Nonmonotonic Causal Theories

Enrico Giunchiglia
DIST — Università di Genova, Italy

Joohyung Lee, Vladimir Lifschitz, Norman McCain
University of Texas, Austin, USA

Hudson Turner
University of Minnesota, Duluth, USA

Abstract

The nonmonotonic causal logic defined in this paper can be used to represent properties of actions, including actions with conditional and indirect effects, nondeterministic actions, and concurrently executed actions. It has been applied to several challenge problems in the theory of commonsense knowledge. We study the relationship between this formalism and other work on nonmonotonic reasoning and knowledge representation, and discuss its implementation, called the Causal Calculator.

1 Introduction

The problem of describing changes caused by the execution of actions is as old as logic-based Artificial Intelligence. In the "advice taker" paper by John McCarthy [1959], effects of actions are described by first-order formulas, such as

$$did(go(desk, car, walking)) \supset at(I, car).$$

Several other approaches to describing effects of actions have been proposed later, including the situation calculus [McCarthy and Hayes, 1969], the STRIPS language [Fikes and Nilsson, 1971] and the event calculus [Kowalski and Sergot, 1986].

A large body of research has been devoted to the frame problem—the problem of describing what does *not* change when actions are performed [Shanahan, 1997]. This work has led to the emergence of several formal theories of nonmonotonic reasoning, including circumscription [McCarthy, 1980], [McCarthy, 1986] and default logic [Reiter, 1980].

Several researchers addressed the frame problem by distinguishing between causal and noncausal propositions. When we say that E is an effect of an action A, we assert not only a material implication ("if A has just been executed then E holds") but also the existence of a causal relationship between A and E ("the execution of A causes E"). In the natural sciences, it is common practice to disregard distinctions like this. In AI, surprisingly, formal theories of causal reasoning that stress this distinction have turned out to be quite useful.

The causal theory of actions presented in this paper takes its origin in the nonmonotonic causal logic introduced by McCain and Turner [1997]. An implementation of this approach to actions, called the Causal Calculator (CCALC)[1], has been applied to several challenge problems in the theory of commonsense knowledge [Lifschitz, 2000], [Lifschitz *et al.*, 2000], [Campbell and Lifschitz, 2003]. The companion paper [Akman *et al.*, 2003] shows how the language of CCALC can be used to describe two commonsense domains due to Erik Sandewall.

In this causal logic, we distinguish between being true and having a cause, but we do not attempt to talk about what a cause may be. Strictly speaking, there is no way to express in our formal notation that the execution of an action A is the cause for E; what we write instead is that *there exists* a cause for E if A has just been executed. Syntactically, this can be expressed by combining material implication $\supset$ with a modal operator C that is used to express the existence of a cause:

$$A \supset \mathsf{C}E \tag{1}$$

[Geffner, 1990], [Turner, 1999]. The syntax defined in this paper is more limited. We use "causal rules" of the form

$$F \Leftarrow G$$

where F and G are formulas of classical logic, in place of the combination

$$G \supset \mathsf{C}F$$

("there is a cause for F to be true if G is true"). In particular, we can express that there is a cause for F to be true by writing the causal rule

$$F \Leftarrow \top.$$

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

The noncausal assertion that F is true can be expressed by

$$\bot \Leftarrow \neg F.$$

(Symbols $\top$ and $\bot$ are 0-place propositional connectives.) Expression (1) can be written as the causal rule

$$E \Leftarrow A.$$

We will define a causal theory to be a set of causal rules.

Our semantics of causal theories can be informally summarized as follows: every fact that is caused obtains, and vice versa. This assertion is made precise in Proposition 1 (Section 2.3). Its second half—every fact that obtains is caused—expresses the "principle of universal causation," a rather strong philosophical commitment that is rewarded by mathematical simplicity in the semantics of causal theories. We will soon see how the principle of universal causation can be "disabled" for selected formulas.

To illustrate the idea of universal causation, we will show how to use it, in an informal way, to find the models of a simple causal theory. Take the following example:

$$\begin{aligned} p &\Leftarrow q, \\ q &\Leftarrow q, \\ \neg q &\Leftarrow \neg q. \end{aligned} \tag{2}$$

Here p, q are atoms. Which interpretations (that is, truth-valued functions on $\{p, q\}$) are models of (2)?

Causal rules (2) assert that, under various conditions, there is a cause for p, there is a cause for q, and there is a cause for $\neg q$. But no rule in (2) may lead to the conclusion that there is a cause for $\neg p$. Consequently,

$$\neg p \text{ is not caused.}$$

Since we require that every true formula be caused, we can conclude that

$$\neg p \text{ is not true}$$

or, equivalently,

$$p \text{ is true.}$$

Again because every true formula is caused, it must be the case that

$$p \text{ is caused.}$$

The only way to use rules (2) to establish that p is caused is to refer to rule $p \Leftarrow q$, which says: p is caused if q is true. Consequently,

$$q \text{ is true.}$$

This informal argument shows that the interpretation making both p and q true is the only possible model of (2). And it is indeed a model, because, under this interpretation, the first two rules of (2) guarantee the existence of causes both for p and for q.

The last two rules of (2) illustrate the point made above about relaxing the principle of universal causation—they "disable" it for formula q. These rules express that if q is true then there is a cause for this, and, on the other hand, if q is false then there is a cause for that.

In the next section, we define the semantics of causal theories and study the special case of "definite" theories. In Section 3 we show how to use definite theories to describe action domains; the Monkey and Bananas problem[2] is used as the main example. "Causal laws"—higher-level notation for describing actions, similar to the action language $\mathcal{C}$ from [Giunchiglia and Lifschitz, 1998]—are introduced in Section 4, and further examples illustrating the expressive possibilities of definite causal laws are discussed in Section 5. Section 6 is a brief introduction to the input language of CCALC. In Section 7 we compare this work with other research on nonmonotonic reasoning and knowledge representation. Proofs of theorems are presented in Section 8. We conclude by discussing the relation of this work to the idea of elaboration tolerance [McCarthy, 1999]. Appendix A relates formulas of "multi-valued propositional signatures" used in the main part of the paper to usual propositional formulas. Appendix B is a list of useful abbreviations for special kinds of causal laws.

We dedicate this paper to the memory of Ray Reiter (1939–2002).

2 Causal Theories

2.1 Formulas

The class of formulas defined below is similar to the class of propositional formulas, but a little bit more general: we will allow atomic parts of a formula to be equalities of the kind found in constraint satisfaction problems.

[2] A monkey wants to get a bunch of bananas hanging from the ceiling. He can reach the bananas by first pushing a box to the empty place under the bananas and then climbing on top of the box.

This is convenient when formulas are used to talk about states of a system. For instance, to describe the location of a person in an apartment, we can use equalities like

$$Loc = Kitchen, \; Loc = LivingRoom, \; Loc = Bathroom, \; Loc = Bedroom.$$

The effect of walking to the kitchen can be described by saying that it makes the first of these atomic formulas true, so that the others become false.

A *(multi-valued propositional) signature* is a set σ of symbols called *constants*, along with a nonempty finite set $Dom(c)$ of symbols, disjoint from σ, 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.

To distinguish formulas of the usual propositional logic from formulas of a multi-valued signature, we will call them "classical."

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.

The following definitions are standard in logic. A *model* of a set X of formulas is an interpretation that satisfies all formulas in X. If X has a model, it is said to be *consistent*, or *satisfiable*. If every model of X satisfies a formula F then we say that X *entails* F and write $X \models F$. Two sets of formulas are *equivalent* to each other if they have the same models.

A *Boolean* constant is one whose domain is the set $\{\mathbf{f}, \mathbf{t}\}$ of truth values. A *Boolean* signature is one whose constants are Boolean. If c is a Boolean constant, we will sometimes use c as shorthand for the atom $c=\mathbf{t}$. When the syntax and semantics defined above are restricted to Boolean signatures and to formulas that do not contain $\mathbf{f}$, they turn into the usual syntax and semantics of classical propositional formulas. Checking satisfiability (and entailment) for formulas of a multi-valued propositional signature can be easily reduced to the satisfiability problem for the classical case, as discussed in Appendix A.

Recall that, according to the definition, an atom is an equality whose left-hand side is a constant c, and whose right-hand side is an element of the domain of c. An expression of the form $c=d$, where both c and d are constants, will be understood as an abbreviation for the disjunction

$$\bigvee_{v \in Dom(c) \cap Dom(d)} (c=v \wedge d=v).$$

The symbol $\neq$ will be used to abbreviate the negation of an equality of either kind.

For example, consider the use of a multi-valued signature to describe possible states in the Monkey and Bananas problem. A state can be described by specifying

- the current locations of the monkey, the bananas, and the box,
- whether or not the monkey is on the box, and
- whether or not the monkey has the bananas.

Assume that the possible locations of the monkey, the bananas, and the box are L_1, L_2, L_3. A signature that would allow us to talk about states consists of the constants

$$Loc(x) \qquad (x \in \{Monkey, Bananas, Box\}) \tag{3}$$

whose domain is $\{L_1, L_2, L_3\}$, and the Boolean constants

$$HasBananas,\ OnBox. \tag{4}$$

This signature has $3^3 \cdot 2^2$ interpretations. The interpretations that satisfy

$$HasBananas \supset Loc(Monkey) = Loc(Bananas)$$

and

$$OnBox \supset Loc(Monkey) = Loc(Box)$$

represent the possible states of the system.

2.2 Causal Rules: Syntax

Begin with a multi-valued propositional signature σ. By a *(causal) rule* we mean an expression of the form $F \Leftarrow G$ ("F is caused if G is true"), where F and G are formulas of σ, called the *head* and the *body* of the rule. Rules with the head $\bot$ are called *constraints*. A *causal theory* is a set of causal rules.

The examples below come from the representation of the Monkey and Bananas domain that will be discussed in Section 3.2. Since that theory deals with a sequence $s_0, \ldots, s_m$ of states rather than an individual state, the supply of constants used in these rules is richer than the signature introduced in Section 2.1. For a fixed nonnegative integer m—the length of "histories" that we want to describe—introduce $m + 1$ copies of the constants in that signature, one corresponding to the each of the states s_i. Each copy is formed by putting a "time stamp" i: in front of each of the constants:

$$i\!:\!Loc(x),\ \ i\!:\!HasBananas,\ \ i\!:\!OnBox \tag{5}$$

($x \in \{Monkey, Bananas, Box\}$; $i \in \{0, \dots, m\}$). We also need Boolean constants representing the execution of actions:

$$i\!:\!Walk(l) \tag{6}$$

(expressing that between states s_i and s_{i+1} the monkey walks to location l),

$$i\!:\!PushBox(l) \tag{7}$$

(the monkey pushes the box to location l), and

$$i\!:\!ClimbOn,\ \ i\!:\!ClimbOff,\ \ i\!:\!GraspBananas, \tag{8}$$

where $l \in \{L_1, L_2, L_3\}$ and $i \in \{0, \dots, m-1\}$.[3] Intuitively, a function mapping constants (6)–(8) to truth values is understood as the execution of all actions that are mapped to $\mathbf{t}$; the actions with the same time stamp i are executed concurrently.

The effects of actions can be described by rules such as

$$\begin{array}{l} i\!+\!1\!:\!Loc(Monkey)\!=\!l \Leftarrow i\!:\!Walk(l), \\ i\!+\!1\!:\!HasBananas \Leftarrow i\!:\!GraspBananas. \end{array} \tag{9}$$

For instance, the last rule says that there is a cause for the monkey to have the bananas in state s_{i+1} if he grasped the bananas between states s_i and s_{i+1}.

Restrictions on the executability of actions can be described by constraints, for instance:

$$\begin{array}{l} \bot \Leftarrow i\!:\!(GraspBananas \wedge \neg OnBox), \\ \bot \Leftarrow i\!:\!(GraspBananas \wedge Loc(Monkey)\!\neq\!Loc(Bananas)). \end{array} \tag{10}$$

In the bodies of the last two rules, we use the following convention:

$$i\!:\!(F \wedge G)$$

stands for

$$(i\!:\!F) \wedge (i\!:\!G)$$

and similarly for the other propositional connectives.

[3] The action of climbing off the box is not required to solve the traditional form of the Monkey and Bananas problem, but is included in the version of the domain discussed in this paper. It is needed, for instance, if the monkey wants to bring the bananas back to his original location.

2.3 Causal Rules: Semantics

Now we will show how to extend the concept of a model, defined in Section 2.1 for sets of formulas, to sets of causal rules.

Let T be a causal theory, and let I be an interpretation of its signature. The *reduct* T^I of T relative to I is the set of the heads of all rules in T whose bodies are satisfied by I. We say that I is a *model* of T if I is the unique model of T^I.

Intuitively, T^I is the set of formulas that are caused, according to the rules of T, under interpretation I. If this set has no models or more than one model, then, according to the definition above, I is not considered a model of T. If T^I has exactly one model, but that model is different from I, then I is not a model of T either. The only case when I is a model of T is when I satisfies every formula in the reduct, and no other interpretation does.

If a causal theory T has a model, we say that it is *consistent*, or *satisfiable*. If every model of T satisfies a formula F then we say that T *entails* F and write $T \models F$.

As an example, take

$$\sigma = \{c\},\ Dom(c) = \{1, \dots, n\}$$

for some positive integer n, and let the only rule of T be

$$c{=}1 \Leftarrow c{=}1. \tag{11}$$

The interpretation I defined by $I(c) = 1$ is a model of T. Indeed,

$$T^I = \{c{=}1\},$$

so that I is the only model of T^I. Furthermore, T has no other models. Indeed, for any interpretation J such that $J(c) \neq 1$, T^J is empty, and I is a model of T^J different from J.

It follows that causal theory (11) entails $c{=}1$.

Consider now what happens if we add the rule

$$c{=}2 \Leftarrow \top \tag{12}$$

to this theory. The reduct of the extended theory relative to any interpretation includes the atom $c{=}2$. Consequently, the interpretation assigning 2 to c is the only possible model of the extended theory. It is easy to see that this is indeed a model.

The extended theory does not entail $c=1$; it entails $c=2$. This example shows that the logic introduced above is nonmonotonic. Intuitively, rule (11) expresses that 1 is "the default value" of c, and rule (12) overrides this default.

If the rule

$$c=2 \Leftarrow c=2 \tag{13}$$

is added to (11) instead of (12), we will get a causal theory with two models. This theory entails $c=1 \vee c=2$.

Let us apply the definition of a model to the causal theory T with rules (2), assuming that the Boolean constants p, q are the only elements of the underlying signature. Consider, one by one, all interpretations of that signature:

- $I_1(p) = I_1(q) = \mathbf{t}$. The reduct consists of the heads of the first two rules of (2): $T^{I_1} = \{p, q\}$. Since I_1 is the unique model of T^{I_1}, it is a model of T.

- $I_2(p) = \mathbf{f}$, $I_2(q) = \mathbf{t}$. The reduct is the same as above, and I_2 is not a model of the reduct. Consequently, I_2 is not a model of T.

- $I_3(p) = \mathbf{t}$, $I_3(q) = \mathbf{f}$. The only element of the reduct is the head of the third rule of (2): $T^{I_3} = \{\neg q\}$. It has 2 models. Consequently, I_3 is not a model of T.

- $I_4(p) = I_4(q) = \mathbf{f}$. The reduct is the same as above, so that I_4 is not a model of T either.

We see that I_1 is the only model of T.

In the discussion of each of the examples above, it was essential that the underlying signature does not include any constants that do not occur in the rules. Adding a constant to the signature of a causal theory without adding any rules containing that constant in the head would render the theory inconsistent (unless the domain of the new constant was a singleton).

The following proposition provides an alternative characterization of the semantics of causal theories:

Proposition 1 *An interpretation I is a model of a causal theory T if and only if for every formula F,*

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

In particular, we see that in a model I of a causal theory, every fact that obtains is caused, in the sense that every formula satisfied by I is entailed by the set of formulas caused under I according to the rules of the theory. This assertion is a precise form of the principle of universal causation discussed in the introduction.

The following fact relates causal rules to corresponding material implications:

Proposition 2 *If a causal theory T contains a causal rule $F \Leftarrow G$ then T entails $G \supset F$.*

The satisfiability problem for multi-valued propositional formulas clearly belongs to class NP. The semantics of causal theories is apparently more complex:

Proposition 3 *The problem of determining that a finite causal theory is consistent is Σ_2^P-complete.*

In Section 2.6 we define a special kind of causal theories for which the satisfiability problem is in class NP.

2.4 Equivalent Transformations of Causal Theories

A negated atom $c \neq v$ is equivalent to the disjunction

$$\bigvee_{w \in Dom(c) \setminus \{v\}} c = w.$$

Consequently, any formula of a multi-valued propositional signature can be rewritten in "positive disjunctive normal form"—as a disjunction of conjunctions of atoms. Similarly, any such formula is equivalent to a conjunction of disjunctions of atoms.

Furthermore, if the head of a causal rule is a conjunction, or if its body is a disjunction, then the rule can be broken into simpler rules:

Proposition 4 *(i) Replacing a rule*

$$F \wedge G \Leftarrow H$$

in a causal theory by the rules

$$F \Leftarrow H, \; G \Leftarrow H$$

does not change the set of models. (ii) Replacing a rule

$$F \Leftarrow G \vee H$$

in a causal theory by the rules

$$F \Leftarrow G,\ F \Leftarrow H$$

does not change the set of models.

Taken together, these facts allow us to replace any rule in a causal theory by several rules whose heads are disjunctions of atoms, and whose bodies are conjunctions of atoms.

2.5 Constraints

As we saw in Section 2.3, adding a rule to a causal theory may affect the theory nonmonotonically—it can get new models. The proposition below shows that the effect of adding constraints (rules with head $\bot$) is monotonic.

We say that an interpretation I *violates* a constraint $\bot \Leftarrow F$ if I satisfies F.

Proposition 5 *Let T_1 and T_2 be causal theories of a signature σ, such that every rule in T_2 is a constraint. An interpretation of σ is a model of $T_1 \cup T_2$ iff it is a model of T_1 and does not violate any of the constraints T_2.*

In other words, the effect of adding a set of constraints to a causal theory is simply to eliminate the models that violate at least one of these constraints.

For instance, the theory whose rules are (11) and (13) has two models: $I_1(c) = 1$ and $I_2(c) = 2$. The constraint

$$\bot \Leftarrow c{=}2 \vee c{=}3 \tag{14}$$

is violated by I_2. Consequently, the theory with rules (11), (13) and (14) has one model, I_1.

Corollary 1 *A causal theory T entails a formula F iff the theory $T \cup \{\bot \Leftarrow F\}$ is inconsistent.*

2.6 Definite Theories

A causal theory T is *definite* if

- the head of every rule of T is an atom or $\bot$, and
- no atom is the head of infinitely many rules of T.

For instance, causal theory (11) is definite. Causal theory (2) is, strictly speaking, not definite, but it can be turned into a definite theory by replacing $\neg q$ in the head of the last rule with the equivalent atom:

$$\begin{array}{l} p \Leftarrow q, \\ q \Leftarrow q, \\ q = \mathbf{f} \Leftarrow \neg q. \end{array} \tag{15}$$

The "completion" process described below reduces the problem of finding a model of a definite causal theory to the problem of finding a model of a set of formulas.

Take a definite causal theory T of a signature σ. We say that an atom $c = v$ of σ is *trivial* if the domain of c is a singleton. For each nontrivial atom A, the *completion formula* for A is the formula

$$A \equiv G_1 \vee \cdots \vee G_n$$

where $G_1, \ldots, G_n$ $(n \geq 0)$ are the bodies of the rules of T with head A. The *completion* of T is obtained by taking the completion formulas for all nontrivial atoms of σ, along with the formula $\neg F$ for each constraint $\bot \Leftarrow F$ in T.

Proposition 6 *The models of a definite causal theory are precisely the models of its completion.*

For instance, the completion of (11) is

$$\begin{array}{ll} c=1 \equiv c=1, & \\ c=v \equiv \bot & \quad (v \in Dom(c) \setminus \{1\}) \end{array} \tag{16}$$

if $|Dom(c)| > 1$. Otherwise the atom $c = 1$ is trivial, and the completion is empty. In both cases, the only model of the completion is defined by $I(c) = 1$. As discussed in Section 2.3, this is the only model of (11).

After adding rule (12), the completion turns into

$$\begin{array}{ll} c=1 \equiv c=1, & \\ c=2 \equiv \top, & \\ c=v \equiv \bot & \quad (v \in Dom(c) \setminus \{1,2\}). \end{array}$$

The only model of these formulas is defined by $I(c) = 2$.

The completion of (15) is

$$\begin{array}{l} p \equiv q, \\ p{=}\mathbf{f} \equiv \bot, \\ q \equiv q, \\ q{=}\mathbf{f} \equiv \neg q. \end{array} \tag{17}$$

The last two equivalences are identically true and can be dropped. The only model of the first two equivalences is the interpretation I_1 (mapping both p and q to $\mathbf{t}$) that was found in Section 2.3 to be the only model of (2).

Here are two more examples of the use of completion. First, we will show how to turn any set X of formulas into a causal theory that has the same models as X. The rules of this theory are

- $A \Leftarrow A$ for every nontrivial atom A, and
- the constraints $\bot \Leftarrow \neg F$ for every $F \in X$.

The completion of this theory consists of the formulas $A \equiv A$ for nontrivial atoms A and the formulas $\neg\neg F$ for all $F \in X$. Clearly, the completion is equivalent to X.

Second, definite theories can be used to express the "closed-world assumption," as follows. Take a Boolean signature σ. The assumption that the elements of σ are false by default can be expressed by the rules

$$\neg c \Leftarrow \neg c \qquad (c \in \sigma) \tag{18}$$

(if c is false then there is a cause for this). If, for some subset S of σ, we combine (18) with the rules

$$c \Leftarrow \top \qquad (c \in S),$$

we will get a causal theory whose only model is the interpretation I that maps the constants in S to $\mathbf{t}$ and all other constants to $\mathbf{f}$. Indeed, the completion of this theory consists of the formulas

$$\begin{array}{ll} c \equiv \top & (c \in S), \\ c \equiv \bot & (c \in \sigma \setminus S), \\ c{=}\mathbf{f} \equiv \neg c & (c \in \sigma), \end{array}$$

and I is the only model of these formulas.

The assertion of Proposition 6 would be incorrect if we did not restrict the completion process to nontrivial atoms. Consider, for instance, the causal theory whose signature consists of one constant c with the domain $\{0\}$, and whose set of rules is empty. If the definition of completion were extended to trivial atoms then the completion of this theory would be $c{=}0 \equiv \bot$, which is inconsistent.

Proposition 6 shows that the satisfiability problem for finite definite causal theories belongs to class NP. It is clearly NP-complete.

3 Actions and Change

3.1 A Very Simple Example

Before presenting a formalization of the Monkey and Bananas domain in causal logic, we consider a much simpler domain: a system whose state is determined by one truth-valued parameter p. The only available action a changes the value of this parameter to $\mathbf{t}$.

Assume first that we are only interested in histories of length 1, that is to say, in pairs of successive states s_0, s_1. In the spirit of the examples from Section 2.2, the system under consideration can be described by a causal theory using the Boolean constants $0\!:\!p$, $1\!:\!p$ and $0\!:\!a$. The effect of executing the action is described by the causal rule

$$1\!:\!p \Leftarrow 0\!:\!a. \tag{19}$$

Rule (19) by itself does not give an adequate description of the system, because it does not tell us

(i) how to determine the value of p in the initial state s_0,

(ii) how to determine whether action a is executed,

(iii) how to determine the value of p in the final state s_1 if a is not executed.

The answer to (i) is that the initial state of the system is arbitrary. We will exempt $0\!:\!p$ from the principle of universal causation by rules similar to the last two rules of (2):

$$\begin{aligned} 0\!:\!p &\Leftarrow 0\!:\!p,\\ 0\!:\!\neg p &\Leftarrow 0\!:\!\neg p. \end{aligned} \tag{20}$$

Whichever causes determine the initial state of the system, they are outside the theory; $0\!:\!p$ is "exogenous." (According to the convention introduced at the end of Section 2.2, $0\!:\!\neg p$ stands for $\neg 0\!:\!p$.)

The answer to (ii) is similar: whichever causes determine whether or not the action is executed, they are outside the theory; $0\!:\!a$ is exogenous as well:

$$\begin{array}{l} 0\!:\!a \Leftarrow 0\!:\!a, \\ 0\!:\!\neg a \Leftarrow 0\!:\!\neg a. \end{array} \tag{21}$$

The answer to (iii) is that, when action a is not executed, the value of p in state s_1 is determined by "commonsense inertia"—it is the same as in state s_0. This idea can be expressed by the rules

$$\begin{array}{l} 1\!:\!p \Leftarrow (0\!:\!p) \wedge (1\!:\!p), \\ 1\!:\!\neg p \Leftarrow (0\!:\!\neg p) \wedge (1\!:\!\neg p). \end{array} \tag{22}$$

The first rule says that if the value of p is $\mathbf{t}$ both in state s_0 and in state s_1 then there is a cause for it to be $\mathbf{t}$ in state s_1. Intuitively, inertia is the cause. (To put it differently, this rule says that p is true by default in state s_1 if it is true in state s_0.) The second rule expresses a similar condition on the value $\mathbf{f}$. Rules (22) illustrate the solution to the frame problem adopted throughout this paper.

To find models of causal theory (19)–(22), we compute the completion of this theory and translate it into classical propositional logic, as this is done for theory (2) in Section 2.6 and Appendix A. The result is

$$\begin{array}{l} 0\!:\!p \equiv 0\!:\!p, \\ \neg\, 0\!:\!p \equiv \neg\, 0\!:\!p, \\ 1\!:\!p \equiv 0\!:\!a \vee (0\!:\!p \wedge 1\!:\!p), \\ \neg\, 1\!:\!p \equiv \neg\, 0\!:\!p \wedge \neg\, 1\!:\!p, \\ 0\!:\!a \equiv 0\!:\!a, \\ \neg\, 0\!:\!a \equiv \neg\, 0\!:\!a. \end{array}$$

The first two lines and the last two lines are tautologies. The third line can be rewritten as the conjunction of two implications:

$$\begin{array}{l} 1\!:\!p \supset 0\!:\!a \vee 0\!:\!p, \\ 0\!:\!a \supset 1\!:\!p. \end{array}$$

The fourth line is equivalent to

$$0\!:\!p \supset 1\!:\!p.$$

The conjunction of these three implications can be rewritten in the form of an explicit definition of $1\!:\!p$ in terms of $0\!:\!a$ and $0\!:\!p$:

$$1\!:\!p \equiv 0\!:\!a \vee 0\!:\!p.$$

(The end value of p is $\mathbf{t}$ iff action a is executed or the initial value of p is $\mathbf{t}$.) This calculation shows that causal theory (19)–(22) has 4 models, corresponding to the possible combinations of the values of $0\!:\!p$ and $0\!:\!a$.

To get a causal theory whose models correspond to the histories of the same simple domain whose length is m ($m \geq 0$), we introduce Boolean constants $i : p$ for $i = 0, ..., m$ and $i : a$ for $i = 0, ..., m-1$. The value of $i\!:\!p$ characterizes state s_i—it gives the value of the parameter p in that state. The value of $i\!:\!a$ characterizes the event occurring between states s_i and s_{i+1}—it tells us whether that event included the execution of action a. Rules (19)–(22) are generalized as follows:

$$\begin{array}{l} i{+}1\!:\!p \Leftarrow i\!:\!a, \\ 0\!:\!p \Leftarrow 0\!:\!p, \\ 0\!:\!\neg p \Leftarrow 0\!:\!\neg p, \\ i\!:\!a \Leftarrow i\!:\!a, \\ i\!:\!\neg a \Leftarrow i\!:\!\neg a, \\ i{+}1\!:\!p \Leftarrow (i\!:\!p) \wedge (i{+}1\!:\!p), \\ i{+}1\!:\!\neg p \Leftarrow (i\!:\!\neg p) \wedge (i{+}1\!:\!\neg p) \end{array} \tag{23}$$

($i = 0, \ldots, m{-}1$). Call this causal theory SD_m (for "simple domain, histories of length m"). For instance, SD_0 consists of just two rules

$$\begin{array}{l} 0\!:\!p \Leftarrow 0\!:\!p, \\ 0\!:\!\neg p \Leftarrow 0\!:\!\neg p. \end{array} \tag{24}$$

Theory SD_0 has two models, corresponding to the possible states of the system. Theory SD_1 is identical to (19)–(22). For any m, the completion of SD_m can be rewritten as m equivalences

$$i{+}1 : p \equiv (i\!:\!a) \vee (i\!:\!p) \qquad (0 \leq i < m).$$

It follows that SD_m has 2^{m+1} models, each characterized by the truth values assigned to the constants $0\!:\!p$ and $i\!:\!a$ ($i = 0, \ldots, m-1$).

3.2 Monkey and Bananas

We will now describe a causal theory MB_m ($m \geq 0$) whose models represent all possible histories of length m in the Monkey and Bananas domain. The signature of this theory is listed in (5)–(8). Recall that it consists of symbols (3) and (4) prefixed by i: and symbols of the forms

$$\mathit{Walk}(l),\ \mathit{PushBox}(l),\ \mathit{ClimbOn},\ \mathit{ClimbOff},\ \mathit{GraspBananas} \tag{25}$$

prefixed by i: with $i < m$. Here and below, l ranges over $\{L_1, L_2, L_3\}$, and i ranges over $\{0, \ldots, m\}$.

As discussed in Section 2.1, the possible states of the Monkey and Bananas system are described by the interpretations of constants (3), (4) that satisfy two conditions: if the monkey has the bananas then the monkey and the bananas are at the same location; if the monkey is on the box then the monkey and the box are at the same location. In the causal theory MB_m we postulate the following causal counterparts of these conditions. If the monkey has the bananas then there is a cause for the bananas to be at the same place where the monkey is:

$$i : Loc(Bananas) = l \Leftarrow i : (HasBananas \wedge Loc(Monkey) = l). \tag{26}$$

If the monkey is on the box then there is a cause for the monkey to be at the same place where the box is:

$$i : Loc(Monkey) = l \Leftarrow i : (OnBox \wedge Loc(Box) = l). \tag{27}$$

We will see soon why including these rules is convenient.

The next group of rules describes the effects and preconditions of walking:

$$\begin{array}{l} i+1 : Loc(Monkey) = l \Leftarrow i : Walk(l), \\ \bot \Leftarrow i : (Walk(l) \wedge Loc(Monkey) = l), \\ \bot \Leftarrow i : (Walk(l) \wedge OnBox) \end{array} \tag{28}$$

($i < m$). We have seen the first of these rules in Section 2.2. The other two rules are similar to constraints (10).

The effect of walking on the location of the monkey is not the only possible effect of this action: if the monkey has the bananas then walking will affect the location of the bananas as well. This second effect of walking can be described by the rules

$$i+1 : Loc(Bananas) = l \Leftarrow i : (HasBananas \wedge Walk(l)).$$

But we will not include these rules, because they would be redundant: in the presence of (26) the change in the location of the bananas is an indirect effect, or "ramification," of walking (and of any other action that affects the location of the monkey). The possibility of this simplification is what makes (26) an attractive postulate. Similarly, in the presence of (27), a change in the location of the monkey is an indirect effect of any action that affects the location of the box, if it is executed when the monkey is on the box. (Scenarios like this are possible in the enhanced version of the Monkey and Bananas domain described in Section 3.4.)

Pushing the box has two effects and three preconditions:

$$\begin{array}{l}
i{+}1{:}\,Loc(Box){=}l \Leftarrow i{:}\,PushBox(l),\\
i{+}1{:}\,Loc(Monkey){=}l \Leftarrow i{:}\,PushBox(l),\\
\bot \Leftarrow i{:}(PushBox(l) \wedge Loc(Monkey) = l),\\
\bot \Leftarrow i{:}(PushBox(l) \wedge OnBox),\\
\bot \Leftarrow i{:}(PushBox(l) \wedge Loc(Monkey) {\neq} Loc(Box))
\end{array} \tag{29}$$

$(i < m)$. The descriptions of the other actions have a similar structure:

$$\begin{array}{l}
i{+}1{:}\,OnBox \Leftarrow i{:}\,ClimbOn,\\
\bot \Leftarrow i{:}(ClimbOn \wedge OnBox),\\
\bot \Leftarrow i{:}(ClimbOn \wedge Loc(Monkey) {\neq} Loc(Box)),\\
\\
i{+}1{:}\neg OnBox \Leftarrow i{:}\,ClimbOff,\\
\bot \Leftarrow i{:}(ClimbOff \wedge \neg OnBox),\\
\\
i{+}1{:}\,HasBananas \Leftarrow i{:}\,GraspBananas,\\
\bot \Leftarrow i{:}(GraspBananas \wedge HasBananas),\\
\bot \Leftarrow i{:}(GraspBananas \wedge \neg OnBox),\\
\bot \Leftarrow i{:}(GraspBananas \wedge Loc(Monkey) {\neq} Loc(Bananas))
\end{array} \tag{30}$$

$(i < m)$. Some of these rules are familiar to us from Section 2.2.

The next group of rules expresses that some combinations of actions cannot be executed concurrently:

$$\begin{array}{l}
\bot \Leftarrow i{:}(Walk(l) \wedge PushBox(l)),\\
\bot \Leftarrow i{:}(Walk(l) \wedge ClimbOn),\\
\bot \Leftarrow i{:}(PushBox(l) \wedge ClimbOn),\\
\bot \Leftarrow i{:}(ClimbOff \wedge GraspBananas)
\end{array} \tag{31}$$

$(i < m)$.[4]

To complete the list of postulates, we need to add

[4]Equivalently, we could prohibit the concurrent execution of *all* possible pairs of actions. For the pairs of actions not included in (31), the fact that their concurrent execution is impossible follows from the other postulates. For instance, actions $Walk(l)$ and $Walk(l_1)$ for $l \neq l_1$ cannot be executed at the same time because their effects are incompatible. Actions $Walk(l)$ and $ClimbOff$ cannot be executed at the same time because their preconditions are incompatible.

- rules expressing that the initial state is exogenous, which are similar to (20),
- rules expressing that the execution of actions is exogenous, which are similar to (21), and
- inertia rules, which are similar to (22).

To say that the initial state is exogenous, we postulate

$$0\!:\!c\!=\!v \Leftarrow 0\!:\!c\!=\!v, \tag{32}$$

where c is any of the symbols (3), (4), and $v \in Dom(0\!:\!c)$. To say that the execution of actions is exogenous, we postulate

$$i\!:\!c\!=\!v \Leftarrow i\!:\!c\!=\!v, \tag{33}$$

where c is any of the symbols (25), $i < m$, and $v \in \{\mathbf{f}, \mathbf{t}\}$. The inertia rules are

$$i\!+\!1\!:\!c\!=\!v \Leftarrow (i\!:\!c\!=\!v) \wedge (i\!+\!1\!:\!c\!=\!v), \tag{34}$$

where c is any of the symbols (3), (4); $i < m$; $v \in Dom(i\!:\!c)$.

We define MB_m as the causal theory whose rules are (26)–(34). The models of this theory correspond to the possible histories of the Monkey and Bananas domain of length m.

3.3 Reasoning and Planning

Since causal theories MB_m are definite, the process of completion described in Section 2.6 can be used to reduce some computational problems related to the Monkey and Bananas domain to the satisfiability problem for propositional logic. This idea is at the heart of the operation of CCalc, which turns a given query into a set of propositional formulas and then invokes a satisfiability solver to find an answer.

Consider a few examples.

Prediction. *Initially, the monkey is at L_1, the bananas are at L_2, and the box is at L_3. The monkey walks to L_3 and then pushes the box to L_2. Does it follow that in the resulting state the monkey, the bananas and the box are at the same location?*

This question can be formalized as follows: Determine whether MB_2 entails the formula

$$\begin{aligned}&[(0\!:\!Loc(Monkey)\!=\!L_1) \wedge (0\!:\!Loc(Bananas)\!=\!L_2) \wedge (0\!:\!Loc(Box)\!=\!L_3)\\ &\quad \wedge (0\!:\, Walk(L_3)) \wedge (1\!:\!PushBox(L_2))]\\ &\qquad\quad \supset 2\!:\!(Loc(Monkey)\!=\!Loc(Bananas) \wedge Loc(Bananas)\!=\!Loc(Box)).\end{aligned} \tag{35}$$

This is equivalent to asking whether (35) is entailed by the completion of MB_2, that is to say, whether the set of formulas obtained by adding the negation of (35) to the completion of MB_2 is unsatisfiable. The answer to this question is yes.

Postdiction. *The monkey walked to location L_3 and then pushed the box. Does it follow that the box was initially at L_3?*

This question can be formalized as follows: Determine whether MB_2 entails the formula

$$\left[(0\!:\, Walk(L_3)) \wedge \left(1\!:\!\bigvee_l PushBox(l)\right)\right] \supset 0\!:\!Loc(Box)\!=\!L_3. \tag{36}$$

It can be reduced to the satisfiability problem in the same way as the prediction problem above. The answer to this question is yes.

Planning. *For the initial state described in the prediction problem above, find the shortest sequence of actions that would allow the monkey to have the bananas.*

This problem can be formalized as follows: Find a model of MB_m that satisfies the initial conditions

$$0\!:\!Loc(Monkey)\!=\!L_1,\ 0\!:\!Loc(Bananas)\!=\!L_2,\ 0\!:\!Loc(Box)\!=\!L_3 \tag{37}$$

and the goal

$$m\!:\!HasBananas \tag{38}$$

where m is the smallest number for which such a model exists. To solve this problem, we take consecutively $m = 0, 1, \ldots$ and look for an interpretation satisfying both the completion of MB_m and formulas (37), (38). Such an interpretation will be first found for $m = 4$. It assigns the value $\mathbf{t}$ to

$$0\!:\, Walk(L_3),\ 1\!:\!PushBox(L_2),\ 2\!:\, ClimbOn,\ 3\!:\, GraspBananas.$$

These constants represent the shortest solution to the Monkey and Bananas Problem.

3.4 Concurrent Actions and Multiple Agents

Causal theories can be used to describe the concurrent execution of actions by several agents. Consider, for instance, the enhancement of the Monkey and Bananas domain that involves several monkeys. Each monkey can walk, push the box, climb on and off, and grasp the bananas, except that two monkeys cannot both have the bananas simultaneously, or be on top of the box simultaneously.

In the modification of MB_m described below, α and β range over a fixed finite nonempty set M of symbols—the names of the monkeys. The signature consists of the symbols

$$i\!:\!Loc(x),\;\; i\!:\!HasBananas(\alpha),\;\; i\!:\!OnBox(\alpha) \tag{39}$$

where $x \in M \cup \{Bananas, Box\}$, and the symbols

$$\begin{array}{c} i\!:\!Walk(\alpha,l),\;\; i\!:\!PushBox(\alpha,l),\\ i\!:\!ClimbOn(\alpha),\;\; i\!:\!ClimbOff(\alpha),\;\; i\!:\!GraspBananas(\alpha) \end{array} \tag{40}$$

for $i < m$. As before, the domain of $i : Loc(x)$ is $\{L_1, L_2, L_3\}$; the other constants are Boolean.

The first group of rules consists of generalizations of (26) and (27)

$$\begin{array}{l} i\!:\!Loc(Bananas)\!=\!l \Leftarrow i\!:\!(HasBananas(\alpha) \wedge Loc(\alpha)\!=\!l),\\ i\!:\!Loc(\alpha)\!=\!l \Leftarrow i\!:\!(OnBox(\alpha) \wedge Loc(Box)\!=\!l) \end{array}$$

and the new constraints

$$\bot \Leftarrow i\!:\!(HasBananas(\alpha) \wedge HasBananas(\beta)), \tag{41}$$

$$\bot \Leftarrow i\!:\!(OnBox(\alpha) \wedge OnBox(\beta)) \tag{42}$$

$(\alpha \neq \beta)$. Next, we generalize rules (28)–(31):

$$\begin{array}{l} i\!+\!1\!:\!Loc(\alpha)\!=\!l \Leftarrow i\!:\!Walk(\alpha,l),\\ \bot \Leftarrow i\!:\!(Walk(\alpha,l) \wedge Loc(\alpha)\!=\!l),\\ \bot \Leftarrow i\!:\!(Walk(\alpha,l) \wedge OnBox(\alpha)), \end{array}$$

$$\begin{array}{l} i\!+\!1\!:\!Loc(Box)\!=\!l \Leftarrow i\!:\!PushBox(\alpha,l),\\ i\!+\!1\!:\!Loc(\alpha)\!=\!l \Leftarrow i\!:\!PushBox(\alpha,l),\\ \bot \Leftarrow i\!:\!(PushBox(\alpha,l) \wedge Loc(\alpha)\!=\!l),\\ \bot \Leftarrow i\!:\!(PushBox(\alpha,l) \wedge OnBox(\alpha)),\\ \bot \Leftarrow i\!:\!(PushBox(\alpha,l) \wedge Loc(\alpha)\!\neq\! Loc(Box)), \end{array}$$

$$
\begin{array}{l}
i{+}1\!:\!\mathit{OnBox}(\alpha) \Leftarrow i\!:\!\mathit{ClimbOn}(\alpha),\\
\bot \Leftarrow i\!:\!(\mathit{ClimbOn}(\alpha) \wedge \mathit{OnBox}(\alpha)),\\
\bot \Leftarrow i\!:\!(\mathit{ClimbOn}(\alpha) \wedge \mathit{Loc}(\alpha) \neq \mathit{Loc}(\mathit{Box})),\\
\\
i{+}1\!:\!\neg \mathit{OnBox}(\alpha) \Leftarrow i\!:\!\mathit{ClimbOff}(\alpha),\\
\bot \Leftarrow i\!:\!(\mathit{ClimbOff}(\alpha) \wedge \neg \mathit{OnBox}(\alpha)),\\
\\
i{+}1\!:\!\mathit{HasBananas}(\alpha) \Leftarrow i\!:\!\mathit{GraspBananas}(\alpha),\\
\bot \Leftarrow i\!:\!(\mathit{GraspBananas}(\alpha) \wedge \mathit{HasBananas}(\beta)),\\
\bot \Leftarrow i\!:\!(\mathit{GraspBananas}(\alpha) \wedge \neg \mathit{OnBox}(\alpha)),\\
\bot \Leftarrow i\!:\!(\mathit{GraspBananas}(\alpha) \wedge \mathit{Loc}(\alpha) \neq \mathit{Loc}(\mathit{Bananas})),\\
\\
\bot \Leftarrow i\!:\!(\mathit{Walk}(\alpha,l) \wedge \mathit{PushBox}(\alpha,l)),\\
\bot \Leftarrow i\!:\!(\mathit{Walk}(\alpha,l) \wedge \mathit{ClimbOn}(\alpha)),\\
\bot \Leftarrow i\!:\!(\mathit{PushBox}(\alpha,l) \wedge \mathit{ClimbOn}(\alpha)),\\
\bot \Leftarrow i\!:\!(\mathit{ClimbOff}(\alpha) \wedge \mathit{GraspBananas}(\alpha))
\end{array}
$$

$(i < m)$. Finally, we include rules (32) and (34) for c of any of the forms

$$\mathit{Loc}(x),\ \mathit{HasBananas}(\alpha),\ \mathit{OnBox}(\alpha)$$

and rule (33) for c of any of the forms

$$\mathit{Walk}(\alpha,l),\ \mathit{PushBox}(\alpha,l),\ \mathit{ClimbOn}(\alpha),\ \mathit{ClimbOff}(\alpha),\ \mathit{GraspBananas}(\alpha).$$

This causal theory MB_m^M entails, for instance, that two monkeys cannot climb the box simultaneously:

$$\neg\, i\!:\!(\mathit{ClimbOn}(\alpha) \wedge \mathit{ClimbOn}(\beta))$$

$(i < m, \alpha \neq \beta)$. To prove this fact, note that, by Proposition 2, MB_m^M entails each of the formulas

$$
\begin{array}{l}
(i\!:\!\mathit{ClimbOn}(\alpha)) \supset (i{+}1\!:\!\mathit{OnBox}(\alpha)),\\
(i\!:\!\mathit{ClimbOn}(\beta)) \supset (i{+}1\!:\!\mathit{OnBox}(\beta)),\\
\neg((i{+}1\!:\!\mathit{OnBox}(\alpha)) \wedge (i{+}1\!:\!\mathit{OnBox}(\beta))).
\end{array}
$$

If M is a singleton then MB_m^M is essentially identical to MB_m.

3.5 Making Rules Defeasible

Using auxiliary constants, we can make causal rules "defeasible." Consider, for instance, the constraint (42) in MB_m^M that prohibits more than one monkey on top of the box at the same time. We can make this constraint defeasible by rewriting it as

$$\bot \Leftarrow i\!:\!(\mathit{OnBox}(\alpha) \wedge \mathit{OnBox}(\beta) \wedge \neg Ab_1(\alpha,\beta)), \tag{43}$$

where $i : Ab_1(\alpha,\beta)$ are "abnormality constants"—new Boolean constants assumed to be false by default:

$$i\!:\!\neg Ab_1(\alpha,\beta) \Leftarrow i\!:\!\neg Ab_1(\alpha,\beta). \tag{44}$$

(Rule (44) is similar to the formalization (18) of the closed-world assumption.) The causal theory obtained from MB_m^M by replacing (42) with (43) and (44) entails the conjunctive terms $\neg Ab_1(\alpha,\beta)$ that were added to the body of (42). In this sense, including these terms did not make (42) weaker. But the limitations expressed by the modified rule can be retracted by adopting additional postulates. Imagine, for instance, that David is a baby monkey, so small that there is enough room for him on top of the box even when another monkey is there. We can express this additional information by adding the rules

$$\begin{array}{l} i\!:\!Ab_1(\mathit{David},\alpha) \Leftarrow \top, \\ i\!:\!Ab_1(\alpha,\mathit{David}) \Leftarrow \top. \end{array} \tag{45}$$

The extended theory entails $i\!:\!\neg Ab_1(\alpha,\beta)$ only when $\alpha,\beta \neq \mathit{David}$.

As another example, consider the rules describing the effect of pushing the box:

$$\begin{array}{l} i\!+\!1\!:\!\mathit{Loc}(\mathit{Box})\!=\!l \Leftarrow i\!:\!\mathit{PushBox}(\alpha,l), \\ i\!+\!1\!:\!\mathit{Loc}(\alpha)\!=\!l \Leftarrow i\!:\!\mathit{PushBox}(\alpha,l). \end{array} \tag{46}$$

We can make them defeasible by rewriting them as

$$\begin{array}{l} i\!+\!1\!:\!\mathit{Loc}(\mathit{Box})\!=\!l \Leftarrow i\!:\!(\mathit{PushBox}(\alpha,l) \wedge \neg Ab_2(\alpha)), \\ i\!+\!1\!:\!\mathit{Loc}(\alpha)\!=\!l \Leftarrow i\!:\!(\mathit{PushBox}(\alpha,l) \wedge \neg Ab_2(\alpha)), \end{array} \tag{47}$$

where $i\!:\!Ab_2(\alpha)$ for $i < m$ are new Boolean constants, and adding the rules

$$i\!:\!\neg Ab_2(\alpha) \Leftarrow i\!:\!\neg Ab_2(\alpha). \tag{48}$$

The modification (47) of rules (46) will be convenient if we decide later to enhance the description of the domain by allowing the action *PushBox* to be unsuccessful. Imagine that one of the monkeys, Goliath, is so heavy that the other monkeys cannot move the box while Goliath sits on it. We can retract this special case of (47) by postulating

$$i\!:\!Ab_2(\alpha) \Leftarrow i\!:\!\mathit{OnBox}(\mathit{Goliath}). \tag{49}$$

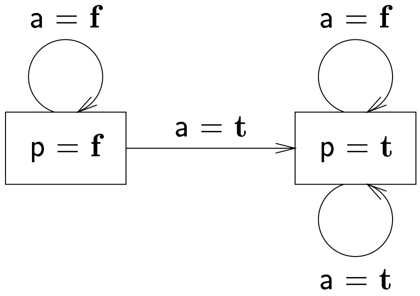

Figure 1: The transition system corresponding to the causal theories SD_m.

Adding this rule does not make the action $PushBox(\alpha, l)$ nonexecutable when Goliath is on the box; it only retracts the assumption about the effect of pushing the box on the locations of the box and the monkey in this special case. The box with Goliath on top will stay, by inertia, where it was, and so will the monkey. This example shows how assumptions about effects of actions can be qualified by adding more postulates.

What if Goliath is trying to climb the box while α is trying to push it? To express that α does not succeed in this case either, we postulate

$$i\!:\!Ab_2(\alpha) \Leftarrow i\!:\!ClimbOn(Goliath). \tag{50}$$

In the next state, Goliath will be on top of the box, but the location of α and the location of the box will not change.

4 Language $\mathcal{C}+$

4.1 Action Description Languages

Let us return to the causal theories SD_m describing a simple action domain in Section 3.1. Models of these theories can be visualized as paths in a "transition system"—the graph shown in Figure 1. The two vertices of the graph represent states; in one state, the value of the parameter p is $\mathbf{f}$, in the other it is $\mathbf{t}$. The edges represent transitions between states; the action a is executed in two transitions, and is not executed in the other two.

There is a simple 1–1 correspondence between the models of SD_m and the paths of length m in this transition system. For instance, the model I of SD_2 defined by

$$I(0\!:\!p) = \mathbf{f},\; I(0\!:\!a) = \mathbf{f},\; I(1\!:\!p) = \mathbf{f},\; I(1\!:\!a) = \mathbf{t},\; I(2\!:\!p) = \mathbf{t} \tag{51}$$

corresponds to the path

$$\langle p = \mathbf{f},\ a = \mathbf{f},\ p = \mathbf{f},\ a = \mathbf{t},\ p = \mathbf{t}\rangle$$

(start at the vertex $p = \mathbf{f}$, follow the loop labeled $a = \mathbf{f}$, and then follow the edge labeled $a = \mathbf{t}$ to the vertex $p = \mathbf{t}$). The models of SD_0 correspond to the vertices of the transition system, and the models of SD_1 correspond to its edges.

The sequence of causal theories MB_m (Section 3.2) can be represented by a graph also. The models of MB_0 correspond to the vertices of that graph, and the models of MB_1 correspond to its edges. More generally, the models of MB_m correspond to the paths of length m. The same is true for the generalization MB_m^M of this sequence, where M is a fixed set of symbols (Section 3.4), and for the modifications of MB_m^M introduced in Section 3.5.

Action description languages are formal languages for describing transition systems. We will introduce here an action description language, called $\mathcal{C}+$, and define how to turn any action description D in this language into an infinite sequence $D_0, D_1, \ldots$ of causal theories. We will see there is an action description in $\mathcal{C}+$, called SD, that corresponds to the sequence SD_m defined in Section 3.1, and that there is also an action description MB that corresponds to the sequence MB_m from Section 3.2. The generalizations and modifications of this sequence introduced in Sections 3.4 and 3.5 can be represented in $\mathcal{C}+$ as well. We will show that for any action description D, the theories D_0 and D_1 characterize the vertices and edges of a transition system such that there is a 1–1 correspondence between the paths of length m in this system and the models of D_m.

The "higher-level" notation of $\mathcal{C}+$ is more concise than the notation of causal theories. This fact, along with the availability of a transition system semantics, makes $\mathcal{C}+$ an attractive formalism for describing actions.

4.2 From Action Descriptions to Causal Theories

Begin with a multi-valued signature (see Section 2.1) partitioned into *fluent* constants and *action* constants. The fluent constants are assumed to be further partitioned into *simple* and *statically determined*.

Intuitively, fluent constants denote "fluents," or parameters characterizing a state. Action constants denote parameters characterizing an event leading from one state to another. For instance, in the action description SD introduced below, p is a simple fluent constant, and a is an action constant. Note that action constants are not required to be Boolean; one possible use of non-Boolean action constants is discussed in Section 5.6. The need to distinguish between simple and statically determined fluent constants is discussed in Section 4.4.

A *fluent formula* is a formula such that all constants occurring in it are fluent constants. 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

$$\textbf{caused } F \textbf{ if } G \tag{52}$$

where F and G are fluent formulas. An *action dynamic law* is an expression of the form (52) in which F is an action formula and G is a formula. A *fluent dynamic law* is an expression of the form

$$\textbf{caused } F \textbf{ if } G \textbf{ after } H \tag{53}$$

where F and G are fluent formulas and H is a formula, provided that F does not contain statically determined constants. A *causal law* is a static law, or an action dynamic law, or a fluent dynamic law.

Fluent dynamic laws (53) are the most important element of the language, because they can be used for describing direct effects of actions. If c is a Boolean action constant, we express that F is an effect of executing c by the fluent dynamic law

$$\textbf{caused } F \textbf{ if } \top \textbf{ after } c,$$

which can be abbreviated as

$$c \textbf{ causes } F.$$

Static laws can be used to talk about causal dependencies between fluents in the same state, and action dynamic laws can express causal dependencies between concurrently executed actions.

An *action description* is a set of causal laws.

The formula F in a causal law (52) or (53) is called the *head*. Note that statically determined constants are allowed in the heads of static laws, but not in the heads of dynamic laws. This explains the term "statically determined."

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

- $i \in \{0, \dots, m\}$ 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. By $i : F$ we denote the result of inserting $i{:}$ in front of every occurrence of every constant in a formula F, and similarly for a set of formulas. The rules of D_m are:

$$i{:}F \Leftarrow i{:}G \tag{54}$$

for every static law (52) in D and every $i \in \{0, \ldots, m\}$, and for every action dynamic law (52) in D and every $i \in \{0, \ldots, m-1\}$;

$$i{+}1{:}F \Leftarrow (i{+}1{:}G) \wedge (i{:}H) \tag{55}$$

for every fluent dynamic law (53) in D and every $i \in \{0, \ldots, m-1\}$;

$$0{:}c{=}v \Leftarrow 0{:}c{=}v \tag{56}$$

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

Note that the definition of D_m treats simple fluent constants and statically determined fluent constants in different ways: rules (56) are included only when c is simple.

In the example (57) below, the following abbreviations are used. If c is an action constant, the expression

exogenous c

stands for the action dynamic laws

caused $c{=}v$ **if** $c{=}v$

for all $v \in Dom(c)$. If c is a simple fluent constant, the expression

inertial c

stands for the fluent dynamic laws

caused $c{=}v$ **if** $c{=}v$ **after** $c{=}v$

for all $v \in Dom(c)$. These abbreviations for causal laws are part of a longer list given in Appendix B.

By SD we denote the following action description:

a **causes** p,
exogenous a,
inertial p. (57)

The causal theory SD_m generated from this action description as defined above is identical to the causal theory (23) denoted by SD_m in Section 3.1 (to be precise, it turns into that theory after dropping the trivial conjunctive terms $\top$ in some formulas, and replacing the atoms of the form $c = \mathbf{f}$ with $\neg c$). Indeed, the first line of (57) corresponds to the rules

$$i+1 : p \Leftarrow i : a,$$

the second line to

$$\begin{array}{l} i : a \Leftarrow i : a, \\ i : \neg a \Leftarrow i : \neg a, \end{array}$$

and the third to

$$\begin{array}{l} i+1 : p \Leftarrow (i : p) \wedge (i+1 : p), \\ i+1 : \neg p \Leftarrow (i : \neg p) \wedge (i+1 : \neg p). \end{array}$$

The remaining two lines of (23)

$$\begin{array}{l} 0 : p \Leftarrow 0 : p, \\ 0 : \neg p \Leftarrow 0 : \neg p \end{array}$$

come from (56).

4.3 Monkey and Bananas in $\mathcal{C}+$

The action description MB is shown in Figures 2, 3. The first two causal laws in this description, corresponding to rules (26) and (27), are examples of static laws (52). According to Appendix B,

$$\textbf{nonexecutable } c \textbf{ if } F$$

stands for the fluent dynamic law

$$\textbf{caused } \bot \textbf{ if } \top \textbf{ after } c \wedge F.$$

The sequence of causal theories corresponding to action description MB is identical to the sequence MB_m defined in Section 3.2.

Figure 4 shows the modification of the signature of MB needed to include a set M of names of monkeys and to make some of the causal rules defeasible, as in Section 3.5. Note that the fluent constants $Ab_1(\alpha, \beta)$ are classified here as statically determined; this is essential because the corresponding causal theories do not contain the rules

$$0 : Ab_1(\alpha, \beta) \Leftarrow 0 : Ab_1(\alpha, \beta)$$

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:
$Walk(l)$, $PushBox(l)$, $ClimbOn$, $ClimbOff$, $GraspBananas$	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** $Loc(Box)\!=\!l$
$PushBox(l)$ **causes** $Loc(Monkey)\!=\!l$
nonexecutable $PushBox(l)$ **if** $Loc(Monkey)\!=\!l$
nonexecutable $PushBox(l)$ **if** $OnBox$
nonexecutable $PushBox(l)$ **if** $Loc(Monkey)\!\neq\!Loc(Box)$

Figure 2: Action description MB, Part 1.

that would have been included, as a special case of (56), if these constants were simple. The symbols $Ab_2(\alpha)$ are classified as action constants, and not as fluent constants; this is essential because the signature of the corresponding causal theory contains the constants $i : Ab_2(\alpha)$ for $i < m$, but not for $i = m$. Like the other action constants, $Ab_2(\alpha)$ describes an event leading from one state to another, rather than a state: its value tells us whether the execution of the action $PushBox(\alpha, l)$ during that event is "abnormal." In one way, however, $Ab_2(\alpha)$ is different from the other action constants in this example: we would not include

$$\textbf{exogenous } Ab_2(\alpha)$$

in the set of postulates. This is because the corresponding causal theories

$ClimbOn$ **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 c for every action constant c

inertial c for every simple fluent constant c

Figure 3: Action description MB, Part 2.

do not contain the rules

$$i : Ab_2(\alpha) \Leftarrow i : Ab_2(\alpha).$$

Using an abbreviation introduced in Appendix B, we can write the causal laws corresponding to causal rules (41) as

$$\textbf{constraint } \neg(HasBananas(\alpha) \wedge HasBananas(\beta))$$

$(\alpha \neq \beta)$. The pair of rules (43), (44) can be written as

$$\textbf{constraint } \neg(OnBox(\alpha) \wedge OnBox(\beta)) \textbf{ unless } Ab_1(\alpha, \beta)$$

$(\alpha \neq \beta)$. The pair (47), (48) becomes

$$PushBox(\alpha, l) \textbf{ causes } Loc(Box) = l \textbf{ unless } Ab_2(\alpha),$$
$$PushBox(\alpha, l) \textbf{ causes } Loc(\alpha) = l \textbf{ unless } Ab_2(\alpha).$$

Notation: α, β range over M; x ranges over $M \cup \{Bananas, Box\}$; l ranges over $\{L_1, L_2, L_3\}$.

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

Statically determined fluent constants:	Domains:
$Ab_1(\alpha, \beta)$	Boolean

Action constants:	Domains:
$Walk(\alpha, l)$, $PushBox(\alpha, l)$, $ClimbOn(\alpha)$, $ClimbOff(\alpha)$, $GraspBananas(\alpha)$, $Ab_2(\alpha)$	Boolean

Figure 4: The signature of the action description corresponding to the sequence of causal theories from Section 3.5.

The "exception" rules (45), (49) and (50) turn into the static laws

$$\begin{array}{l}\textbf{caused } Ab_1(David, \alpha),\\ \textbf{caused } Ab_1(\alpha, David)\end{array}$$

and the action dynamic laws

$$\begin{array}{l}\textbf{caused } Ab_2(\alpha) \textbf{ if } OnBox(Goliath),\\ \textbf{caused } Ab_2(\alpha) \textbf{ if } ClimbOn(Goliath).\end{array}$$

4.4 From Action Descriptions to Transition Systems

Consider an action description D with a set σ^{fl} of fluent constants and a set σ^{act} of action constants. We will define now the transition system represented by D, using the first two members D_0, D_1 of the sequence of causal theories corresponding to D. In this definition, we identify an interpretation I in the sense of Section 2.1 with the set of atoms that are satisfied by this interpretation, that is to say, with the set of atoms of the form $c = I(c)$. This convention allows us to represent any interpretation of the signature of D_m in the form

$$(0\!:\!s_0) \cup (0\!:\!e_0) \cup (1\!:\!s_1) \cup (1\!:\!e_1) \cup \cdots \cup (m\!:\!s_m) \tag{58}$$

where $s_0, \ldots, s_m$ are interpretations of σ^{fl}, and $e_1, \ldots, e_{m-1}$ are interpretations of σ^{act}. For instance, for the interpretation I defined by (51),

$$\begin{aligned} I &= \{0{:}p = \mathbf{f},\ 0{:}a = \mathbf{f},\ 1{:}p = \mathbf{f},\ 1{:}a = \mathbf{t},\ 2{:}p = \mathbf{t}\} \\ &= 0{:}\{p = \mathbf{f}\} \cup 0{:}\{a = \mathbf{f}\} \cup 1{:}\{p = \mathbf{f}\} \cup 1{:}\{a = \mathbf{t}\} \cup 2{:}\{p = \mathbf{t}\}. \end{aligned}$$

A *state* is an interpretation s of σ^{fl} such that $0 : s$ is a model of D_0. States are the vertices of the transition system represented by D. A *transition* is a triple $\langle s, e, s' \rangle$, where s and s' are interpretations of σ^{fl} and e is an interpretation of σ^{act}, such that $0{:}s \cup 0{:}e \cup 1{:}s'$ is a model of D_1. Transitions correspond to the edges of the transition system: for every transition $\langle s, e, s' \rangle$, it contains an edge from s to s' labeled e. These labels e will be called *events*.

For example, the vertices of the transition system in Figure 1 are $\{p = \mathbf{f}\}$ and $\{p = \mathbf{t}\}$; accordingly, $0{:}\{p = \mathbf{f}\}$ and $0{:}\{p = \mathbf{t}\}$ are the models of causal theory (24). The transitions of that transition system are

$$\begin{array}{l} \langle \{p = \mathbf{f}\}, \{a = \mathbf{f}\}, \{p = \mathbf{f}\} \rangle, \\ \langle \{p = \mathbf{f}\}, \{a = \mathbf{t}\}, \{p = \mathbf{t}\} \rangle, \\ \langle \{p = \mathbf{t}\}, \{a = \mathbf{f}\}, \{p = \mathbf{t}\} \rangle, \\ \langle \{p = \mathbf{t}\}, \{a = \mathbf{t}\}, \{p = \mathbf{t}\} \rangle; \end{array}$$

accordingly, the interpretations

$$\begin{array}{l} 0{:}\{p = \mathbf{f}\} \cup 0{:}\{a = \mathbf{f}\} \cup 1{:}\{p = \mathbf{f}\}, \\ 0{:}\{p = \mathbf{f}\} \cup 0{:}\{a = \mathbf{t}\} \cup 1{:}\{p = \mathbf{t}\}, \\ 0{:}\{p = \mathbf{t}\} \cup 0{:}\{a = \mathbf{f}\} \cup 1{:}\{p = \mathbf{t}\}, \\ 0{:}\{p = \mathbf{t}\} \cup 0{:}\{a = \mathbf{t}\} \cup 1{:}\{p = \mathbf{t}\} \end{array}$$

are the models of causal theory (19)–(22).

The definition of the transition system above implicitly relies on the following property of transitions:

Proposition 7 *For any transition $\langle s, e, s' \rangle$, s and s' are states.*

The validity of this proposition depends on the fact that in the definition of a fluent dynamic law (Section 4.2) the head is not allowed to contain statically determined fluent constants. Indeed, imagine that this limitation is lifted. Then we will be able to form an action description D that consists of two causal laws: the static law

$$\textbf{caused } p \textbf{ if } p$$

and the fluent dynamic law

$$\textbf{caused } \neg p \textbf{ if } \top \textbf{ after } p$$

where p is a statically determined Boolean fluent constant. For this D, causal theory D_0 is

$$0\!:\!p \Leftarrow 0\!:\!p$$

and causal theory D_1 is

$$\begin{array}{l} 0\!:\!p \Leftarrow 0\!:\!p, \\ 1\!:\!p \Leftarrow 1\!:\!p, \\ 1\!:\!\neg p \Leftarrow 0\!:\!p. \end{array}$$

The last member $\{p = \mathbf{f}\}$ of the transition $\langle\{p = \mathbf{t}\}, \emptyset, \{p = \mathbf{f}\}\rangle$ is not a state.

This fact explains the need to distinguish between fluent constants of two kinds—simple, that are allowed in the heads of dynamic laws of D, and statically determined, for which causal rules (56) are not automatically included in D_m.

The correspondence between the models of D_m and paths in the transition system represented by D can be described as follows:

Proposition 8 *For any $m > 0$, an interpretation (58) of the signature of D_m is a model of D_m iff each triple $\langle s_i, e_i, s_{i+1}\rangle$ $(0 \le i < m)$ is a transition.*

4.5 Causal Logic as a Subset of $\mathcal{C}+$

In Section 4.2 we defined the semantics of $\mathcal{C}+$ by showing how to turn any action description into a sequence of causal theories. A reduction in the opposite direction is possible also. Any causal theory T can be turned into an action description by treating every constant of T as a statically determined fluent constant, and rewriting every causal rule

$$F \Leftarrow G$$

as the static law

$$\textbf{caused } F \textbf{ if } G.$$

It is easy to see that the states of the transition system represented by this action description are identical to the models of T.

We will agree to identify a causal theory with the corresponding action description. We will treat any causal theory, in other words, as an action description without simple fluent constants, without action constants, and without dynamic laws. This convention allows us to specify causal theories using the abbreviations for static laws introduced in Appendix B. For instance, (2) can be now written as

$$\begin{array}{l}\textbf{caused } p \textbf{ if } q,\\ \textbf{exogenous } q,\end{array} \tag{59}$$

and (11) can be written as

$$\textbf{default } c{=}1.$$

4.6 Rigid Constants

A fluent constant c in the signature of an action description D is *rigid* (relative to D) if, for every transition $\langle s, e, s' \rangle$ in the transition system represented by D, $s'(c) = s(c)$. Intuitively, rigid constants represent the fluents whose values are not affected by any events.

Imagine, for instance, that the monkey in the Monkey and Bananas domain is unable to push the box. This assumption can be expressed by adding the causal laws

$$\textbf{nonexecutable } \mathit{PushBox}(l)$$

to the action description MB shown in Figures 2, 3. This modification makes the fluent constant $\mathit{Loc}(\mathit{Box})$ rigid. The corresponding transition system consists of 3 disconnected parts: $s(\mathit{Loc}(\mathit{Box}))$ is L_1 for the states s in the first part, L_2 in the second part, and L_3 in the third. No edges of the transition system lead from one part to another.

Other examples of action descriptions with rigid constants are given by the extensions of MB that contain symbols for the size and the material of the box, or for the species and gender of the monkey.

According to Appendix B, the expression

$$\textbf{rigid } c$$

stands for the set of causal laws

$$\textbf{constraint } c{=}v \textbf{ after } c{=}v,$$

that is to say,

$$\textbf{caused } \bot \textbf{ if } \neg(c{=}v) \textbf{ after } c{=}v$$

for all $v \in Dom(c)$. It is clear that c is rigid relative to any action description containing these laws.

One of the reasons why rigid constants are interesting is that, under some conditions, their presence allows us to make the causal theories D_m more compact, which can be computationally advantageous. Let R be a set of fluent constants that are rigid relative to D. Denote by D_m^R the causal theory whose constants and causal rules are obtained from the constants and causal rules of D_m by dropping the time stamps before each constant from R. For any interpretation I of the signature of D_m, by I^R we denote the interpretation of the signature of D_m^R defined by the formulas

$$\begin{array}{ll} I^R(c) = I(0:c) & \text{if } c \in R, \\ I^R(i\!:\!c) = I(i\!:\!c) & \text{if } c \notin R. \end{array}$$

Proposition 9 *If*

(i) every constant in R is statically determined, and

(ii) for every causal law in D that contains a constant from R in the head, all constants occurring in this law belong to R,

then the mapping $I \mapsto I^R$ is a 1-1 correspondence between the models of D_m and the models of D_m^R.

Thus dropping the time stamps in front of the rigid constants from R does not affect the meaning of D_m if, first, R contains no simple constants, and second, no constant from R "causally depends" on a constant that does not belong to R.

The following example shows that the assertion of Proposition 9 would be incorrect without the first condition. Take D to be

$$\begin{array}{l} \textbf{rigid } p, \\ \textbf{default } p \end{array}$$

where p is a Boolean simple fluent, and let $R = \{p\}$. Then D_1 is

$$\begin{array}{l} \bot \Leftarrow (1\!:\!p) \wedge \neg(0\!:\!p), \\ \bot \Leftarrow \neg(1\!:\!p) \wedge (0\!:\!p), \\ 0\!:\!p \Leftarrow 0\!:\!p, \\ 1\!:\!p \Leftarrow 1\!:\!p, \\ 0\!:\!\neg p \Leftarrow 0\!:\!\neg p \end{array}$$

and D_1^R is

$$\begin{array}{l}\bot \Leftarrow p \wedge \neg p,\\ \bot \Leftarrow \neg p \wedge p,\\ p \Leftarrow p,\\ \neg p \Leftarrow \neg p.\end{array}$$

The interpretation $\{p = \mathbf{f}\}$ is a model of D_1^R, but it does not have the form I^R for any model I of D_1.

The following example shows that the assertion of Proposition 9 would be incorrect without the second condition. Take D to be (59), where p and q are statically determined fluent constants, and let $R = \{p\}$. Then D_1 is

$$\begin{array}{l}0\!:\!p \Leftarrow 0\!:\!q,\\ 1\!:\!p \Leftarrow 1\!:\!q,\\ 0\!:\!q \Leftarrow 0\!:\!q,\\ 1\!:\!q \Leftarrow 1\!:\!q,\\ \neg 0\!:\!q \Leftarrow \neg 0\!:\!q,\\ \neg 1\!:\!q \Leftarrow \neg 1\!:\!q\end{array}$$

and D_1^R is

$$\begin{array}{l}p \Leftarrow 0\!:\!q,\\ p \Leftarrow 1\!:\!q,\\ 0\!:\!q \Leftarrow 0\!:\!q,\\ 1\!:\!q \Leftarrow 1\!:\!q,\\ \neg 0\!:\!q \Leftarrow \neg 0\!:\!q,\\ \neg 1\!:\!q \Leftarrow \neg 1\!:\!q.\end{array}$$

The interpretation $\{p = \mathbf{t}, 0\!:\!q = \mathbf{f}, 1\!:\!q = \mathbf{t}\}$ is a model of D_1^R, but it does not have the form I^R for any model I of D_1.

5 Expressive Possibilities of the Definite Fragment of $\mathcal{C}+$

An action description D is *definite* if

- the head of every causal law of D is an atom or $\bot$, and
- no atom is the head of infinitely many causal laws of D.

This is similar to the definition of a definite causal theory in Section 2.6. For any definite action description D, the corresponding causal theories D_m are definite also, so that many questions about transition systems described in the definite fragment of $\mathcal{C}+$ can be answered using the computational methods discussed in Sections 2.6 and 3.3.

The examples of action descriptions discussed in Section 4 are definite (or can be made definite by replacing formulas $\neg c$ in the heads of causal laws by atoms $c = \mathbf{f}$). In this section we give a few other examples illustrating the expressive possibilities of the definite fragment of $\mathcal{C}+$.

5.1 Actions with Conditional Effects

Effects of an action can be "conditional": they may be caused by executing the action in some states, but not in others. As an example of representing conditional effects in $\mathcal{C}+$, we formalize in Figure 5 an enhancement of the well-known "Yale shooting" example [Hanks and McDermott, 1987] in which the effect of shooting depends on how the gun is aimed. As defined in Appendix B, the expression

$$\mathit{Shoot}\ \mathbf{causes}\ \neg \mathit{Alive}(x)\ \mathbf{if}\ \mathit{Target} = x$$

in this action description stands for the fluent dynamic law

$$\mathbf{caused}\ \neg \mathit{Alive}(x)\ \mathbf{if}\ \top\ \mathbf{after}\ \mathit{Shoot} \wedge \mathit{Target} = x.$$

5.2 Nondeterministic Actions

When Jack goes to work, he either walks there or drives his car. We view walking and driving as two ways of executing the same action. The effect of that action on Jack's location is deterministic, but its effect on the location of his car is not.

The representation of this example in Figure 6 describes the nondeterministic effect of $Go(l)$ by the expression

$$Go(l)\ \mathbf{may\ cause}\ \mathit{Loc}(\mathit{Car}) = l\ \mathbf{if}\ \mathit{Loc}(\mathit{Car}) = \mathit{Loc}(\mathit{Jack}).$$

According to Appendix B, it stands for the fluent dynamic law

$$\begin{array}{l}\mathbf{caused}\ \mathit{Loc}(\mathit{Car}) = l\ \mathbf{if}\ \mathit{Loc}(\mathit{Car}) = l \\ \qquad\qquad\qquad\qquad \mathbf{after}\ Go(l) \wedge \mathit{Loc}(\mathit{Car}) = \mathit{Loc}(\mathit{Jack}).\end{array}$$

By placing a copy of the head $\mathit{Loc}(\mathit{Car}) = l$ after **if** we say that this is not a necessary effect of the action, but merely a possible one. If this condition holds in the resulting state then there is a cause for this.

Notation: x ranges over $\{Turkey1, Turkey2\}$.

Simple fluent constants:	Domains:
$Loaded$, $Alive(x)$	Boolean
$Target$	$\{Turkey1, Turkey2, None\}$
Action constants:	Domains:
$Load$, $Aim(x)$, $Shoot$	Boolean

Causal laws:

$Load$ **causes** $Loaded$
$Load$ **causes** $Target = None$

$Aim(x)$ **causes** $Target = x$

$Shoot$ **causes** $\neg Alive(x)$ **if** $Target = x$
$Shoot$ **causes** $\neg Loaded$
nonexecutable $Shoot$ **if** $\neg Loaded$

nonexecutable $Aim(x) \wedge Shoot$

exogenous c for every action constant c

inertial c for every simple fluent constant c

Figure 5: Shooting turkeys.

In the transition system represented by this action description we can find two different edges that start at the same state and are labeled by the same event. For instance, there are two edges that start at

$$\{Loc(Jack) = Home, Loc(Car) = Home\}$$

and have the label

$$\{Go(Home) = \mathbf{f}, Go(Work) = \mathbf{t}\}.$$

One of them leads to

$$\{Loc(Jack) = Work, Loc(Car) = Home\}$$

Notation: x ranges over $\{Jack, Car\}$; l ranges over $\{Home, Work\}$.

Simple fluent constants:	Domains:
$Loc(x)$	$\{Home, Work\}$
Action constants:	Domains:
$Go(l)$	Boolean

Causal laws:

$Go(l)$ **causes** $Loc(Jack)\!=\!l$
$Go(l)$ **may cause** $Loc(Car)\!=\!l$ **if** $Loc(Car)\!=\!Loc(Jack)$
nonexecutable $Go(l)$ **if** $Loc(Jack)\!=\!l$

exogenous c for every action constant c

inertial c for every simple fluent constant c

Figure 6: Going to work.

(walking), and the other to

$$\{Loc(Jack) = Work, Loc(Car) = Work\}$$

(driving).

5.3 Interaction between Concurrently Executed Actions

In a standard example of interacting actions, two agents lift the opposite ends of a table upon which various objects have been placed [Pednault, 1987, Section 3]. If one end of the table has been raised, the objects on the table will fall off. But if both ends are lifted simultaneously, the objects on the table will remain fixed.

A formalization of this domain in $\mathcal{C}+$ is shown in Figure 7. The change in the value of the fluent *OnTable* is treated here as an indirect effect of changes in the positions of the ends of the table.

Notation: x ranges over $\{LeftEnd, RightEnd\}$.

Simple fluent constants:	Domains:
$Level(x)$	$\{Low, High\}$
$OnTable$	Boolean

Action constants:	Domains:
$Lift(x)$	Boolean

Causal laws:

$Lift(x)$ **causes** $Level(x) = High$
nonexecutable $Lift(x)$ **if** $Level(x) = High$

caused $\neg OnTable$ **if** $Level(LeftEnd) \neq Level(RightEnd)$

exogenous c for every action constant c

inertial c for every simple fluent constant c

Figure 7: Lifting the ends of the table.

5.4 Noninertial Fluents

Consider a pendulum that moves from its leftmost position to the rightmost and back, with each swing taking one unit of time. We would like to describe the effect of holding the pendulum steady in its current position for the duration of one unit of time.

The action description in Figure 8 describes the "default" behavior of the pendulum by the expressions

default $Right$ **after** $\neg Right$
default $\neg Right$ **after** $Right$

that stand for the dynamic laws

caused $Right$ **if** $Right$ **after** $\neg Right$,
caused $\neg Right$ **if** $\neg Right$ **after** $Right$

(Appendix B). They are used here instead of the dynamic laws

inertial $Right$,

Simple fluent constant: Domain:
Right Boolean

Action constant: Domain:
Hold Boolean

Causal laws:

Hold **causes** *Right* **if** *Right*
Hold **causes** $\neg$*Right* **if** $\neg$*Right*

default *Right* **after** $\neg$*Right*
default $\neg$*Right* **after** *Right*

exogenous *Hold*

Figure 8: Holding the pendulum.

that is to say, instead of

caused *Right* **if** *Right* **after** *Right*,
caused $\neg$*Right* **if** $\neg$*Right* **after** $\neg$*Right*.

This is an example of a simple fluent constant that is not postulated to be inertial.

5.5 Defined Fluents

In the process of designing an action description, we may want to introduce a fluent constant that is defined in terms of other fluent constants. To see how this can be done, imagine that we want to expand action description *MB* (Figures 2, 3) by the new Boolean fluent constant *NextToBox* that can serve as shorthand for the formula $Loc(Monkey) = Loc(Box)$.

In the extended action description MB' we make the new fluent constant *NextToBox* statically determined, and include two static laws containing that constant in the head:

$$\begin{array}{l}\textbf{caused } NextToBox \textbf{ if } Loc(Monkey) = Loc(Box),\\ \textbf{default } \neg NextToBox\end{array} \tag{60}$$

(there is a cause for *NextToBox* to be true if the monkey and the box are at the same location; otherwise, *NextToBox* is false).

It is easy to see that the completion of MB'_m can be obtained from the completion of MB_m by adding the formulas

$$i : (NextToBox \equiv Loc(Monkey) = Loc(Box))$$

$(i = 0, \dots, m)$. This observation shows that the transition system represented by *MB* is isomorphic to the transition system represented by MB': every state of the former can be turned into the corresponding state of the latter by assigning to *NextToBox* the truth value equal to the truth value of the formula $Loc(Monkey) = Loc(Box)$.

In the presence of (60), any occurrences of $Loc(Monkey) = Loc(Box)$ in the bodies of the causal laws of *MB* can be equivalently replaced with *NextToBox*. For instance,

$$\textbf{nonexecutable } PushBox(l) \textbf{ if } Loc(Monkey) \neq Loc(Box)$$

can be rewritten as

$$\textbf{nonexecutable } PushBox(l) \textbf{ if } \neg NextToBox.$$

Proposition 4(ii) shows that the first of the causal laws (60) can be equivalently replaced with

$$\textbf{caused } NextToBox \textbf{ if } Loc(Monkey) = l \wedge Loc(Box) = l$$

$(l = L_1, L_2, L_3)$.

The fact that the transition systems represented by *MB* and MB' are isomorphic to each other depends on the assumption that the new fluent constant is statically determined. If we made *NextToBox* a simple fluent constant then the causal theories MB' would have been different because of the instance

$$0 : NextToBox \Leftarrow 0 : NextToBox$$

of (56).

5.6 Describing Actions by Attributes

Executing the action *Publish* causes the Boolean fluent *HasPublications* to become true:

$$Publish \textbf{ causes } HasPublications. \tag{61}$$

Imagine that we want to enhance an action description containing this causal law by distinguishing between publications of different kinds. The fluent *HasPublications* becomes true no matter whether a conference paper or a journal article has been published, but in the second case the action has one more effect—the fluent *HasJournalPublications* becomes true also. This distinction can be expressed by switching from the notation *Publish* for the action in question to the more elaborate notation $Publish(k)$, where k ranges over the types of publications *Conference*, *Journal*. In the modified action description, (61) turns into

$$Publish(k) \textbf{ causes } HasPublications \tag{62}$$

for both possible values of k, and additionally we postulate

$$Publish(Journal) \textbf{ causes } HasJournalPublications. \tag{63}$$

If we later decide to distinguish between publications in terms of their length, the notation for the action will need to be amended again. Instead of $Publish(k)$, we can denote the action, for instance, by $Publish(k,l)$, where the number of pages l ranges over an initial segment of positive integers. Causal laws (62) and (63) will turn then into

$$\begin{array}{l} Publish(k,l) \textbf{ causes } HasPublications, \\ Publish(Journal,l) \textbf{ causes } HasJournalPublications, \end{array} \tag{64}$$

and we also will be able to say

$$Publish(k,l) \textbf{ causes } HasLongPublications \qquad (l > 30)$$

to express that publishing a paper that is over 30 pages causes the fluent *HasLongPublications* to become true.

This example illustrates the need to modify notation for actions by introducing additional arguments, and to change the form of causal laws, that often arises when we want to introduce additional distinctions. An alternative way to represent such distinctions, which is sometimes preferable, is based on expressing them by "attributes." In Figure 9, the kind and length of a publication are treated as attributes of the action *Publish*. The attributes *Kind* and *Length* are action constants—their values, like the value of *Publish*, characterize an event occurring between two states. The domain of every attribute of an action includes the special value *None*, and the attribute is required to take that value (to be "undefined") if and only if the action is not executed. For instance, we postulate

$$\textbf{always } Kind = None \equiv \neg Publish.$$

Notation: l ranges over $\{1, \ldots, 100\}$.

Simple fluent constants:	Domains:
HasPublications, *HasJournalPublications*, *HasLongPublications*	Boolean

Action constants:	Domains:
Publish	Boolean
Kind	$\{Conference, Journal, None\}$
Length	$\{1, \ldots, 100, None\}$

Causal laws:

Publish **causes** *HasPublications*

always $Kind = None \equiv \neg Publish$
Publish **causes** *HasJournalPublications* **if** $Kind = Journal$

always $Length = None \equiv \neg Publish$
Publish **causes** *HasLongPublications* **if** $Length = l$ for $l > 30$

exogenous c for every action constant c

inertial c for every simple fluent constant c

Figure 9: Publishing papers.

According to Appendix B, this expression stands for the fluent dynamic law

$$\textbf{caused} \perp \textbf{if} \top \textbf{after} \neg(Kind = None \equiv \neg Publish).$$

6 Action Descriptions and Queries in the Language of the Causal Calculator

The original version of the Causal Calculator was written by one of the authors as part of his dissertation [McCain, 1997]. Now the system is being maintained by Texas Action Group at Austin.[5] It can be downloaded from its home page

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

To illustrate the syntax of the input language of CCalc, we show in Figures 10 and 11 how to rewrite in that language the $\mathcal{C}+$ description of the Monkey and Bananas domain given in Figures 2 and 3.

CCalc is written in Prolog, and it follows the Prolog tradition of capitalizing variables. This is why, in the language of CCalc, $Loc(Monkey)=l$ turns into `loc(monkey)=L`.

The ranges of variables, described at the beginning of Figure 2, are given names in the sort declarations at the beginning of Figure 10. The extent of each sort is defined in object declarations.

The constant declaration

```
loc(thing)              :: inertialFluent(location)
```

has the same meaning as the declaration

```
loc(thing)              :: simpleFluent(location)
```

accompanied by the fluent dynamic law

```
inertial loc(X)
```

where `X` is a variable for `thing`s. That is to say, this declaration says that any expression consisting of the symbol `loc` followed by a `thing` in parentheses is a simple fluent constant, that the domain of each of these constants is the set of `location`s, and that these constants are postulated to be inertial. The symbol `inertialFluent` is used more often than `simpleFluent` in CCalc action descriptions, because the inertia assumption is required for almost every simple fluent constant. (There are exceptions, however, as we saw in Section 5.4.) When the symbol `inertialFluent` in the declaration of a constant is not followed by a sort, the constant is assumed to be Boolean. This convention is used in the declarations of `onBox` and `hasBananas`. Similarly, the action constants declared in Figure 10 are understood to be Boolean, since no other domain is specified.

The declaration

```
walk(location)              :: exogenousAction
```

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

```
:- sorts
  thing;
  location.

:- objects
  monkey,bananas,box     :: thing;
  l1,l2,l3               :: location.

:- variables
  L                      :: location.

:- constants
  loc(thing)             :: inertialFluent(location);
  hasBananas,onBox       :: inertialFluent;

  walk(location),
  pushBox(location),
  climbOn,
  climbOff,
  graspBananas           :: exogenousAction.

caused loc(bananas)=L if hasBananas & loc(monkey)=L.
caused loc(monkey)=L if onBox & 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.
pushBox(L) causes loc(monkey)=L.
nonexecutable pushBox(L) if loc(monkey)=L.
nonexecutable pushBox(L) if onBox.
nonexecutable pushBox(L) if loc(monkey)\=loc(box).
```

Figure 10: Monkey and Bananas in the language of CCalc, Part 1.

```
climbOn causes onBox.
nonexecutable climbOn if onBox.
nonexecutable climbOn if loc(monkey)\=loc(box).

climbOff causes -onBox.
nonexecutable climbOff if -onBox.

graspBananas causes hasBananas.
nonexecutable graspBananas if hasBananas.
nonexecutable graspBananas if -onBox.
nonexecutable graspBananas if loc(monkey)\=loc(bananas).

nonexecutable walk(L) & pushBox(L).
nonexecutable walk(L) & climbOn.
nonexecutable pushBox(L) & climbOn.
nonexecutable climbOff & graspBananas.
```

Figure 11: Monkey and Bananas in the language of CCALC, Part 2.

has the same meaning as the declaration

```
walk(location)              :: action
```

accompanied by the action dynamic law

```
exogenous walk(L)
```

where `L` is a variable for `locations`. That is to say, this declaration says that any expression consisting of the symbol `walk` followed by a `location` in parentheses is a Boolean-valued action constant, and that these constants are postulated to be exogenous. The symbol `exogenousAction` is used more often than `action` in CCALC action descriptions, because most actions are exogenous. (There are exceptions, however, such as $Ab_2(\alpha)$ in Section 4.3.)

The rest of the CCALC encoding of the Monkey and Bananas domain consists of causal laws. It is almost identical to the list of causal laws in Figures 2, 3.

Figure 12 shows how a planning problem can be represented in the language of the Causal Calculator. This piece of code instructs CCALC to test successively $m = 1, \dots, 10$ and find the smallest value for which MB_m has a model satisfying both the initial conditions (37) and the goal (38). In response to this query, CCALC produces the following output:

```
:- query
  maxstep :: 1..10;
  0: loc(monkey)=l1,
     loc(bananas)=l2,
     loc(box)=l3;
  maxstep: hasBananas.
```

Figure 12: A planning problem in the language of CCalc.

```
0:  loc(monkey)=l1  loc(bananas)=l2  loc(box)=l3

ACTIONS:  walk(l3)

1:  loc(monkey)=l3  loc(bananas)=l2  loc(box)=l3

ACTIONS:  pushBox(l2)

2:  loc(monkey)=l2  loc(bananas)=l2  loc(box)=l2

ACTIONS:  climbOn

3:  onBox  loc(monkey)=l2  loc(bananas)=l2  loc(box)=l2

ACTIONS:  graspBananas

4:  hasBananas  onBox  loc(monkey)=l2  loc(bananas)=l2
loc(box)=l2
```

7 Related Work

7.1 Default Logic

A causal theory of a Boolean signature can be viewed as a default theory in the sense of [Reiter, 1980]. (The syntax and semantics of propositional default theories are reviewed in Section 8.10.) Let us agree to identify a causal rule $F \Leftarrow G$ with the default

$$\frac{:\ G}{F}. \tag{65}$$

In the statement of the following proposition, we identify an interpretation I with the set of formulas satisfied by I.

Proposition 10 *Let T be a causal theory of a Boolean signature. An interpretation I is a model of T iff I is an extension for T in the sense of default logic.*

This theorem shows that causal rules are essentially propositional defaults without prerequisites and with a single justification, as long as we are only interested in the extensions that correspond to interpretations (that is to say, in the extensions that are consistent and complete).

For instance, causal theory (2) corresponds to the default theory

$$\frac{:\ q}{p},\ \frac{:\ q}{q},\ \frac{:\ \neg q}{\neg q}.$$

This theory has two extensions: the set of all consequences of p, q and the set of all consequences of $\neg q$. The first extension is complete, and it corresponds to the only model of (2).

This translation into default logic can be applied, in particular, to causal theories of the form D_m (Section 4). Given an action description D whose signature is Boolean, and a nonnegative integer m, consider the following set of defaults:[6]

$$\frac{:\ (i\!:\!G)}{i\!:\!F}$$

for every static law (52) in D and every $i \in \{0, \ldots, m\}$, and for every action dynamic law (52) in D and every $i \in \{0, \ldots, m-1\}$;

$$\frac{:\ (i\!+\!1\!:\!G) \wedge (i\!:\!H)}{i\!+\!1\!:\!F} \tag{66}$$

for every fluent dynamic law (53) in D and every $i \in \{0, \ldots, m-1\}$;

$$\frac{:\ (0\!:\!c\!=\!v)}{0\!:\!c\!=\!v}$$

for every simple fluent constant c and every $v \in Dom(c)$. According to Proposition 10, an interpretation I is a model of D_m if and only if it is an extension for this set of defaults. Alternatively, the second conjunctive term

[6] Note that in these defaults the colon is used in two ways: to separate justifications from prerequisites, as in default logic, and to separate time stamps from formulas, as described in Section 2.2.

in the justification of (66) can be turned into a prerequisite, so that this default becomes

$$\frac{(i\!:\!H)\ :\ (i\!+\!1\!:\!G)}{i\!+\!1\!:\!F}.$$

As it turns out, if the translation of the fluent dynamic laws of D is modified in this way, the complete, consistent extensions for the default theory remain the same. (Proof is straightforward using the Splitting Sequence Theorem for default logic from [Turner, 1996].)

It is interesting to apply the modified translation to the dynamic laws **inertial** c (Section 4). For Boolean c, this expression stands for the pair of dynamic laws

$$\begin{array}{l} \textbf{caused}\ c\ \textbf{if}\ c\ \textbf{after}\ c, \\ \textbf{caused}\ \neg c\ \textbf{if}\ \neg c\ \textbf{after}\ \neg c. \end{array}$$

In application to these laws, the modified translation gives the normal defaults

$$\frac{(i\!:\!c)\ :\ (i\!+\!1\!:\!c)}{i\!+\!1\!:\!c},\ \frac{(i\!:\!\neg c)\ :\ (i\!+\!1\!:\!\neg c)}{i\!+\!1\!:\!\neg c}.$$

This observation shows that the approach to the frame problem adopted in this paper is closely related to the "frame default" from [Reiter, 1980, Section 1.1.4].

7.2 Logic Programming

The embedding of causal logic into default logic (Section 7.1) assumes that the underlying signature is Boolean. If we assume, in addition, that the head of every rule is an atom ($c=\mathbf{t}$ or $c=\mathbf{f}$), then a causal theory can be also translated into the language of logic programs under the answer set semantics [Gelfond and Lifschitz, 1991]. Using the equivalent transformations discussed in Section 2.4, such a theory can be rewritten as a set of rules of the form

$$l_0 \Leftarrow l_1 \wedge \cdots \wedge l_n \tag{67}$$

where $l_0, \ldots, l_n$ ($n \geq 0$) are literals (c or $\neg c$). Let us agree to identify causal rule (67) with the logic programming rule

$$l_0 \leftarrow not\ \overline{l_1}, \ldots, not\ \overline{l_n} \tag{68}$$

($\bar{l}$ stands for the literal complementary to l). In the statement of the following proposition, we identify an interpretation I with the set of literals satisfied by I.

Proposition 11 *Let T be a causal theory whose rules have the form (67). An interpretation I is a model of T iff I is an answer set for T in the sense of logic programming.*

This theorem shows that causal rules of the form (67) can be viewed as nondisjunctive rules under the answer set semantics, as long as we are only interested in consistent and complete answer sets.

Recall that an expression in the body of a rule in a logic program corresponds to a justification in a default if that expression contains negation as failure, and it corresponds to a conjunctive term of the prerequisite if it does not [Gelfond and Lifschitz, 1991, Section 5]. From this perspective, rule (68) is similar to default (65): the body of (68) does not include expressions without negation as failure, and (65) does not have a prerequisite. On the other hand, the body of (68) may have several occurrences of negation as failure, but (65) has only one justification.

The completion process defined in Section 2.6 is a modification of the syntactic transformation proposed in [Clark, 1978] as a semantics of negation as failure. For a Boolean signature, this process differs from the propositional case of Clark's completion in that it introduces completion formulas not only for atoms in the sense of classical propositional logic, but also for negative literals (atoms of the form $c = \mathbf{f}$). This is why the completion of causal theories was called "literal completion" when it was first described in [McCain and Turner, 1997].

This observation shows that it is easy to reduce the propositional case of Clark's completion to completion in the sense of causal logic. Let T be a finite set of rules of the form

$$a \Leftarrow l_1 \wedge \cdots \wedge l_n$$

where a is an atom in the sense of classical propositional logic, and $l_1, \ldots, l_n$ ($n \geq 0$) are literals. Clark's completion of T can be obtained by applying the completion procedure defined in Section 2.6 to the union of T with the set of rules

$$\neg a \Leftarrow \neg a$$

for all atoms a. Indeed, in the presence of these additional rules, the completion formulas for the negative literals are tautologies $\neg a \equiv \neg a$.

As a semantics of logic programming, completion is known to lead sometimes to unintuitive results. For instance, under the completion semantics, the rules

$$\begin{array}{l} q(x,x) \Leftarrow \top \\ q(x,y) \Leftarrow p(x,z) \wedge q(z,y) \end{array} \tag{69}$$

fail to express that q is the reflexive transitive closure of p.[7] For this reason, the analogy between causal theories and logic programs can be misleading. For instance, the union of (69) with the closed-world assumption

$$\neg q(x,y) \Leftarrow \neg q(x,y)$$

fails to express in causal logic that q is the reflexive transitive closure of p.

7.3 Action Languages

Language $\mathcal{C}+$ is a new addition to the family of formal languages for describing actions, which started with STRIPS [Fikes and Nilsson, 1971] and ADL [Pednault, 1994]. Gelfond and Lifschitz [1993] related language $\mathcal{A}$ (which is essentially the propositional fragment of ADL) to logic programming. Baral and Gelfond [1997] extended this work to a language that permits the concurrent execution of actions (including the "difficult" cases, such as the example discussed in Section 5.3).

A similar result for a language with static causal laws is proved in [Turner, 1997]. That work, along with the theory of nonmonotonic causal reasoning presented in [McCain and Turner, 1997], has led to the design of language $\mathcal{C}$ [Giunchiglia and Lifschitz, 1998], which is the immediate predecessor of $\mathcal{C}+$.

7.4 Other Research on Representing Actions

The formalism discussed in this paper is part of a long line of research on representing actions, and, in particular, of research on the frame problem in the presence of actions with indirect effects. A critical discussion of this work and a comprehensive bibliography can be found in [Shanahan, 1997].

The explicit definition of $1\!:\!p$ in terms of $0\!:\!p$ and $0\!:\!a$ in Section 3.1 is similar to successor state axioms introduced by Reiter [1991] as a generalization of earlier work by Pednault [1989].

Among the research on the nonmonotonic logic of causation, besides the papers mentioned in the introduction, Fangzhen Lin's proposal to circumscribe the predicate *Caused* is particularly relevant [Lin, 1995, Lin, 1996]. Although that approach uses a different model of nonmonotonic reasoning, our formalism is similar to it in the sense that both theories talk about "being caused," but not about what a cause is. It is also interesting to note

[7] Example: combine (69) with rules $p(1,1) \Leftarrow \top$ and $p(2,2) \Leftarrow \top$, and let x, y, z range over $\{1,2\}$. Clark's completion of this program has an unintended model in which q is identically true.

that the circumscriptions used by Lin can be often reduced to a form of completion, just like definite causal theories (Section 2.6).

An alternative way to apply causation to describing indirect effects of actions is proposed in [Thielscher, 1997].

Nonmonotonic causal theories are similar in some ways to temporal action logics [Doherty *et al.*, 1998, Kvarnström and Doherty, 2001], which were inspired by the ideas of [Sandewall, 1994]. The Causal Calculator can be compared to VITAL ("Visualization and Implementation of Temporal Action Logics")[8]—a powerful software system for planning and for query answering in action domains.

The use of action attributes (Section 5.6) is an old idea in linguistic representations, originally due to Davidson [1967].

7.5 Satisfiability Planning and Answer Set Programming

Computationally, the operation of CCALC as a planner is a form of satisfiability planning [Kautz and Selman, 1992]. Since causal theories are closely related to logic programs, it can be also viewed as a form of answer set programming [Marek and Truszczyński, 1999], [Niemelä, 1999] applied to plan generation [Dimopoulos *et al.*, 1997], [Lifschitz, 2002]. Since CCALC employs satisfiability solvers for search, its operation is particularly close to the operation of the answer set solver CMODELS.[9]

8 Proofs of Theorems

8.1 Proof of Proposition 1

Proposition 1 *An interpretation I is a model of a causal theory T if and only if, for every formula F,*

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

Proof

$$\begin{array}{lll} I \text{ is a model of } T & \text{iff} & I \text{ is the unique model of } T^I \\ & \text{iff} & \text{for every atom } c{=}v,\ I \models c{=}v \text{ iff } T^I \models c{=}v \\ & \text{iff} & \text{for every formula } F,\ I \models F \text{ iff } T^I \models F. \end{array}$$

[8] `http://www.ida.liu.se/~jonkv/vital/` .

[9] `http://www.cs.utexas.edu/users/tag/cmodels.html` .

8.2 Proof of Proposition 2

Proposition 2 *If a causal theory T contains a causal rule $F \Leftarrow G$ then T entails $G \supset F$.*

Proof Let I be a model of T. If $I \models G$ then $F \in T^I$, and consequently $I \models F$.

8.3 Proof of Proposition 3

Our proof of Σ_2^P-hardness is based on an embedding of disjunctive logic programs (without classical negation) into causal theories, and on the investigation of the complexity of disjunctive logic programs in [Eiter and Gottlob, 1993]. We will describe the reduction first.

Consider a Boolean signature σ. A *disjunctive logic program* over σ is a set of *(dlp) rules* of the form

$$a_1; \cdots ; a_k \leftarrow b_1, \ldots, b_m, not\ c_1, \ldots, not\ c_n \tag{70}$$

where $a_1, \ldots, a_k, b_1, \ldots, b_m, c_1, \ldots, c_n \in \sigma$, $k \geq 1$, $m, n \geq 0$. Let us identify each dlp rule (70) with the causal rule

$$b_1 \wedge \cdots \wedge b_m \supset a_1 \vee \cdots \vee a_k \Leftarrow \neg c_1 \wedge \cdots \wedge \neg c_n\,. \tag{71}$$

Let P be a disjunctive logic program over σ. If we agree to identify an interpretation I of σ with the set $\{c \in \sigma : I(c) = \mathsf{t}\}$, then we can say that I is an *answer set* for P if I is a minimal model of P^I. Although this is an unusual way of stating the definition of an answer set, its equivalence to the standard definition from [Gelfond and Lifschitz, 1991] (in the absence of classical negation) is easily verified.

Lemma *Let P be a disjunctive logic program over σ. An interpretation I of σ is an answer set for P iff I is a model of the causal theory*

$$P \cup \{\, \neg c \Leftarrow \neg c \,:\, c \in \sigma \,\}\,.$$

This lemma too is easily verified. It is similar to an embedding of disjunctive programs into default logic due to Sakama and Inoue [1993].

Proposition 3 *The problem of determining that a finite causal theory is consistent is Σ_2^P-complete.*

Proof We first establish Σ_2^P membership. Let T be a causal theory. If we know that $I \models T^I$, then it is an NP problem to determine that I is not a model of T: simply guess an interpretation J, and verify (in polynomial time) that $J \neq I$ and $J \models T^I$. Hence determining that T is consistent belongs to Σ_2^P: guess an interpretation I, verify (in polynomial time) that $I \models T^I$, and then use an NP-oracle to determine that I is a model of T.

Σ_2^P-hardness (for causal theories with Boolean signatures) follows from the lemma above, since the problem of existence of an answer set for a finite disjunctive logic program is Σ_2^P-hard [Eiter and Gottlob, 1993, Corollary 3.8].

8.4 Proof of Proposition 4

Proposition 4 *(i) Replacing a rule*

$$F \wedge G \Leftarrow H \tag{72}$$

in a causal theory by the rules

$$F \Leftarrow H,\ G \Leftarrow H \tag{73}$$

does not change its models. (ii) Replacing a rule

$$F \Leftarrow G \vee H \tag{74}$$

in a causal theory by the rules

$$F \Leftarrow G,\ F \Leftarrow H \tag{75}$$

does not change its models.

Proof Let causal theory T_r be obtained from a causal theory T by replacing a rule of form (72) with the rules (73). Take any interpretation I, and consider two cases.

Case 1: $I \models H$. Then rule (72) guarantees that $F \wedge G \in T^I$, while rules (73) guarantee that $F, G \in T_r^I$. Otherwise T^I and T_r^I are identical. Hence I is the unique model of T^I iff I is the unique model of T_r^I.

Case 2: $I \not\models H$. Then $T^I = T_r^I$.

In both cases, I is a model of T iff I is a model of T_r.

Now let T_r be obtained from a causal theory T by replacing a rule of form (74) with the rules (75). For any interpretation I, $T^I = T_r^I$, and so T and T_r have the same models.

8.5 Proof of Proposition 5

Proposition 5 *Let T_1 and T_2 be causal theories of a signature σ, such that every rule in T_2 is a constraint. An interpretation of σ is a model of $T_1 \cup T_2$ iff it is a model of T_1 and does not violate any of the constraints T_2.*

Proof Let I be an interpretation of σ. Consider two cases.

Case 1: I violates a constraint of T_2. Then $T_2^I = \{\bot\}$ and so

$$I \not\models T_1^I \cup T_2^I = (T_1 \cup T_2)^I.$$

Hence I is not a model of $T_1 \cup T_2$, which proves the claim in this case.

Case 2: I violates no constraint of T_2. Then $T_2^I = \emptyset$, so

$$T_1^I = T_1^I \cup T_2^I = (T_1 \cup T_2)^I.$$

Consequently $T_1 \cup T_2$ and T_1 have the same models, which proves the claim in this case.

8.6 Proof of Proposition 6

Proposition 6 *The models of a definite causal theory are precisely the models of its completion.*

Proof Let T be a definite causal theory. Assume that I is a model of T. It follows that I violates no constraint of T, and thus satisfies every formula in the completion of T that is obtained from a constraint. It remains to show that I satisfies the completion formula for every nontrivial atom A. Consider two cases.

Case 1: $A \in T^I$. Since T is definite and $I \models T^I$, T^I is a set of atoms true in I. So I satisfies A, which is the left-hand side of the completion formula for A. Since $A \in T^I$, there is a rule with head A whose body is true in I. Hence I also satisfies the right-hand side of the completion formula for A.

Case 2: $A \notin T^I$. So there is no rule in T with head A whose body is true in I, which shows that I doesn't satisfy the right-hand side of the completion formula for A. It remains to show that $I \not\models A$. Since T^I is a set of atoms whose unique model is I, every nontrivial atom true in I belongs to T^I. Since A is a nontrivial atom that doesn't belong to T^I, we can conclude that A is false in I.

Proof in the other direction is similar.

8.7 Proof of Proposition 7

Proposition 7 *For any transition $\langle s, e, s' \rangle$, s and s' are states.*

Proof Let $X = 0\!:\!s \cup 0\!:\!e \cup 1\!:\!s'$ be a model of D_1. We need to show that $0\!:\!s$ and $0\!:\!s'$ are models of D_0. By $i\!:\!\sigma^{fl}$ we denote the set of constants of the form $i\!:\!c$ where $c \in \sigma^{fl}$.

To show that $0\!:\!s$ is a model of D_0, observe that D_0 is the part of D_1 consisting of rules (54) for static laws with $i = 0$ and rules (56). The reduct D_0^X is a set of formulas over $0 : \sigma^{fl}$ and every formula from D_1^X with a constant from $0\!:\!\sigma^{fl}$ belongs to D_0^X. Since X is the unique model of D_1^X, we can conclude that $0\!:\!s$ is the unique model of D_0^X. But $D_0^X = D_0^{0:s}$, so that $0\!:\!s$ is a model of D_0.

Next we show that $0 : s'$ is a model of D_0. Let T be the part of D_1 consisting of rules (54) for static laws with $i = 1$, rules (54) for action dynamic laws with $i = 0$, and rules (55) with $i = 0$. Let $\Gamma = T^X$. It is straightforward to verify that Γ is a set of formulas over $1 : \sigma^{fl}$ and that every formula from D_1^X with a constant from $1 : \sigma^{fl}$ belongs to Γ. Since X is the unique model of D_1^X, we can conclude that $1 : s'$ is the unique model of Γ. Let Γ_0 be the set of formulas over $0\!:\!\sigma^{fl}$ obtained from Γ by replacing each time stamp $1:$ with $0:$. Then $0 : s'$ is the unique model of Γ_0. We need to show that $0\!:\!s'$ is also the unique model of $D_0^{0:s'}$. Observe first that every formula in $D_0^{0:s'}$ that does not belong to Γ_0 is an atom from $0\!:\!s'$ that came to the reduct from one of the rules (56) of D_0. Hence $0\!:\!s'$ satisfies $D_0^{0:s'}$. Due to the presence of rules (56) in D_0, any interpretation that satisfies $D_0^{0:s'}$ must agree with $0\!:\!s'$ on simple fluent constants. On the other hand, the formulas in Γ_0 that do not belong to $D_0^{0:s'}$ do not contain statically determined constants, because their counterparts in Γ came from the heads of dynamic laws. Consequently any interpretation that satisfies $D_0^{0:s'}$ must agree with $0 : s'$ on statically determined fluent constants. It follows that $0\!:\!s'$ is the unique model of $D_0^{0:s'}$, so that $0\!:\!s'$ is a model of D_0.

8.8 Proof of Proposition 8

Proposition 8 *For any $m > 0$, an interpretation (58) of the signature of D_m is a model of D_m iff each triple $\langle s_i, e_i, s_{i+1} \rangle$ $(0 \le i < m)$ is a transition.*

Proof We understand the notation $i\!:\!\sigma^{fl}$ as in the previous proof, and the meaning of $i\!:\!\sigma^{act}$ is similar.

Take a model X of D_m, represent it in the form (58), and take any $j \in \{0, \ldots, m-1\}$. We need to show that $0\!:\!s_j \cup 0\!:\!e_j \cup 1\!:\!s_{j+1}$ is a model of D_1.

Let T be the subset of D_m consisting of rules (54) for static laws with $i=j+1$, rules (54) for action dynamic laws with $i=j$, and rules (55) with $i=j$. Let $\Gamma = T^X$. It is straightforward to verify that Γ is a set of formulas over $j:\sigma^{act} \cup\ j+1:\sigma^{fl}$, and that every formula from D_m^X with a constant from $j:\sigma^{act} \cup\ j+1:\sigma^{fl}$ belongs to Γ. Since X is the unique model of D_m^X, it follows that $j:e_j \cup\ j+1:s_{j+1}$ is the unique model of Γ. Let Γ_0 be the set of formulas over $0:\sigma^{act} \cup\ 1:\sigma^{fl}$ obtained from Γ by replacing $j:$ with $0:$ and $j+1:$ with $1:$. Then $0:e_j \cup\ 1:s_{j+1}$ is the only interpretation of $0:\sigma^{act} \cup\ 1:\sigma^{fl}$ that satisfies Γ_0.

The proof of the previous proposition is easily adapted to show that s_j is a state, which is to say that $0:s_j$ is a model of D_0. That is, $0:s_j$ is the unique model of $D_0^{0:s_j} = D_0^{0:s_j \cup\ 0:e_j \cup\ 1:s_{j+1}}$.

It remains to observe that $D_0^{0:s_j \cup\ 0:e_j \cup\ 1:s_{j+1}} \cup\ \Gamma_0 = D_1^{0:s_j \cup\ 0:e_j \cup\ 1:s_{j+1}}$, so that $0:s_j \cup 0:e_j \cup 1:s_{j+1}$ is the unique model of this set of formulas, and, consequently, a model of D_1.

For the other direction, assume that, for each $j \in \{0, \ldots, m-1\}$, the triple $\langle s_j, e_j, s_{j+1} \rangle$ is a transition. We need to show that the corresponding interpretation X of form (58) is a model of D_m.

For each j, let T_j be the subset of D_m consisting of rules (54) for static laws with $i=j+1$, rules (54) for action dynamic laws with $i=j$, and rules (55) with $i=j$. Notice that $D_m = D_0 \cup T_0 \cup \cdots \cup T_{m-1}$. Let $\Gamma_j = T_j^X$. Of course $D_m^X = D_0^X \cup \Gamma_0 \cup \cdots \cup \Gamma_{m-1}$.

For any such j, since $\langle s_j, e_j, s_{j+1} \rangle$ is a transition, $0:s_j \cup 0:e_j \cup 1:s_{j+1}$ is the unique model of $D_1^{0:s_j\ \cup\ 0:e_j\ \cup\ 1:s_{j+1}} = D_0^{0:s_j} \cup T_0^{0:s_j\ \cup\ 0:e_j\ \cup\ 1:s_{j+1}}$. Reasoning much as before, it follows that $0:e_j \cup 1:s_{j+1}$ is the unique model $T_0^{0:s_j\ \cup\ 0:e_j\ \cup\ 1:s_{j+1}}$. This is equivalent to saying that $j:e_j \cup j+1:s_{j+1}$ is the unique model $T_j^{j:s_j\ \cup\ j:e_j\ \cup\ j+1:s_{j+1}} = \Gamma_j$.

From the previous proposition, we can conclude also that $0:s_0$ is the unique model of D_0^X.

Finally, since $D_m^X = D_0^X \cup \Gamma_0 \cup \cdots \cup \Gamma_{m-1}$, we can conclude that X is the unique model of D_m^X, and thus a model of D_m.

8.9 Proof of Proposition 9

Proposition 9 *If*

(i) every constant in R is statically determined, and

(ii) for every causal law in D that contains a constant from R in the head, all constants occurring in this law belong to R,

then the mapping $I \mapsto I^R$ is a 1-1 correspondence between the models of D_m and the models of D_m^R.

Proof Using the fact that $s'(c) = s(c)$ for every $c \in R$ and for every transition $\langle s, e, s' \rangle$, it is easy to verify that if I is a model of D_m, then I^R is a model of D_m^R. In other words, the mapping $I \mapsto I^R$ is a function from the set of models of D_m into the set of models of D_m^R. It is also easy to see that the function is 1-1. It remains to show that the function is onto.

Take any model J of D_m^R. Define the interpretation I of the signature of D_m as follows:

$$I(i\!:\!c) = \begin{cases} J(c) & \text{if } c \in R, \\ J(i\!:\!c) & \text{otherwise.} \end{cases}$$

Then

$$I(0\!:\!c) = \ldots = I(m\!:\!c) \tag{76}$$

for all $c \in R$, and $I^R = J$. We will check that I is a model of D_m, that is to say, the only model of D_m^I.

For any formula F of the signature of D_m, let F^R be the result of dropping the time stamps in front of all constants from R in F. It is easy to see that I satisfies a formula F iff J satisfies F^R. It follows that

$$(D_m^R)^J = \{F^R : F \in D_m^I\}. \tag{77}$$

Since J is a model of $(D_m^R)^J$, it follows that I is a model of D_m^I. It remains to show that D_m^I has no other models.

Take any model I_1 of D_m^I. By (77), I_1^R is a model of $(D_m^R)^J$. Since J is the only model of $(D_m^R)^J$, $I_1^R = J$. Let i_0 be a number from $\{0, \ldots, m\}$. Define the interpretation I_2 of the signature of D_m as follows:

$$I_2(i\!:\!c) = \begin{cases} I_1(i_0\!:\!c) & \text{if } c \in R, \\ I_1(i\!:\!c) & \text{otherwise.} \end{cases}$$

We want to show that I_2 is a model of D_m^I as well. Since I_1 is a model of D_m^I, I_2 satisfies all formulas from D_m^I that do not contain constants from R. Consider a formula from D_m^I that contains at least one constant from R. Since all constants in R are statically determined, this formula has the form $i : F$ for some static causal law (52) in D and some time stamp i such that I satisfies $i : G$. By condition (ii), all constants occurring in F, G belong to R. For every constant c from R, I assigns to $i : c$ and $i_0 : c$

the same value; consequently, I satisfies $i_0 : G$, so that $i_0 : F$ belongs to D^I_m. It follows that I_1 satisfies $i_0 : F$. Since I_2 does not differ from I_1 on the constants occurring in this formula, it follows that I_2 satisfies $i_0 : F$. For every constant c from R, I_2 assigns to $i : c$ and $i_0 : c$ the same value; consequently, I_2 satisfies $i : F$. We have established that I_2 indeed satisfies D^I_m.

In view of this fact, it follows from (77) that I^R_2 satisfies $(D^R_m)^J$. We can conclude that $I^R_2 = J$. So, for every $c \in R$, $I_2(0\!:\!c) = J(c)$, and consequently $I_1(i_0\!:\!c) = J(c)$. Since i_0 was arbitrary, we have proved that

$$I_1(0\!:\!c) = \ldots = I_1(m\!:\!c) \tag{78}$$

for every $c \in R$. And since $I^R = J = I^R_1$, it follows from (76) and (78) that $I_1 = I$. This shows that D^I_m has no models other than I.

8.10 Proof of Proposition 10

A *(propositional) default theory* is a set of *defaults* of the form

$$\frac{F : G_1, \ldots, G_k}{H} \tag{79}$$

where each of $F, G_1, \ldots, G_k, H$ is a classical propositional formula and $k \geq 0$. If formula F in (79) is $\top$, it is often omitted. If in addition $k = 0$, we identify the default (79) with the formula H.

Let D be a default theory and E a logically closed set of classical formulas. By D^E we denote the *reduct* of D relative to E, defined as follows.

$$D^E = \left\{ \frac{F}{H} : \text{for some default (79) in } D \text{, none of } \neg G_1, \ldots, \neg G_k \text{ belong to } E \right\}$$

E is an *extension* for D if E is the least logically closed set of classical formulas such that for every $\frac{F}{H} \in D^E$ if $F \in E$ then $H \in E$.

Where appropriate below, we identify an interpretation I with the set of formulas true in it, and we identify a causal rule $F \Leftarrow G$ with the default $\frac{:G}{F}$.

Proposition 10 *Let T be a causal theory of a Boolean signature. An interpretation I is a model of T iff I is an extension for T in the sense of default logic.*

Proof Let I be an interpretation. Notice that for any formula G, $I \models G$ iff $\neg G \notin I$. It follows that the reduct of causal theory T relative to I is the same as the reduct of default theory T relative to I. It remains to observe that I is the unique model of T^I iff I is the least logically closed set of formulas that contains T^I, that is to say, iff I is an extension for T.

8.11 Proof of Proposition 11

Here we are concerned with a different class of logic programs than in the proof of Proposition 3. These programs consist of rules of the form (68). For such programs, there is a well-known correspondence between answer sets and the extensions of a corresponding default theory, obtained by replacing each logic programming rule (68) with the default

$$\frac{: l_1, \ldots, l_n}{l_0}$$

[Gelfond and Lifschitz, 1991, Section 5]. We are interested in a special case of this correspondence. For convenience, where appropriate, we identify an interpretation with either the set of literals true in it or the set of formulas true in it. Under this convention, we can observe that an interpretation I is an answer set for a logic program whose rules have form (68) iff I is an extension for the corresponding default theory.

Proposition 11 *Let T be a causal theory whose rules have the form (67). An interpretation I is a model of T iff I is an answer set for T in the sense of logic programming.*

Proof In light of the previous observation, Proposition 11 follows from Proposition 10.

9 Conclusion: Elaboration Tolerant Representations of Action Domains

Elaboration tolerance is a property of knowledge representation formalisms described by John McCarthy [1999] as follows:

> A formalism is elaboration tolerant to the extent that it is convenient to modify a set of facts expressed in the formalism to take into account new phenomena or changed circumstances... Human-level AI will require representations with much more elaboration tolerance than those used by present AI programs,

> because human-level AI needs to be able to take new phenomena into account.
>
> The simplest kind of elaboration is the addition of new formulas. Next comes changing the values of parameters. Adding new arguments to functions and predicates represents more of a change.

Achieving a high degree of elaboration tolerance was one of our primary goals in the design of the formalisms described in this paper—the language of causal theories, action language $\mathcal{C}+$ and the input language of the Causal Calculator. If the description of an action domain in one of these languages needs to be enhanced to reflect additional effects of an action or additional restrictions on its executability, this can be accomplished by adding postulates. Any postulate can be partially or completely retracted by adding other postulates, provided that it is expressed in a defeasible form. Adding new arguments to the name of an action can be often replaced by adding postulates describing attributes of that action.

Finding ways to further increase the degree of elaboration tolerance of languages for describing actions is a topic for future work.

Acknowledgements

We are grateful to Varol Akman, Ernest Davis, Esra Erdem, Neelakantan Kartha, Leora Morgenstern, Maurice Pagnucco and the anonymous referees for useful comments. Joohyung Lee and Vladimir Lifschitz were partially supported by the National Science Foundation under Grant IIS-9732744 and by the Texas Higher Education Coordinating Board under Grant 003658-0322-2001. Hudson Turner was partially supported by the National Science Foundation under CAREER Grant 0091773.

References

[Akman *et al.*, 2003] Varol Akman, Selim Erdoğan, Joohyung Lee, Vladimir Lifschitz, and Hudson Turner. Representing the Zoo World and the Traffic World in the language of the Causal Calculator. *Artificial Intelligence*, 2003. To appear.

[Baral and Gelfond, 1997] Chitta Baral and Michael Gelfond. Reasoning about effects of concurrent actions. *Journal of Logic Programming*, 31, 1997.

[Campbell and Lifschitz, 2003] Jonathan Campbell and Vladimir Lifschitz. Reinforcing a claim in commonsense reasoning.[10] In *Working Notes of the AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, 2003.

[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.

[Davidson, 1967] Donald Davidson. The logical form of action sentences. In *The Logic of Decision and Action*, pages 81–120. University of Pittsburgh Press, 1967.

[Dimopoulos *et al.*, 1997] Yannis Dimopoulos, Bernhard Nebel, and Jana Koehler. Encoding planning problems in non-monotonic logic programs. In Sam Steel and Rachid Alami, editors, *Proc. European Conf. on Planning 1997*, pages 169–181. Springer-Verlag, 1997.

[Doherty *et al.*, 1998] Patrick Doherty, Joakim Gustafsson, Lars Karlsson, and Jonas Kvarnström. TAL: Temporal action logics language specification and tutorial.[11] *Linköping Electronic Articles in Computer and Information Science ISSN 1401-9841*, 3(015), 1998.

[Eiter and Gottlob, 1993] Thomas Eiter and Georg Gottlob. Complexity results for disjunctive logic programming and application to nonmonotonic logics. In Dale Miller, editor, *Proc. ILPS-93*, pages 266–278, 1993.

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

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

[Gelfond and Lifschitz, 1991] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

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

[10] `http://www.cs.utexas.edu/users/vl/papers/sams.ps` .
[11] `http://www.ep.liu.se/ea/cis/1998/015/` .

[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.

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

[Kautz and Selman, 1992] Henry Kautz and Bart Selman. Planning as satisfiability. In *Proc. ECAI-92*, pages 359–363, 1992.

[Kowalski and Sergot, 1986] Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[Kvarnström and Doherty, 2001] Jonas Kvarnström and Patrick Doherty. Talplanner: A temporal logic based forward chaining planner. *Annals of Mathematics and Artificial Intelligence*, 30:119–169, 2001.

[Lifschitz *et al.*, 2000] Vladimir Lifschitz, Norman McCain, Emilio Remolina, and Armando Tacchella. Getting to the airport: The oldest planning problem in AI. In Jack Minker, editor, *Logic-Based Artificial Intelligence*, pages 147–165. Kluwer, 2000.

[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.

[Lifschitz, 2002] Vladimir Lifschitz. Answer set programming and plan generation. *Artificial Intelligence*, 138:39–54, 2002.

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

[Lin, 1996] Fangzhen Lin. Embracing causality in specifying the indeterminate effects of actions. In *Proc. AAAI-96*, pages 670–676, 1996.

[Marek and Truszczyński, 1999] Victor Marek and Mirosław Truszczyński. Stable models and an alternative logic programming paradigm. In *The Logic Programming Paradigm: a 25-Year Perspective*, pages 375–398. Springer Verlag, 1999.

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

[McCain, 1997] Norman McCain. *Causality in Commonsense Reasoning about Actions.*[12] PhD thesis, University of Texas at Austin, 1997.

[McCarthy and Hayes, 1969] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh, 1969.

[McCarthy, 1959] John McCarthy. Programs with common sense. In *Proc. Teddington Conf. on the Mechanization of Thought Processes*, pages 75–91, London, 1959. Her Majesty's Stationery Office.

[McCarthy, 1980] John McCarthy. Circumscription—a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39,171–172, 1980. Reproduced in [McCarthy, 1990].

[McCarthy, 1986] John McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 26(3):89–116, 1986. Reproduced in [McCarthy, 1990].

[McCarthy, 1990] John McCarthy. *Formalizing Common Sense: Papers by John McCarthy.* Ablex, Norwood, NJ, 1990.

[McCarthy, 1999] John McCarthy. Elaboration tolerance.[13] In progress, 1999.

[Niemelä, 1999] Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25:241–273, 1999.

[Pednault, 1987] Edwin Pednault. Formulating multi-agent, dynamic world problems in the classical planning framework. In Michael Georgeff and Amy Lansky, editors, *Reasoning about Actions and Plans*, pages 47–82. Morgan Kaufmann, San Mateo, CA, 1987.

[Pednault, 1989] Edwin Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In Ronald Brachman, Hector Levesque, and Raymond Reiter, editors, *Proc. First Int'l Conf. on Principles of Knowledge Representation and Reasoning*, pages 324–332, 1989.

[12] `ftp://ftp.cs.utexas.edu/pub/techreports/tr97-25.ps.Z` .

[13] `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.

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

[Reiter, 1991] Raymond Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, 1991.

[Sakama and Inoue, 1993] Chiaki Sakama and Katsumi Inoue. Relating disjunctive logic programs to default theories. In Luis Moniz Pereira and Anil Nerode, editors, *Logic Programming and Non-monotonic Reasoning: Proceedings of the Second Int'l Workshop*, pages 166–282, 1993.

[Sandewall, 1994] Erik Sandewall. *Features and Fluents*, volume 1. Oxford University Press, 1994.

[Shanahan, 1997] Murray Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia.* MIT Press, 1997.

[Thielscher, 1997] Michael Thielscher. Ramification and causality. *Artificial Intelligence*, 89(1–2):317–364, 1997.

[Turner, 1996] Hudson Turner. Splitting a default theory. In *Proc. AAAI-96*, pages 645–651, 1996.

[Turner, 1997] Hudson Turner. Representing actions in logic programs and default theories: a situation calculus approach. *Journal of Logic Programming*, 31:245–298, 1997.

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

A Reducing Multi-Valued Formulas to Classical Formulas

If we disregard the internal structure of atoms $c = v$ of a multi-valued propositional signature σ (Section 2.1) then any formula of this signature becomes a classical formula, and it can be evaluated relative to any truth-valued function on the set of atoms. To find a model of a set X of formulas of signature σ means to find a truth-valued function I on the set of atoms that is a model of X in the sense of classical propositional logic and has the following additional property: for each $c \in \sigma$ there exists a unique $v \in Dom(c)$ such that $I(c{=}v) = \mathbf{t}$. This property can be expressed by saying that I satisfies the formulas

$$\bigvee_v c{=}v \;\wedge\; \bigwedge_{v \neq w} \neg(c{=}v \wedge c{=}w) \tag{80}$$

for all $c \in \sigma$. Consequently, to find a model of X means to find a model of the union of X with formulas (80) in the sense of classical propositional logic.

If some constants in σ are Boolean then the translation to classical propositional logic can be made more economical: for a Boolean constant c, we can replace every occurrence of $c = \mathbf{f}$ with $\neg c$, drop the corresponding formula (80), and then drop $c = \mathbf{f}$ from the set of atoms. For instance, the models of (17) can be computed without adding any of the formulas (80) if we rewrite (17) in the form

$$\begin{array}{l} p \equiv q, \\ \neg p \equiv \bot, \\ q \equiv q, \\ \neg q \equiv \neg q. \end{array}$$

A similar simplification can be used for any constant with a two-element domain.

The number of conjunctive terms

$$\neg(c{=}v \wedge c{=}w) \qquad (v \neq w) \tag{81}$$

in formula (80) is quadratic in the cardinality of $Dom(c)$, which makes this formula computationally unattractive when the domain of c is large. Instead of (81), we can use a definitional extension of this set of formulas whose size is linear in the number of elements $v_1, \dots, v_n$ of $Dom(c)$:

$$\begin{array}{ll} p_1 \equiv c{=}v_1 & \qquad \neg(p_1 \wedge c{=}v_2) \\ p_2 \equiv p_1 \vee c{=}v_2 & \qquad \neg(p_2 \wedge c{=}v_3) \\ \cdots & \qquad \cdots \\ p_{n-1} \equiv p_{n-2} \vee c{=}v_{n-1} & \qquad \neg(p_{n-1} \wedge c{=}v_n) \end{array}$$

where $p_1, \dots, p_{n-1}$ are new Boolean constants.

B Abbreviations for Causal Laws

1. A static law or an action dynamic law of the form

$$\textbf{caused } F \textbf{ if } \top$$

can be written as

$$\textbf{caused } F.$$

2. A fluent dynamic law of the form

$$\textbf{caused } F \textbf{ if } \top \textbf{ after } H$$

can be written as

$$\textbf{caused } F \textbf{ after } H.$$

3. A static law of the form

$$\textbf{caused } \bot \textbf{ if } \neg F$$

can be written as

$$\textbf{constraint } F.$$

4. A fluent dynamic law of the form

$$\textbf{caused } \bot \textbf{ if } \neg F \textbf{ after } G$$

can be written as

$$\textbf{constraint } F \textbf{ after } G.$$

5. An expression of the form

$$\textbf{rigid } c$$

where c is a fluent constant stands for the set of causal laws

$$\textbf{constraint } c = v \textbf{ after } c = v$$

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

6. A fluent dynamic law of the form

$$\textbf{caused } \bot \textbf{ after } \neg F$$

can be written as

$$\textbf{always } F.$$

7. A fluent dynamic law of the form

$$\textbf{caused } \bot \textbf{ after } F \wedge G$$

where F is an action formula can be written as

$$\textbf{nonexecutable } F \textbf{ if } G. \tag{82}$$

8. An expression of the form

$$F \textbf{ causes } G \textbf{ if } H \tag{83}$$

where F is an action formula stands for the fluent dynamic law

$$\textbf{caused } G \textbf{ after } F \wedge H$$

if G is a fluent formula,[14] and for the action dynamic law

$$\textbf{caused } G \textbf{ if } F \wedge H$$

if G is an action formula.

9. An expression of the form

$$\textbf{default } F \textbf{ if } G \tag{84}$$

stands for the causal law

$$\textbf{caused } F \textbf{ if } F \wedge G.$$

10. An expression of the form

$$\textbf{default } F \textbf{ if } G \textbf{ after } H$$

stands for the fluent dynamic law

$$\textbf{caused } F \textbf{ if } F \wedge G \textbf{ after } H.$$

The part

$$\textbf{if } G$$

[14]It is clear that the expression in the previous line is a fluent dynamic law only when G does not contain statically determined constants. Similar remarks can be made in connection with many of the abbreviations introduced below.

in this abbreviation can be dropped if G is $\top$.

11. An expression of the form

$$\textbf{exogenous } c \textbf{ if } G \tag{85}$$

where c is a constant stands for the set of causal laws

$$\textbf{default } c{=}v \textbf{ if } G$$

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

12. An expression of the form

$$F \textbf{ may cause } G \textbf{ if } H \tag{86}$$

where F is an action formula stands for the fluent dynamic law

$$\textbf{default } G \textbf{ after } F \wedge H$$

if G is a fluent formula, and for the action dynamic law

$$\textbf{default } G \textbf{ if } F \wedge H$$

if G is an action formula.

13. An expression of the form

$$\textbf{inertial } c \textbf{ if } G \tag{87}$$

where c is a fluent constant stands for the set of fluent dynamic laws

$$\textbf{default } c{=}v \textbf{ after } c{=}v \wedge G$$

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

14. If any of the abbreviations (83)–(87) ends with

$$\textbf{if } \top$$

then this part of the expression can be dropped.

15. The expression obtained by appending

$$\textbf{unless } c$$

where c is a Boolean statically determined fluent constant to a static law

$$\textbf{caused } F \textbf{ if } G \tag{88}$$

stands for the pair of static laws

$$\begin{array}{l}\textbf{caused } F \textbf{ if } G \wedge \neg c,\\ \textbf{default } \neg c.\end{array} \tag{89}$$

16. The expression obtained by appending

$$\textbf{unless } c$$

where c is a Boolean action constant to an action dynamic law (88) stands for the pair of action dynamic laws (89).

17. The expression obtained by appending

$$\textbf{unless } c$$

where c is a Boolean action constant to a fluent dynamic law

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

stands for the pair of dynamic laws

$$\begin{array}{l}\textbf{caused } F \textbf{ if } G \textbf{ after } H \wedge \neg c,\\ \textbf{default } \neg c.\end{array}$$