# Variables in Action Descriptions: Merging $\mathcal{C}+$ with ADL

**Vladimir Lifschitz and Wanwan Ren**
University of Texas, 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 $\mathcal{C}+$.

## 1 Introduction

Current research on the design of action description languages continues the line of work that started with the invention of ADL [Pednault, 1994]. Semantically, an action description represents a transition system ("state-transition model," in Pednault's terminology), that 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

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

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 this feature will bring significant advantages when applied to more complex action languages.

The central part of this paper is Section 5, which describes a new way to represent action descriptions by causal theories. It is preceded by the discussion of 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.

**sorts**
  *Room*;
**objects**
  *Room1*, *Room2*: *Room*;
**constants**
  *Location*: **fluent**(*Room*);
  *GoTo*(*Room*): **action**;
**variables**
  *r*: *Room*;
**axioms**
  **caused** $Location = r$ **if** $Location = r$
    **after** $Location = r$;
  **caused** $GoTo(r)$ **if** $GoTo(r)$;
  **caused** $\neg GoTo(r)$ **if** $\neg GoTo(r)$;
  **caused** $Location = r$ **if** $\top$ **after** $GoTo(r)$;

Figure 1: Action description $R$

## 2 Example

We assume that the reader has some familiarity with the syntactic constructs of $\mathcal{C}+$ [Giunchiglia *et al.*, 2004, Section 4.2].

An example illustrating the syntax of action descriptions used in this paper is shown in Figure 1. The abbreviations for causal laws of special kinds that are introduced in [Giunchiglia *et al.*, 2004, Appendix B] would allow us to write the last four lines of Figure 1 more concisely:

  **inertial** *Location*;
  **exogenous** $GoTo(r)$;
  $GoTo(r)$ **causes** $Location = r$;

but in the list of axioms all causal laws are written in full, because, for simplicity, we do not introduce these abbreviations in our grammar (see Section 3).

A definition in Section 5 below shows how to turn this action description $R$ into a sequence $R_0, R_1, \ldots$ of causal theories. It turns out that for any $m$, a model $I$ of the causal theory $R_m$ can be specified by selecting

- its universe $|I|$ — a non-empty set;
- a subset $I[Room]$ of $|I|$ — the set of rooms in the model;
- distinct elements $I[Room1]$, $I[Room2]$ of $|I|$ — the rooms represented by the object names — such that
$$I[Room] = \{I[Room1], I[Room2]\};$$
- for each $i \in \{0, \ldots, m\}$, an element $I[i : Location]$ of $I[Room]$ — the location of the agent at time $i$;
- for each $i \in \{0, \ldots, m-1\}$, a subset $I[i : GoTo]$ of $I[Room]$ — the set of rooms to which the agent goes between times $i$ and $i+1$ — such that
  - $I[i : GoTo]$ is either empty or a singleton, because actions in this domain cannot be executed concurrently,
  - if it is empty then $I[i+1 : Location] = I[i : Location]$, because the fluent *Location* is inertial,
  - if it is a singleton then its element equals $I[i+1 : Location]$, because after going to a room, the agent is in that room.

## 3 Syntax of Action Descriptions

An action description consists of five parts, as in the example above. The first of them is a list of sort names:

<sort declarations> ::= **sorts** {<sort name>';'}

The object declaration part is a list of object specifications. An object specification is a list of object names followed by a sort name:

<object declarations> ::= **objects** {<object spec>';'}
<object spec> ::= {<object name>','}
  <object name> ':' <sort name>

The constant declaration part is a list of constant specifications. Each constant represents either a simple fluent or an action; for simplicity, we do not allow statically determined fluent constants (see [Giunchiglia *et al.*, 2004, Section 4.2]). A fluent constant can be Boolean or non-Boolean. For simplicity, we do not allow non-Boolean action constants; also, non-Boolean fluent constants are not allowed to take arguments:

<constant declarations>
  ::= **constants** {<constant spec>';'}
<constant spec> ::= <Boolean fluent spec> |
  <non-Boolean fluent spec> |
  <action spec>
<Boolean fluent spec>
  ::= {<Boolean fluent schema>','}
  <Boolean fluent schema> ':' **fluent**
<non-Boolean fluent spec>
  ::= {<non-Boolean fluent name>','}
  <non-Boolean fluent name>
  ':' **fluent** '('<sort name>')'
<action spec> ::= {<action schema>','}
  <action schema> ':' **action**
<Boolean fluent schema>
  ::= <Boolean fluent name>
  ['('{<sort name>','} <sort name>')']
<action schema>
  ::= <action name>
  ['('{<sort name>','} <sort name>')']

The form of the variable declaration part is similar to the form of the object declaration part.

The axiom part is a list of causal laws:

<axioms> ::= **axioms** {<causal law>';'}
<causal law> ::= <static law> |
  <action dynamic law> |
  <fluent dynamic law>
<static law> ::= **caused** <fluent formula>
  **if** <fluent formula>
<action dynamic law> ::= **caused** <action formula>
  **if** <formula>
<fluent dynamic law> ::= **caused** <fluent formula>
  **if** <fluent formula>
  **after** <formula>

<fluent formula> ::= <formula>
<action formula> ::= <formula>

Formulas are formed using propositional connectives; for simplicity, quantifiers are not allowed:

<formula> ::= <atom> | '('<term>'='<term>')' |
 $\bot$ | $\top$ | '$\neg$'<formula> |
 '(' <formula> <binary connective>
 <formula> ')'
<atom>
 ::= (<Boolean fluent name> | <action name>)
 ['('{<argument>,} <argument>')']
<argument> ::= <object name> | <variable name>
<term> ::= <non-Boolean fluent name> |
 <object name> | <variable name>

Any name occurring in an action description should be declared exactly once. In declarations, atoms and terms, a name can only be used in accordance with its declaration.

A fluent formula cannot contain action names. An action formula should contain an action name, but it cannot contain fluent names.

## 4 Review of Causal Logic

The review of the syntax and semantics of causal theories in this section follows [Lifschitz, 1997, Section 2].

A *causal rule* is an expression of the form

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

where $F$ and $G$ are first-order formulas, called the *head* and the *body* of the rule. Expression (2) reads: there is a cause for $F$ if $G$ holds. A *causal theory* is defined by

- a finite subset of the signature[1] 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[2] to $E$, the result of replacing all occurrences of the constants $E$ in $F(E)$ by the variables $e$ can be denoted by $F(e)$.

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

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

Note that the occurrences of explainable symbols in the heads are replaced here by variables, and the occurrences in the bodies are not. We will 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

$$\begin{array}{l} \mathit{Room}(\mathit{Room1}) \Leftarrow \top, \\ \mathit{Room}(\mathit{Room2}) \Leftarrow \top, \\ \neg \mathit{Room}(x) \Leftarrow \neg \mathit{Room}(x), \end{array} \tag{4}$$

where the predicate constant *Room* is explainable, and the object constants *Room1*, *Room2* are not explainable. Intuitively, the last line of (4) expresses the closed-world assumption for *Room* in the language of causal logic: if $x$ is not a room then there is a cause for this. In this case, $E$ is *Room*, $e$ is a unary predicate variable *room*, and $T^*(\mathit{room})$ is

$$\begin{array}{l} \mathit{room}(\mathit{Room1}) \wedge \mathit{room}(\mathit{Room2}) \\ \qquad \wedge \forall x (\neg \mathit{Room}(x) \rightarrow \neg \mathit{room}(x)). \end{array}$$

The second-order sentence

$$\forall \mathit{room} (T^*(\mathit{room}) \leftrightarrow \mathit{room} = \mathit{Room})$$

can be equivalently rewritten as the first-order sentence

$$\forall x (\mathit{Room}(x) \leftrightarrow x = \mathit{Room1} \vee x = \mathit{Room2}). \tag{5}$$

## 5 Semantics of Action Descriptions

Given an action description $D$ and a nonnegative integer $m$, the corresponding causal theory $D_m$ is formed as follows.

Its signature $\sigma^{D_m}$ consists of

- an explainable unary predicate constant $S$ for each sort name $S$ declared in $D$;
- a non-explainable object constant $V$ for each object name $V$ declared in $D$;
- an explainable predicate constant $i : P$ for each Boolean fluent name $P$ declared in $D$, and every $i \in \{0, \ldots, m\}$; the arity of $i : P$ is the same as the arity of $P$;

[1] 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).

[2] 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.

- an explainable object constant $i : C$ for each non-Boolean fluent name $C$ declared in $D$, and every $i \in \{0, \dots, m\}$;
- an explainable predicate constant $i : P$ for each action name $P$ declared in $D$, and every $i \in \{0, \dots, m-1\}$; the arity of $i : P$ is the same as the arity of $P$.

For instance, the signature $\sigma^{R_m}$ corresponding to the action description $R$ shown in Figure 1 consists of the object constants

$$\mathit{Room1},\ \mathit{Room2},\ i : \mathit{Location}$$

and the unary predicate constants

$$\mathit{Room},\ i : \mathit{GoTo}.$$

Among these, *Room1* and *Room2* are non-explainable.

For any object name $V$ declared in $D$, $\mathit{SORT}_V$ stands for the sort name assigned to $V$ in the object declaration part, and similarly for variable names and for non-Boolean fluent names. By $i : F$ we denote the result of prepending $i :$ to all fluent names and action names in $F$.

The causal theory $D_m$ consists of the following rules:

(i) $\neg S(x) \Leftarrow \neg S(x)$ for each sort name $S$, where $x$ is an object variable;

(ii) $\mathit{SORT}_V(V) \Leftarrow \top$ for each object name $V$;

(iii) $V_1 \neq V_2 \Leftarrow \top$ for each pair of distinct object names $V_1$, $V_2$;

(iv) the rules

$$\begin{aligned} 0 : P(x_1, \dots, x_n) &\Leftarrow 0 : P(x_1, \dots, x_n) \\ &\qquad \wedge S_1(x_1) \wedge \dots \wedge S_n(x_n), \\ \neg\, 0 : P(x_1, \dots, x_n) &\Leftarrow \neg\, 0 : P(x_1, \dots, x_n) \\ &\qquad \wedge S_1(x_1) \wedge \dots \wedge S_n(x_n) \end{aligned}$$

for each Boolean fluent schema $P(S_1, \dots, S_n)$ from $D$, where $x_1, \dots, x_n$ are distinct object variables;

(v) the rules

$$\neg\, i : P(x_1, \dots, x_n) \Leftarrow \neg\, S_j(x_j) \qquad (1 \leq j \leq n)$$

for each Boolean fluent schema $P(S_1, \dots, S_n)$ from $D$ and $0 \leq i \leq m$, and for each action schema $P(S_1, \dots, S_n)$ from $D$ and $0 \leq i < m$, where $x_1, \dots, x_n$ are distinct object variables;

(vi) the rules

$$0 : C = x \Leftarrow 0 : C = x \wedge \mathit{SORT}_C(x)$$

and

$$\neg\, (i : C = x) \Leftarrow \neg\, \mathit{SORT}_C(x) \qquad (0 \leq i \leq m)$$

for each non-Boolean fluent name $C$, where $x$ is an object variable;

(vii) the rules

$$i : F \Leftarrow i : G \wedge \bigwedge_x \mathit{SORT}_x(x) \qquad (0 \leq i \leq m)$$

for each static causal law

**caused** $F$ **if** $G$

in $D$, where the conjunction is over all variables $x$ occurring in $F$ or $G$;

$$\begin{aligned} \neg \mathit{Room}(x) &\Leftarrow \neg \mathit{Room}(x), \\ \mathit{Room}(\mathit{Room1}) &\Leftarrow \top, \\ \mathit{Room}(\mathit{Room2}) &\Leftarrow \top, \\ \mathit{Room1} \neq \mathit{Room2} &\Leftarrow \top, \\ \neg\, i : \mathit{GoTo}(x) &\Leftarrow \neg \mathit{Room}(x) \qquad (0 \leq i < m), \\ 0 : \mathit{Location} = x &\Leftarrow 0 : \mathit{Location} = x \wedge \mathit{Room}(x), \\ \neg\, (i : \mathit{Location} = x) &\Leftarrow \neg \mathit{Room}(x) \qquad (0 \leq i \leq m), \\ i+1 : \mathit{Location} = r &\Leftarrow i+1 : \mathit{Location} = r \\ &\qquad \wedge i : \mathit{Location} = r \wedge \mathit{Room}(r), \\ i : \mathit{GoTo}(r) &\Leftarrow i : \mathit{GoTo}(r) \wedge \mathit{Room}(r), \\ \neg\, i : \mathit{GoTo}(r) &\Leftarrow \neg\, i : \mathit{GoTo}(r) \wedge \mathit{Room}(r), \\ i+1 : \mathit{Location} = r &\Leftarrow i : \mathit{GoTo}(r) \wedge \mathit{Room}(r) \\ &\qquad (0 \leq i < m). \end{aligned}$$

Figure 2: Rules of causal theory $R_m$

(viii) the rules

$$i : F \Leftarrow i : G \wedge \bigwedge_x \mathit{SORT}_x(x) \qquad (0 \leq i < m)$$

for each action dynamic causal law

**caused** $F$ **if** $G$

in $D$, where the conjunction is over all variables $x$ occurring in $F$ or $G$;

(ix) the rules

$$i+1 : F \Leftarrow i+1 : G \wedge i : H \wedge \textstyle\bigwedge_x \mathit{SORT}_x(x) \qquad (0 \leq i < m)$$

for each fluent dynamic causal law

**caused** $F$ **if** $G$ **after** $H$

in $D$, where the conjunction is over all variables $x$ occurring in $F$, $G$ or $H$.

Clauses (vii)–(ix) in this definition generalize the process of translating causal laws of $\mathcal{C}+$ into propositional causal logic described in [Giunchiglia *et al.*, 2004, Section 4.2].

Intuitively, the models of $D_m$ in the sense of Section 4 represent the possible behaviors, or "histories," of the state-transition system described by $D$ over successive time instants $0, 1, \dots, m$.

For instance, the causal theory $R_m$ corresponding to the action description $R$ from Figure 1 is shown in Figure 2. The models of this theory are described in Section 2 above.

For any sort name $S$, by $|S|$ we denote the set of all object names of sort $S$.

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

$$\forall x \left( S(x) \leftrightarrow \bigvee_{V \in |S|} x = V \right).$$

In other words, the extent of any sort in a model of $D_m$ is the set of elements of the universe representing the objects of that sort. For instance, any model of $R_m$ satisfies (5).

## 6 States and Transitions

The models of $D_0$ will be called *states*; the models of $D_1$ are *transitions*.

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 $\sigma^{D_1}$, by $I^0$ and $I^1$ we denote the interpretations of $\sigma^{D_0}$ 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 name $C$
$(i = 0, 1)$.

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

For any interpretation $I$ of $\sigma^{D_m}$, by $I^{(i)}$ $(0 \leq i < m)$ we denote the interpretations of $\sigma^{D_1}$ 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 $C$.

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

## 7 Reduction to $\mathcal{C}+$

As discussed in the introduction, our semantics treats variables in essentially the same way as the semantics of classical logic. On the other hand, when actions are described in $\mathcal{C}+$, variables can be only used in schematic expressions that represent groups of causal laws obtained from these expressions by grounding. To relate these two views to each other, we show in this section that grounding allows us to characterize the new semantics in terms of the semantics of $\mathcal{C}+$, at least under the assumption that the "extent" $|S|$ of every sort $S$ is non-empty. This assumption corresponds to the requirement, in the semantics of $\mathcal{C}+$, that the domain of every constant be a non-empty set.

Given an action description $D$ in the sense of Section 3 such that $|S| \neq \emptyset$ for each of its sort names $S$, the corresponding $\mathcal{C}+$ action description $D'$ is formed as follows. Its signature consists of

- Boolean simple fluent constants $C(V_1, \ldots, V_n)$ for each Boolean fluent schema $C(S_1, \ldots, S_n)$ in the constant declaration part of $D$, where $V_i$ $(1 \leq i \leq n)$ is an object name of sort $S_i$;
- a simple fluent constant $C$ with domain $|SORT_C|$, for each non-Boolean fluent name $C$ declared in $D$;
- Boolean action constants $C(V_1, \ldots, V_n)$ for each action schema $C(S_1, \ldots, S_n)$ in the constant declaration part of $D$, where $V_i$ $(1 \leq i \leq n)$ is an object name of sort $S_i$.

For instance, the signature of the $\mathcal{C}+$ action description $R'$ corresponding to action description $R$ (Figure 1) consists of three constants: the simple fluent constant *Location* with domain $\{Room1, Room2\}$ and the Boolean action constants $GoTo(Room1)$ and $GoTo(Room2)$.

The presence of variables in causal laws in the sense of Section 3 is not the only feature that makes them more general than the causal laws of $\mathcal{C}+$. In a formula of the form $t_1 = t_2$, we allow each of the terms $t_1$, $t_2$ to be an arbitrary object name or an arbitrary non-Boolean fluent name. In atoms in the sense of $\mathcal{C}+$, on the other hand, the left-hand side must be a constant, and the right-hand side must be an element of the domain of that constant. (Also, Boolean constants and equalities between two constants can be used as abbreviations; see [Giunchiglia *et al.*, 2004, Section 2.1].) For this reason, the causal laws of $D'$ are obtained from the causal laws in the axioms part of $D$ in two steps: first grounding, then adapting the form of equalities to the requirements of the syntax of $\mathcal{C}+$.

By $D'$ we denote the $\mathcal{C}+$ action description obtained from $D$ by

- grounding all causal laws of $D$ so that the symbols substituted for each variable $x$ are arbitrary object names of the sort $SORT_x$,
- then modifying the parts $t_1 = t_2$ in the expressions obtained after result of grounding, as follows:
  - whenever $t_1$ and $t_2$ are object names, replace $t_1 = t_2$ with $\top$ if $t_1$ equals $t_2$, and with $\bot$ otherwise;
  - then, whenever $t_1$ is an object name and $t_2$ is a fluent name, replace $t_1 = t_2$ with $t_2 = t_1$;
  - then, whenever $t_1$ is a fluent name and $t_2$ is an object name of a sort different from $SORT_{t_1}$, replace $t_1 = t_2$ with $\bot$.

(See Figure 3 for an example.)

Proposition 4 below shows how models of $D_m$ in the sense of Section 5 can be characterized in terms of models of $D'_m$ in the sense of $\mathcal{C}+$. In its statement, we refer to the following two conditions on an interpretation $I$ of $\sigma^{D_m}$:

(a) $I \models V_1 \neq V_2$ for any distinct object names $V_1$, $V_2$;

(b) $I \models \bigvee_{V \in |SORT_C|} i : C = V$ for any non-Boolean fluent name $C$ and any $i \in \{0, \ldots, m\}$.

For any interpretation $I$ of $\sigma^{D_m}$ satisfying these conditions, by $I'$ we denote the interpretation (in the sense of $C+$) of the signature $\sigma^{D'_m}$ such that

**caused** $Location = Room1$
  **if** $Location = Room1$ **after** $Location = Room1$,
**caused** $Location = Room2$
  **if** $Location = Room2$ **after** $Location = Room2$,
**caused** $GoTo(Room1)$ **if** $GoTo(Room1)$,
**caused** $GoTo(Room2)$ **if** $GoTo(Room2)$,
**caused** $\neg GoTo(Room1)$ **if** $\neg GoTo(Room1)$,
**caused** $\neg GoTo(Room2)$ **if** $\neg GoTo(Room2)$,
**caused** $Location = Room1$
  **if** $\top$ **after** $GoTo(Room1)$,
**caused** $Location = Room2$
  **if** $\top$ **after** $GoTo(Room2)$.

Figure 3: $\mathcal{C}+$ action description $R'$

- for each Boolean constant $i : C(V_1, \dots, V_n)$,
$$I'[i : C(V_1, \dots, V_n)] = I[i : C](I[V_1], \dots, I[V_n]);$$
- for each non-Boolean constant $i : C$, $I'[i : C]$ is the object name $V$ such that $I[i : C] = I[V]$.

(Conditions (a) and (b) guarantee the existence and uniqueness of such $V$.)

**Proposition 4** *An interpretation $I$ of $\sigma^{D_m}$ is a model of $D_m$ iff*

- $I \models \forall x \left( S(x) \leftrightarrow \bigvee_{V \in |S|} x = V \right)$,
- *$I$ satisfies conditions (a) and (b),*
- *$I'$ is a model of $D'_m$ (in the sense of $\mathcal{C}+$).*

Thus for the interpretations satisfying the formula from Proposition 1 and the conditions needed to define the mapping $I \mapsto I'$, the new semantics of action descriptions can be reduced to the semantics of $\mathcal{C}+$ by grounding.

## 8 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.

We expect that the advantages of the new approach to the semantics of action descriptions will become essential when we extend it to additional syntactic constructs, important for the purposes of knowledge representation. Here are two examples of such features, both implemented in the input language of the Causal Calculator (CCALC)[3].

The syntax defined in Section 3 allows the list of arguments in an atom to include object names and variable names, but not constant names. But it is sometimes convenient to write, for instance, $C_1(C_2)$, where $C_1$ is a Boolean fluent name and $C_2$ is a non-Boolean fluent name; this expression has the same meaning as
$$\exists x(C_2 = x \wedge C_1(x)),$$
where $x$ is a variable of the same sort as $C_2$. A semantics based on grounding has to be explicit about "expansion steps" like this; the semantics defined in Section 5 applies to the extended syntax without any changes.

Second, it is often convenient to declare one sort to be a subsort of another. In the new approach, the assertion that $S_1$ is a subsort of $S_2$ can be understood as
$$S_1(x) \rightarrow S_2(x) \Leftarrow \top.$$
Explaining subsort declarations in terms of grounding is more cumbersome.

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.

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


## 9 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, Lee, & Lifschitz, 2007] Ferraris, P.; Lee, J.; and Lifschitz, V. 2007. A new perspective on stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*. To appear.

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

[Lifschitz & Turner, 1999] 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, 1997] Lifschitz, V. 1997. On the logic of causal explanation. *Artificial Intelligence* 96:451–465.

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

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