The Semantics of Variables in Action Descriptions

Vladimir Lifschitz and **Wanwan Ren**
The University of Texas at Austin, USA
{vl,rww6}@cs.utexas.edu

Abstract

Action description language $\mathcal{C}+$ is more expressive than ADL in many ways; for instance, it addresses the ramification problem. On the other hand, ADL is based on first-order logic, while $\mathcal{C}+$ is only propositional; expressions with variables, which are frequently used when action domains are described in $\mathcal{C}+$, are merely schemas describing finite sets of causal laws that are formed according to the same pattern. In this paper we propose a new approach to the semantics of action descriptions with variables that combines attractive features of ADL and $\mathcal{C}+$.

Introduction

The problem of describing changes caused by the execution of actions plays an important rule in knowledge representation. Some of the best known approaches to representing actions are the situation calculus (McCarthy & Hayes 1969), the STRIPS language (Fikes & Nilsson 1971) and the event calculus (Kowalski & Sergot 1986).

Current research on the design of action description languages (Gelfond & Lifschitz 1998) continues the line of work that started with the invention of ADL (Pednault 1994). Action description languages are attractive because they are concise, and because their semantics is based on the simple idea of a transition system ("state-transition model," in Pednault's terminology). A transition system is a directed graph with vertices corresponding to states of the world, and edges corresponding to transitions that may be caused by the execution of actions.

Modern action description languages, such as $\mathcal{C}+$ (Giunchiglia *et al.* 2004), are more expressive than ADL in many ways. In particular, they solve the ramification problem, that is, allow the user to characterize effects of actions indirectly. But in one sense ADL is more expressive than $\mathcal{C}+$: the former is based on first-order logic, and the latter is only propositional. In (Pednault 1994), *state-transition models for a first-order language* are defined (Definition 2.3); their states are semantic structures, or interpretations, in the sense of first-order logic. In $\mathcal{C}+$, on the other hand, a state is an interpretation of a (multi-valued) propositional signature (Giunchiglia *et al.* 2004, Section 4.4). There are no variables in $\mathcal{C}+$, strictly speaking. Expressions with variables, which are frequently used when action domains are described in $\mathcal{C}+$, are merely schemas describing finite sets of causal laws that are formed according to the same pattern.

For example, the description of the blocks world action *Put*(b, l) in (Pednault 1994, Figure 2) has the formula *On*(b, l) on its add list. In this formula, b and l are object variables in the sense of first-order logic, and *On* is a binary predicate constant. In $\mathcal{C}+$ we can express the same idea by writing

$$\textit{Put}(b, l) \textbf{ causes } \textit{On}(b, l). \tag{1}$$

But here b and l are *meta*variables, and we need to specify their possible values when we say that (1) is part of an action description. We can say, for instance, that b stands for any of the symbols *Block1*, *Block2*, *Block3*, and that l stands for *Block1*, *Block2*, *Block3* or *Table*. Expression (1) will denote then a set of 12 causal laws, obtained from (1) by grounding. The expression *On*(*Block2*, *Table*), occurring in one of them, is a fluent constant, according to the syntax of $\mathcal{C}+$, but the three parts that this expression is built from — *On*, *Block2* and *Table* — have no syntactic status in the definition of the language.

In this paper we show how to define a semantics of action descriptions that is similar to the semantics of $\mathcal{C}+$ and, at the same time, allows us to use genuine object variables. Like the semantics of ADL, it is based on state-transition models for first-order languages.

The tool that helps us achieve this is the first-order causal logic proposed in (Lifschitz 1997). Recall that the semantics of $\mathcal{C}+$ is characterized in (Giunchiglia *et al.* 2004, Section 4.2) by a translation that turns any action description D into a sequence of propositional causal theories $D_0, D_1, \ldots$. Models of D_m correspond to the possible behaviors of the state-transition system described by D over successive time instants $0, 1, \ldots, m$. In particular, models of D_0 are the states of the system, and models of D_1 are its transitions. In our modification of this approach, D_m becomes a *first-order* causal theory. As a result, the new semantics of causal laws with variables avoids any references to grounding. We argue that it has significant conceptual advantages in comparison with the grounding approach.

The central part of this paper is the section on the semantics, which describes a new way to represent action descriptions by causal theories. It is preceded by the discussion of

sorts
Robot; *Box*; *Thing*; *Place*;
inclusions
$\mathit{Robot} \ll \mathit{Thing}$;
$\mathit{Box} \ll \mathit{Thing}$;
objects
R_1, R_2: *Robot*;
B_1, B_2: *Box*;
P_1, P_2: *Place*;
constants
$\mathit{Location}(\mathit{Thing})$: **fluent**$(\mathit{Place})$;
$\mathit{Go}(\mathit{Robot}, \mathit{Place}), \mathit{Carry}(\mathit{Robot}, \mathit{Box}, \mathit{Place})$: **action**;
variables
r: *Robot*; b: *Box*; x: *Thing*; p: *Place*;
axioms
inertial $\mathit{Location}(x)$;
exogenous $\mathit{Go}(r, p)$;
exogenous $\mathit{Carry}(r, b, p)$;

$\mathit{Go}(r, p)$ **causes** $\mathit{Location}(r) = p$;
$\mathit{Carry}(r, b, p)$ **causes** $\mathit{Location}(r) = p$;
$\mathit{Carry}(r, b, p)$ **causes** $\mathit{Location}(b) = p$;

nonexecutable $\mathit{Carry}(r, b, p) \wedge \mathit{Go}(r, p)$;
nonexecutable $\mathit{Carry}(r, b, p)$
if $\neg(\mathit{Location}(r) = \mathit{Location}(b))$;

Figure 1: Action description *RBP*

the syntax of action descriptions with variables adopted in this paper and a review of first-order causal logic, and followed by the investigation of mathematical properties of the new semantics.

Syntax of Action Descriptions

The action description language defined in this section is a simplified version of the input language of the implementation of $\mathcal{C}+$ called the Causal Calculator, or CCALC.[1] In the example shown in Figure 1 we talk about robots R_1, R_2; boxes B_1, B_2; places P_1, P_2; finally, things, which include both robots and boxes. The location of a thing is a place-valued fluent. There are actions of two kinds: a robot can go to a place or carry a box to a place. The axioms are syntactically similar to causal laws in the sense of $\mathcal{C}+$ (Giunchiglia *et al.* 2004, Section 4.2), except that they contain variables. Locations are inertial: they don't change without a cause. The actions are exogenous: they can be executed or not executed at will. The last five axioms describe the effects and preconditions of the actions.

Generally, an action description consists of six parts, as in Figure 1. The *sort declaration part* is a list of sort names. The *sort inclusion part* is a list of subsort expressions, such as $\mathit{Robot} \ll \mathit{Thing}$. The *object declaration part* is a list of object specifications, such as

$$R_1, R_2\text{: } \mathit{Robot}.$$

[1] http://www.cs.utexas.edu/users/tag/ccalc/

Each object specification is a list of object names followed by a sort name.

The *constant declaration part* is a list of constant specifications, such as

$$\mathit{Location}(\mathit{Thing})\text{: } \textbf{fluent}(\mathit{Place}).$$

Each constant specification begins with a list of constant schemas. A constant schema is a constant name (in this case, *Location*) followed by a list of sort names describing its arguments (*Thing*). The expression after the colon shows whether the symbol that is being declared is a fluent constant or an action constant. The sort name after the reserved word **fluent** shows the sort of values of the fluent; it is dropped if the fluent is Boolean.

The *variable declaration part* is syntactically similar to the object declaration part.

The *axiom part* is a list of causal laws. As in $\mathcal{C}+$ (Giunchiglia *et al.* 2004, Section 4.2), we distinguish between three kinds of causal laws: static, action dynamic, and fluent dynamic.

In a syntactically correct action description, each name is declared exactly once. In declarations and in axioms, a name can only be used in accordance with its declaration.

Review of Causal Logic

The review of the syntax and semantics of causal theories in this section follows (Lifschitz 1997, Section 2). The main idea of nonmonotonic causal logic (McCain & Turner 1997) is to distinguish between the claim that a proposition is true and the stronger claim that there is a *cause* for it to be true. Causal dependencies are described by *causal rules*—expressions of the form

$$F \Leftarrow G, \tag{2}$$

where F and G are first-order formulas. Rule (2) expresses that there is a cause for the head formula F to hold if the body formula G holds, or, in other words, that G provides a "causal explanation" for F.

A *causal theory* is defined by

- a finite subset of the signature[2] of the underlying language, called the *explainable symbols* of the theory, and
- a finite set of causal rules.

In the definition of the semantics of causal theories below, we use the substitution of variables for the explainable symbols in a formula. In connection with this, it is convenient to denote formulas by expressions like $F(E)$, where E is the list of all explainable symbols. Then, for any tuple e of variables that is similar[3] to E, the result of replacing all occurrences of the constants E in $F(E)$ by the variables e can be denoted by $F(e)$.

[2] The *signature* of a (nonsorted) first-order language is the set of its function constants and predicate constants (other than equality). This includes, in particular, object constants (function constants of arity 0) and propositional constants (predicate constants of arity 0).

[3] The similarity condition means that (i) e has the same length as E, (ii) if the k-th member of E is a function constant then the k-th member of e is a function variable of the same arity, and (iii) if the k-th member of E is a predicate constant then the k-th member of e is a predicate variable of the same arity.

Consider a causal theory T with the explainable symbols E and the causal rules

$$F_i(E, x^i) \Leftarrow G_i(E, x^i) \qquad (i = 1, \ldots),$$

where x^i is the list of all free variables of the i-th rule. Take a tuple e of new variables similar to E. By $T^*(e)$ we denote the formula

$$\bigwedge_i \forall x^i (G_i(E, x^i) \to F_i(e, x^i)).$$

Note that the occurrences of explainable symbols in the heads are replaced here by variables, and the occurrences in the bodies are not. As suggested in (Lifschitz 1997), we view T as shorthand for the sentence

$$\forall e(T^*(e) \leftrightarrow e = E). \tag{3}$$

(The expression $e = E$ stands for the conjunction of the equalities between the members of e and the corresponding members of E.) For instance, by a *model* of T we mean a model of (3); a formula is *entailed* by T if it is entailed by (3). Note that the tuple e may contain function and predicate variables, so that (3) is, generally, a second-order formula.

Intuitively, the condition $T^*(e)$ expresses that the possible values e of the explainable symbols E are "causally explained" by the rules of T. Sentence (3) says that the actual values of these symbols are the only ones that are explained by the rules of T.

For instance, let T be the causal theory with the rules

$$\begin{array}{l} \mathit{Robot}(R_1) \Leftarrow \top, \\ \mathit{Robot}(R_2) \Leftarrow \top, \\ \neg \mathit{Robot}(x) \Leftarrow \neg \mathit{Robot}(x), \end{array} \tag{4}$$

where the predicate constant *Robot* is explainable, and the object constants R_1, R_2 are not explainable. Intuitively, the last line of (4) expresses the closed-world assumption: if x is not a robot then there is a cause for this. In this case, E is *Robot*, e is a unary predicate variable, and $T^*(e)$ is

$$e(R_1) \wedge e(R_2) \wedge \forall x(\neg \mathit{Robot}(x) \to \neg e(x)).$$

The second-order sentence (3) is equivalent in this case to the first-order sentence

$$\forall x(\mathit{Robot}(x) \leftrightarrow x = R_1 \vee x = R_2). \tag{5}$$

Semantics of Action Descriptions

Given an action description D and a nonnegative integer m (the length of the behaviors that we are interested in), the causal theory D_m is formed as follows.

Its signature σ^{D_m} consists of

- an explainable unary predicate constant S for each sort name S;[4]
- a non-explainable object constant V for each object name V;
- a predicate constant $i\!:\!P$ for each Boolean fluent name P and every $i \in \{0, \ldots, m\}$; the arity of $i\!:\!P$ is the same as the arity of P; this constant is explainable if $i > 0$;
- a function constant $i\!:\!P$ for each non-Boolean fluent name P and every $i \in \{0, \ldots, m\}$; the arity of $i\!:\!P$ is the same as the arity of P; this constant is explainable if $i > 0$;
- an explainable predicate constant $i : P$ for each action name P and every $i \in \{0, \ldots, m-1\}$; the arity of $i\!:\!P$ is the same as the arity of P;
- a new non-explainable object constant *None*.

For instance, the signature σ^{RBP_m}, corresponding to the action description *RBP* shown in Figure 1, consists of the non-explainable object constants

$$R_1, R_2, B_1, B_2, P_1, P_2, \mathit{None};$$

the explainable predicate constants

$$\mathit{Robot}, \mathit{Box}, \mathit{Thing}, \mathit{Place}, i\!:\!\mathit{Go}, i\!:\!\mathit{Carry} \quad (0 \le i < m);$$

the function constants

$$i\!:\!\mathit{Location} \quad (0 \le i \le m)$$

that are explainable when $i > 0$.

The prefixes i: are "time stamps." For instance, the value of

$$5\!:\!\mathit{Location}(R_1)$$

is the location of robot R_1 at time 5; the truth value of

$$5\!:\!\mathit{Go}(R_1, P_2)$$

shows whether R_1 goes to place P_2 between times 5 and 6.

In the list of the causal rules of D_m below, the following notation is used. For any object name V, $SORT_V$ stands for the sort name assigned to V in the object declaration part of D, and similarly for variable names and for non-Boolean fluent names. For any formula F, $i\!:\!F$ stands for the result of prepending i: to all fluent names and action names in F. By $x, x_1, \ldots, x_n$ we denote distinct object variables.

The causal rules of D_m are

- $\neg S(x) \Leftarrow \neg S(x)$ for each sort name S (the closed-world assumption for sorts);
- $S_1(x) \to S_2(x) \Leftarrow \top$ for each subsort expression $S_1 \ll S_2$ in the sort inclusion part of D;
- $SORT_V(V) \Leftarrow \top$ for each object name V;
- $\neg(V = \mathit{None}) \Leftarrow \top$ for each object name V;
- $\neg(V_1 = V_2) \Leftarrow \top$ for each pair of distinct object names V_1, V_2 (the unique name assumption for objects);
- $\neg\, i\!:\!P(x_1, \ldots, x_n) \Leftarrow \neg\, S_j(x_j) \qquad (1 \le j \le n)$ for all Boolean fluent schemas and action schemas $P(S_1, \ldots, S_n)$ in the constant declaration part of D, and the rules

  $$i\!:\!P(x_1, \ldots, x_n) = \mathit{None} \Leftarrow \neg\, S_j(x_j) \quad (1 \le j \le n)$$

  for each non-Boolean fluent schema $P(S_1, \ldots, S_n)$ in the constant declaration part of D (all arguments should be of appropriate sorts);

[4] These predicate constants are needed because the target language of our translation—the language of causal theories—is non-sorted.

- the rules

$$\neg(i{:}P(x_1,\dots,x_n)=x) \Leftarrow \neg \mathit{SORT}_P(x) \wedge S_1(x_1) \wedge \cdots \wedge S_n(x_n)$$

for each non-Boolean fluent schema $P(S_1,\dots,S_n)$ in the constant declaration part of D (the value should be of the appropriate sort);

- $i{:}F \Leftarrow i{:}G \wedge \bigwedge_x SORT_x(x)$, where the conjunction extends over all variables of F and G, for each static law and each action dynamic law of the form

caused F **if** G

in D (this clause and all following clauses generalize the process of translating causal laws of $\mathcal{C}+$ into propositional causal logic described in (Giunchiglia *et al.* 2004, Section 4.2));

- the rules

$$i{:}F \Leftarrow i{:}F \wedge \bigwedge_x SORT_x(x),$$
$$i{:}\neg F \Leftarrow i{:}\neg F \wedge \bigwedge_x SORT_x(x),$$

where the conjunction extends over all variables of F, for each action dynamic law in D of the form

exogenous F;

- $i+1{:}F \Leftarrow i+1{:}G \wedge i{:}H \wedge \bigwedge_x SORT_x(x)$, where the conjunction extends over all variables of F, G and H, for each fluent dynamic law in D of the form

caused F **if** G **after** H;

- $i+1:F \Leftarrow i:G \wedge \bigwedge_x SORT_x(x)$, where the conjunction extends over all variables of F and G, for each fluent dynamic law in D of the form

G **causes** F;

- $i+1{:}F \Leftarrow i+1{:}F \wedge i{:}F \wedge \bigwedge_x SORT_x(x)$, where the conjunction extends over all variables of F, for each fluent dynamic law of the form

inertial F

in D such that F begins with a Boolean fluent name;

- $i+1{:}(t=y) \Leftarrow i+1{:}(t=y) \wedge i{:}(t=y) \wedge S(y) \wedge \bigwedge_x SORT_x(x)$, where the conjunction extends over all variables occurring in t, and y is a variable that doesn't occur in t, for each fluent dynamic law of the form

inertial t

in D such that t begins with a non-Boolean fluent name of a sort S;

- $\bot \Leftarrow i{:}(F \wedge G) \wedge \bigwedge_x SORT_x(x)$, where the conjunction extends over all variables of F and G, for each fluent dynamic law of the form

nonexecutable F **if** G

in D (if the part **if** G is missing, drop the conjunctive term G).

This concludes the definition of our translation from the language of action descriptions into the language of causal theories, and thus the definition of the semantics of action descriptions. Each model of the causal theory D_m represents a possible "history" of the state-transition system described by D over the time instants $0,\dots,m$. In particular, the models of D_0 are the *states* of D, and the models of D_1 are the *transitions* of D.

The causal theory RBP_m, corresponding to the action description *RBP* from Figure 1, is shown in Figure 2.

In the rest of the paper we argue in favor of the new approach to the semantics of action descriptions with variables by showing, first, that it has some desirable, intuitively expected mathematical properties, and second, that it has essential advantages in comparison with the approach based on grounding.

Three Theorems

Proposition 1 below shows that in any model of D_m the extent of any sort S corresponds to the set of the object names that one would expect to belong to S according to the object declarations and sort inclusions of D.

To make this claim precise, we will use the following notation. For any action description D, by $\mathcal{E}_D$ we denote the smallest set such that

- $(V,S) \in \mathcal{E}_D$ if V is declared in D as an object name of sort S;
- $(V,S) \in \mathcal{E}_D$ if $(V,S') \in \mathcal{E}_D$ and the sort inclusion part of D contains the subsort expression $S' \ll S$.

Proposition 1 *For any sort name S, D_m entails*

$$\forall x\,(\,S(x) \leftrightarrow \bigvee_{(V,S)\in\mathcal{E}_D} x = V\,).$$

For instance, any model of RBP_m satisfies (5).

The proof of Proposition 1 is based on the possibility of breaking D_m into disjoint parts in the sense of (Lifschitz 1997, Section 6) and on the relationship between causal logic and circumscription discussed in (Lifschitz 1997, Section 3).

In the theory of $\mathcal{C}+$, the view that histories of length m can be thought of as paths in a transition system is justified by two theorems, Propositions 7 and 8 from (Giunchiglia *et al.* 2004). The first of them shows that any transition "starts" in a state and "ends" in a state. According to the second theorem, an interpretation of the signature of D_m is a model of D_m if and only if it "consists of m transitions." Propositions 2 and 3 below are similar to these theorems.

Let D be an action description. For any interpretation I of σ^{D_1}, by I^0 and I^1 we denote the interpretations of σ^{D_0} defined as follows. (Here $|I|$ stands for the universe of I.)

$|I^i| = |I|$,
$I^i[S] = I[S]$ for every sort name S,
$I^i[V] = I[V]$ for every object name V,
$I^i[0{:}C] = I[i{:}C]$ for every fluent name C,
$I^i[\mathit{None}] = I[\mathit{None}]$.

Intuitively, I^0 and I^1 are the "endpoints" of transition I.

Notation: $V, V' \in \{R_1, R_2, B_1, B_2, P_1, P_2\}$, $k \in \{1, 2\}$, $i \in \{0, \dots, m-1\}$, $j \in \{0, \dots, m\}$.

$$
\begin{aligned}
\neg \mathit{Robot}(x) &\Leftarrow \neg \mathit{Robot}(x),\\
\neg \mathit{Box}(x) &\Leftarrow \neg \mathit{Box}(x),\\
\neg \mathit{Thing}(x) &\Leftarrow \neg \mathit{Thing}(x),\\
\neg \mathit{Place}(x) &\Leftarrow \neg \mathit{Place}(x),\\
\\
\mathit{Robot}(x) \rightarrow \mathit{Thing}(x) &\Leftarrow \top,\\
\mathit{Box}(x) \rightarrow \mathit{Thing}(x) &\Leftarrow \top,\\
\\
\mathit{Robot}(R_k) &\Leftarrow \top,\\
\mathit{Box}(B_k) &\Leftarrow \top,\\
\mathit{Place}(P_k) &\Leftarrow \top,\\
\\
\neg(V = \mathit{None}) &\Leftarrow \top,\\
\neg(V = V') &\Leftarrow \top \qquad (V \neq V'),\\
\\
\neg\, i\!:\!\mathit{Go}(x, y) &\Leftarrow \neg \mathit{Robot}(x),\\
\neg\, i\!:\!\mathit{Go}(x, y) &\Leftarrow \neg \mathit{Place}(y),\\
\neg\, i\!:\!\mathit{Carry}(x, y, z) &\Leftarrow \neg \mathit{Robot}(x),\\
\neg\, i\!:\!\mathit{Carry}(x, y, z) &\Leftarrow \neg \mathit{Box}(y),\\
\neg\, i\!:\!\mathit{Carry}(x, y, z) &\Leftarrow \neg \mathit{Place}(z),\\
j\!:\!\mathit{Location}(x) = \mathit{None} &\Leftarrow \neg \mathit{Thing}(x),\\
\neg\,(j\!:\!\mathit{Location}(x) = y) &\Leftarrow \neg \mathit{Place}(y) \wedge \mathit{Thing}(x),\\
\\
i+1\!:\!\mathit{Location}(x) = y &\Leftarrow i+1\!:\!\mathit{Location}(x) = y\\
&\quad \wedge i\!:\!\mathit{Location(x)} = y\\
&\quad \wedge \mathit{Place}(y) \wedge \mathit{Thing}(x),\\
i\!:\!\mathit{Go}(r, p) &\Leftarrow i\!:\!\mathit{Go}(r, p)\\
&\quad \wedge \mathit{Robot}(r) \wedge \mathit{Place}(p),\\
\neg\, i\!:\!\mathit{Go}(r, p) &\Leftarrow \neg\, i\!:\!\mathit{Go}(r, p)\\
&\quad \wedge \mathit{Robot}(r) \wedge \mathit{Place}(p),\\
i\!:\!\mathit{Carry}(r, b, p) &\Leftarrow i\!:\!\mathit{Carry}(r, b, p) \wedge \mathit{Box}(b)\\
&\quad \wedge \mathit{Robot}(r) \wedge \mathit{Place}(p),\\
\neg\, i\!:\!\mathit{Carry}(r, b, p) &\Leftarrow \neg\, i\!:\!\mathit{Carry}(r, b, p)\\
&\quad \wedge \mathit{Box}(b) \wedge \mathit{Robot}(r)\\
&\quad \wedge \mathit{Place}(p),\\
\\
i+1\!:\!\mathit{Location}(r) = p &\Leftarrow i\!:\!\mathit{Go}(r, p)\\
&\quad \wedge \mathit{Robot}(r) \wedge \mathit{Place}(p),\\
i+1\!:\!\mathit{Location}(r) = p &\Leftarrow i\!:\!\mathit{Carry}(r, b, p) \wedge \mathit{Box}(b)\\
&\quad \wedge \mathit{Robot}(r) \wedge \mathit{Place}(p),\\
i+1\!:\!\mathit{Location}(b) = p &\Leftarrow i\!:\!\mathit{Carry}(r, b, p) \wedge \mathit{Box}(b)\\
&\quad \wedge \mathit{Robot}(r) \wedge \mathit{Place}(p),\\
\\
\bot &\Leftarrow i\!:\!\mathit{Carry}(r, b, p)\\
&\quad \wedge i\!:\!\mathit{Go}(r, p) \wedge \mathit{Box}(b)\\
&\quad \wedge \mathit{Robot}(r) \wedge \mathit{Place}(p),\\
\bot &\Leftarrow i\!:\!\mathit{Carry}(r, b, p)\\
&\quad \wedge \neg(i\!:\!\mathit{Location}(r)\\
&\qquad = i\!:\!\mathit{Location}(b))\\
&\quad \wedge \mathit{Box}(b) \wedge \mathit{Robot}(r)\\
&\quad \wedge \mathit{Place}(p).
\end{aligned}
$$

Figure 2: Rules of causal theory RBP_m

Proposition 2 *For any transition I, the interpretations I^0 and I^1 are states.*

For any interpretation I of σ^{D_m}, by $I^{(i)}$ $(0 \leq i < m)$ we denote the interpretations of σ^{D_1} defined as follows:

$|I^{(i)}| = |I|$,
$I^{(i)}[S] = I[S]$ for every sort name S,
$I^{(i)}[V] = I[V]$ for every object name V,
$I^{(i)}[0\!:\!C] = I[i\!:\!C]$ for every fluent or action name C,
$I^{(i)}[1\!:\!C] = I[i+1\!:\!C]$ for every fluent name C,
$I^{(i)}[\mathit{None}] = I[\mathit{None}]$.

Intuitively, $I^{(i)}$ is the i-th "component transition" of "history" I.

Proposition 3 *For any positive integer m and any interpretation I of σ^{D_m}, I is a model of D_m iff every $I^{(i)}$ $(0 \leq i < m)$ is a transition.*

Proofs of Propositions 2 and 3 use Lemma 3 from (Lifschitz 1997).

Comparison with Grounding

Traditionally, the meaning of action descriptions with variables is defined in terms of a procedure based on grounding. In particular, such a procedure can be used to translate action descriptions discussed in this note into $\mathcal{C}+$. But this approach would lead to several complications that we have avoided.

First, to ground an expression with variables, we need to calculate, for each variable, the set of values that can be substituted for it. In our case, this would require that the definition of $\mathcal{E}_D$ above, or a similar definition, be made part of the semantics. The simple modular translation of object declarations and sort inclusions into causal logic that we have used is more attractive.

The syntax of $\mathcal{C}+$ treats an equality $c_1 = c_2$ as an atom only in the case when c_1 is a constant name and c_2 is an object name ("value"). If both c_1 and c_2 are object names, or both are constant names, additional postprocessing steps are required. In our approach, there is no need to distinguish between these cases.

Atoms in which a constant serves as an argument of another constant are particularly unpleasant for the grounding method. For instance, if the result of grounding contains the atom $\mathit{Go}(R_1, \mathit{Location}(R_2))$, expressing that robot R_1 goes to the place where robot R_2 is, then a postprocessing step will have to turn it into a long $\mathcal{C}+$ formula, such as

$$\bigvee_{i \in \{1,2\}} (\mathit{Location}(R_2) = P_i \ \wedge \ \mathit{Go}(R_1, P_i)).$$

Our semantics does not require such "expansion steps."

We expect that the advantages of the new approach to the semantics of action descriptions will become even more essential when we extend it to other syntactic constructs. Grounding is important as an implementation method, but it should be best avoided in the definition of semantics.

Conclusion

The semantics of action descriptions proposed in this paper combines attractive features of ADL and $\mathcal{C}+$. Like the former, it is based on state-transition models for languages with

variables and does not refer to grounding; like the latter, it uses a nonmonotonic causal logic to solve the ramification problem.

Our semantics of action descriptions is somewhat similar to the semantics of logic programming proposed in (Ferraris, Lee, & Lifschitz 2007): both refer to nonmonotonic translations into classical second-order logic and are, in this sense, similar to circumscription (McCarthy 1986). We expect that these parallel approaches to action descriptions and to stable models will help us extend the results on representing actions by logic programs from (Lifschitz & Turner 1999) to action descriptions with variables, and thus justify the use of answer set programming for the implementation of the new semantics. Our plans for the future include also extending the semantics to a modular action description language (Lifschitz & Ren 2006).

Acknowledgements

We are grateful to Selim Erdoğan, Paolo Ferraris, Joohyung Lee and Hudson Turner for comments on a draft of this paper. This work was partially supported by the National Science Foundation under Grant IIS-0412907.

References

Ferraris, P.; Lee, J.; and Lifschitz, V. 2007. A new perspective on stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 372–379.

Fikes, R., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2(3–4):189–208.

Gelfond, M., and Lifschitz, V. 1998. Action languages.[5] *Electronic Transactions on Artificial Intelligence* 3:195–210.

Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; and Turner, H. 2004. Nonmonotonic causal theories. *Artificial Intelligence* 153(1–2):49–104.

Kowalski, R., and Sergot, M. 1986. A logic-based calculus of events. *New Generation Computing* 4:67–95.

Lifschitz, V., and Ren, W. 2006. A modular action description language. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, 853–859.

Lifschitz, V., and Turner, H. 1999. Representing transition systems by logic programs. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 92–106.

Lifschitz, V. 1997. On the logic of causal explanation. *Artificial Intelligence* 96:451–465.

McCain, N., and Turner, H. 1997. Causal theories of action and change. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, 460–465.

McCarthy, J., and Hayes, P. 1969. Some philosophical problems from the standpoint of artificial intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence*, volume 4. Edinburgh: Edinburgh University Press. 463–502.

McCarthy, J. 1986. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence* 26(3):89–116.

Pednault, E. 1994. ADL and the state-transition model of action. *Journal of Logic and Computation* 4:467–512.

[5] `http://www.ep.liu.se/ea/cis/1998/016/` .