Causal Laws and Multi-Valued Fluents

Enrico Giunchiglia, DIST — Università di Genova
Joohyung Lee and Vladimir Lifschitz, University of Texas at Austin
Hudson Turner, University of Minnesota at Duluth

Abstract

This paper continues the line of work on representing properties of actions in nonmonotonic formalisms that stresses the distinction between being *true* and being *caused*, as in the system of causal logic introduced by McCain and Turner and in the action language $\mathcal{C}$ proposed by Giunchiglia and Lifschitz. The only fluents directly representable in language $\mathcal{C}$ are truth-valued fluents, which is often inconvenient. We show that both causal logic and language $\mathcal{C}$ can be extended to allow values from arbitrary nonempty sets. Our extension of language $\mathcal{C}$, called $\mathcal{C}+$, also makes it possible to describe actions in terms of their attributes, which is important from the perspective of elaboration tolerance. We describe an embedding of $\mathcal{C}+$ in causal theories with multi-valued constants, relate $\mathcal{C}+$ to Pednault's action language ADL, and show how multi-valued constants can be eliminated in favor of Boolean constants.

1 Introduction

This paper continues the line of work on representing properties of actions in nonmonotonic formalisms that stresses the distinction between being *true* and being *caused*. Fangzhen Lin [1995] extended the situation calculus by atomic formulas $Caused(p, v, s)$ (fluent p is caused to have value v in situation s). Causal theories in the sense of McCain and Turner [1997] have postulates of the form $F \Rightarrow G$ (if F is true then G is caused). The semantics of the action language $\mathcal{C}$ [Giunchiglia and Lifschitz, 1998] is based on the concept of a "causally explained" transition. Solutions to the frame problem expressed in these formalisms are applicable to domains in which actions may have indirectly specified effects and qualifications. This work has led to the development of the Causal Calculator[1]—an automated system for planning and reasoning about actions in such domains.

The work described above is limited, however, to propositional (that is, truth-valued) fluents. Lin's predicate *Caused* expects a propositional fluent as the first argument and a truth value as the second. Formulas F, G in the causal logic of McCain and Turner are propositional formulas. Fluent symbols in language $\mathcal{C}$ can represent propositional fluents only. In applications to representing specific domains, this is often inconvenient. If we want to express, for instance, that the current location of box B_1 is L_2, we have to write $Loc(B_1, L_2)$ rather than $Loc(B_1) = L_2$; then the fact that a box, when moved to another location, "disappears" from its previous location can be treated as an indirect effect of the move action, provided that appropriate causal laws are postulated. In the notation of $\mathcal{C}$, these laws are

$$\textbf{caused}\ \neg Loc(b, l)\ \textbf{if}\ Loc(b, l') \quad (l \neq l')\,. \quad (1)$$

Similar causal laws are needed in many other cases.

In this paper we show how the system of causal logic from [McCain and Turner, 1997] and language $\mathcal{C}$ can be extended to allow values from arbitrary nonempty sets, besides the set of Boolean truth values. Our extension of $\mathcal{C}$, denoted by $\mathcal{C}+$, makes it possible to represent many action domains more concisely. Language $\mathcal{C}+$ allows us also to describe actions in terms of their attributes. For instance, in $\mathcal{C}+$ we can express that robot r moves box b to location l by conjoining $Move(b)$ with formulas $Mover(b) = r$ and $Destination(b) = l$, instead of using action names with many arguments, such as $Move(r, b, l)$. As argued in [Lifschitz, 2000], the use of action attributes can help make a representation more elaboration tolerant in the sense of [McCarthy, 1999].

As a preliminary step, in Section 2 we define the syntax and semantics of propositional combinations of equalities; every equality contains a nonlogical constant whose value may come from a certain nonempty domain. Classical propositional logic corresponds precisely to the case when all constants are Boolean (that is, have domain $\{\mathbf{f}, \mathbf{t}\}$). On the basis of that language, Section 3 defines causal theories with multi-valued constants. A simple characterization of the meaning of "definite" causal theories is provided by the concept of multi-valued completion. Section 4 introduces the action language $\mathcal{C}+$, shows an example of its use, and defines

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

a simple embedding of $\mathcal{C}+$ into causal logic with multi-valued constants. Section 5 relates $\mathcal{C}+$ to the language ADL from [Pednault, 1994]. Finally, Section 6 describes methods of eliminating multi-valued constants with finite domains in favor of Boolean constants. The results of that section show that, in the finite case, the extensions of causal logic and language $\mathcal{C}$ proposed in this paper can be reduced to the original versions of these systems.

2 Propositional Logic of Multi-Valued Constants

A *(multi-valued propositional) signature* is a set of symbols called *constants*, along with a nonempty set $Dom(c)$ of symbols assigned to each constant c. We call $Dom(c)$ the *domain* of c. An *atom* of a signature σ 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* of σ is a propositional combination of atoms. An *interpretation* of σ is a function that maps every element of σ to an element of its domain. An interpretation I *satisfies* an atom $c=v$ (symbolically, $I \models c=v$) if $I(c)=v$. The satisfaction relation is extended from atoms to arbitrary formulas according to the usual truth tables for the propositional connectives. A *model* of a set X of formulas is an interpretation that satisfies all formulas in X. If every model of X satisfies a formula F then we say that X *entails* F and write $X \models F$.

A *Boolean* constant is one whose domain is $\{\mathbf{f}, \mathbf{t}\}$. When all constants are Boolean, these definitions correspond to the usual syntax and semantics of propositional formulas, if we agree to use only atoms of the form $c=\mathbf{t}$ and to use c as shorthand for $c=\mathbf{t}$.

We will often identify an interpretation I of a multi-valued propositional signature with the set of atoms true in I.

3 Causal Theories with Multi-Valued Constants

Causal theories are defined here in precisely the same manner as in [McCain and Turner, 1997], except that formulas in the sense of the previous section are allowed in place of propositional formulas.

3.1 Syntax

Begin with a multi-valued propositional signature σ. By a *causal law* we mean an expression of the form

$$F \Rightarrow G \tag{2}$$

where F and G are formulas of σ. By the *antecedent* and *consequent* of (2), we mean the formulas F and G, respectively.

Note that (2) is not the material conditional $F \supset G$. The intended reading of (2) is: *Necessarily, if F then G is caused.*

A *causal theory* is a set of causal laws.

3.2 Semantics

For any causal theory T and interpretation I, let

$$T^I = \{\, G : \text{for some } F,\ F \Rightarrow G \in T \text{ and } I \models F \,\}.$$

So T^I is the set of consequents of causal laws in T whose antecedents are true in I. Intuitively then, T^I entails exactly the formulas that are caused to be true in I according to T.

An interpretation I is *causally explained* according to a causal theory T if I is the unique model of T^I.

It follows that an interpretation I is causally explained according to T if and only if, for every formula F,

$$I \models F \quad \text{iff} \quad T^I \models F\,.$$

That is, I is causally explained according to T if and only if the formulas that are true in I are exactly the formulas caused to be true in I according to T. For further discussion of the so-called "principle of universal causation" that motivates this definition, see [McCain and Turner, 1997; Turner, 1999].

As an example, let $\sigma = \{c\}$ and $Dom(c) = \{1, 2, \ldots\}$, and let the only causal law in T be

$$c=1 \Rightarrow c=1\,. \tag{3}$$

The interpretation I defined by $I(c) = 1$ is causally explained according to T. Indeed, $T^I = \{c=1\}$, so I is the only model of T^I. Furthermore, T has no other causally explained interpretations. Indeed, for any interpretation J such that $J(c) \neq 1$, T^J is empty and so has models different from J.

3.3 Multi-Valued Completion

A causal theory T is *definite* if

- no constant in the signature of T has a singleton domain,
- the consequent of every causal law of T is an atom or $\bot$, and
- no atom is the consequent of infinitely many causal laws of T.

Due to the first two conditions, an interpretation I is causally explained according to a definite causal theory T if and only if $I = T^I$.

Definite causal theories have a concise translation into sets of formulas, as follows. For each atom A in the language of T, the *completion* of A is the formula

$$A \equiv (F_1 \vee \cdots \vee F_n)$$

where $F_1, \ldots, F_n$ ($n \geq 0$) are the antecedents of the causal laws with consequent A. The *multi-valued completion* of T is obtained by taking the completion of each atom in the language of T, along with the formula $\neg F$ for each causal law of the form $F \Rightarrow \bot$ that belongs to T.

Proposition 1 *Let T be a definite causal theory. The causally explained interpretations according to T are precisely the models of the multi-valued completion of T.*

For instance, causal theory (3) is definite, and its multi-valued completion is

$$\begin{array}{ll} c=1 \equiv c=1\,, & \\ c=v \equiv \bot & (v>1)\,. \end{array}$$

The interpretation causally explained according to (3) is the only model of these formulas.

Multi-valued completion is a straightforward generalization of the "literal completion" method introduced in [McCain and Turner, 1997], which resembles the well-known Clark completion method for logic programming [Clark, 1978]. When all constants are Boolean, multi-valued completion corresponds precisely to literal completion.

4 Action Language $\mathcal{C}+$

Action language $\mathcal{C}+$ is a multi-valued extension of the action language $\mathcal{C}$ [Giunchiglia and Lifschitz, 1998], and includes $\mathcal{C}$ as the special case in which all constants are Boolean.

4.1 Syntax

Consider a multi-valued signature σ partitioned into *fluent symbols* σ^{fl} and *action symbols* σ^{act}. A *state formula* is a formula of signature σ^{fl}. An *action* is an interpretation of signature σ^{act}.

As an example, we will show how to use $\mathcal{C}+$ to describe several boxes that can be moved between various locations. Take two collections of symbols, *Boxes* and *Locations*. For each $b \in \mathit{Boxes}$,

- $\mathit{Loc}(b)$ is a fluent symbol with the domain *Locations*,
- $\mathit{Move}(b)$ is a Boolean action symbol, and
- $\mathit{Destination}(b)$ is an action symbol with the domain $\mathit{Locations} \cup \{\mathit{None}\}$,

where *None* is a symbol that does not belong to *Locations*. To formalize the enhancement of this example mentioned in the introduction, we would need also the action symbols $\mathit{Mover}(b)$ with the domain $\mathit{Robots} \cup \{\mathit{None}\}$.

There are two kinds of propositions in $\mathcal{C}+$: *static laws* of the form

$$\textbf{caused } F \textbf{ if } G \tag{4}$$

and *dynamic laws* of the form

$$\textbf{caused } F \textbf{ if } G \textbf{ after } H\,, \tag{5}$$

where F and G are state formulas and H is any formula of σ. In a proposition of either kind, formula F is called its *head*.

For instance,

$$\begin{array}{l} \textbf{caused } \bot \textbf{ if } \top \\ \qquad \textbf{after } \mathit{Move}(b)=\mathbf{t} \equiv \mathit{Destination}(b)=\mathit{None} \end{array} \tag{6}$$

is a dynamic law. It expresses that $\mathit{Destination}(b)$ is an attribute of action $\mathit{Move}(b)$: whenever a box b is moved, it is moved to a certain location, and the other way around.

An *action description* is a set of propositions.

4.2 Special Cases

For any Boolean action symbol α, state formula F and formula H,

$$\alpha \textbf{ causes } F \textbf{ if } H$$

is shorthand for the dynamic law

$$\textbf{caused } F \textbf{ if } \top \textbf{ after } \alpha=\mathbf{t} \wedge H\,.$$

For instance,

$$\mathit{Move}(b) \textbf{ causes } \mathit{Loc}(b)=l \textbf{ if } \mathit{Destination}(b)=l \tag{7}$$

is a dynamic law. Such expressions are used to describe direct effects of actions. More generally, a direct effect of the concurrent execution of $\alpha_1,\dots,\alpha_k$ can be described by an expression of the form

$$\alpha_1,\dots,\alpha_k \textbf{ causes } F \textbf{ if } H\,,$$

which stands for

$$\textbf{caused } F \textbf{ if } \top \textbf{ after } \alpha_1=\mathbf{t} \wedge \dots \wedge \alpha_k=\mathbf{t} \wedge H\,.$$

A dynamic law of the form

$$\alpha_1,\dots,\alpha_k \textbf{ causes } \bot \textbf{ if } H$$

can be further abbreviated as

$$\textbf{nonexecutable } \alpha_1,\dots,\alpha_k \textbf{ if } H\,.$$

For instance,

$$\begin{array}{l} \textbf{nonexecutable } \mathit{Move}(b) \\ \qquad \textbf{if } \mathit{Loc}(b)=l \wedge \mathit{Destination}(b)=l \end{array} \tag{8}$$

is a dynamic law that asserts, intuitively, that it is impossible to move b to l if b is already at l.

For any state formula F,

$$\textbf{inertial } F$$

stands for the dynamic law

$$\textbf{caused } F \textbf{ if } F \textbf{ after } F\,.$$

This proposition expresses that F "tends to remain true," in the sense that if F is true before an action occurs then there will be a cause for F if it remains true after the action occurs. For instance, the propositions

$$\textbf{inertial } \mathit{Loc}(b)=l \tag{9}$$

for all boxes b and locations l express that if a box does not change its location then there is a cause for it to be where it is. As observed by McCain and Turner, such propositions can be used to express the commonsense law of inertia.

For any state formula F, we write

$$\textbf{never } F$$

for the static law

$$\textbf{caused } \bot \textbf{ if } F\,.$$

It represents a "qualification constraint" in the sense of [Lin and Reiter, 1994]. For instance, the expressions

$$\textbf{never } \mathit{Loc}(b)=l \wedge \mathit{Loc}(b')=l \qquad (b \neq b') \tag{10}$$

are static laws; they allow us to conclude that two boxes cannot be simultaneously moved to the same location, and that a box can be moved to a location occupied by another box only if the latter is moved also.

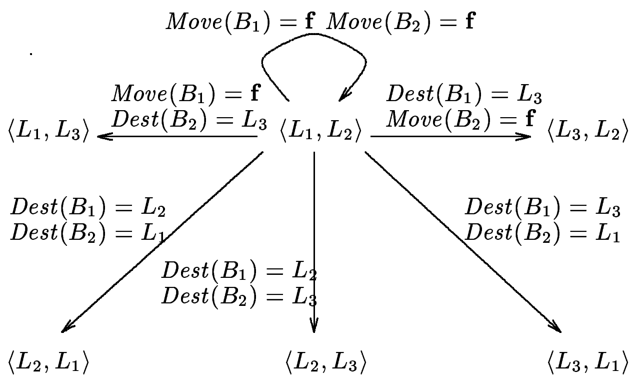
$Move(B_1) = \mathbf{f}$ $Move(B_2) = \mathbf{f}$
$Move(B_1) = \mathbf{f}$
$Dest(B_2) = L_3$
$Dest(B_1) = L_3$
$Move(B_2) = \mathbf{f}$
$\langle L_1, L_3\rangle$
$\langle L_1, L_2\rangle$
$\langle L_3, L_2\rangle$
$Dest(B_1) = L_2$
$Dest(B_2) = L_1$
$Dest(B_1) = L_2$
$Dest(B_2) = L_3$
$Dest(B_1) = L_3$
$Dest(B_2) = L_1$
$\langle L_2, L_1\rangle$
$\langle L_2, L_3\rangle$
$\langle L_3, L_1\rangle$

Figure 1: Part of the transition diagram for (6)–(10).

4.3 Semantics

Consider an action description D. A *state* is an interpretation of σ^{fl} that satisfies $G \supset F$ for every static law (4) in D. A *transition* is a triple $\langle s, a, s'\rangle$ where s, s' are states and a is an action; s is the *initial* state of the transition, and s' is its *resulting* state. A formula F is *caused* in transition $\langle s, a, s'\rangle$ if it is

- the head of a static law (4) from D such that $s' \models G$, or
- the head of a dynamic law (5) from D such that $s' \models G$ and $s \cup a \models H$.

A transition $\langle s, a, s'\rangle$ is *causally explained* according to D if its resulting state s' is the only interpretation of σ^{fl} that satisfies all formulas caused in this transition.

As with other action languages, we associate a transition diagram to D: the *transition diagram* represented by D is the labeled directed graph which has the states of D as nodes, and which includes an edge from s to s' labeled with action a if $\langle s, a, s'\rangle$ is a causally explained transition of D.

Consider the action description (6)–(10) with boxes B_1, B_2 and locations L_1, L_2, L_3. This system has six states, corresponding to the possible ways of placing the two boxes in two of the three locations. Each state is reachable from any of the states by executing one action: that is, for each pair of states s, s' there is an action a such that $\langle s, a, s'\rangle$ is causally explained according to (6)–(10). Figure 1 shows the six states, and the causally explained transitions $\langle s, a, s'\rangle$ in which $s(Loc(B_1)) = L_1$ and $s(Loc(B_2)) = L_2$. Each state s is represented here by the pair of locations $\langle s(Loc(B_1)), s(Loc(B_2))\rangle$. Each action is represented by two of the four atoms true in it.

4.4 Embedding in Causal Theories

A $\mathcal{C}+$ action description can be translated as a causal theory.

Given a signature σ and number $n \in \{0,1\}$, define $\sigma(n) = \{c_n : c \in \sigma\}$, with $Dom(c_n) = Dom(c)$. For any formula F of σ, let $F(n)$ be the formula of $\sigma(n)$ obtained by replacing each occurrence of each atom $c\!=\!v$ in F by $c_n\!=\!v$. For any set T of formulas of σ, $T(n) = \{F(n) : F \in T\}$. In particular, if I is an interpretation of σ then $I(n)$ is an interpretation of $\sigma(n)$.

For any action description D, we obtain causal theory $ct(D)$ over signature $\sigma(0) \cup \sigma^{fl}(1)$ as follows. For each atom A of $\sigma(0)$, include the causal law

$$A \Rightarrow A\,. \tag{11}$$

For each static law (4) in D, include the causal laws

$$G(n) \Rightarrow F(n) \tag{12}$$

for $n = 0, 1$, and for each dynamic law (5) in D, include the causal law

$$H(0) \wedge G(1) \Rightarrow F(1)\,. \tag{13}$$

For instance, if D is action description (6)–(10) then $ct(D)$ consists of the following causal laws:

$$\begin{array}{l}
Loc_0(b)\!=\!l \Rightarrow Loc_0(b)\!=\!l\,,\\
Move_0(b)\!=\!\mathbf{f} \Rightarrow Move_0(b)\!=\!\mathbf{f},\\
Move_0(b)\!=\!\mathbf{t} \Rightarrow Move_0(b)\!=\!\mathbf{t}\,,\\
Destination_0(b)\!=\!v \Rightarrow Destination_0(b)\!=\!v\,,\\
Move_0(b)\!=\!\mathbf{t} \equiv Destination_0(b)\!=\!None \Rightarrow \bot\,,\\
Move_0(b)\!=\!\mathbf{t} \wedge Destination_0(b)\!=\!l \Rightarrow Loc_1(b)\!=\!l\,,\\
Move_0(b)\!=\!\mathbf{t} \wedge Loc_0(b)\!=\!l \wedge Destination_0(b)\!=\!l \Rightarrow \bot\,,\\
Loc_0(b)\!=\!l \wedge Loc_1(b)\!=\!l \Rightarrow Loc_1(b)\!=\!l\,,\\
Loc_n(b)\!=\!l \wedge Loc_n(b')\!=\!l \Rightarrow \bot \qquad (b\!\neq\!b',\ n\!=\!0,1)\,.
\end{array}$$

Proposition 2 *For any action description D, a transition $\langle s, a, s'\rangle$ is causally explained according to D iff the interpretation $s(0) \cup a(0) \cup s'(1)$ is causally explained according to $ct(D)$. Moreover, every interpretation causally explained according to $ct(D)$ can be written in the form $s(0) \cup a(0) \cup s'(1)$ for some transition $\langle s, a, s'\rangle$.*

4.5 Definite Action Descriptions

An action description D is *definite* if

- no constant in the signature of D has a singleton domain,
- the head of each proposition of D is an atom or $\bot$, and
- no atom is the head of infinitely many propositions of D.

Notice that if an action description D is definite, so is the corresponding causal theory $ct(D)$. Hence, by Propositions 1 and 2, definite action descriptions have a concise translation into multi-valued propositional logic.

For example, if D is action description (6)–(10) then its translation into propositional logic with multi-valued constants is equivalent to

$$\begin{array}{l}
Move_0(b)\!=\!\mathbf{f} \equiv Destination_0(b)\!=\!None\,,\\
\neg(Destination_0(b)\!=\!l \wedge Loc_0(b)\!=\!l)\,,\\
\neg(Loc_n(b)\!=\!l \wedge Loc_n(b')\!=\!l) \quad (b\!\neq\!b',\ n\!=\!0,1)\,,\\
Loc_1(b)\!=\!l \equiv\\
\qquad (Destination_0(b)\!=\!l \vee (Loc_0(b)\!=\!l \wedge Move_0(b)\!=\!\mathbf{f}))\,.
\end{array}$$

These formulas entail, for instance, the two assertions mentioned at the end of Section 4.2:

$$\begin{array}{l}
\neg(Destination_0(b)\!=\!l \wedge Destination_0(b')\!=\!l) \qquad (b\!\neq\!b')\,,\\
Destination_0(b)\!=\!l \wedge Loc_0(b')\!=\!l \supset Move_0(b')\!=\!\mathbf{t}\,.
\end{array}$$

5 Relation of $\mathcal{C}+$ to ADL

Language ADL [Pednault, 1994] describes effects of actions on multi-valued fluents in terms of "update conditions." We will show how this idea can be incorporated in the syntactic framework of Section 2 and then describe its relationship to language $\mathcal{C}+$.

As a preliminary step, consider a multi-valued propositional signature σ whose constants have the same finite domain *Dom*. The concept of a formula of a signature σ can be extended as follows. A *term* is a constant of σ, a value (an element of *Dom*) or a variable (from a fixed infinitely countable set). An *extended atom* is an expression of the form $t = v$ where t is a term and v is a value. *Extended formulas* are formed from atoms using propositional connectives and quantifiers, as in first-order logic. We will sometimes identify a closed extended formula F with the formula in the sense of Section 2 that is obtained from F as follows: first, eliminate from F all quantifiers by replacing each subformula of the form $\forall x G(x)$ with $\bigwedge_v G(v)$, where v ranges over *Dom*, and each $\exists x G(x)$ with $\bigvee_v G(v)$; second, replace all occurrences of atoms of the form $v = v$ with $\top$, and all occurrences of atoms of the form $v_1 = v_2$, where v_1 is a value different from v_2, with $\bot$. This convention allows us, for instance, to talk about the satisfaction relation between interpretations and closed extended formulas.

5.1 ADL Action Descriptions

Consider a multi-valued signature σ partitioned into *fluent symbols* σ^{fl} and *action symbols* σ^{act}, such that all fluent symbols have the same finite domain, and all action symbols are Boolean. An *ADL action description* consists of

- a closed extended formula $Precond^{\alpha}$ of signature σ^{fl} for every action symbol α, and
- an extended formula $Update_c^{\alpha}(x)$ of signature σ^{fl}, with no free variables other than x, for every action symbol α and every fluent symbol c.

An ADL action description is *consistent* if, for every action symbol α, every fluent symbol c, and every pair of distinct values v_1, v_2,

$$Precond^{\alpha} \models \neg(Update_c^{\alpha}(v_1) \wedge Update_c^{\alpha}(v_2)).$$

Let D be a consistent ADL action description, let s and s' be interpretations of σ^{fl}, and let α be an action symbol. We say that s' is the *result of executing α in s according to D* if

$$s \models Precond^{\alpha}$$

and, for every fluent symbol c,

$$s'(c) = \begin{cases} v, & \text{if } s \models Update_c^{\alpha}(v), \\ s(c), & \text{if } s \models \neg\exists x\, Update_c^{\alpha}(x). \end{cases}$$

5.2 Reduction to $\mathcal{C}+$

The counterpart of an ADL action description D in language $\mathcal{C}+$ consists of the propositions

$$\begin{array}{l} \textbf{inertial } c = v \\ \textbf{nonexecutable } \alpha \textbf{ if } \neg Precond^{\alpha} \\ \alpha \textbf{ causes } c = v \textbf{ if } Update_c^{\alpha}(v) \end{array} \tag{14}$$

for every fluent symbol c, value v and action symbol α.

In the following theorem, we identify a Boolean action symbol α with the action that maps α to $\mathbf{t}$ and maps every other action symbol to $\mathbf{f}$.

Proposition 3 *For any consistent ADL action description D, s' is the result of executing α in s according to D iff transition $\langle s, \alpha, s' \rangle$ is causally explained according to the counterpart of D in language $\mathcal{C}+$.*

We see that the version of ADL described above is significantly less expressive than $\mathcal{C}+$: ADL is mapped here into the subset of $\mathcal{C}+$ that does not include static laws (and consequently does not address the ramification problem) and has no concurrent actions or noninertial fluents.

6 Replacing Multi-Valued Constants with Boolean Constants

If the domain of a constant c is finite, we can replace c with a family of Boolean constants, one for each element of $Dom(c)$. We first observe that this is easily done in propositional logic with multi-valued constants. This suggests a general method applicable to causal theories. We then introduce a less obvious method that can be used to preserve definiteness. Finally, we specify similar elimination methods for action language $\mathcal{C}+$.

6.1 Eliminating Multi-Valued Constants from Formulas

Let σ be a multi-valued propositional signature, and let $c \in \sigma$. Let σ_c be the signature obtained from σ by replacing constant c with Boolean constants $c(v)$ for all $v \in Dom(c)$. For any formula F of σ, let F_c be the formula obtained by replacing each occurrence of each atom $c = v$ with $c(v) = \mathbf{t}$. For each interpretation I of σ there is a corresponding interpretation I_c of σ_c such that for all atoms A common to both signatures

$$I \models A \quad \text{iff} \quad I_c \models A\,,$$

and for all $v \in Dom(c)$

$$I \models c = v \quad \text{iff} \quad I_c \models c(v) = \mathbf{t}\,.$$

The following lemma is easily proved by structural induction.

Lemma 1 *For any formula F, interpretation I, and constant c of σ, $I \models F$ iff $I_c \models F_c$.*

Let X be a set of formulas of signature σ. We will say that a set X_c of formulas of signature σ_c *correctly reduces* c to a family of Boolean constants (relative to X) if the following holds: an interpretation of σ_c is a model of X_c iff it corresponds to a model of X.

Assuming that $Dom(c)$ is finite, let $elim_c$ be the formula

$$\bigvee_{v} c(v) = \mathbf{t} \;\wedge\; \bigwedge_{v \neq v'} (c(v) = \mathbf{f} \vee c(v') = \mathbf{f})\,. \tag{15}$$

The *elimination* of c from X is the set of formulas of signature σ_c obtained by replacing each occurrence of an atom $c\!=\!v$ in X with $c(v) = \mathbf{t}$, and adding the formula $elim_c$.

Notice that the models of $elim_c$ are precisely the interpretations of σ_c that correspond to an interpretation of σ. This observation and Lemma 1 yield the following result.

Proposition 4 *Let X be a set of formulas, and let c be a constant with a finite domain. The elimination of c from X correctly reduces c to a family of Boolean constants.*

6.2 Eliminating Multi-Valued Constants from Causal Theories

Begin with a causal theory T with signature σ. We will say that a causal theory T_c with signature σ_c *correctly reduces* c to a family of Boolean constants (relative to T) if the following holds: an interpretation of σ_c is causally explained according to T_c iff it corresponds to an interpretation causally explained according to T.

General Elimination Method for Causal Theories

The *general elimination* of c from T is the causal theory with signature σ_c obtained by replacing each occurrence of an atom $c\!=\!v$ in T with $c(v)\!=\!\mathbf{t}$, and adding the causal law

$$\top \Rightarrow elim_c\,. \tag{16}$$

Proposition 5 *Let T be a causal theory with constant c whose domain is finite. The general elimination of c from T correctly reduces c to a family of Boolean constants.*

This simple elimination method can be applied whenever $Dom(c)$ is finite, but it does not preserve definiteness. Since definiteness is quite useful, we next introduce a less general elimination method that preserves it, and can be applied to any definite causal theory in which $Dom(c)$ is finite.

Definite Elimination Method for Causal Theories

The *definite elimination* of c from T is the causal theory with signature σ_c obtained by replacing each occurrence of an atom $c\!=\!v$ in T with $c(v)\!=\!\mathbf{t}$, and adding the causal laws

$$c(v)\!=\!\mathbf{t} \Rightarrow c(v')\!=\!\mathbf{f} \tag{17}$$

for all $v, v' \in Dom(c)$ such that $v \neq v'$, and also adding

$$\bigwedge_v c(v)\!=\!\mathbf{f} \Rightarrow \bot\,. \tag{18}$$

Proposition 6 *Let T be a causal theory with constant c such that (i) $Dom(c)$ is finite, with at least two elements, and (ii) any consequent in which c occurs is an atom. The definite elimination of c from T correctly reduces c to a family of Boolean constants.*

Perhaps we should say a word about the requirement in Proposition 6 that $|Dom(c)|$ exceed one. For any $c \in \sigma$ such that $Dom(c) = \{v\}$, $I(c) = v$ for every interpretation I of σ. Accordingly, if we wish to replace c with a new Boolean constant $c(v)$, we can add the causal law $\top \Rightarrow c(v)\!=\!\mathbf{t}$, which is precisely what is done in the general elimination method. By comparison, the definite elimination method would add only $c(v)\!=\!\mathbf{f} \Rightarrow \bot$. For example, take $\sigma = \{c\}$, $Dom(c) = \{v\}$, and $T = \emptyset$. Then σ has only one interpretation, and it is causally explained according to T. On the other hand, $\sigma_c = \{c(v)\}$, $Dom(c(v)) = \{\mathbf{f}, \mathbf{t}\}$, and the definite elimination of c from T is $\{c(v)\!=\!\mathbf{f} \Rightarrow \bot\}$, which has no causally explained interpretations.

Recall that constants with singleton domains are also not allowed in definite causal theories. We now see that such constants are of little use. Indeed, the preceding observations imply that if a causal theory T includes a constant c with $Dom(c) = \{v\}$, then every occurrence of atom $c\!=\!v$ in T can be replaced with $\top$ without affecting the causally explained interpretations. In fact, constant c can then be dropped from the signature σ, and the resulting causally explained interpretations will be precisely the restrictions to $\sigma \setminus \{c\}$ of the interpretations causally explained according to T.

6.3 Eliminating Multi-Valued Constants from $\mathcal{C}+$

If an action description includes a constant c whose domain is finite, c can be replaced by a family of Boolean constants using methods similar to those introduced for causal theories.

Eliminating Multi-Valued Fluent Constants

Let D be an action description with fluent symbol c. We will say that an action description D_c with action symbols σ^{act} and fluent symbols σ_c^{fl} *correctly reduces* c to a family of Boolean fluent constants (relative to D) if the following hold.

- If $\langle s, a, s' \rangle$ is causally explained according to D, then $\langle s_c, a, s'_c \rangle$ is causally explained according to D_c.
- Any transition causally explained according to D_c can be written in the form $\langle s_c, a, s'_c \rangle$ where $\langle s, a, s' \rangle$ is causally explained according to D.

General Elimination for Fluents in Action Descriptions

The *general elimination* of fluent symbol c from D is the action description with action symbols σ^{act} and fluent symbols σ_c^{fl} obtained by replacing each occurrence of an atom $c\!=\!v$ in D with $c(v)\!=\!\mathbf{t}$, and adding the static law

$$\textbf{caused } elim_c \textbf{ if } \top\,. \tag{19}$$

Proposition 7 *Let D be an action description with fluent symbol c whose domain is finite. The general elimination of c from D correctly reduces c to a family of Boolean fluent constants.*

Definite Elimination for Fluents in Action Descriptions

The *definite elimination* of fluent symbol c from D is the action description with action symbols σ^{act} and fluent symbols σ_c^{fl} obtained by replacing each occurrence of an atom $c\!=\!v$ in D with $c(v)\!=\!\mathbf{t}$, and adding the static laws

$$\textbf{caused } c(v)\!=\!\mathbf{f} \textbf{ if } c(v')\!=\!\mathbf{t} \tag{20}$$

for all $v, v' \in Dom(c)$ such that $v \neq v'$, and also adding the static law

$$\textbf{caused } \bot \textbf{ if } \bigwedge_v c(v)\!=\!\mathbf{f}. \tag{21}$$

Note that the static laws (20) generalize example (1) from the introduction.

Proposition 8 *Let D be an action description with fluent symbol c such that (i) $Dom(c)$ is finite, with at least two elements, and (ii) any head in which c occurs is an atom. The definite elimination of c from D correctly reduces c to a family of Boolean fluent constants.*

Eliminating Multi-Valued Action Constants

Let D be an action description with action symbol c. We'll say that an action description D_c with action symbols σ_c^{act} and fluent symbols σ^{fl} *correctly reduces* c to a family of Boolean action constants (relative to D) if the following hold.

- If $\langle s, a, s' \rangle$ is causally explained according to D, then $\langle s, a_c, s' \rangle$ is causally explained according to D_c.
- Any transition causally explained according to D_c can be written in the form $\langle s, a_c, s' \rangle$ where $\langle s, a, s' \rangle$ is causally explained according to D.

The *elimination* of action symbol c from D is the action description with action symbols σ_c^{act} and fluent symbols σ^{fl} obtained by replacing each occurrence of an atom $c\!=\!v$ in D with $c(v)\!=\!\mathbf{t}$, and adding the dynamic law

$$\textbf{caused } \bot \textbf{ if } \top \textbf{ after } \neg elim_c\,. \tag{22}$$

Proposition 9 *Let D be an action description with action symbol c whose domain is finite. The elimination of c from D correctly reduces c to a family of Boolean action constants.*

Example

Consider the action description (6)–(10). Let's eliminate all non-Boolean constants.

For fluents $Loc(b)$ add propositions (1), as discussed in the introduction, and also add the static law

$$\textbf{caused } \bot \textbf{ if } \bigwedge_l \neg Loc(b,l)\,.$$

For actions $Destination(b)$, add

$$\textbf{caused } \bot \textbf{ if } \top \textbf{ after } \bigwedge_v \neg Destination(b,v)$$

and, for all $v, v' \in Dom(Destination(b))$ such that $v \neq v'$, also add

$$\textbf{nonexecutable } Destination(b,v), Destination(b,v') \textbf{ if } \top\,.$$

(These propositions are obtained by simplifying (22).)

Finally, replacing non-Boolean constants in (6)–(10) yields the following propositions.

$\textbf{caused } \bot \textbf{ if } \top \textbf{ after } Move(b) \equiv Destination(b, None)$
$Move(b) \textbf{ causes } Loc(b,l) \textbf{ if } Destination(b,l)$
$\textbf{nonexecutable } Move(b) \textbf{ if } Loc(b,l) \wedge Destination(b,l)$
$\textbf{inertial } Loc(b,l)$
$\textbf{never } Loc(b,l) \wedge Loc(b',l) \qquad (b \neq b')$

7 Conclusion

The extension of the action language $\mathcal{C}$ proposed in this paper is an improvement of that language in two ways. First, we can now represent multi-valued fluents directly, without replacing them by Boolean fluents. Second, actions can be now described in terms of their attributes. This work provides mathematical background for adding these features to the input language of the Causal Calculator, which will turn it into a better knowledge representation tool.

8 Acknowledgements

We are grateful to Esra Erdem for comments on an earlier draft. The first author was partially supported by ASI and MURST. The second and third authors were partially supported by NSF under grant IIS-9732744. The fourth author was partially supported by NSF CAREER Grant #0091773.

References

[Clark, 1978] Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

[Giunchiglia and Lifschitz, 1998] Enrico Giunchiglia and Vladimir Lifschitz. An action language based on causal explanation: Preliminary report. In *Proc. AAAI-98*, pages 623–630. AAAI Press, 1998.

[Lifschitz, 2000] Vladimir Lifschitz. Missionaries and cannibals in the causal calculator. In *Principles of Knowledge Representation and Reasoning: Proc. Seventh Int'l Conf.*, pages 85–96, 2000.

[Lin and Reiter, 1994] Fangzhen Lin and Raymond Reiter. State constraints revisited. *Journal of Logic and Computation*, 4:655–678, 1994.

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

[McCain and Turner, 1997] Norman McCain and Hudson Turner. Causal theories of action and change. In *Proc. AAAI-97*, pages 460–465, 1997.

[McCarthy, 1999] John McCarthy. Elaboration Tolerance. In progress, 1999. Available at `http://www-formal.stanford.edu/jmc/elaboration.html`.

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

[Turner, 1999] Hudson Turner. A logic of universal causation. *Artificial Intelligence*, 113:87–123, 1999.