Representing Actions in Default Logic: A Situation Calculus Approach

Hudson Turner
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712, USA
hudson@cs.utexas.edu

DRAFT

November 2, 1995

1 Introduction

In this paper we address the problem of representing commonsense knowledge about action domains in the formalism of default logic. We employ a methodology proposed by Gelfond and Lifschitz [GL93] which involves first defining a high-level language for representing commonsense knowledge about actions, and then specifying a translation from the high-level action language into a general-purpose formalism, such as default logic. The advantage of starting with a high-level action language is that it can utilize a restricted syntax and a relatively simple semantics which is nonetheless adequate to capture our commonsense intuitions about the whole family of action domains expressible in the language. Moreover, given a suitable action language and a translation from that language into a general-purpose formalism, the correctness of our representations of knowledge about actions in the general-purpose formalism becomes a precise mathematical question, to be settled by proving the translation sound and complete.

Accordingly, we define a high-level action language $\mathcal{AC}$, and specify sound and complete translations of portions of $\mathcal{AC}$ into default logic.

The action language $\mathcal{AC}$ is modeled after the language $\mathcal{AR}$ of Giunchiglia, Kartha and Lifschitz [GKL95]. $\mathcal{AR}$ was introduced as an improvement on the language $\mathcal{AR}_0$ [KL94], adding non-boolean fluents to $\mathcal{AR}_0$ and replacing the so-called "release" propositions in $\mathcal{AR}_0$ with "possibly changes" propositions. Since $\mathcal{AC}$ does not allow non-boolean fluents, but does include "possibly changes" propositions, $\mathcal{AC}$ essentially builds on propositional $\mathcal{AR}$.

The language $\mathcal{AC}$ improves on propositional $\mathcal{AR}$ by adopting the method for representing causal background knowledge that was proposed by McCain and Turner in [MT95b]. In $\mathcal{AR}$, background knowledge is represented in the form of state constraints, which have been traditionally used for this purpose. Recently, many authors have argued that state constraints are inadequate for representing background knowledge in action domains, because they do not adequately represent causal relations [Gef90, Elk92, BH93, Bar95, Lin95, MT95b, Thi95]. Accordingly, $\mathcal{AC}$ includes propositions that express causal background knowledge of the following kind: in the action domain being described, one can make a fluent formula ψ true by making a fluent formula ϕ true. In [MT95b] we argued for the usefulness of such propositions, and showed that they are more expressive than state constraints.

Consider for example yet another variant of the Yale Shooting domain [HM87]. There is a pilgrim and a turkey. The pilgrim has two guns. If the pilgrim fires a loaded gun, the turkey will not be alive afterwards. If the turkey is not alive it cannot be trotting. Suppose that initially the turkey is trotting and at least one of the two guns is loaded. We can conclude that the turkey will not be trotting if the pilgrim shoots his two guns, one after the other.

In the action language $\mathcal{AC}$, this action domain can be formalized as follows.[1]

$$\begin{array}{l} \textbf{initially } \mathit{Trotting} \\ \textbf{initially } \mathit{Loaded}(\mathit{Gun}_1) \vee \mathit{Loaded}(\mathit{Gun}_2) \\ \neg \mathit{Alive} \textbf{ determines } \neg \mathit{Trotting} \\ \mathit{Shoot}(x) \textbf{ causes } \neg \mathit{Alive} \textbf{ if } \mathit{Loaded}(x) \end{array}$$

This $\mathcal{AC}$ domain description entails, for instance, the $\mathcal{AC}$ proposition

$$\neg \mathit{Trotting} \textbf{ after } \mathit{Shoot}(\mathit{Gun}_1); \mathit{Shoot}(\mathit{Gun}_2)$$

which says that $\neg \mathit{Trotting}$ holds in the situation that results from performing the action sequence $\mathit{Shoot}(\mathit{Gun}_1); \mathit{Shoot}(\mathit{Gun}_2)$ in the initial situation.

The above domain description includes one proposition in particular, namely

$$\neg \mathit{Alive} \textbf{ determines } \neg \mathit{Trotting}$$

that should be remarked on further at this point. This $\mathcal{AC}$ proposition describes a causal relationship between two fluent formulas. It says that, in

[1]Although $\mathcal{AC}$ domain descriptions are propositional, we sometimes use meta-variables in our representations of them. For instance, the meta-variable x in the fourth expression below ranges over $\{\mathit{Gun}_1, \mathit{Gun}_2\}$.

the action domain we are describing, $\neg Trotting$ can be made true by making $\neg Alive$ true. We emphasize that in the semantics of $\mathcal{AC}$ it does not follow from this that *Alive* can be made true by making *Trotting* true!

This action domain is correctly formalized by the following default theory, which can be obtained from the above domain description by a translation defined in this paper.

$$\frac{\neg Holds(Trotting, S_0)}{False} \qquad \frac{\neg(Holds(Loaded(Gun_1), S_0) \vee Holds(Loaded(Gun_2), S_0))}{False}$$

$$\frac{\neg Holds(Alive, s)}{\neg Holds(Trotting, s)} \qquad \frac{Holds(Loaded(x), s)}{\neg Holds(Alive, Result(Shoot(x), s))}$$

$$\frac{: Holds(f, S_0)}{Holds(f, S_0)} \qquad \frac{: \neg Holds(f, S_0)}{\neg Holds(f, S_0)}$$

$$\frac{Holds(f, s) : Holds(f, Result(a, s))}{Holds(f, Result(a, s))} \qquad \frac{\neg Holds(f, s) : \neg Holds(f, Result(a, s))}{\neg Holds(f, Result(a, s))}$$

The first rule reflects the assertion that the turkey is initially trotting, by ensuring that there can be no consistent extension of the default theory in which the turkey is initially not trotting. Similarly, the second rule says that at least one of the pilgrim's guns is initially loaded. The third rule says that the turkey can be made to stop trotting by making it not alive. The fourth rule says that the turkey is not alive after the pilgrim fires a gun, if that gun is loaded. The fifth and sixth rules encode the fact that each fluent is either true or false in the initial situation.[2] The seventh and eighth rules express the "commonsense law of inertia" — the principle that things don't change unless they are made to.

Recall that one consequence of this action domain is that the turkey is not trotting after the pilgrim shoots his two guns. Accordingly, this default theory entails the following literal.

$$\neg Holds(Trotting, Result(Shoot(Gun_2), Result(Shoot(Gun_1), S_0)))$$

After preliminary definitions (Section 2), we define in Section 3 the action language $\mathcal{AC}$. In Section 4 we specify a sound and complete translation into default logic of those $\mathcal{AC}$ domain descriptions in which all action preconditions are explicit. We also specify a second, simpler embedding for those $\mathcal{AC}$ domain descriptions in which there are no action preconditions whatsoever. We conclude in Section 5 with a brief comparison of our translation of the Yale Shooting domain with the default theories considered by Hanks and McDermott [HM87] and Morris [Mor88]. All proofs are omitted.

[2]This obvious fact about the initial situation must be made explicit in our default theory, since the law of excluded middle does not hold for default logic extensions.

2 Preliminary Definitions

Given a propositional language, we represent interpretations of the language by maximal consistent sets of literals. We write inference rules as expressions of the form

$$\frac{\phi}{\psi}$$

where ϕ and ψ are formulas. We often find it convenient to identify a formula ϕ with the inference rule $\frac{True}{\phi}$.

Let R be a set of inference rules and Γ a set of formulas. We say Γ is *closed under* R if for every rule $\frac{\phi}{\psi}$ in R, if ϕ belongs to Γ then ψ does too. By $Cn(R)$ we denote the least logically closed set of formulas that is closed under R.

A *default rule* is an expression of the form

$$\frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma}$$

where all of $\alpha, \beta_1, \ldots, \beta_n, \gamma$ are formulas ($n \geq 0$). Let r be such a default rule. We call α the *prerequisite* of r, and denote it by $pre(r)$. We call the formulas $\beta_1, \ldots, \beta_n$ the *justifications* of r, and write $just(r)$ to denote the set $\{\beta_1, \ldots, \beta_n\}$. We call γ the *consequent* of r, and denote it by $cons(r)$. If $pre(r) = True$ we often omit it and write $\frac{: \beta_1, \ldots, \beta_n}{\gamma}$ instead. If $just(r)$ is empty, we identify r with the corresponding inference rule $\frac{\alpha}{\gamma}$.

A *default theory* is a set of default rules. Let D be a default theory and let E be a set of formulas. We define the *reduct of* D *by* E, denoted by D^E, as follows.

$$D^E = \left\{ \frac{pre(r)}{cons(r)} \ : \ r \in D \ \textit{and for all} \ \beta \in just(r) \ , \ \neg\beta \notin E \right\}$$

We say E is an *extension* of D if $E = Cn(D^E)$. We say D is *consistent* if it has at least one consistent extension. We say that a formula is a *consequence* of D if it belongs to every extension of D. The definition of a default theory is due to Reiter [Rei80]. The definition of an extension given above follows [GLPT91], and is equivalent to Reiter's definition.

Although the definitions in this section are stated for the propositional case, they are taken, in the standard way, to apply naturally in the non-propositional case as well, by taking each non-ground expression to stand for all of its ground instances.

3 The Action Language $\mathcal{AC}$

The high-level action language $\mathcal{AC}$ is patterned after the languages $\mathcal{AR}_0$ [KL94] and $\mathcal{AR}$ [GKL95]. It also incorporates the approach to representing causal background knowledge that was introduced in [MT95b]. An $\mathcal{AC}$ action domain description consists of propositions of the following five kinds:

1. value propositions, which restrict the values of fluents in situations resulting from the performance of sequences of actions;
2. determination propositions, which say that a fluent formula can be made true by making another fluent formula true;
3. effect propositions, which say that under certain conditions a fluent formula holds after an action is executed;
4. influence propositions, which say that under certain conditions the execution of an action may "nondeterministically" change the value of a fluent;
5. executability propositions, which say that under certain conditions an action cannot be executed.

After specifying the syntax and semantics of $\mathcal{AC}$, we very briefly discuss its properties.

3.1 Syntax of $\mathcal{AC}$

We begin with two disjoint nonempty sets of symbols, a set of *fluent names* and a set of *action names*. We designate a subset of the fluent names as the *frame fluents* and call the remaining fluents *nonframe fluents*. A *fluent formula* is a formula whose atoms are fluent names. A *frame fluent formula* is a formula whose atoms are frame fluents.

An *atomic value proposition* is an expression of the form

$$\phi \textbf{ after } \overline{A}$$

where ϕ is a fluent formula and $\overline{A}$ is a string of action names. If $\overline{A}$ is the empty string, we may write instead

$$\textbf{initially } \phi\,.$$

A *value proposition* is a propositional combination of atomic value propositions.

A *determination proposition* is an expression of the form

$$\phi \textbf{ determines } \psi$$

where ϕ and ψ are fluent formulas. Such a proposition says that, in the action domain being described, one can make the fluent formula ψ true by making the fluent formula ϕ true. We write

$$\textbf{always } \phi$$

as an abbreviation for the proposition *True* **determines** ϕ. Given a nonframe fluent F, an expression of the form

$$\textbf{always } F \equiv \phi$$

where ϕ is a frame fluent formula, is called an *explicit definition of* F. An $\mathcal{AC}$ domain description must include an explicit definition of every nonframe fluent.

An *effect proposition* is an expression of the form

$$A \textbf{ causes } \phi \textbf{ if } \psi$$

where A is an action name, and ϕ and ψ are fluent formulas. If ψ is the formula *True*, we may drop the **if** and simply write A **causes** ϕ.

An *influence proposition* is an expression of the form

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

where A is an action name, F is a fluent name, and ψ is a fluent formula. If ψ is the formula *True*, we may drop the **if** and simply write A **possibly changes** F.

An *executability proposition* is an expression of the form

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

where A is an action name and ψ is a fluent formula. In the semantics of $\mathcal{AC}$ such a proposition has the same meaning as the effect proposition A **causes** *False* **if** ψ, but the syntactic distinction becomes convenient in Section 4 when we specify the translations from $\mathcal{AC}$ into default logic.

An $\mathcal{AC}$ *domain description* is a set of $\mathcal{AC}$ propositions that includes an explicit definition for each nonframe fluent.

3.2 Semantics of $\mathcal{AC}$

Let D be an $\mathcal{AC}$ domain description. A *structure for* D is a partial function from action strings to interpretations of the fluent atoms, whose domain is nonempty and prefix-closed. By $Dom(\Psi)$ we denote the domain of a structure Ψ. Notice that for every structure Ψ, $Dom(\Psi)$ includes the empty string (denoted by ϵ).

Let R be the set of inference rules $\frac{\phi}{\psi}$ such that the determination proposition

$$\phi \textbf{ determines } \psi$$

is in D. An interpretation S of the fluent atoms is a *state* if $Cn(S)$ is closed under R.

Let A be action name and S a state. We say that A is *prohibited in* S if there is an executability proposition

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

in D such that S satisfies ψ. Let $E(A,S)$ be the set of fluent formulas ϕ for which there is an effect proposition

$$A \textbf{ causes } \phi \textbf{ if } \psi$$

in D such that S satisfies ψ. Similarly, let $F(A,S)$ be the set of fluent names F for which there is an influence proposition

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

in D such that S satisfies ψ. A set E of fluent formulas is an *explicit effect of* A *in* S if:

1. A is not prohibited in S, and
2. there is an interpretation I of the atoms in $F(A,S)$ such that $E = I \cup E(A,S)$.

Let $\mathbf{L}_f$ denote the set of frame fluent literals. A state S' *may result from doing* A *in* S if there is an explicit effect E of A in S such that

$$Cn(S') = Cn[\,(\,S \cap S' \cap \mathbf{L}_f\,) \cup E \cup R\,]\,.$$

This fixpoint definition guarantees that S' may result from doing A in S if and only if the value of every frame fluent in S' is explained — either it held the same value in S and was not made to change, or its value was changed (directly or indirectly) by the action. Thus, this definition reflects the commonsense law of inertia — the assumption that things don't change unless they're made to.[3] Let $Res(A, S)$ denote the set of states that may result from doing A in S.

Given a structure Ψ, we say that an atomic value proposition ϕ **after** $\overline{A}$ is *true in* Ψ if $\overline{A} \in Dom(\Psi)$ and $\Psi(\overline{A})$ satisfies ϕ. The general truth definition for value propositions is then given by the standard recursion over the propositional connectives.

A structure Ψ for D is called a *model of* D if it satisfies the following four conditions.

1. $\Psi(\epsilon)$ is a state.
2. For all $\overline{A} \in Dom(\Psi)$ and all action names A, if $Res(A, \Psi(\overline{A})) \neq \emptyset$ then $\overline{A}; A \in Dom(\Psi)$.
3. For all $\overline{A}; A \in Dom(\Psi)$, $\Psi(\overline{A}; A) \in Res(A, \Psi(\overline{A}))$.
4. Every value proposition in D is true in Ψ.

A value proposition is *entailed* by D if it is true in every model of D.

3.3 Remarks on $\mathcal{AC}$

The action description language $\mathcal{AC}$ is modeled after the language $\mathcal{AR}$ of Giunchiglia, Kartha and Lifschitz [GKL95]. Unlike the language $\mathcal{AC}$, $\mathcal{AR}$ allows non-boolean fluents; but if we restrict our consideration to the propositional portion of $\mathcal{AR}$, we find that the model structures for the two languages are identical.

Syntactically, $\mathcal{AC}$ and $\mathcal{AR}$ are very similar. One difference is that $\mathcal{AR}$ does not include determination propositions for representing background knowledge, which is instead represented by means of state constraints of the form **always** ϕ, where ϕ is a fluent formula. In $\mathcal{AC}$ we understand such an expression as an abbreviation of the corresponding determination proposition *True* **determines** ϕ. Thus $\mathcal{AR}$ state constraints are well-formed $\mathcal{AC}$ propositions. Another significant syntactic difference between $\mathcal{AC}$ and $\mathcal{AR}$ is that $\mathcal{AR}$ includes only atomic value propositions, whereas $\mathcal{AC}$ allows propositional combinations of atomic value propositions. A third difference

[3] See [MT95b] and [PT95] for further discussion related to this fixpoint definition.

is that in $\mathcal{AR}$ the expression **impossible** A **if** ψ is an abbreviation for the effect proposition A **causes** *False* **if** ψ, whereas in $\mathcal{AC}$ these are distinct propositions.

As the preceding observations suggest, the set of well-formed propositional $\mathcal{AR}$ expressions is a proper subset of the set of well-formed $\mathcal{AC}$ expressions. Given this, the relationship between $\mathcal{AR}$ and $\mathcal{AC}$ is captured in the following theorem.

$\mathcal{AR}$ **Theorem.** *Let D be a propositional $\mathcal{AR}$ domain description such that every nonframe fluent in D has an explicit definition in terms of frame fluents. D is an $\mathcal{AC}$ domain description, and the $\mathcal{AC}$ models of D are exactly the $\mathcal{AR}$ models of D.*

Determination propositions were introduced in [MT95b], in a simpler setting with a different syntax. There we showed, for instance, that determination propositions of the form *True* **determines** ϕ correspond to ramification constraints, which are state constraints used to derive indirect effects, or ramifications, of actions. Traditionally, state constraints have played the role of ramification constraints, as they in fact do in $\mathcal{AR}$. We also showed in [MT95b] that determination propositions of the form $\neg\phi$ **determines** *False* play the role of qualification constraints, which are state constraints used to derive action preconditions.[4] More generally, we argued that a determination proposition ϕ **determines** ψ turns out to be an intuitively correct representation of the corresponding causal background knowledge — the knowledge that, in the action domain itself, one can make ψ true by making ϕ true.

The idea for the fixpoint definition of *Res* adopted in the semantics of $\mathcal{AC}$ was introduced in a pair of related papers [MT95b, PT95]. In [MT95b] we show how the fixpoint definition can be derived as a natural extension of the corresponding definition from Winslett's [Win88] possible models approach to reasoning about the effects of actions. In [PT95] the idea is explored in the more abstract setting of theory update. There we introduce a fixpoint definition of "rule update" — update by means of arbitrary sets of inference rules — and show that rule update not only extends Winslett's update definition but also extends the notion of revision programming, introduced by Marek and Truszczyński [MT94, MT95a].

[4]The terminology of "ramification" and "qualification" constraints is due to Lin and Reiter [LR94].

4 Embedding $\mathcal{AC}$ in Default Logic

We say that an $\mathcal{AC}$ domain description is *qualification-free* if for all action names A and states S, A is prohibited in S whenever $Res(A, S)$ is empty. Thus, in qualification-free domain descriptions, all action preconditions are stated in the form of executability propositions. In this section we specify a sound and complete translation of qualification-free $\mathcal{AC}$ domain descriptions into default theories. We also specify a second, simpler translation for those $\mathcal{AC}$ domain descriptions in which there are no action preconditions whatsoever. These embeddings build on previous work in [Tur94, LT95, PT95].

Our default theories for reasoning about action use the situation calculus. For any fluent formula ϕ, we write $Holds(\phi, S)$ as an abbreviation for the formula obtained by replacing each fluent atom F in ϕ by $Holds(F, S)$. (Notice that the special constants *True* and *False* are not replaced.) Given an action string $A_1; \cdots; A_m$, we write

$$[A_1; \cdots; A_m]$$

to stand for the term

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

Given an atomic value proposition ϕ **after** $\overline{A}$, we write

$$[\phi \textbf{ after } \overline{A}]$$

to stand for the formula

$$(\ Holds(\phi, [\overline{A}]) \wedge Reachable([\overline{A}])\ ) \, .$$

Given an arbitrary value proposition V, we write $[V]$ to stand for the formula obtained by simultaneously replacing each atomic value proposition V' in V by the formula $[V']$.

Below we define a translation δ that takes an $\mathcal{AC}$ domain description D to a default theory $\delta(D)$. For each value proposition V in D, include the rule

$$\frac{\neg[V]}{False} \, .$$

For each determination proposition ϕ **determines** ψ in D, include the rule

$$\frac{Holds(\phi, s) \wedge Reachable(s)}{Holds(\psi, s)} \, .$$

For each effect proposition A **causes** ϕ **if** ψ in D, include the rule

$$\frac{Holds(\psi, s) \wedge Reachable(Result(A, s))}{Holds(\phi, Result(A, s))}.$$

For each influence proposition A **possibly changes** F **if** ψ in D, include the pair of rules

$$\frac{Holds(\psi, s) \wedge Reachable(Result(A, s)) : Holds(F, Result(A, s))}{Holds(F, Result(A, s))}$$

and

$$\frac{Holds(\psi, s) \wedge Reachable(Result(A, s)) : \neg Holds(F, Result(A, s))}{\neg Holds(F, Result(A, s))}.$$

For each executability proposition **impossible** A **if** ψ in D, include the rule

$$\frac{Holds(\psi, s)}{\neg Reachable(Result(A, s))}.$$

We also include a number of additional rules, as specified below.

Reachability axioms. Include the rules

$$\frac{: Reachable(s)}{Reachable(s)} \quad \text{and} \quad \frac{\neg Reachable(s)}{\neg Reachable(Result(a, s))}.$$

Initial situation axioms. For each fluent literal L, include the rule

$$\frac{: Holds(L, S_0)}{Holds(L, S_0)}.$$

Inertia axioms. For each frame fluent literal L, include the rule

$$\frac{Holds(L, s) \wedge Reachable(Result(a, s)) : Holds(L, Result(a, s))}{Holds(L, Result(a, s))}.$$

The following theorem shows that the translation specified above is sound and complete for qualification-free $\mathcal{AC}$ domain descriptions.

Embedding Theorem. *Let D be a qualification-free $\mathcal{AC}$ domain description. A value proposition V is entailed by D if and only if the formula $[V]$ is entailed by the default theory $\delta(D)$.*

The Embedding Theorem is an immediate consequence of the following stronger theorem.

Correspondence Theorem. *Let D be a qualification-free $\mathcal{AC}$ domain description. There is a one-to-one correspondence between models of D and consistent extensions of $\delta(D)$ such that, for every model Ψ of D and its corresponding extension E, a value proposition V is true in Ψ if and only if $[V] \in E$.*

Consider the following domain description D_1, in which *Dead* is a nonframe fluent.

$$\begin{array}{l}
\textbf{always } \mathit{Dead} \equiv \neg \mathit{Alive} \\
\textbf{initially } \mathit{Alive} \\
\mathit{Dead} \textbf{ after } \mathit{Spin}; \mathit{Shoot} \\
\mathit{Shoot} \textbf{ causes } \mathit{Dead} \\
\textbf{impossible } \mathit{Shoot} \textbf{ if } \neg \mathit{Loaded} \\
\mathit{Spin} \textbf{ possibly changes } \mathit{Loaded}
\end{array}$$

Domain description D_1 entails, for instance, the value proposition

$$\mathit{Loaded} \textbf{ after } \mathit{Spin}$$

and the Embedding Theorem guarantees that the corresponding formula

$$\mathit{Holds}(\mathit{Loaded}, \mathit{Result}(\mathit{Spin}, S_0)) \wedge \mathit{Reachable}(\mathit{Result}(\mathit{Spin}, S_0))$$

is entailed by the corresponding default theory $\delta(D_1)$, which appears in Figure 1 (next page).

If a domain description includes no executability propositions, we can eliminate the *Reachable* atoms in the corresponding default theory, thus obtaining a simpler translation, as follows. Let D be an $\mathcal{AC}$ domain description. By $\delta'(D)$ we denote the default theory obtained from $\delta(D)$ by first eliminating the reachability axioms and then replacing each *Reachable* atom in the remaining rules by the special atom *True*. Of course it is then straightforward to eliminate occurrences of *True* in the resulting default theory.

For each atomic value proposition ϕ **after** $\overline{A}$, let

$$[\![\phi \textbf{ after } \overline{A}]\!]$$

denote the formula

$$\mathit{Holds}(\phi, [\overline{A}])$$

and for every value proposition V, let $[\![V]\!]$ be the formula obtained from V by simultaneously replacing each atomic value proposition V' that occurs in V by the formula $[\![V']\!]$.

$$\frac{True \wedge Reachable(s)}{Holds(Dead, s) \equiv \neg Holds(Alive, s)} \qquad \frac{\neg(Holds(Alive, S_0) \wedge Reachable(S_0))}{False}$$

$$\frac{\neg(Holds(Dead, Result(Shoot, Result(Spin, S_0))) \wedge Reachable(Result(Shoot, Result(Spin, S_0))))}{False}$$

$$\frac{True \wedge Reachable(Result(Shoot, s))}{Holds(Dead, Result(Shoot, s))} \qquad \frac{\neg Holds(Loaded, s)}{\neg Reachable(Result(Shoot, s))}$$

$$\frac{True \wedge Reachable(Result(Spin, s)) : Holds(Loaded, Result(Spin, s))}{Holds(Loaded, Result(Spin, s))}$$

$$\frac{True \wedge Reachable(Result(Spin, s)) : \neg Holds(Loaded, Result(Spin, s))}{\neg Holds(Loaded, Result(Spin, s))}$$

$$\frac{: Holds(f, S_0)}{Holds(f, S_0)} \qquad \frac{: \neg Holds(f, S_0)}{\neg Holds(f, S_0)} \qquad \frac{: Reachable(s)}{Reachable(s)} \qquad \frac{\neg Reachable(s)}{\neg Reachable(Result(a, s))}$$

$$\frac{Holds(Alive, s) \wedge Reachable(Result(a, s)) : Holds(Alive, Result(a, s))}{Holds(Alive, Result(a, s))}$$

$$\frac{\neg Holds(Alive, s) \wedge Reachable(Result(a, s)) : \neg Holds(Alive, Result(a, s))}{\neg Holds(Alive, Result(a, s))}$$

$$\frac{Holds(Loaded, s) \wedge Reachable(Result(a, s)) : Holds(Loaded, Result(a, s))}{Holds(Loaded, Result(a, s))}$$

$$\frac{\neg Holds(Loaded, s) \wedge Reachable(Result(a, s)) : \neg Holds(Loaded, Result(a, s))}{\neg Holds(Loaded, Result(a, s))}$$

Figure 1: The translation $\delta(D_1)$ of $\mathcal{AC}$ domain description D_1.

Reachability Corollary. *Let D be a qualification-free $\mathcal{AC}$ domain description with no executability propositions. There is a one-to-one correspondence between models of D and consistent extensions of $\delta'(D)$ such that, for every model Ψ of D and its corresponding extension E, a value proposition V is true in Ψ if and only if $[\![V]\!] \in E$.*

For example, the default theory that appears in the introduction can be obtained by the translation δ'. Another example, adapted from [Lin95], involves a spring-loaded briefcase with two clasps. We have actions that unfasten the clasps, one at a time. If both clasps are unfastened, the briefcase pops open. Initially the briefcase is not open. This action domain can be formalized in $\mathcal{AC}$ by the following domain description D_2.

$$\textbf{initially } \neg Open$$

$$Unfasten(x) \textbf{ causes } \neg Fastened(x)$$

$$\neg Fastened(Clasp_1) \wedge \neg Fastened(Clasp_2) \textbf{ determines } Open$$

Domain description D_2 entails the $\mathcal{AC}$ value proposition

$$(\, Open \textbf{ after } Unfasten(Clasp_1)\,) \equiv (\, \textbf{initially } \neg Fastened(Clasp_2)\,)$$

and, accordingly, we know by the Reachability Corollary that the formula

$$Holds(Open, Result(Unfasten(Clasp_1), S_0)) \equiv \neg Holds(Fastened(Clasp_2), S_0)$$

is a consequence of the default theory $\delta'(D_2)$, listed below.

$$\frac{Holds(Open, S_0)}{False} \qquad \frac{True}{\neg Holds(Fastened(x), Result(Unfasten(x), s))}$$

$$\frac{\neg Holds(Fastened(Clasp_1), s) \wedge \neg Holds(Fastened(Clasp_2), s)}{Holds(Open, s)}$$

$$\frac{: Holds(f, S_0)}{Holds(f, S_0)} \qquad \frac{: \neg Holds(f, S_0)}{\neg Holds(f, S_0)}$$

$$\frac{Holds(f, s) : Holds(f, Result(a, s))}{Holds(f, Result(a, s))} \qquad \frac{\neg Holds(f, s) : \neg Holds(f, Result(a, s))}{\neg Holds(f, Result(a, s))}$$

5 Concluding Remarks

At this point it may be interesting to consider how our default theory for the Yale Shooting domain [HM87] differs from the one found inadequate by Hanks and McDermott, and from the more adequate solution later proposed by Morris [Mor88].

We represent the Yale Shooting domain in $\mathcal{AC}$ by the domain description D_3 below.

$$\begin{array}{l} \textbf{initially } Alive \\ Load \textbf{ causes } Loaded \\ Shoot \textbf{ causes } \neg Alive \textbf{ if } Loaded \end{array}$$

Of course D_3 entails

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

and accordingly, we know by the Reachability Corollary that the corresponding literal

$$\neg Holds(Alive, Result(Shoot, Result(Wait, Result(Load, S_0))))$$

is entailed by the corresponding default theory $\delta'(D_3)$, listed below.

$$\frac{\neg Holds(Alive, S_0)}{False}$$

$$\frac{True}{Holds(Loaded, Result(Load, s))} \quad \frac{Holds(Loaded, s)}{\neg Holds(Alive, Result(Shoot, s))}$$

$$\frac{: Holds(f, S_0)}{Holds(f, S_0)} \quad \frac{: \neg Holds(f, S_0)}{\neg Holds(f, S_0)}$$

$$\frac{Holds(f, s) : Holds(f, Result(a, s))}{Holds(f, Result(a, s))} \quad \frac{\neg Holds(f, s) : \neg Holds(f, Result(a, s))}{\neg Holds(f, Result(a, s))}$$

By comparison, the default theory of Hanks and McDermott is essentially as follows.

$$\frac{True}{Holds(Alive, S_0)}$$

$$\frac{True}{Ab(Loaded, Load, s) \wedge Holds(Loaded, Result(Load, s))}$$

$$\frac{True}{Holds(Loaded, s) \supset (Ab(Alive, Shoot, s) \wedge \neg Holds(Alive, Result(Shoot, s)))}$$

$$\frac{True}{(Holds(f, s) \wedge \neg Ab(f, a, s)) \supset Holds(f, Result(a, s))}$$

$$\frac{True}{(\neg Holds(f, s) \wedge \neg Ab(f, a, s)) \supset \neg Holds(f, Result(a, s))}$$

$$\frac{: \neg Ab(f, a, s)}{\neg Ab(f, a, s)}$$

In this default theory, the well-known technical difficulty results from the fact that we can reason backwards in time, using the rule describing the effect of *Shoot* and the inertia rules. In particular, we can derive the literals

$$Ab(Loaded, Wait, Result(Load, S_0))$$

and

$$\neg Holds(Loaded, Result(Wait, Result(Load, S_0)))$$

from the default "supposition"

$$\neg Ab(Alive, Shoot, Result(Wait, Result(Load, S_0)))$$

thus obtaining the famous anomaly: according to this default theory the gun may become mysteriously unloaded during the wait action.

It may be worth noting that there is another peculiarity here, related to the fact that in the Yale Shooting domain (as described and formalized by

Hanks and McDermott) we do not know whether or not the gun is initially loaded. Accordingly, domain description D_3 entails neither

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

nor

$$\textbf{initially } \neg \mathit{Loaded}\ .$$

Default theory $\delta'(D_3)$ includes Initial Situation axioms ensuring that all initial states consistent with the domain description are taken into account. Thus for instance, as we know from the Reachability Corollary, default theory $\delta'(D_3)$ does not entail the formula

$$\neg \mathit{Holds}(\mathit{Loaded}, S_0)\ .$$

As it turns out though, the default theory of Hanks and McDermott does. Thus we see that their default theory is not only incomplete for the Yale Shooting domain, but also unsound.

Next consider the default theory proposed by Morris [Mor88], which is essentially as follows.

$$\frac{\mathit{True}}{\mathit{Holds}(\mathit{Alive}, S_0)}$$

$$\frac{\mathit{True}}{\mathit{Ab}(\mathit{Loaded}, \mathit{Load}, s) \wedge \mathit{Holds}(\mathit{Loaded}, \mathit{Result}(\mathit{Load}, s))}$$

$$\frac{\mathit{True}}{\mathit{Holds}(\mathit{Loaded}, s) \supset (\mathit{Ab}(\mathit{Alive}, \mathit{Shoot}, s) \wedge \neg \mathit{Holds}(\mathit{Alive}, \mathit{Result}(\mathit{Shoot}, s)))}$$

$$\frac{\mathit{Holds}(f, s) : \neg \mathit{Ab}(f, a, s)}{\mathit{Holds}(f, \mathit{Result}(a, s))} \qquad \frac{\neg \mathit{Holds}(f, s) : \neg \mathit{Ab}(f, a, s)}{\neg \mathit{Holds}(f, \mathit{Result}(a, s))}$$

Even in Morris' default theory it is not clear that we cannot reason backwards in time using the rule describing the effect of the action *Shoot*. But notice that there is now no default rule allowing us to "suppose" literals of the form $\neg \mathit{Ab}(f, a, s)$. Moreover, there is no opportunity for "inappropriately" deriving atoms of the form $\mathit{Ab}(f, a, s)$ by reasoning backwards in time. Thus the famous anomaly is eliminated. On the other hand, it turns out that the formula $\neg \mathit{Holds}(\mathit{Loaded}, S_0)$ is once again inappropriately entailed; so Morris' default theory for the Yale Shooting domain is apparently complete but unsound.

In our translations from $\mathcal{AC}$ into default logic, every default rule is either justification-free or normal. In this way our approach is simpler than that of Morris, which uses non-normal defaults, as well as an auxiliary predicate Ab. Moreover, our method has the considerable advantage of guaranteeing correct representations of commonsense knowledge for a wide variety of action domains.

Acknowledgements

Special thanks to Vladimir Lifschitz and Norman McCain. Thanks also to Chitta Baral, Michael Gelfond, Enrico Giunchiglia, G. Neelakantan Kartha, Teodor Przymusinski and Michael Thielscher for helpful discussions. This work is partially supported by NSF grant #IRI-9306751.

References

[Bar95] Chitta Baral. Reasoning about actions: non-deterministic effects, constraints, and qualifications. In *Proc. of IJCAI-95*, pages 2017–2023, 1995.

[BH93] Gerhard Brewka and Joachim Hertzberg. How to do things with worlds: On formalizing actions and plans. *Journal of Logic and Computation*, 3(5), 1993.

[Elk92] Charles Elkan. Reasoning about action in first-order logic. In *Proc. of the 1992 Canadian Conf. on Artificial Intelligence*, 1992.

[Gef90] Hector Geffner. Causal theories of nonmonotonic reasoning. In *Proc. of AAAI-90*, pages 524–530, 1990.

[GKL95] Enrico Giunchiglia, G. Neelakantan Kartha, and Vladimir Lifschitz. Actions with indirect effects (extended abstract). In *Working Notes of the AAAI Spring Symposium on Extending Theories of Action*, pages 80–85, 1995.

[GL93] Michael Gelfond and Vladimir Lifschitz. Representing action and change by logic programs. *Journal of Logic Programming*, 17:301–322, 1993.

[GLPT91] Michael Gelfond, Vladimir Lifschitz, Halina Przymusińska, and Miroslaw Truszczyński. Disjunctive defaults. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Principles of Knowledge Representation and Reasoning: Proc. of the 2nd Int'l Conference*, pages 230–237, 1991.

[HM87] Steve Hanks and Drew McDermott. Nonmonotonic logic and temporal projection. *Artificial Intelligence*, 33(3):379–412, 1987.

[KL94] 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.

[Lin95] Fangzhen Lin. Embracing causality in specifying the indirect effects of actions. In *Proc. of IJCAI-95*, pages 1985–1991, 1995.

[LR94] Fangzhen Lin and Raymond Reiter. State constraints revisited. *Journal of Logic and Computation*, 4(5):655–678, 1994. Special Issue on Actions and Processes.

[LT95] Vladimir Lifschitz and Hudson Turner. From disjunctive programs to abduction. In Jürgen Dix, Luis Pereira, and Teodor Przymusinski, editors, *Non-Monotonic Extensions of Logic Programming (Lecture Notes in Artificial Intelligence 927)*, pages 23–42. Springer-Verlag, 1995.

[Mor88] Paul Morris. The anomalous extension problem in default reasoning. *Artificial Intelligence*, 35(3):383–399, 1988.

[MT94] Wiktor Marek and Miroslaw Truszczyński. Revision specifications by means of programs. In *Logics in AI. Proceedings of JELIA '94*, 1994.

[MT95a] Wiktor Marek and Miroslaw Truszczyński. Revision programming, database updates and integrity constraints. In *Proc. of the 5th Int'l Conf. on Database Theory*, pages 368–382, 1995.

[MT95b] Norman McCain and Hudson Turner. A causal theory of ramifications and qualifications. In *Proc. of IJCAI-95*, pages 1978–1984, 1995.

[PT95] Teodor Przymusinski and Hudson Turner. Update by means of inference rules. In *Proc. of the 3rd Int'l Conf. on Logic Programming and Nonmonotonic Reasoning*, pages 156–174, 1995.

[Rei80] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1,2):81–132, 1980.

[Thi95] Michael Thielscher. Computing ramifications by postprocessing. In *Proc. of IJCAI-95*, pages 1994–2000, 1995.

[Tur94] Hudson Turner. Signed logic programs. In Maurice Bruynooghe, editor, *Logic Programming: Proc. of the 1994 Int'l Symposium*, pages 61–75. MIT Press, 1994.

[Win88] Marianne Winslett. Reasoning about action using a possible models approach. In *Proc. of AAAI–88*, pages 89–93, 1988.