Actions as Special Cases

Selim T. Erdoğan and Vladimir Lifschitz
Department of Computer Sciences
The University of Texas at Austin
1 University Station C0500
Austin, TX 78712-0233 USA
{selim,vl}@cs.utexas.edu

Abstract

This paper is motivated by the idea of interaction between two directions of research in knowledge representation: the design of action description languages and the development of libraries of reusable, general-purpose knowledge components. Writing an action description that characterizes actions in terms of their effects, as common today, can be compared to writing a program that does not use standard subroutines. We conjecture that a library of standard descriptions for a number of "basic" actions can facilitate writing, understanding and modifying action descriptions. In this paper, we take some steps towards determining how such a library, written in the action language $\mathcal{C}+$, can be used. When using an instance of a library action description, we relate the library constants to the domain-specific constants by providing definitions. Therefore, a theory of explicit definitions in $\mathcal{C}+$ is developed. To illustrate the use of the library, we show how the action *PushBox* in the Monkey and Bananas domain can be described as a special case of the "library action" *Move*.

Introduction

Research on describing actions started with the invention of STRIPS (Fikes & Nilsson 1971) and ADL (Pednault 1994) and led in recent years to the design of very expressive action languages, such as $\mathcal{C}+$ (Giunchiglia *et al.* 2004). The heart of every action language is a syntactic mechanism for describing effects of actions on fluents. When we define, for instance, the Monkey and Bananas domain in STRIPS, we can specify how pushing the box affects the location of the box by including appropriate atoms in the add list and delete list of the operator $PushBox(l)$. In $\mathcal{C}+$ the same idea can be expressed by the causal law

$$PushBox(l) \textbf{ causes } Loc(Box) = l \tag{1}$$

(quoted from (Giunchiglia *et al.* 2004), Figure 2, reproduced below).

Descriptions like these are common in knowledge representation, but they are strikingly different from the descriptions of actions that humans give to each other informally. The dictionary says, for instance, that pushing is *moving by steady pressure*. This phrase explains the meaning of the word *push* not by listing the effects of this action, but by presenting it as a special case of another action, *move*, that is supposed to be already familiar to the reader. Some actions may need to be described directly in terms of the changes that they cause; to move, for instance, means *to cause to change position*, according to the dictionary. But in most cases the easiest way to describe an action is to relate it to more basic actions.

We would like to apply the idea of a library of reusable, general-purpose knowledge components (Barker, Porter, & Clark 2001) to the design of action languages. We conjecture that a library of standard descriptions for "basic" actions, such as *move* and *give*, can facilitate writing, understanding and modifying action descriptions. Additionally, we conjecture that the constructs of language $\mathcal{C}+$ are rich enough to capture the features needed to build and use such a library.

In this paper, we take some steps towards determining how such a $\mathcal{C}+$ library of standard actions can be used when writing action descriptions. Specifically, we introduce a general form of causal laws for relating special-case actions and fluents to the library constants. These laws "explicitly define" a constant in terms of other constants. Therefore, after a brief review of $\mathcal{C}+$, we develop a theory of explicit definitions in $\mathcal{C}+$.

The laws used to define constants in terms of others can be called "bridge rules" because they provide a connection between the library and the specific domain description. With the theory of explicit definitions in $\mathcal{C}+$, bridge rules can be used to completely eliminate all references to the library and thus obtain an alternative action description in terms of the domain-specific constants.

The causal laws used in the bridge rules are nondefinite (see the review of $\mathcal{C}+$ below). Since the Causal Calculator[1] is an implementation of the definite fragment of $\mathcal{C}+$, it will not be possible to use that system, at least directly, to process action descriptions containing bridge rules. However, one of the propositions from our theory of explicit definitions shows that, under certain conditions, bridge rules may be replaced by definite causal laws.

In the rest of the paper, a specific example is used to illustrate how action domains can be specified with libraries. First we give a $\mathcal{C}+$ description of the action *Move* that can be included, in principle, in a library of general-purpose action

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

descriptions. Then we review the formalization of the Monkey and Bananas domain from (Giunchiglia *et al.* 2004) and show how to replace some of the $\mathcal{C}+$ laws in that formalization with a group of $\mathcal{C}+$ laws that characterizes *PushBox* as a special case of *Move*. It turns out that this reformulation is essentially equivalent to the original formalization. Finally, we demonstrate how this nondefinite reformulation involving the library may be turned into an equivalent definite action description.

Proofs are relegated to the appendix.

Review of $\mathcal{C}+$

The review of $\mathcal{C}+$ in this section follows (Giunchiglia *et al.* 2004).

A *(multi-valued) signature* is a set σ of symbols, called *(multi-valued) constants*, along with a non-empty finite set $Dom(c)$ of symbols, disjoint from σ, assigned to each constant c. The set $Dom(c)$ is the *domain* of c. Each constant belongs to one of three groups: *action* constants, *simple fluent* constants and *statically determined fluent* constants. For example, in the formalization of the Monkey and Bananas domain $Loc(Box)$ is a simple fluent constant with domain $\{L_1, L_2, L_3\}$.

Consider a fixed multi-valued signature σ. An *atom* is an expression of the form $c = v$ ("the value of c is v") where $c \in \sigma$ and $v \in Dom(c)$. A *formula* is a propositional combination of atoms. An *interpretation* maps every constant in σ to an element of its domain. A formula which is true under all interpretations is called *tautological*.

An *action description* is a set of *(causal) laws*—expressions of the form

$$\textbf{caused } F \textbf{ if } G \tag{2}$$

or

$$\textbf{caused } F \textbf{ if } G \textbf{ after } H \tag{3}$$

where F, G and H are formulas satisfying certain syntactic conditions. (See the appendix for details.) Depending on the form of F, G and H, each causal law belongs to one of three types: *static*, *action dynamic*, *fluent dynamic*. We say that two causal laws are *similar* if they are of the same type. Formula F is called the *head* of the law. If its head is an atom or $\bot$, a law is called *definite*.

Many useful constructs are defined as abbreviations for the basic forms (2) and (3) shown above.[2] For instance, law (1) stands for

$$\textbf{caused } Loc(Box) = l \textbf{ if } \top \textbf{ after } PushBox(l).$$

According to the semantics of $\mathcal{C}+$, every action description represents a transition system—a directed graph whose vertices are states, and whose edges are labeled by events. A state is characterized by the values of fluent constants; an event is characterized by the values of action constants. (See Figure 1 for an example.)

[2] The reader is referred to (Giunchiglia *et al.* 2004, Appendix B) for a detailed list.

Explicit Definitions in $\mathcal{C}+$

In classical logic, an explicit definition of a predicate constant P is an axiom of the form

$$P(x) \equiv \phi(x) \tag{4}$$

where ϕ is a formula that does not contain P. Such a definition has two properties. First, due to the equivalent replacement theorem of classical logic, if a theory contains axiom (4), any occurrences of P in other axioms may be eliminated. Second, adding axiom (4) to any theory which does not contain P yields a "conservative extension" of the original theory; any model of the new theory can be turned into a model of the original theory by dropping the predicate representing P.

Our goal is to develop a similar theory of explicit definitions in $\mathcal{C}+$.

An *explicit definition* of a multi-valued constant c, *in terms of* a multi-valued signature σ which does not contain c, is a set of causal laws of the form

$$\textbf{caused } c = v \equiv F_v \ , \tag{5}$$

one for each $v \in Dom(c)$, where

- each F_v is a formula of σ such that
  - if c is an action constant then F_v does not contain fluent constants;
  - if c is a statically determined fluent constant then F_v does not contain action constants;
  - if c is a simple fluent constant then F_v contains neither action constants nor statically determined fluent constants;
- the formulas
  $$\bigvee_{v \in Dom(c)} F_v$$
  and
  $$\bigwedge_{v,w \in Dom(c), v \neq w} \neg(F_v \wedge F_w)$$
  are tautological.

Intuitively, in view of the second condition, there is exactly one value of c corresponding to any interpretation of σ.

For example, the causal laws

$$\textbf{caused } Clear = L_1 \equiv (Loc(Box) = L_2 \vee Loc(Box) = L_3)$$

$$\textbf{caused } Clear = L_2 \equiv Loc(Box) = L_1$$

$$\textbf{caused } Clear = L_3 \equiv \bot$$

provide an explicit definition of the simple fluent constant *Clear* with domain $\{L_1, L_2, L_3\}$. Intuitively, *Clear* is the "first" location which is clear of the box.

The following counterpart of the equivalent replacement theorem from classical logic allows us to eliminate all occurrences of an explicitly defined constant except its occurrences in the definition:

Proposition 1 *Let F, G be formulas, let D be an action description, and let L, L' be similar causal laws such that L'*

is obtained from L by replacing an occurrence of F by G. Then the action description

$$D$$
$$L$$
$$\textbf{caused } F \equiv G$$

represents the same transition system as

$$D$$
$$L'$$
$$\textbf{caused } F \equiv G.$$

Proposition 2 below shows that adding an explicit definition of a new constant yields a "conservative extension." Let D and D' be action descriptions such that the signature of D is a part of the signature of D'. We say that D is a *residue* of D' if restricting the states and events of the transition system for D' to the signature of D establishes an isomorphism between the transition system for D' and the transition system for D.

Proposition 2 *Let D be an action description of a signature σ, and let c be a constant that does not belong to σ. If D' is an action description of the signature $\sigma \cup \{c\}$ obtained from D by adding an explicit definition of c in terms of σ, then D is a residue of D'.*

For instance, if D is an action description of a signature containing the fluent constant *Loc*(*Box*), c is *Clear*, and D' is obtained from D by adding the explicit definition of *Clear* shown above, then the transition system for D' is isomorphic to that for D. The latter can be obtained by restricting the states of the transition system for D' to the fluent constants other than *Clear*.

The causal laws used in explicit definitions are nondefinite because their heads are equivalences. In the general case, there is no known way to express definitions (with the two properties we would like them to have) using definite laws. However, if an action description does not refer to such a defined constant c in the heads of any laws other than the definition itself, then the definition may be equivalently expressed using definite causal laws:

Proposition 3 *Let σ be a signature and c be a constant that does not belong to σ. Let D be an action description of signature $\sigma \cup \{c\}$ which does not contain c in the heads of laws. Let D' be an action description of signature $\sigma \cup \{c\}$ obtained from D by adding an explicit definition (5) of c in terms of σ. Then the action description of signature $\sigma \cup \{c\}$ obtained from D by adding the rules*

$$\textbf{caused } c = v \textbf{ if } F_v \qquad (v \in Dom(c))$$

represents the same transition system as D'.

Explicit definitions will play an essential role in relating special-case actions and fluents to actions and fluents in a general-purpose library. Such definitions constitute the "bridge rules" providing a connection between the library and the specific domain description.

The rest of the paper focuses on an example of using a library description of action *Move* to reformalize the Monkey and Bananas domain from (Giunchiglia *et al.* 2004).

Moving Things

Our "general-purpose" formalization of the action *Move* is a family of $\mathcal{C}+$ action descriptions depending on two parameters. For any nonempty finite sets P, L of symbols, the action description $MOVE(P, L)$ below represents the properties of moving physical objects (elements of P) to locations (elements of L).

The signature and the causal laws of $MOVE(P, L)$ are as follows:

Notation: p, p_1 range over P; l ranges over L.

Simple fluent constants:	Domains:
$Location(p)$	L

Action constants:	Domains:
$Move(p)$	Boolean
$Mover(p)$	$P \cup \{None\}$
$Destination(p)$	$L \cup \{None\}$

Causal laws:

$$\textbf{always } Mover(p) = None \equiv \neg Move(p) \tag{6}$$

$$\textbf{always } Destination(p) = None \equiv \neg Move(p) \tag{7}$$

$$Move(p) \textbf{ causes } Location(p) = l \textbf{ if } Destination(p) = l \tag{8}$$

$$Move(p) \textbf{ causes } Location(p_1) = l \textbf{ if } Mover(p) = p_1 \wedge Destination(p) = l \tag{9}$$

$$\textbf{nonexecutable } Move(p) \textbf{ if } Location(p) = Destination(p) \tag{10}$$

$$\textbf{nonexecutable } Move(p) \textbf{ if } Mover(p) = p_1 \wedge Location(p_1) \neq Location(p) \tag{11}$$

$$\textbf{exogenous } Move(p) \tag{12}$$

$$\textbf{exogenous } Mover(p)$$

$$\textbf{exogenous } Destination(p)$$

$$\textbf{inertial } Location(p) \tag{13}$$

The constants $Mover(p)$ and $Destination(p)$ are used here as attributes of the action $Move(p)$ in the sense of (Giunchiglia *et al.* 2004, Section 5.6). When the action $Move(p)$ is executed, the value of $Mover(p)$ is the agent executing that action, and the value of $Destination(p)$ is the location to which p is being moved; otherwise the value of each attribute is *None* ("undefined"). Executing $Move(p)$ causes the location of p and the location of $Mover(p)$ to be equal to $Destination(p)$. The action is not executable if $Destination(p)$ is the current location of p, and also if p and $Mover(p)$ are in different places.

Consider, for example, the transition system represented by the action description

$$MOVE(\{Monkey, Box, Bananas\}, \{L_1, L_2, L_3\}). \tag{14}$$

(This choice of "actual parameters," substituted for the "formal parameters" P, L, corresponds to the use of *MOVE* in the next section.) This graph has 27 vertices, corresponding to the states—assignments of locations L_1, L_2, L_3 to fluents

$Location(Monkey)$, $Location(Box)$ and $Location(Bananas)$. Every edge of this graph is labeled by an event—an assignment of values to the action constants. In one of these events, for instance, the monkey is moving the box from L_2 to L_3, where the bananas are. The corresponding edge of the graph is shown in Figure 1.

Pushing the Box as a Special Case of Moving

Here are the signature and the causal laws of the action description *MB*, proposed in (Giunchiglia *et al.* 2004, Figure 2) as a description of the Monkey and Bananas domain in $\mathcal{C}+$:

Notation: x ranges over $\{Monkey, Bananas, Box\}$; l ranges over $\{L_1, L_2, L_3\}$.

Simple fluent constants:	Domains:
$Loc(x)$	$\{L_1, L_2, L_3\}$
$HasBananas, OnBox$	Boolean

Action constants:	Domains:
$Walk(l), PushBox(l)$	Boolean
$ClimbOn, ClimbOff, GraspBananas$	Boolean

Causal laws:

caused $Loc(Bananas) = l$
  if $HasBananas \wedge Loc(Monkey) = l$

caused $Loc(Monkey) = l$ **if** $OnBox \wedge Loc(Box) = l$

$Walk(l)$ **causes** $Loc(Monkey) = l$

nonexecutable $Walk(l)$ **if** $Loc(Monkey) = l$

nonexecutable $Walk(l)$ **if** $OnBox$

$PushBox(l)$ **causes** $Loc(Box) = l$ (15)

$PushBox(l)$ **causes** $Loc(Monkey) = l$ (16)

nonexecutable $PushBox(l)$ **if** $Loc(Monkey) = l$ (17)

nonexecutable $PushBox(l)$ **if** $OnBox$

nonexecutable $PushBox(l)$
  if $Loc(Monkey) \neq Loc(Box)$ (18)

$ClimbOn$ **causes** $OnBox$

nonexecutable $ClimbOn$ **if** $OnBox$

nonexecutable $ClimbOn$ **if** $Loc(Monkey) \neq Loc(Box)$

$ClimbOff$ **causes** $\neg OnBox$

nonexecutable $ClimbOff$ **if** $\neg OnBox$

$GraspBananas$ **causes** $HasBananas$

nonexecutable $GraspBananas$ **if** $HasBananas$

nonexecutable $GraspBananas$ **if** $\neg OnBox$

nonexecutable $GraspBananas$
  if $Loc(Monkey) \neq Loc(Bananas)$

nonexecutable $Walk(l) \wedge PushBox(l)$

nonexecutable $Walk(l) \wedge ClimbOn$

nonexecutable $PushBox(l) \wedge ClimbOn$

nonexecutable $ClimbOff \wedge GraspBananas$

exogenous $Walk(l)$

exogenous $PushBox(l)$ (19)

exogenous $ClimbOn$

exogenous $ClimbOff$

exogenous $GraspBananas$

inertial $Loc(x)$ (20)

inertial $HasBananas$

inertial $OnBox$

Action *PushBox* is a special case of *Move*, in which the object that is being moved is the box, the mover is the monkey, and the destination may be any one of the locations L_1, L_2, L_3. On the right margin we assigned numbers to the causal laws of *MB* that have counterparts in $MOVE(P, L)$. Our goal is to find a collection of causal laws ("bridge rules") relating *MB* to $MOVE(P, L)$ that will make (15)–(20) redundant. Causal laws (15) and (16), describing the effects of *PushBox*, will become "special cases" of (8) and (9), which describe the effects of *Move*. Causal laws (17) and (18), describing some of the preconditions of *PushBox*, will become redundant in the presence of the general preconditions (10) and (11) of *Move*. (The other precondition of the action *PushBox*—the fact that it cannot be executed if the monkey is on the box—is domain-specific and has no counterpart in the "library description" $MOVE(P, L)$.) Finally, (19) and (20) will become redundant in the presence of (12) and (13).

Our reformulation MB^* of *MB* is defined as follows. Its signature is the union of the signature of *MB* with the signature of the instance (14) of the "library description" of *Move*. Its causal laws are

- the causal laws of *MB*, except (15)–(20),
- the causal laws of (14), and
- the following causal laws, connecting (14) with *MB*:

caused $Location(p) = Loc(p)$ (21)

caused $Move(Box) \equiv \bigvee_l PushBox(l)$ (22)

caused $\neg Move(p)$ $(p \neq Box)$ (23)

caused $Mover(Box) = Monkey \equiv Move(Box)$ (24)

caused $Destination(Box) = l \equiv PushBox(l)$ (25)

where p ranges over $\{Monkey, Box, Bananas\}$, and l over $\{L_1, L_2, L_3\}$.

Laws (21)–(25) are the bridge rules, connecting the domain-specific description (*MB* without laws (15)–(20)) with the library (14). Law (21) says that *Location* is synonymous with *Loc*. Laws (22) and (23) tell us that moving the box amounts to pushing it to some location, and that no object other than the box is ever moved. According to (24),

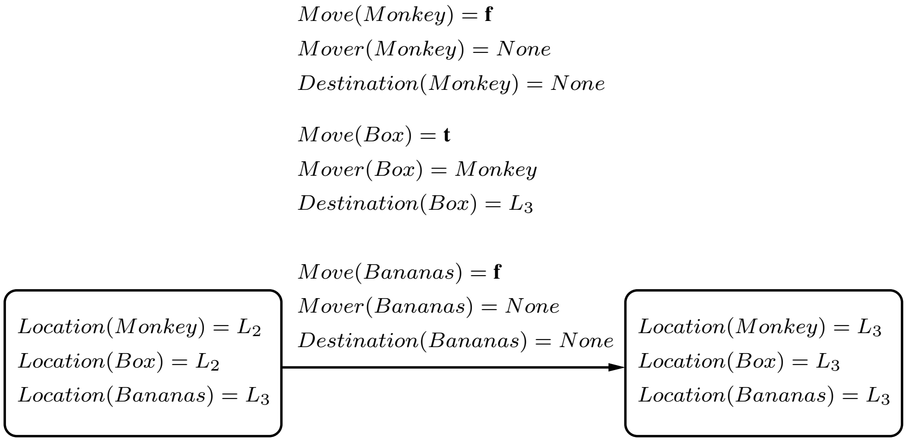

Figure 1: An edge of the graph represented by action description (14)

the mover is the monkey whenever the box is being moved. According to (25), the destination is l whenever the box is pushed to l.

We mentioned earlier that our bridge rules would take the form of explicit definitions. Specifically, every bridge rule defines a constant from a library instance in terms of the domain-specific signature. For example, the first, (21) is in fact equivalent to

$$\textbf{caused } \mathit{Location}(p)\!=\!l \equiv \mathit{Loc}(p)\!=\!l \qquad (l \in \{L_1, L_2, L_3\}) \quad (26)$$

which is an explicit definition of *Location* in terms of the signature of *MB*. Laws (22)–(25) can be rewritten as explicit definitions of the other constants from (14), using simple equivalent transformations.

Action description *MB** is not exactly equivalent to *MB*, because its signature is different. The proposition below shows, however, that the transition systems represented by *MB* and *MB** are isomorphic to each other. In this sense, our reformulation of *MB* based on the "toy library" is adequate.

Proposition 4 *MB is a residue of MB**.

The proof of this proposition, outlined in the appendix, relies on Propositions 1 and 2.

Whenever we have an action description containing bridge rules that explicitly define all constants from the library (such as *MB**), we may obtain a residue for it that does not contain the library constants, in two steps. First, we apply Proposition 1 to turn the causal laws coming from the library into equivalent laws involving only domain-specific constants. Then the bridge rules will be the only laws referring to the library. Second, we drop the bridge rules to obtain an action description for the domain, which doesn't refer to the library at all. By Proposition 2, this new description will be a residue. Applying this procedure to *MB** will yield an action description which has the same transition system as *MB*.

Turning *MB** into a Definite Theory

We have shown that the formalization of the Monkey and Bananas domain may be reformulated using our "toy library" *MOVE*(P, L), as the nondefinite action description *MB**. As discussed in the introduction, from an implementation point of view it is important to be able to turn a nondefinite action description into a definite one. Here we show how to do this for *MB**.

The first nondefinite causal law in *MB** is (21), which is equivalent to (26). We would like to use Proposition 3 to make it definite. However, Proposition 3 is not directly applicable because laws (8), (9) and (13) contain *Location* in their heads. Therefore we first use Proposition 1 in the presence of (21) to replace *Location* by *Loc* in the heads of (8), (9) and (13). Now we may use Proposition 3 to replace (21) with the definite causal laws

$$\textbf{caused } \mathit{Location}(p)\!=\!l \textbf{ if } \mathit{Loc}(p)\!=\!l \quad (l \in \{L_1, L_2, L_3\}).$$

The remaining nondefinite laws (22)–(25) contain only action constants. They may be transformed into definite laws using Proposition 5 below.

A constant c is said to be *exogenous* in an action description D if the action description contains the causal laws

$$\textbf{caused } c\!=\!v \textbf{ if } c\!=\!v$$

for all values $v \in \mathit{Dom}(c)$.

Proposition 5 *Let D be an action description and F be a formula such that all constants in F are action constants which are exogenous in D. Then*

$$\begin{array}{l} D \\ \textbf{caused } F \textbf{ if } G \end{array}$$

represents the same transition system as

$$\begin{array}{l} D \\ \textbf{caused } \bot \textbf{ after } \neg F \wedge G. \end{array}$$

This proposition is often applicable to a causal law containing only action constants, such as (22)–(25), because action constants are usually exogenous. For instance, in the presence of (12), we can replace (23) with **caused** $\bot$ **if** *Move*(p).

Conclusion

The premise for this work is that having a library of descriptions of "basic" actions can facilitate the process of writing domain-specific action descriptions. To illustrate this idea, we presented a general-purpose $\mathcal{C}+$ description of the action *Move*, and showed that a group of causal laws in the $\mathcal{C}+$ description of the Monkey and Bananas domain could be replaced by a reference to this "library."

When using a general-purpose library together with a domain-specific description, we need a set of causal laws relating the constants from either side to each other. These laws, which we call "bridge rules," can be seen as defining constants from an instance of a library action description in terms of domain-specific constants. Therefore we developed a theory of explicit definitions in $\mathcal{C}+$, which includes three theorems. One is a counterpart of the equivalent replacement theorem from classical logic. The second corresponds to the theorem asserting that definitional extensions are conservative. The third states that, under certain conditions, explicit definitions may be replaced by definite laws.

Making nondefinite action descriptions definite is useful for implementation purposes. We provided an example of how this may be done, by turning our nondefinite formalization of the Monkey and Bananas domain into an equivalent definite description.

This paper illustrated the suitability of $\mathcal{C}+$ for building a library of general-purpose action descriptions by showing how to use a general purpose description of a single action to describe another action which is a special case. In practice it is essential to be able to use the same general-purpose action description from the library multiple times in a new action description. For instance, of the actions in *MB*, three others besides *PushBox* can be expressed as special cases of *Move*. The actions *Walk*, *ClimbOn* and *ClimbOff* may be viewed as the monkey moving itself. Lifschitz and Ren (2006) introduce a modular language for describing actions, based on $\mathcal{C}+$, which provides the capability of "importing" (possibly several) instances of action descriptions into other action descriptions.

Acknowledgments

We are grateful to Paolo Ferraris, Joohyung Lee and Wanwan Ren for useful discussions and to the anonymous reviewers for their comments. This work is partially supported by the National Science Foundation under Grant IIS-0412907.

Appendix: Proofs

We begin by providing a little more information about the syntax and semantics of action descriptions, in addition to what we said at the beginning of the paper. Then, since the semantics of an action description is defined in terms of a causal theory, we prove several lemmas about causal theories. These lemmas are then used in the proofs of Propositions 1–5.

Syntax and Semantics of Action Descriptions

Given a $\mathcal{C}+$ signature σ a *fluent formula* is a formula such that all constants occurring in it are fluent constants, and an *action formula* is a formula that contains at least one action constant and no fluent constants. A *static law* is an expression of the form (2) in which F and G are fluent formulas. An *action dynamic law* is an expression of the form (2) in which F is an action formula and G is a formula. A *fluent dynamic law* is an expression of the form (3) in which F and G are fluent formulas, H is a formula, and F does not contain any statically determined constants. An *action description* is a set consisting of static laws, action dynamic laws and fluent dynamic laws.

The transition system $TS(D)$ represented by an action description D is defined in terms of a sequence $D_0, D_1, \ldots$ of causal theories. A detailed description of the syntax and semantics of causal theories may be found in (Giunchiglia *et al.* 2004, Secs. 2.1–2.3). Recall that a rule of a causal theory has the form $F \Leftarrow G$. The *reduct* T^I of a causal theory T relative to an interpretation I is the set of the heads of all rules in T whose bodies are satisfied by I. If I is the unique model of T^I, then it is a *model* of T.

For any action description D and any nonnegative integer m, the causal theory D_m is defined as follows. The signature σ_m of D_m consists of the pairs $i : c$ such that

- $i \in \{0, \ldots, m\}$ and c is a fluent constant of D, or
- $i \in \{0, \ldots, m-1\}$ and c is an action constant of D.

The domain of $i : c$ is the same as the domain of c. The expression $i : F$ denotes the result of inserting $i :$ in front of every occurrence of every constant in a formula F. Intuitively, if c is a fluent constant, $i : c$ represents the value of c at time i; if c is an action constant, $i : c$ represents the value of c between times i and $i+1$.

The rules of D_m are:

$$i : F \Leftarrow i : G$$

for every static law (2) in D and every $i \in \{0, \ldots, m\}$, and for every action dynamic law (2) in D and every $i \in \{0, \ldots, m-1\}$;

$$i+1 : F \Leftarrow (i+1 : G) \wedge (i : H)$$

for every fluent dynamic law (3) in D and every $i \in \{0, \ldots, m-1\}$;

$$0 : c = v \Leftarrow 0 : c = v$$

for every simple fluent constant c and every $v \in Dom(c)$.

The transition system $TS(D)$ is completely characterized by the first two members D_0, D_1 of the sequence of causal theories corresponding to D, as follows:

- A state in $TS(D)$ is an interpretation s of the fluent constants such that the corresponding interpretation $0 : s$ of the signature of D_0 is a model of D_0.

- A transition in $TS(D)$ is a triple $\langle s, e, s' \rangle$ where s and s' are interpretations of the fluent constants and e is an interpretation of the action constants, such that the corresponding interpretation $(0 : s) \cup (0 : e) \cup (1 : s')$ of the signature of D_1 is a model of D_1.

Some Properties of Causal Theories

The following is a counterpart of Proposition 2 for causal theories. It is a restatement of Proposition 1 from (Turner 2004).

Lemma 1 *Let T be a causal theory containing a rule of the form*

$$F \equiv G \Leftarrow \top.$$

The causal theory obtained by replacing an occurrence of F by G in any other rule of T has the same models as T.

Since Propositions 2 and 3 are about explicit definitions in $\mathcal{C}+$, we define the counterpart of this concept for causal theories. An *explicit definition* of a multi-valued constant c, *in terms of* a multi-valued signature σ which does not contain c, is a set of causal laws of the form

$$c = v \equiv F_v \;\Leftarrow \top, \tag{27}$$

one for each $v \in Dom(c)$, where

- each F_v is a formula of σ, and
- the formulas
$$\bigvee_{v \in Dom(c)} F_v$$
and
$$\bigwedge_{v,w \in Dom(c), v \neq w} \neg(F_v \wedge F_w)$$
are tautological.

The following is a counterpart of Proposition 2 for causal theories.

Lemma 2 *Let T be a causal theory of a signature σ, and let c be a constant that does not belong to σ. If T' is a causal theory of the signature $\sigma \cup \{c\}$ obtained from T by adding an explicit definition of c in terms of σ, then $X \mapsto X|_\sigma$ is a 1-1 correspondence between the models of T' and the models of T.*

Proof: Let the set of formulas in the heads of rules (27) be called C. The task of proving that $X \mapsto X|_\sigma$ is a 1-1 correspondence between the models of T' and the models of T can be divided into three parts.

Part I: Showing that if X is a model of T' then $X|_\sigma$ is a model of T.

Assume that X is a model of T'. Then $X \models (T')^X$, and consequently $X \models T^X$. Since T doesn't contain c, $X \models T^{X|_\sigma}$. Since $T^{X|_\sigma}$ doesn't contain c, it follows that $X|_\sigma \models T^{X|_\sigma}$.

We also need to show that $X|_\sigma$ is the unique model of $T^{X|_\sigma}$. Let Y be any model of $T^{X|_\sigma}$. Define a new interpretation Y' of the signature $\sigma \cup \{c\}$ such that $Y'|_\sigma = Y$ and $Y'(c)$ is $v \in Dom(c)$ for which Y satisfies F_v. (Under the assumptions of the theorem, such a v is unique.) Clearly, Y' satisfies $T^{X|_\sigma}$ because Y does. Also, Y' was defined in a way that ensures it satisfies C. Therefore Y' is a model of $(T^{X|_\sigma} \cup C) = (T')^X$. Since X is the unique model of $(T')^X$, it follows that $Y' = X$, and $Y = Y'|_\sigma = X|_\sigma$. Thus $X|_\sigma$ is a model of T.

Part II: Showing that every model of T can be represented in the form $X|_\sigma$, where X is a model of T'.

Let Y be a model of T. Define a new interpretation X of the signature $\sigma \cup \{c\}$ such that $X|_\sigma = Y$ and $X(c)$ is $v \in Dom(c)$ for which Y satisfies F_v. Clearly, X satisfies T^Y because Y does. Also, X is defined to satisfy C. Therefore X is a model of $(T^Y \cup C) = (T')^X$. Now we need to show that X is the unique model of $(T')^X$.

Let X' be any model of $(T')^X$. Then X' satisfies $(T')^X = T^Y \cup C$. Since T doesn't contain c, $X'|_\sigma$ satisfies T^Y. But Y is the unique model of T^Y so $X'|_\sigma = Y = X|_\sigma$. Since X' satisfies C,

$$X' \models c = X'(c) \equiv F_{X'(c)}$$

so that $X' \models F_{X'(c)}$. Then $Y \models F_{X'(c)}$. By the choice of $X(c)$, $Y \models F_{X(c)}$. It follows that $X(c) = X'(c)$. We have shown that $X' = X$, so X is the unique model of $(T')^X$, and therefore a model of T', such that $X|_\sigma = Y$.

Part III: Showing that no model of T can be represented in the form $X|_\sigma$, where X is a model of T', in more than one way.

Let X and Z be models of T' such that $X|_\sigma = Z|_\sigma$. Then X is the unique model of $(T')^X$ and Z is the unique model of $(T')^Z$. Since the bodies of rules in T' don't contain c, $(T')^Z = (T')^X$. So Z is the unique model of $(T')^X$. Therefore $Z = X$.

The following is a counterpart of Proposition 3 for causal theories.

Lemma 3 *Let σ be a signature and c be a constant that does not belong to σ. Let T be a causal theory of signature $\sigma \cup \{c\}$ which does not contain c in the heads of rules. Let T' be a causal theory of signature $\sigma \cup \{c\}$ obtained from T by adding an explicit definition (27) of c in terms of σ. Then the causal theory T'' of signature $\sigma \cup \{c\}$ obtained from T by adding the rules*

$$c = v \Leftarrow F_v \qquad (v \in Dom(c))$$

has the same models as T'.

Proof: *Left to right:* Let X be a model for T'. Then X is the unique model of $T^X \cup (T' \setminus T)^X$. Since X is a model of $(T' \setminus T)^X$, $X \models F_{X(c)}$ and, for all $w \neq X(c)$, $X \not\models F_w$. Therefore, $(T'' \setminus T)^X = \{c = X(c)\}$ and X is a model for $(T'' \setminus T)^X$. Consequently, X is a model for $(T'')^X$. We need to show that it is the unique model.

Let $Y \models (T'')^X$. Since $(T'' \setminus T)^X = \{c = X(c)\}$, $Y(c) = X(c)$. Since (27) is an explicit definition, for some $w \in Dom(c)$, $Y \models F_w$ and, for all $x \neq w$, $Y \not\models F_x$. Take the interpretation Y' of $\sigma \cup \{c\}$ such that $Y'|_\sigma = Y|_\sigma$ and $Y'(c) = w$. Then $Y' \models (T' \setminus T)^X$ and $Y' \models T^X$ (since $Y \models T^X$ and T^X doesn't contain c). By the fact that X is the unique model of $(T')^X$, $Y' = X$. So $Y|_\sigma = Y'|_\sigma = X|_\sigma$. Consequently, $Y = X$.

Right to left: Let X be a model for T''. Then X is the unique model of $T^X \cup (T'' \setminus T)^X$. Since (27) is an explicit definition, $(T'' \setminus T)^X$ is a singleton, $\{c = X(c)\}$. Consequently $X \models F_{X(c)}$ and for all $w \neq X(c)$, $X \not\models F_w$. Then $X \models (T' \setminus T)^X$ and X is a model for $(T')^X$. We need to show that it is the unique model.

Let $Y \models (T')^X$. Take the interpretation Y' of $\sigma \cup \{c\}$ such that $Y'|_\sigma = Y|_\sigma$ and $Y'(c) = X(c)$. Then $Y' \models T^X$ (since $Y \models T^X$ and T^X doesn't contain c) and $Y' \models (T'' \setminus T)^X$. Since X is the unique model of $(T'')^X$, $Y' = X$ so $Y|_\sigma = Y'|_\sigma = X|_\sigma$. Since X satisfies only $F_{X(c)}$ among formulas F_v ($v \in Dom(c)$) and these formulas don't contain c, Y also satisfies only $F_{X(c)}$. Since $Y \models (T' \setminus T)^X$, $Y(c) = X(c)$.

To prove Proposition 3 we will also need the following modification of Lemma 3.

Lemma 4 *Let σ be a signature and c be a constant that does not belong to σ. Let T be a causal theory of signature $\sigma \cup \{c\}$ which does not contain c in the heads of rules. Let T' be a causal theory of signature $\sigma \cup \{c\}$ obtained from T by adding an explicit definition (27) of c in terms of σ and the rules*

$$c = v \quad \Leftarrow \quad c = v \tag{28}$$

$$F_v \quad \Leftarrow \quad F_v \tag{29}$$

for all v from $Dom(c)$. Then the causal theory T'' of signature $\sigma \cup \{c\}$ obtained from T by adding rules (28), (29) and

$$c = v \Leftarrow F_v \qquad (v \in Dom(c))$$

has the same models as T'.

The proof is very similar to that of Lemma 3. (Instead of constructing Y', we can simply use Y.)

The other lemmas that we need are related to the concept of strong equivalence, which was originally introduced for logic programs in (Lifschitz, Pearce, & Valverde 2001) and was extended to causal theories in (Turner 2004); also see (Sergot & Craven 2005).

Causal theories T_1 and T_2 of the same signature σ are *equivalent* if they have the same models. They are *strongly equivalent* if, for every causal theory T of a signature σ' containing σ, the theories $T_1 \cup T$ and $T_2 \cup T$ of the signature σ' are equivalent.

Lemma 5 *Let T_1 and T_2 be causal theories with a common signature, such that for any interpretation J of their signature, T_1^J is equivalent to T_2^J. Then T_1 and T_2 are strongly equivalent.*

This lemma is slightly weaker than Theorem 1 from (Turner 2004), which gives a complete characterization of strong equivalence in terms of pairs of interpretations.

Proof: Let T be a causal theory. For any interpretation J of the signature of T,

$$\begin{array}{ll} & J \text{ is a model of } T_1 \cup T \\ \text{iff} & J \text{ is the unique model of } T_1^J \cup T^J \\ \text{iff} & J \text{ is a model of } T_1^J \cup T^J \\ & \text{ and for any model } I \text{ of } T_1^J \cup T^J, I = J \\ \text{iff} & J \text{ is a model of } T_2^J \cup T^J \\ & \text{ and for any model } I \text{ of } T_2^J \cup T^J, I = J \\ \text{iff} & J \text{ is the unique model of } T_2^J \cup T^J \\ \text{iff} & J \text{ is a model of } T_2 \cup T. \end{array}$$

Lemma 6 *Let F be a formula of the signature σ. The causal theory T consisting of rules of the form*

$$a \Leftarrow a$$

for all atoms a of σ, is strongly equivalent to the theory obtained from T by adding the rule

$$F \Leftarrow F.$$

Proof: Call the second theory T'. By Lemma 5, all we need to check is that, for any interpretation J of T, T^J and $(T')^J$ are equivalent. Note that due to the form of the rules in T, for any interpretation J of σ, T^J is satisfied only by J. Similarly, $(T')^J$ is also satisfied by J only.

The following is a counterpart of Proposition 5 for causal theories.

Lemma 7 *Let F be a formula of a signature σ and G be a formula of a signature σ' containing σ. Let T be the causal theory of consisting of the rules*

$$a \Leftarrow a$$

for all atoms a of σ. Then

$$\begin{array}{l} T \\ F \Leftarrow G \end{array}$$

is strongly equivalent to

$$\begin{array}{l} T \\ \bot \Leftarrow \neg F \wedge G. \end{array}$$

Proof: Call the first theory T_1, the second T_2. By Lemma 5, all we need to check is that, for any interpretation J of σ', T_1^J and T_2^J are equivalent. Due to the form of the rules in T, for any interpretation I of σ', if I satisfies T^J, then $I|_\sigma = J|_\sigma$ and consequently $I \models F$ iff $J \models F$. It follows that

$$\begin{array}{ll} & I \models T_1^J \\ \text{iff} & I \models T^J \text{ and } (I \models F \text{ if } J \models G) \\ \text{iff} & I \models T^J \text{ and } (J \models F \text{ if } J \models G) \\ \text{iff} & I \models T^J \text{ and } J \models G \supset F \\ \text{iff} & I \models T^J \text{ and } J \not\models \neg F \wedge G \\ \text{iff} & I \models T_2^J. \end{array}$$

Proofs of Propositions 1–5

Proposition 1 *Let F, G be formulas, let D be an action description, and let L, L' be similar causal laws such that L' is obtained from L by replacing an occurrence of F by G. Then the action description*

$$D$$
$$L$$
$$\textbf{caused } F \equiv G$$

represents the same transition system as

$$D$$
$$L'$$
$$\textbf{caused } F \equiv G.$$

Proof: Let the first action description be A and the second A'. Since the transition systems $TS(A)$ and $TS(A')$ are characterized by A_0, A_1 and A'_0, A'_1, it suffices to show that A_m has the same models as A'_m.

First, note that, due to its form, the last law in A and A' above must be an action dynamic law or a static law. The causal theories A_m and A'_m will contain rules

$$i : F \equiv i : G \Leftarrow \top$$

where i ranges over $\{0, \dots, m-1\}$ or $\{0, \dots, m\}$, depending on whether the causal law is an action dynamic law or a static law. Theory A'_m can be obtained from A_m by replacing some formulas of the form $i : F$ by $i : G$. Therefore, by Lemma 1, the theories A_m and A'_m have the same models.

Proposition 2 *Let D be an action description of a signature σ, and let c be a constant that does not belong to σ. If D' is an action description of the signature $\sigma \cup \{c\}$ obtained from D by adding an explicit definition of c in terms of σ, then D is a residue of D'.*

Proof: Action description D is a residue of D' if the mapping

$$s \mapsto s|_\sigma \tag{30}$$

is a 1-1 correspondence between the states of $TS(D')$ and the states of $TS(D)$, and the mapping

$$\langle s_0, e, s_1 \rangle \mapsto \langle s_0|_\sigma, e|_\sigma, s_1|_\sigma \rangle \tag{31}$$

is a 1-1 correspondence between the transitions of $TS(D')$ and the transitions of $TS(D)$.

Since the semantics of an action description D is characterized in terms of the corresponding causal theories D_0 and D_1, to prove that (30) is a 1-1 correspondence between the states of $TS(D')$ and the states of $TS(D)$ we need to check that

$$0{:}s \mapsto 0{:}(s|_\sigma) \tag{32}$$

is a 1-1 correspondence between the models of D'_0 and the models of D_0, and to prove that (31) is a 1-1 correspondence between the transitions of $TS(D')$ and the transitions of $TS(D)$ we need to check that

$$0{:}s_0 \cup 0{:}e \cup 1{:}s_1 \mapsto 0{:}(s_0|_\sigma) \cup 0{:}(e|_\sigma) \cup 1{:}(s_1|_\sigma) \tag{33}$$

is a 1-1 correspondence between the models of D'_1 and the models of D_1.

First consider the case when the explicitly-defined constant c is a simple fluent constant. For every simple fluent constant d from $\sigma \cup \{c\}$, by $R(d)$ we denote the set of rules

$$0 : d = v \Leftarrow 0 : d = v$$

for all $v \in Dom(d)$. Clearly D'_m is obtained from D_m by adding the rules

$$i : c = v \equiv i : F_v \Leftarrow \top \qquad (v \in Dom(c)) \tag{34}$$

where i ranges over $\{0, \dots, m\}$ and the rules

$$0 : c = v \Leftarrow 0 : c = v \qquad (v \in Dom(c)), \tag{35}$$

that is, $R(c)$. Note first that dropping the rules $R(c)$ from D'_m does not change the set of models. Indeed, according to Lemma 1, in the presence of (34) the rules $R(c)$ can be replaced by

$$0 : F_v \Leftarrow 0 : F_v \qquad (v \in Dom(c)). \tag{36}$$

Since all constants occurring in F_v are simple fluent constants from σ, D_m contains the rules $R(d)$ for all such constants d. By Lemma 6, in the presence of these rules (36) can be dropped.

To conclude the proof for the case when c is a simple fluent constant, it remains to observe that rules (34) can be viewed as an explicit definition of $i : c$ in terms of $i : \sigma$. By Lemma 2, (32) is a 1-1 correspondence between the models of D_0 and the models of D'_0, and (33) is a 1-1 correspondence between the models of D_1 and the models of D'_1.

When the constant c is a statically determined fluent constant or an action constant, the proof is similar but simpler, since there are no rules (35), so we don't need to use Lemma 6.

Proposition 3 *Let σ be a signature and c be a constant that does not belong to σ. Let D be an action description of signature $\sigma \cup \{c\}$ which does not contain c in the heads of laws. Let D' be an action description of signature $\sigma \cup \{c\}$ obtained from D by adding an explicit definition (5) of c in terms of σ. Then the action description of signature $\sigma \cup \{c\}$ obtained from D by adding the rules*

$$\textbf{caused } c = v \textbf{ if } F_v \qquad (v \in Dom(c))$$

represents the same transition system as D'.

Proof: First consider the case when the explicitly-defined constant c is a simple fluent constant. Call the second action description D''. The difference between D'_m and D''_m is that the former includes rules

$$i : c = v \equiv i : F_v \Leftarrow \top \qquad (v \in Dom(c)) \tag{37}$$

whereas the latter includes

$$i : c = v \Leftarrow i : F_v \qquad (v \in Dom(c)), \tag{38}$$

where i ranges over $\{0, \dots, m\}$. In addition to these, both contain D_m and the rules

$$0 : c = v \Leftarrow 0 : c = v \qquad (v \in Dom(c)) \tag{39}$$

because c is a simple fluent constant. Since F_v contains only simple fluent constants in σ, causal theories D'_m and D''_m

will contain rules of the same form as (39) for any fluent constants in $0 : F_v$. Therefore, using Lemma 6, we may add the rules

$$0 : F_v \Leftarrow 0 : F_v \qquad (v \in Dom(c)) \tag{40}$$

to D'_m and D''_m without changing their models.

By one application of Lemma 4 and $m - 1$ applications of Lemma 3, causal theories D'_m and D''_m have the same models.

When the constant c is a statically determined fluent constant or an action constant, the proof is similar but simpler, since there are no rules (39), so we don't need to use Lemma 6 or Lemma 4.

Proposition 4 *MB is a residue of MB*.*

The proof uses the concept of strong equivalence of action descriptions. Two action descriptions D and D' of the same signature are *equivalent* if $TS(D) = TS(D')$. They are *strongly equivalent* if for any action description D'' (of a possibly larger signature), action descriptions $D \cup D''$ and $D' \cup D''$ are equivalent. A similar definition appears in Section 5 of (Sergot & Craven 2005).

The lemma below refers to the following explicit definitions of constants in (14).

$$\textbf{caused } Location(p)\!=\!l \equiv Loc(p)\!=\!l \tag{41}$$

$$\textbf{caused } Move(Box)\!=\!true \equiv \bigvee_{l_0 \in L} PushBox(l_0) \tag{42}$$

$$\textbf{caused } Move(Box)\!=\!false \equiv \neg \bigvee_{l_0 \in L} PushBox(l_0) \tag{43}$$

$$\textbf{caused } Move(p)\!=\!true \equiv \bot \qquad (p \neq Box) \tag{44}$$

$$\textbf{caused } Move(p)\!=\!false \equiv \top \qquad (p \neq Box) \tag{45}$$

$$\textbf{caused } Mover(Box)\!=\!Monkey \equiv \bigvee_{l_0 \in L} PushBox(l_0) \tag{46}$$

$$\textbf{caused } Mover(Box)\!=\!None \equiv \neg \bigvee_{l_0 \in L} PushBox(l_0) \tag{47}$$

$$\textbf{caused } Mover(Box)\!=\!Bananas \equiv \bot \tag{48}$$

$$\textbf{caused } Mover(Box)\!=\!Box \equiv \bot \tag{49}$$

$$\textbf{caused } Mover(p)\!=\!None \equiv \top \qquad (p \neq Box) \tag{50}$$

$$\textbf{caused } Mover(p)\!=\!Monkey \equiv \bot \qquad (p \neq Box) \tag{51}$$

$$\textbf{caused } Mover(p)\!=\!Bananas \equiv \bot \qquad (p \neq Box) \tag{52}$$

$$\textbf{caused } Mover(p)\!=\!Box \equiv \bot \qquad (p \neq Box) \tag{53}$$

$$\textbf{caused } Destination(Box)\!=\!l \equiv PushBox(l) \wedge \bigwedge_{\substack{l_0 \in L\} \\ l_0 < l}} \neg PushBox(l_0) \tag{54}$$

$$\textbf{caused } Destination(Box)\!=\!None \equiv \neg \bigvee_{l_0 \in L} PushBox(l_0) \tag{55}$$

$$\textbf{caused } Destination(p)\!=\!l \equiv \bot \qquad (p \neq Box) \tag{56}$$

$$\textbf{caused } Destination(p)\!=\!None \equiv \top \qquad (p \neq Box) \tag{57}$$

(In formulas (54) and (55), L stands for $\{L_1, L_2, L_3\}$ and the relation $<$ on this set is defined by $L_1 < L_2 < L_3$.)

Lemma 8 *MB* is strongly equivalent to the action description which consists of MB and (41)–(57).*

The proof of this lemma is given by a long series of strongly equivalent transformations. We do not include them here.

Proposition 4 may be derived from the lemma by applying Proposition 2 to each of the constants in (14).

Proposition 5 *Let D be an action description and F be a formula such that all constants in F are action constants which are exogenous in D. Then*

D

caused F **if** G

represents the same transition system as

D

caused $\bot$ **after** $\neg F \wedge G$.

Proof: Let the first action description be A and the second A'. Let σ be the set of constants occurring in F. Causal theory A_m will contain

$$i : F \Leftarrow i : G$$

and A'_m will contain

$$\bot \Leftarrow \neg i : F \wedge i : G$$

where i ranges over $\{0, \ldots, m-1\}$. Due to the requirement that constants in F are action constants which are exogenous, both A_m and A'_m will contain rules of the form

$$i : a \Leftarrow i : a$$

for all atoms a of σ, where i ranges over $\{0, \ldots, m-1\}$. By Lemma 7, the theories A_m and A'_m have the same models.

References

Barker, K.; Porter, B.; and Clark, P. 2001. A library of generic concepts for composing knowledge bases. In *First International Conference on Knowledge Capture*, 14–21.

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.

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

Lifschitz, V., and Ren, W. 2006. Towards a modular action description language.[3] In *Working Notes of the AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*.

Lifschitz, V.; Pearce, D.; and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Transactions on Computational Logic* 2:526–541.

[3] `http://www.cs.utexas.edu/users/vl/papers/tmad.ps` .

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

Sergot, M., and Craven, R. 2005. Some logical properties of nonmonotonic causal theories. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 198–210.

Turner, H. 2004. Strong equivalence for causal theories. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, 289–301.