A Logic Program Characterization of Causal Theories*

Paolo Ferraris
Department of Computer Sciences
University of Texas at Austin
1 University Station C0500
Austin, TX 78705
otto@cs.utexas.edu

Abstract

Nonmonotonic causal logic, invented by McCain and Turner, is a formalism well suited for representing knowledge about actions, and the definite fragment of that formalism has been implemented in the reasoning and planning system called CCalc. A 1997 theorem due to McCain shows how to translate definite causal theories into logic programming under the answer set semantics, and thus opens the possibility of using answer set programming for the implementation of such theories. In this paper we propose a generalization of McCain's theorem that extends it in two directions. First, it is applicable to arbitrary causal theories, not only definite. Second, it covers causal theories of a more general kind, which can describe non-Boolean fluents.

1 Introduction

Causal logic [McCain and Turner, 1997] is a formalism for knowledge representation, especially suited for representing effects of actions. Causal theories are syntactically simple but also very general: they consist of causal rules of the form

$$F \Leftarrow G \tag{1}$$

where F and G are propositional formulas. Intuitively, rule (1) says that there is a cause for F to be true if G is true. For instance, the causal rule

$$p_{t+1} \Leftarrow a_t \tag{2}$$

can be used to describe the effect of an action a on a Boolean fluent p: if a is executed at time t then there is a cause for p to hold at time $t+1$. Other important concepts in commonsense reasoning can be easily expressed by rules of this kind too. For instance, a rule of the form

$$F \Leftarrow F$$

("if F is true then there is a cause for this") expresses, intuitively, that F is true by default. In particular, the causal rule

$$p_t \Leftarrow p_t \tag{3}$$

says that Boolean fluent p is normally true. The frame problem [McCarthy and Hayes, 1969] is solved in causal logic using the rules

$$\begin{aligned} p_{t+1} &\Leftarrow p_t \wedge p_{t+1} \\ \neg p_{t+1} &\Leftarrow \neg p_t \wedge \neg p_{t+1}. \end{aligned} \tag{4}$$

These rules express inertia: if a fluent p is true (false) at time t then normally it remains true (false) at time $t+1$.

The equivalence of two fluents or actions can be expressed by equivalences in the head. For instance, to express that two actions constants a and a' are synonymous, we can use causal rule

$$a_t \equiv a'_t \Leftarrow \top. \tag{5}$$

Rules of those kind are used to semantically characterize the relationship between modules in the Modular Action Description language MAD [Lifschitz and Ren, 2006]. For instance, [Erdoğan and Lifschitz, 2006; Erdoğan *et al.*, 2007] use an equivalence of this kind to state that pushing the box in the Monkey and Bananas domain is a specialization of a more general action "move".

The language of causal theories has been extended in [Giunchiglia *et al.*, 2004] to handle multi-valued constants, in which a constant may assume values different from true and false. For instance, we can express the fact that object x is in location l with $loc(x) = l$. One advantage of using $loc(x) = l$ instead of the Boolean fluent $loc(x, l)$ is that $loc(x) = l$ implicitly expresses the commonsense fact that every object is exactly in one position.

In many useful causal rules, such as (2)–(4), the formula before the "$\Leftarrow$" is a Boolean literal, a non-Boolean atom (such as $loc(x) = l$) or $\bot$. Rules of this kinds are called definite. Such rules are important because a causal theory consisting of definite rules can be converted into an equivalent set of propositional formulas [McCain and Turner, 1997; Giunchiglia *et al.*, 2004], so that its models can be computed using a satisfiability solver. That translation is used in an implementation of the definite fragment of causal logic, called the Causal Calculator, or CCALC.[1] The Causal Calculator has been applied to several problems in the theory of commonsense reasoning [Lifschitz *et al.*, 2000; Lifschitz, 2000; Akman *et al.*, 2004; Campbell and Lifschitz, 2003; Lee and Lifschitz, 2006].

*This research was partially supported by the National Science Foundation under Grant IIS-0412907.

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

On the other hand, a rule of the form (5) is not definite, so that the method of computing models described above is not applicable in the presence of such rules.

Another method of computing the models of a causal theory uses converting it into a logic program under the answer set semantics [Gelfond and Lifschitz, 1988; 1991]. This translation, discovered by McCain [1997], is modular (i.e., can be applied to a causal theory rule-by-rule). It converts, for instance, causal rule (3) into

$$p_t \leftarrow \mathit{not}\ \neg p_t,$$

which is the usual way of expressing that p_t is true by default in logic programs [Gelfond and Lifschitz, 1991, Section 3]. Answer set solvers — systems that find the answer sets for logic programs — can then be used to find the models of definite causal theories [Doğandağ *et al.*, 2001]. On the other hand, this translation is only applicable to Boolean causal theories consisting of definite rules.

In this paper we propose a translation that is more general than McCain's: it is applicable to arbitrary causal theories, including multi-valued theories containing nondefinite rules. The price that we pay for this generality is that the logic programs produced by our translation are programs with nested expressions. Such programs, defined in [Lifschitz *et al.*, 1999], can be converted into disjunctive programs in polynomial time at the price of introducing additional atoms [Pearce *et al.*, 2002]; this process has been implemented.[2] Consequently, our translation may allow us to compute the models of arbitrary causal theories using answer set solvers applicable to disjunctive programs, such as DLV [3], GNT [4] and CMODELS, [5].

Our translation, like the one due to McCain, is modular. It is defined in two steps. First, every rule of the given causal theory is converted into a set of rules in clausal form

$$l_1 \vee \cdots \vee l_n \Leftarrow G \tag{6}$$

where each l_i is a literal, using [Giunchiglia *et al.*, 2004, Proposition 4]. Then each rule (6) is replaced by a logic program with nested expressions; we will see that this program is "small" – usually linear in the size of the input.

In addition, we show how both steps can be done more efficiently. First of all, the "clausification" of a causal theory can be done without much increase in size if we agree to introduce auxiliary atoms. About converting a clausified causal theory into a logic program, we will show that many rules and atoms can be dropped in special cases. For instance, in case of causal theories with Boolean signature, we get a much simpler definition of the translation. Our optimizations also allow us to translate causal theories whose heads are literals or $\bot$ into nondisjunctive logic programs. It follows that, for this class of causal theories, the problem of the existence of a model is in class NP.

We review the syntax and semantics of causal theories and logic programs in Sections 2 and 3, respectively. The transformation from causal theories in clausal form into logic program is shown in Section 4, and Section 5 describes how to make it more compact. Clausifying causal theories is discussed in Section 6.

[2] http://www.cs.uni-potsdam.de/~torsten/nlp/

[3] http://www.dbai.tuwien.ac.at/proj/dlv/ .

[4] http://www.tcs.hut.fi/Software/gnt/ .

[5] http://www.cs.utexas.edu/~tag/cmodels/ .

2 Causal Theories

We review the more general syntax and semantics of causal theories — which allow multi-valued constants — from [Giunchiglia *et al.*, 2004].

A *(multi-valued) signature* is a set σ of symbols c, called *constants*, with a set of symbols $Dom(c)$ (the *domain* of c) associated to each of them. A *(multi-valued) atom* is a string of the form $c = v$, where $c \in \sigma$ and $v \in Dom(c)$. A *(multi-valued) formula* is built from atoms using the connectives $\wedge$, $\vee$, $\neg$, $\top$ and $\bot$. Formulas of the forms $F \supset G$ and $F \equiv G$ can be seen as abbreviations in the usual way.

A *(multi-valued) causal rule* is an expression of the form $F \Leftarrow G$, where F and G are formulas. These formulas are called the *head* and the *body* of the rule respectively. A *(multi-valued) causal theory* is a set of causal rules.

A *(multi-valued) interpretation* over σ is a (total) function that maps each constant c of σ to an element of $Dom(c)$. An interpretation I *satisfies* (or is a *model of*) an atom $c = v$ if $I(c) = v$. The definition of satisfaction and model of formulas of more general form follows the usual rules of propositional logic.

The semantics of causal theories of [Giunchiglia *et al.*, 2004] defines when an interpretation I of σ is a model of a causal theory T, as follows. The *reduct* T^I of T relative to I is the set of the heads of the rules of T whose bodies are satisfied by I. We say that I is a *model* of T if I is the only model of T^I. It is clear that replacing the head or the body of a causal rule by an equivalent formula doesn't change the models of a causal theory.

Take, for instance, the following causal theory with $Dom(c) = \{1, 2, 3\}$:

$$\begin{aligned} \neg(c = 1) \vee c = 2 &\Leftarrow \top \\ \neg(c = 2) \vee c = 1 &\Leftarrow \top, \end{aligned} \tag{7}$$

The reduct relative to any I is always

$$\{\neg(c = 1) \vee c = 2, \neg(c = 2) \vee c = 1\},$$

which is equivalent to $c = 1 \equiv c = 2$. The only model of the reduct is the interpretation J such that $J(c) = 3$. It is then clear that J is a model of (7), while no other interpretation I is a model of this causal theory because I is not a model of the reduct.

A *literal* is either an atom a or its negation $\neg a$. A rule of the form (6), where $n \geq 0$ and $l_1, \ldots, l_n$ are literals, is said to be in *clausal form*. It is also *semi-definite* if $n \leq 1$, and *definite* if either the head is $\bot$ ($n = 0$) or an atom. A causal theory is *in clausal form* (*semi-definite*, *definite*) if all its rules are in clausal form (respectively *semi-definite*, *definite*).

A constant c is *binary* if $|Dom(c)| = 2$. It is also *Boolean* if $Dom(c) = \{\mathbf{t}, \mathbf{f}\}$. Signatures, formulas, causal rules and causal theories are *binary* (*Boolean*), if they contain binary (respectively, Boolean) constants only. In case of a binary signature, the difference between definite and semi-definite

causal rules is not essential, because every negative literal can be rewritten as an atom. For instance, if the underlying signature is Boolean then $\neg(c = \mathbf{t})$ is equivalent to $c = \mathbf{f}$. In case of Boolean constants c, we will often write $c = \mathbf{t}$ simply as c. If a causal theory of a Boolean signature doesn't contain atoms of the form $c = \mathbf{f}$ then the heads and bodies of its rules are essentially classical, as in the original definition of causal theories [McCain and Turner, 1997]. We call such theories MCT theories.

Take, for instance, the following MCT theory T of signature $\{p, q\}$:

$$\begin{aligned} p \vee \neg q &\Leftarrow \top \\ q &\Leftarrow p. \end{aligned} \tag{8}$$

The interpretation I defined by $I(p) = I(q) = \mathbf{t}$ is a model of T. Indeed, in this case $T^I = \{p \vee \neg q, q\}$, and its only model is I. No other interpretation is a model of T: if $I(p) = \mathbf{t}$ and $I(q) = \mathbf{f}$ then I is not a model of the reduct $T^I = \{p \vee \neg q, q\}$, while if $I(p) = \mathbf{f}$ then the reduct $T^I = \{p \vee \neg q\}$ has more than one model.

3 Logic programs

The answer set semantics was originally defined in [Gelfond and Lifschitz, 1988] for logic programs of a very simple form and has been generalized several times. Here we review the syntax and semantics of programs with nested expressions [Lifschitz *et al.*, 1999]. In this section, the words "atom" and "literal" are understood as in classical propositional logic. A *nested expression* is built from literals using the 0-place connectives $\top$ and $\bot$, the unary connective "*not*" (negation as failure) and the binary connectives "," (conjunction) and ";" (disjunction).

A *logic program rule (with nested expressions)* has the form

$$F \leftarrow G$$

where F and G are nested expressions. As in causal rules, F is called the *head* of the rule and G its *body*. Finally, a *logic program (with nested expressions)* is a set of logic program rules.

The answer set semantics defines when a consistent set of literals (a set that doesn't contain both a and $\neg a$ for the same atom a) is an answer set for a logic program. In the rest of this section X stands for a consistent set of literals, l for a literal, F and G for nested expressions and Π for a logic program.

We define when X *satisfies* F (symbolically, $X \models F$) recursively as follows:

- $X \models l$ if $l \in X$,
- $X \models \top$ and $X \not\models \bot$,
- $X \models \mathit{not}\ F$ if $X \not\models F$,
- $X \models F, G$ if $X \models F$ and $X \models G$, and
- $X \models F; G$ if $X \models F$ or $X \models G$.

Finally, X satisfies Π ($X \models \Pi$) if, for all rules $F \leftarrow G$ in Π, $X \models F$ whenever $X \models G$.

The *reduct* Π^X of Π relative to X is the result of replacing every maximal subexpression of Π that has the form $\mathit{not}\ F$ with $\bot$ if $X \models F$, and with $\top$ otherwise. A set X is an *answer set* for Π if X is a minimal set (in the sense of set inclusion) satisfying Π^X.

An expression of the form $F \leftrightarrow G$ will stand for two rules $F \leftarrow G$ and $G \leftarrow F$.

4 Main translation

Consider a multi-valued signature σ. For any formula F of that signature, we define F^{ne} as the nested expression obtained from F by replacing each $\wedge$ with a comma, $\vee$ with a semicolon and $\neg$ with *not*.

We are now ready to define our translation. Given any causal theory T in clausal form, we define Π_T as the program with nested expressions obtained from T

- by replacing each causal rule (6) by

$$l_1; \ldots; l_n \leftarrow \mathit{not\ not}\ G^{ne}, (\overline{l_1}; \mathit{not}\ \overline{l_1}), \ldots, (\overline{l_n}; \mathit{not}\ \overline{l_n}) \tag{9}$$

  where each $\overline{l_i}$ stands for the literal complementary to l_i, and

- by adding, for every constant $c \in \sigma$ and every distinct $v, v' \in Dom(c)$, rules

$$c = v \leftrightarrow \bigsqcup_{w \in Dom(c) \setminus \{v\}} \neg(c = w) \tag{10}$$

$$\neg(c = v); \neg(c = v') \leftarrow \mathit{not}\ (c = v), \mathit{not}\ (c = v') \tag{11}$$

  where the "big comma" is used in the same way as big conjunctions.

According to this definition, each rule (9) of Π_T can be obtained from the corresponding rule of T in three steps: by

- replacing each propositional connective with the corresponding "logic program connective", with the exception of negation in the head,
- prepending *not not* to the body of the rule, and
- adding some "excluded middle hypotheses" to the body of the rule.

This last step "compensates" the replacement of $\vee$ with the corresponding "stronger" logic program connective. It is clear that this translation is linear if there is an upper bound on the size of the domain for each constant in T (for instance, when T is binary).

Rules (10) and (11) relate literals containing the same constant. They are needed to establish a 1–1 relationship between the models of T^I and the subsets of I that satisfy Π_T^I.

For instance, if T is (7) then Π_T is

$$\begin{aligned} &\neg(c = 1); c = 2 \leftarrow \mathit{not\ not}\ \top, (c = 1; \mathit{not}\ (c = 1)), \\ &\qquad\qquad (\neg(c = 2); \mathit{not}\ \neg(c = 2)) \\ &\neg(c = 2); c = 1 \leftarrow \mathit{not\ not}\ \top, (c = 2; \mathit{not}\ (c = 2)), \\ &\qquad\qquad (\neg(c = 1); \mathit{not}\ \neg(c = 1)) \\ &\qquad c = 1 \leftrightarrow \neg(c = 2), \neg(c = 3) \\ &\qquad c = 2 \leftrightarrow \neg(c = 1), \neg(c = 3) \\ &\qquad c = 3 \leftrightarrow \neg(c = 1), \neg(c = 2) \\ &\neg(c = 1); \neg(c = 2) \leftarrow \mathit{not}\ (c = 1), \mathit{not}\ (c = 2) \\ &\neg(c = 1); \neg(c = 3) \leftarrow \mathit{not}\ (c = 1), \mathit{not}\ (c = 3) \\ &\neg(c = 2); \neg(c = 3) \leftarrow \mathit{not}\ (c = 2), \mathit{not}\ (c = 3). \end{aligned} \tag{12}$$

If T is (8) then Π_T is

$$\begin{aligned} p; \neg q \leftarrow \mathit{not\ not}\ \top, (\neg p; \mathit{not}\ \neg p), (q; \mathit{not}\ q) \\ q \leftarrow \mathit{not\ not}\ p, (\neg q; \mathit{not}\ \neg q) \\ p \leftrightarrow \neg(p = \mathbf{f}) \\ p = \mathbf{f} \leftrightarrow \neg p \\ q \leftrightarrow \neg(q = \mathbf{f}) \\ q = \mathbf{f} \leftrightarrow \neg q \\ \neg p; \neg(p = \mathbf{f}) \leftarrow \mathit{not}\ p, \mathit{not}\ (p = \mathbf{f}) \\ \neg q; \neg(q = \mathbf{f}) \leftarrow \mathit{not}\ q, \mathit{not}\ (q = \mathbf{f}). \end{aligned} \tag{13}$$

The theorem below expresses the soundness of this translation. We identify each interpretation with the (complete) set of literals over σ that are satisfied by the interpretation.

Theorem 1 *For any causal theory T in clausal form, the models of T are identical to the answer sets for Π_T.*

For instance, the only answer set for (12) is $\{\neg(c = 1), \neg(c = 2), c = 3\}$, and indeed it is the only model of (7). The only answer set for (13) is $\{p, \neg(p = \mathbf{f}), q, \neg(q = \mathbf{f})\}$, which is the only model of (8).

For each causal rule (6) that has the form $l_1 \Leftarrow G$ (i.e., $n = 1$), we can drop the "excluded middle hypothesis" from the corresponding rule (9) of Π_T. Two logic programs Π_1 and Π_2 are *strongly equivalent* if, for any program Π, $\Pi_1 \cup \Pi$ has the same answer sets of $\Pi_2 \cup \Pi$ [Lifschitz *et al.*, 2001].

Proposition 1 *For any literal l and any nested expression F, the one-rule logic program*

$$l \leftarrow F, \left(\overline{l}; \mathit{not}\ \overline{l}\right)$$

is strongly equivalent to

$$l \leftarrow F.$$

For instance, the second rule of (13) can be rewritten as

$$q \leftarrow \mathit{not\ not}\ p$$

and the answer sets don't change.

However, dropping terms of the form $\overline{l_i}; \mathit{not}\ \overline{l_i}$ from (9) is usually not sound when $n > 1$. Take, for instance, the one-rule MCT causal theory:

$$p \vee \neg p \Leftarrow \top,$$

which has no models. As we expect, the corresponding logic program Π_T:

$$\begin{aligned} p; \neg p \leftarrow & \mathit{not\ not}\ \top, (\neg p; \mathit{not}\ \neg p), (p; \mathit{not}\ p) \\ p \leftrightarrow & \neg(p = \mathbf{f}) \\ p = \mathbf{f} \leftrightarrow & \neg p \\ \neg p; \neg(p = \mathbf{f}) \leftarrow & \mathit{not}\ p, \mathit{not}\ (p = \mathbf{f}) \end{aligned} \tag{14}$$

has no answer sets. If we drop the two disjunctions in the body of the first rule of (14) we get a logic program with two answer sets $\{p, \neg(p = \mathbf{f})\}$ and $\{\neg p, p = \mathbf{f}\}$ instead.

5 Reducing the translation

Our simplification of Π_T depends on two parameters:

- a set S of atoms of σ such that every atom occurring in T belongs to S, and
- a set C of constants of σ such that every rule of T containing a constant from C in the head is semidefinite.

For each constant c, let N_c denote the number of atoms containing c that do not occur in S. We define the logic program $\Delta_T(S, C)$ as obtained from Π_T by:

- dropping all rules (11) such that $c \in C$ or $\{c = v, c = v'\} \not\subseteq S$,
- replacing, for each constant c such that $N_c > 0$, rules (10) with the set of rules

$$\underset{w\,:\,c=w \in S, w \neq v}{,} \neg(c = w) \leftarrow c = v \tag{15}$$

  for all $v \in Dom(c)$ such that $c = v \in S$, and
- adding

$$\bot \leftarrow \underset{w\,:\,c=w \in S}{,} \mathit{not}\ (c = w) \tag{16}$$

  for each constant c such that $N_c > 1$.

We will denote $\Delta_T(S, \emptyset)$ by simply $\Delta_T(S)$. We can easily notice that $\Delta_T(S, \emptyset)$ contains atoms from S only. Clearly, when S contains all atoms, $\Delta_T(S) = \Pi_T$. Taking S smaller and C larger makes $\Delta_T(S, C)$ contain less and generally simpler rules.

Rules (15) impose a condition similar to the left-to-right half of (10), but they are limited to atoms of S. Rule (16) expresses, in the translation, the following fact about causal theories: if neither of two distinct atoms $c = v_1$ and $c = v_2$ occurs in a causal theory T then no model of T maps c to either v_1 or v_2. For instance, if $Dom(c) = \{1, 2, 3\}$ and only $c = 1$ occurs in T then every model of T maps c to 1. However, if $c = 2$ occurs in T as well then c can be mapped to 3, as shown by example (7).

For instance, if T is (7) then $\Delta_T(\{c = 1, c = 2\})$ is

$$\begin{aligned} \neg(c = 1); c = 2 \leftarrow & \mathit{not\ not}\ \top, (c = 1; \mathit{not}\ (c = 1)), \\ & (\neg(c = 2); \mathit{not}\ \neg(c = 2)) \\ \neg(c = 2); c = 1 \leftarrow & \mathit{not\ not}\ \top, (c = 2; \mathit{not}\ (c = 2)), \\ & (\neg(c = 1); \mathit{not}\ \neg(c = 1)) \\ \neg(c = 2) \leftarrow & c = 1 \\ \neg(c = 1) \leftarrow & c = 2 \\ \neg(c = 1); \neg(c = 2) \leftarrow & \mathit{not}\ (c = 1), \mathit{not}\ (c = 2) \end{aligned} \tag{17}$$

If S is a set of atoms, a subset of $\{a, \neg a : a \in S\}$ is *complete over S* if it contains exactly one of the two literals a or $\neg a$ for each $a \in S$.

Theorem 2 *Let T be a causal theory over σ. Let S be a set of atoms of σ such that every atom occurring in T belongs to S, and let C be a set of constants of σ such that every rule of T containing a constant from C in the head is semidefinite. Then $I \mapsto I \cap \{a, \neg a : a \in S\}$ is a 1–1 correspondence between the models of T and the answer sets of $\Delta_T(S, C)$ that are complete over S.*

We get the models of the original causal theory by looking at the unique interpretation that satisfies each complete answer set for $\Delta_T(S, C)$. (The uniqueness of the interpretation is guaranteed by the theorem.) For instance, $\{\neg(c = 1), \neg(c = 2)\}$ is the only complete answer set for (17); it corresponds to the interpretation that maps c to 3, and this is indeed the only model of (7).

Program $\Delta_T(S, C)$ may have incomplete answer sets, and those don't correspond to any interpretation. They can be eliminated from $\Delta_T(S, C)$ by adding the constraint

$$\bot \leftarrow \mathit{not}\ a, \mathit{not}\ \neg a \tag{18}$$

for every $a \in S$.

We can notice that no constant $c \in C$ occurs in the head of "intrinsically disjunctive" rules of $\Delta_T(S, C)$. Indeed, if $c \in C$ then each rule (9) with c in the head is nondisjunctive because it comes from a semi-definite causal rule, and $\Delta_T(S, C)$ doesn't contain rules (11) whose head contains c. Moreover, rules (10) and (16) can be strongly equivalently rewritten as nondisjunctive rules. In particular, it is possible to translate semi-definite causal theories into nondisjunctive programs of about the same size. As a consequence, the problem of the existence of a model of a semi-definite causal theory is in class NP.

When, for a binary constant c, only one of the two atoms belongs to S, all rules (10) and (11) in Π_T for such constant c are replaced in $\Delta_T(S, C)$ by a single rule (15) whose head is $\top$, which can be dropped. In particular, an MCT theory T over σ can be translated into logic program $\Delta_T(\sigma)$, basically consisting just of rules (9) for all rules (6) in T.

For instance, if T is (8) then $\Delta_T(\{p, q\})$ is

$$p; \neg q \leftarrow \mathit{not}\ \mathit{not}\ \top, (\neg p; \mathit{not}\ \neg p), (q; \mathit{not}\ q)$$
$$q \leftarrow \mathit{not}\ \mathit{not}\ p, (\neg q; \mathit{not}\ \neg q)$$

whose only complete answer set is $\{p, q\}$ as expected.

6 Clausifying a causal theory

As we mentioned in the introduction, the translations from the previous sections can also be applied to arbitrary causal theories, by first converting them into clausal form. One way to do that is by rewriting the head of each rule in conjunctive normal form, and then by breaking each rule

$$C_1 \wedge \cdots \wedge C_n \Leftarrow G, \tag{19}$$

where $C_1, \ldots, C_n$ $(n \geq 0)$ are clauses, into n rules

$$C_i \Leftarrow G \tag{20}$$

$(i = 1, \ldots, n)$ [Giunchiglia *et al.*, 2004, Proposition 4]. However, this reduction may lead to an exponential increase in size unless we assume an upper bound on the number of atoms that occur in the head of each single rule.

We propose a reduction from an arbitrary causal theory to a causal theory where the head of each rule has at most three atoms. This translation can be computed in polynomial time and requires the introduction of auxiliary Boolean atoms. The translation is similar to the one for logic programs from [Pearce *et al.*, 2002] mentioned in the introduction.

We denote each auxiliary atom by d_F, where F is a formula. For any causal theory T, the causal theory T' is obtained by T by

- replacing the head of each rule $F \Leftarrow G$ in T by d_F, and
- adding, for each subformula F that occurs in the head of rules of T, ($\otimes$ denotes a binary connective)
  - $d_F \equiv F \Leftarrow \top$, if F is an atom, $\top$ and $\bot$,
  - $d_F \equiv \neg d_G \Leftarrow \top$, if F has the form $\neg G$, and
  - $d_F \equiv d_G \otimes d_H \Leftarrow \top$, if F has the form $G \otimes H$.

Intuitively, the equivalences in the heads of the rules above recursively define each atom d_F occurring in T' to be equivalent to F. This translation is clearly modular.

If T is an MCT theory then T' is an MCT theory also. For instance, MCT rule

$$p \vee (q \wedge \neg r) \Leftarrow r$$

is transformed into the following 7 MCT rules:

$$\begin{aligned} d_{p\vee(q\wedge\neg r)} &\Leftarrow r \\ d_{p\vee(q\wedge\neg r)} \equiv d_p \vee d_{q\wedge\neg r} &\Leftarrow \top \\ d_{q\wedge\neg r} \equiv d_q \wedge d_{\neg r} &\Leftarrow \top \\ d_{\neg r} \equiv \neg d_r &\Leftarrow \top \\ d_a \equiv a &\Leftarrow \top \qquad (a \in \{p, q, r\}) \end{aligned}$$

Theorem 3 *For any causal theory T over a signature σ, $I \mapsto I|_\sigma$ is a 1–1 correspondence between the models of T' and the models of T.*

We can see that every rule in causal theories of the form T' is either already in clausal form, or has the body $\top$ and at most three atoms in the head. It is not hard to see that the clausification process described at the beginning of the section is linear when applied to T'.

7 Related work and conclusions

McCain's translation [McCain and Turner, 1997] is a linear and modular translation applicable to (semi-)definite MCT theories T (over σ) whose bodies are conjunction of literals. Our translation $\Delta_T(\sigma)$ is similar to McCain's translation for causal theories of those kinds. Also, the two outputs are — in presence of rules of the form (18) — strongly equivalent to each others. In this sense, our translation is a generalization of McCain's translation.

Another translation [Doğandağ *et al.*, 2004] turns MCT theories called "almost definite" into logic programs. For a causal theory T that is both almost definite and in normal form, $\Delta_T(\sigma)$ is essentially strongly equivalent to the output of their translation. Causal theories in normal form seem more general than almost definite causal theories, as we don't know if there is any simple, modular and almost linear trasformation from arbitrary causal theories (or even from MCT theories) to almost definite causal theories.

Future work will include studying how we can turn a theory in the Modular Action Description language MAD into a logic program. This will require extending the translation to causal theories with variables [Lifschitz, 1997] and finding other simplifications of the translation.

We will also like to investigate the relationship between causal theories and logic programs with aggregates. It turns

out that, for instance, all rules (10) and (11) for the same constant c can be strongly equivalently replaced by a single aggregate — as defined in [Ferraris, 2005] — in the head of a rule.

Acknowledgments

We thank Selim Erdoğan and Hudson Turner for comments on this topic. We are grateful to Vladimir Lifschitz for many discussions and suggestions.

References

[Akman *et al.*, 2004] 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*, 153(1–2):105–140, 2004.

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

[Doğandağ *et al.*, 2001] Semra Doğandağ, Ferda N. Alpaslan, and Varol Akman. Using stable model semantics (SMODELS) in the Causal Calculator (CCALC). In *Proc. 10th Turkish Symposium on Artificial Intelligence and Neural Networks*, pages 312–321, 2001.

[Doğandağ *et al.*, 2004] Semra Doğandağ, Paolo Ferraris, and Vladimir Lifschitz. Almost definite causal theories. In *Proc. 7th Int'l Conference on Logic Programming and Nonmonotonic Reasoning*, pages 74–86, 2004.

[Erdoğan and Lifschitz, 2006] Selim T. Erdoğan and Vladimir Lifschitz. Actions as special cases. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 377–387, 2006.

[Erdoğan *et al.*, 2007] Selim T. Erdoğan, Paolo Ferraris, Vladimir Lifschitz, and Wanwan Ren. Why the monkey needs the box: A serious look at a toy domain. Submitted for review. [6], 2007.

[Ferraris, 2005] Paolo Ferraris. Answer sets for propositional theories. In *Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 119–131, 2005.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Proceedings of International Logic Programming Conference and Symposium*, pages 1070–1080, 1988.

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

[Giunchiglia *et al.*, 2004] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner. Nonmonotonic causal theories. *Artificial Intelligence*, 153(1–2):49–104, 2004.

[Lee and Lifschitz, 2006] Joohyung Lee and Vladimir Lifschitz. A knowledge module: buying and selling. In *Working Notes of the AAAI Symposium on Formalizing Background Knowledge*, 2006.

[Lifschitz and Ren, 2006] Vladimir Lifschitz and Wanwan Ren. A modular action description language. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 853–859, 2006.

[Lifschitz *et al.*, 1999] Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25:369–389, 1999.

[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 *et al.*, 2001] Vladimir Lifschitz, David Pearce, and Agustin Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2:526–541, 2001.

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

[Lifschitz, 2000] Vladimir Lifschitz. Missionaries and cannibals in the Causal Calculator. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 85–96, 2000.

[McCain and Turner, 1997] Norman McCain and Hudson Turner. Causal theories of action and change. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 460–465, 1997.

[McCain, 1997] Norman McCain. *Causality in Commonsense Reasoning about Actions.*[7] 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.

[Pearce *et al.*, 2002] David Pearce, Torsten Schaub, Vladimir Sarsakov, Hans Tompits, and Stefan Woltran. A polynomial translation of logic programs with nested expressions into disjunctive logic programs. In *Proc. NMR-02*, 2002.

[6] `http://www.cs.utexas.edu/~otto/papers/serious.pdf` .

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