigma,\sigma'') \subset New^A_{D\cup S^{\sigma'}}(\sigma,\sigma').$$

Assume that such a σ'' exists. Then, by (27),

$$New^A_{D\cup S^{\sigma'}}(\sigma,\sigma'')\cup\Gamma\subset New^A_{D\cup S^{\sigma'}}(\sigma,\sigma')\cup\Gamma.$$

By (26) we conclude that

$$New^A_{D\cup S}(\sigma,\sigma'')\subset New^A_{D\cup S}(\sigma,\sigma'),$$

which contradicts the assumption that $\sigma'\in Res_{D\cup S}(A,\sigma)$.

Now assume that $\sigma'\in Res_{D\cup S^{\sigma'}}(A,\sigma)$. Then $\sigma'\in Res^0_{D\cup S}(A,\sigma)$, because every determinate effect proposition in $D\cup S$ belongs to $D\cup S^{\sigma'}$. We need to show that there is no $\sigma''\in Res^0_{D\cup S}(A,\sigma)$ such that

$$New^A_{D\cup S}(\sigma,\sigma'')\subset New^A_{D\cup S}(\sigma,\sigma'). \tag{28}$$

Assume that such a σ'' exists. We want to use (26) to simplify both parts of this inclusion. To this end, we need to prove that $\sigma''\in Res^0_{D\cup S^{\sigma'}}(A,\sigma)$. Consider any determinate effect proposition (2) which is in $D\cup S^{\sigma'}$ but not in $D\cup S$, such that σ satisfies P. This proposition has the form (25) for some indeterminate effect proposition (3) in S. Furthermore, F **is** $\sigma''(F)$ belongs to the left-hand side of (28), and consequently to the right-hand side also. It follows that $\sigma''(F)=\sigma'(F)$, so that σ'' satisfies F **is** $\sigma'(F)$. We showed that σ'' indeed belongs to $Res^0_{D\cup S^{\sigma'}}(A,\sigma)$. Now from (28) and (26) we can conclude that

$$New^A_{D\cup S^{\sigma'}}(\sigma,\sigma'')\cup\Gamma\subset New^A_{D\cup S^{\sigma'}}(\sigma,\sigma')\cup\Gamma.$$

Then, by (27),

$$New^A_{D\cup S^{\sigma'}}(\sigma,\sigma'')\subset New^A_{D\cup S^{\sigma'}}(\sigma,\sigma'),$$

which contradicts the assumption that $\sigma'\in Res_{D\cup S^{\sigma'}}(A,\sigma)$.

A.8 Proof of Theorem 8 (Section 6.3)

Theorem 8. *For any finite action description D, any set I of initial conditions and any value proposition Q, $\widehat{Q}$ is a consequence of the nested abnormality theory $NAT(D,I)$ iff Q is a consequence of the domain description $D\cup I$.*

This theorem is technically more difficult than the others, and what is presented in this section is only an outline of the proof. A complete proof of a similar result can be found in Chapter 8 of [13].

Recall that the result of circumscribing a predicate constant P, with object, function and/or predicate constants Z varied, in a sentence A is denoted by CIRC$[A; P; Z]$. The semantics of nested abnormality theories [18] is characterized by a map φ that translates blocks into sentences of the underlying language of classical logic. The map is defined recursively:

$$\varphi\{C_1, \ldots, C_m : A_1, \ldots, A_n\} = \exists ab F(ab),$$

where

$$F(Ab) = \mathrm{CIRC}[\varphi A_1 \wedge \ldots \wedge \varphi A_n; Ab; C_1, \ldots, C_m].$$

The first in the series of lemmas needed to prove Theorem 8 relates the inner block of Axiom Group 5

$$\begin{array}{l} \{F_1^R, \ldots, F_l^R, \max Poss : \\ \quad \neg Poss(a, \bot), \\ \quad \widehat{Q} \qquad (Q \in D_d), \\ \quad C_R(a, s) \qquad (C \in D_c) \\ \} \end{array} \tag{29}$$

to the function *Res*. The statement of the lemma uses the following notation. For any valuation σ, $\overline{\sigma}$ stands for the conjunction of all domain formulas of the form F **is** $\sigma(F)$. By Σ we denote the set of pairs $\langle A, \sigma\rangle$ such that $Res(A, \sigma) = \emptyset$. Note that, in this definition, the set $Res(A, \sigma)$ can be equivalently replaced by its superset $Res^0(A, \sigma)$, because, for a finite action description, the former cannot be empty unless the latter is empty also.

Lemma 1. *Assume that the set of states of D is nonempty. Axiom Groups 1–3 entail that the result of applying φ to block (29) is equivalent to the conjunction (of the universal closures) of the formulas*

$$\neg Poss(a, \bot), \tag{30}$$

$$\widehat{Q} \qquad (Q \in D_d), \tag{31}$$

$$C_R(a, s) \qquad (C \in D_c) \tag{32}$$

and

$$Poss(a, s) \equiv \neg \left[s = \bot \vee \bigvee_{\langle A, \sigma\rangle \in \Sigma} (a = A \wedge \overline{\sigma}(s)) \right]. \tag{33}$$

The proof of the lemma consists of two steps. First, introduce a new predicate constant *Imposs*, and let $A(\mathit{Imposs}, F_1^R, \ldots, F_l^R)$ be the conjunction of formulas (30)–(32) with $\neg\mathit{Imposs}$ substituted for *Poss*. Propositions 1 and 2 from [18] allow us to reduce the result of applying φ to block (29) to the circumscription of *Imposs*, with $F_1^R, \ldots, F_l^R$ varied, in $A(\mathit{Imposs}, F_1^R, \ldots, F_l^R)$. Second, this circumscription can be computed using the following general fact (Proposition 3.4.1 from [17]):

Theorem. *Let E be a predicate expression without parameters, containing neither P nor Z. If the sentences*

$$A(P, Z) \supset \exists z A(E, z)$$

and

$$A(P, Z) \supset E \leq P$$

are universally valid, then so is the sentence

$$\mathrm{CIRC}[A(P, Z); P; Z] \equiv A(P, Z) \wedge P = E. \tag{34}$$

In our case, P is *Imposs*, Z is $F_1^R, \ldots, F_l^R$, and we take E to be

$$\lambda as \left[s = \bot \vee \bigvee_{\langle A, \sigma \rangle \in \Sigma} (a = A \wedge \overline{\sigma}(s)) \right].$$

To verify the condition

$$A(\mathit{Imposs}, F_1^R, \ldots, F_l^R) \supset \exists f_1, \ldots, f_l A(E, f_1, \ldots, f_l),$$

consider the following formulas $U_i(a, s, v)$:

$$\begin{array}{rcl} U_i(a, s, v) & = & \bigvee_{\langle A, \sigma \rangle \in \Sigma} (a = A \wedge \overline{\sigma}(s) \wedge v = \sigma(F)) \vee \\ & & \bigvee_{\langle A, \sigma \rangle \notin \Sigma} (a = A \wedge \overline{\sigma}(s) \wedge v = r_{A,\sigma}(F)), \end{array}$$

where $r_{A,\sigma}$ is a fixed element of $\mathit{Res}^0(A, \sigma)$. The function f_i is selected to satisfy the condition

$$\forall as\, U_i(a, s, f_i(a, s)).$$

Using Lemma 1, we can prove the following two lemmas that together establish the validity of Theorem 8 in both directions:

Lemma 2. *For every history* $\sigma_0, A_1, \sigma_1, \ldots, A_n, \sigma_n$ *that satisfies all initial conditions in* I *there exists a model* M *of* $NAT(D, I)$ *that satisfies*

$$\overline{\sigma_n}([A_1, \ldots, A_n]) \wedge [A_1, \ldots, A_n] \neq \perp.$$

Lemma 3. *For any model* M *of of* $NAT(D, I)$, *any actions* $A_1, \ldots, A_n, A_{n+1}$ *and any states* σ *and* σ', *if* M *satisfies*

$$\overline{\sigma}([A_1, \ldots, A_n]) \wedge \overline{\sigma'}([A_1, \ldots, A_n, A_{n+1}]) \wedge [A_1, \ldots, A_n, A_{n+1}] \neq \perp$$

then $\sigma' \in Res(A_{n+1}, \sigma)$.

References

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

[2] Chitta Baral and Michael Gelfond. Representing concurrent actions in extended logic programming. In *Proc. of IJCAI-93*, pages 866–871, 1993.

[3] Patrick Doherty, Witold L ukaszewicz, and Andrzey Szał as. Computing circumscription revisited: A reduction algorithm. *Journal of Automated Reasoning*, 1995. To appear.

[4] Phan Minh Dung. Representing actions in logic programming and its applications in database updates. In *Logic Programming: Proceedings of the Tenth Int'l Conf. on Logic Programming*, pages 222–238, 1993.

[5] Charles Elkan. Reasoning about action in first-order logic. In *Proc. of the 1992 Canadian Conf. on Artificial Intelligence*, 1992.

[6] 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.

[7] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.

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

[9] Enrico Giunchiglia and Vladimir Lifschitz. Dependent fluents. In *Proc. IJCAI-95*, pages 1964–1969, 1995.

[10] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, 1987.

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

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

[13] G. Neelakantan Kartha. *A Mathematical Investigation of Reasoning about Actions*. PhD thesis, University of Texas at Austin, 1995. (Available by anonymous ftp from ftp.cs.utexas.edu as /pub/techreports/tr95-17.ps).

[14] 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.

[15] G. Neelakantan Kartha and Vladimir Lifschitz. A simple formalization of actions using circumscription. In *Proc. IJCAI-95*, pages 1970–1975, 1995.

[16] Vladimir Lifschitz. Restricted monotonicity. In *Proc. AAAI-93*, pages 432–437, 1993.

[17] Vladimir Lifschitz. Circumscription. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *The Handbook of Logic in AI and Logic Programming*, volume 3, pages 298–352. Oxford University Press, 1994.

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

[19] Vladimir Lifschitz. Two components of an action language. *Annals of Mathematics and Artificial Intelligence*, 1997. To appear.

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

[21] Norman McCain and Hudson Turner. A causal theory of ramifications and qualifications. In *Proc. of IJCAI-95*, pages 1978–1984, 1995.

[22] John McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 26(3):89–116, 1986. Reproduced in [23].

[23] John McCarthy. *Formalizing common sense: papers by John McCarthy.* Ablex, Norwood, NJ, 1990.

[24] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh, 1969. Reproduced in [23].

[25] 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.

[26] 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.

[27] Erik Sandewall. *Features and fluents*, volume 1. Oxford University Press, 1995.

[28] Erik Sandewall and Yoav Shoham. Non-monotonic temporal reasoning. In D.M. Gabbay, C.J. Hogger, and J.A. Robinson, editors, *The Handbook of Logic in AI and Logic Programming*, volume 4, pages 439–498. Oxford University Press, 1995.

[29] Lenhart Schubert. Monotonic solution of the frame problem in the situation calculus: an efficient method for worlds with fully specified actions. In H.E. Kyburg, R. Loui, and G. Carlson, editors, *Knowledge Representation and Defeasible Reasoning*, pages 23–67. Kluwer, 1990.

[30] Hudson Turner. Representing actions in logic programs and default theories: a situation calculus approach. *Journal of Logic Programming*, 1997. To appear.

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