Causal Theories as Logic Programs

Paolo Ferraris

University of Texas at Austin, Austin TX 78712, USA `otto@cs.utexas.edu`

Abstract. We show how we can rewrite any causal theory — under the semantics of causal logic due to McCain and Turner — as a logic program in the answer set semantics. Using this translation the models of any causal theory can be computed by answer set solvers.

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 $t+1$.

In many useful causal rules, such as (2)–(4), the formula before the "$\Leftarrow$" is a literal or $\bot$. Rules of this kinds are called definite. Other important rules are not definite: for instance, the equivalence of two fluents p and q at some time t is expressed by

$$p_t \equiv q_t \Leftarrow \top.$$

This rule cannot be (strongly) equivalently replaced by any set of definite causal rules.

Search for models of a causal theory is an important computational problem. In fact, the semantics of causal logic defines when an interpretation is a model of a causal theory without offering any search method better than the exhaustive one. Two search techniques exist in the special case when the theory is definite, i.e., all its rules are definite. In this case, the causal theory can be converted into a propositional theory, called its "literal completion" [McCain and Turner, 1997], that has the same models. Turning a causal theory into its literal completion doesn't lead to an increase in size. We can then use a satisfiability solver to find the models of the causal theory. 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, 2005].

The second method uses the translation from definite causal theories into logic programs under the answer set semantics [Gelfond and Lifschitz, 1988], [Gelfond and Lifschitz, 1991] that was discovered by McCain [1997]. This translation is linear as well, and, unlike literal completion, it is modular (i.e., can be applied to a causal theory rule-by-rule). For instance, causal rule (3) is turned into

$$p_t \leftarrow not\ \neg p_t,$$

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].

In this paper we propose a translation that is more general than McCain's one. We can convert a causal theory in clausal form — consisting of rules of the form

$$l_1 \vee \cdots \vee l_n \Leftarrow G \tag{5}$$

where each l_i is a literal — into a logic program. This translation is modular and linear.

Unlike definite causal theories, the class of causal theories in clausal form is very general since every causal theory can be converted into a theory in clausal form with the same models in a modular way [Giunchiglia *et al.*, 2004, Proposition 4]. In this paper we will show that this "clausification" can be even done without much increase in size, at the price of introducing auxiliary atoms.

On the other hand, the output of our translation has a more general form than the output of McCain's translation: it is a logic program with nested expressions [Lifschitz *et al.*, 1999]. Any rule with nested expressions can be replaced by a set of rules of the form

$$l_1; \dots ; l_m \leftarrow l_{m+1}, \dots , l_n, not\ l_{n+1}, \dots , not\ l_p, \tag{6}$$

$(0 \leq m \leq n \leq p)$ in polynomial time as explained in [Pearce *et al.*, 2002], again at the price of introducing auxiliary atoms. The output of McCain's translation consists of rules of the form (6) that are, moreover, nondisjunctive ($m \leq 1$). The need to use disjunctive rules in a polynomial time reduction of causal theories to logic programs follows from computational complexity considerations.

Our translation, together with the "clausification" procedure of causal theories and the reduction from [Pearce *et al.*, 2002], may allow some answer set solvers — the ones that understand rules of the form (6), such as dlv [2], GnT [3] and cmodels, [4] — to compute the models of an arbitrary causal theory.

We review the syntax and semantics of causal theories and logic programs in Sections 2 and 3, respectively. Our translation is described in Section 4, while the clausification process in Section 5. The proofs of the theorems are in Section 6. Related work is discussed in Section 7.

2 Causal Theories

We begin with a propositional signature σ, i.e., a collection of symbols called *atoms*. A *(propositional) 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.

[1] `http://www.cs.utexas.edu/users/tag/ccalc/` .
[2] `http://www.dbai.tuwien.ac.at/proj/dlv/` .
[3] `http://www.tcs.hut.fi/Software/gnt/` .
[4] `www.cs.utexas.edu/users/tag/cmodels/` .

A *causal rule* is an expression of the form $F \Leftarrow G$, where F and G are propositional formulas.[5] These formulas are called the *head* and the *body* of the rule respectively. A *causal theory* is a set of causal rules.

The semantics of causal theories of [McCain and Turner, 1997] defines when an interpretation I of σ (that is, a total function from the atoms of σ to truth values) 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 in the sense of propositional logic. 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 T of signature $\{p, q\}$:

$$\begin{aligned} p \vee \neg q &\Leftarrow \top \\ q &\Leftarrow p. \end{aligned} \tag{7}$$

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.

A rule of the form (5), where $n \geq 0$ and $l_1, \ldots, l_n$ are literals, is said to be in *clausal form.* A causal theory is in *clausal form* if all its rules are in clausal form.

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]. A *literal* is an atom a or its negation $\neg a$. 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 $X \not\models \bot$,
- $X \models not\ F$ 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 *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.

[5] In [Giunchiglia *et al.*, 2004] the syntax of causal rules allows F and G to be "multi-valued propositional formulas." Lee [2005, Section 6.4.2] showed that causal theories in this more general sense can be reduced to causal theories in the sense of [McCain and Turner, 1997] — the kind considered in this paper.

Two logic programs Π_1 and Π_2 are *strongly equivalent* if, for any logic program Π, $\Pi_1 \cup \Pi$ and $\Pi_2 \cup \Pi$ have the same answer sets [Lifschitz *et al.*, 2001]. In this paper we are actually interested in sets of literals that are complete — those that contain, for each atom a, either a or $\neg a$. (The reason for that will be clear from next section.) Then it is convenient to define a condition weaker than strong equivalence: Π_1 and Π_2 are *c-strongly equivalent*, if, for any logic program Π, $\Pi_1 \cup \Pi$ and $\Pi_2 \cup \Pi$ have the same complete answer sets. It can be shown that Π_1 and Π_2 are c-strongly equivalent iff the result of extending Π_1 by the rules

$$\bot \leftarrow not\ a, not\ \neg a \tag{8}$$

for all atoms a is strongly equivalent to the result of extending Π_2 by the same rules.

4 Translation

For any formula F, F^{ne} stands for the nested expression obtained from F by replacing each $\wedge$ with a comma, each $\vee$ with a semicolon and $\neg$ with *not*.

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 (5) by

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

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

According to this definition, each rule 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. This translation is clearly linear.

For instance, if T is (7) then Π_T is

$$\begin{aligned} p; \neg q &\leftarrow not\ not\ \top, (\neg p; not\ \neg p), (q; not\ q) \\ q &\leftarrow not\ not\ p, (\neg q; not\ \neg q). \end{aligned} \tag{10}$$

This program can be equivalently rewritten, using transformations from [Lifschitz *et al.*, 1999, Section 4], as

$$\begin{aligned} p; \neg q &\leftarrow \neg p, q \\ p; \neg q &\leftarrow \neg p, not\ q \\ p; \neg q &\leftarrow not\ \neg p, q \\ p; \neg q &\leftarrow not\ \neg p, not\ q \\ q; not\ p &\leftarrow \neg q \\ q; not\ p &\leftarrow not\ \neg q. \end{aligned}$$

The theorem below expresses the soundness of this translation. We identify each interpretation with the corresponding complete set of literals.

Theorem 1. *For any causal theory T in clausal form, an interpretation I is a model of T iff I is an answer set for Π_T.*

For instance, the only answer set of (10) is $\{p, q\}$. It is a complete set of literals and indeed it is the only model of (7). In general, Π_T may have answer sets that are not complete sets of literals, and those don't correspond to any model of T. The incomplete answer sets of a logic program can be eliminated by adding rules (8) for all atoms a.

The following proposition shows that, in case of causal rules of the form $l_1 \Leftarrow G$ (causal rule (5) with $n = 1$), we can drop $(\overline{l_1}; \mathit{not}\ \overline{l_1})$ from logic program rule (9).

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

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

is strongly equivalent to

$$l \leftarrow F.$$

For instance, the second rule of (10) can be rewritten as

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

and the answer sets don't change.

We will see in Section 7 that Proposition 1 brings us to a translation that is similar to McCain's translation in case of definite causal theories. 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 causal theory:

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

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

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

has no complete answer sets. If we drop the two disjunctions in the body we get a logic program with two complete answer sets $\{p\}$ and $\{\neg p\}$ instead.

5 Clausifying a causal theory

As we mentioned in the introduction, the translation from the previous section 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

$$C_1 \wedge \cdots \wedge C_n \Leftarrow G, \tag{11}$$

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

$$C_i \Leftarrow G \tag{12}$$

$(i = 1, \ldots, n)$ [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 atoms. The translation is similar to the one for logic programs from [Pearce *et al.*, 2002] mentioned in the introduction.

Consider any causal theory T over a signature σ, and let $\mathit{form}(T)$ be the collection of all subformulas of the heads of rules of T. The translation T' of T will contain atoms from σ and auxiliary atoms that we denote by d_F, where $F \in \mathit{form}(T)$.

For any formula $F \in \mathit{form}(T)$, we define $\mathit{def}(F)$ as

– formula $d_G \otimes d_H$, if F has the form $G \otimes H$ where $\otimes$ is $\vee$ or $\wedge$,

– formula $\neg d_G$, if F has the form $\neg G$, and
– formula F, if F is an atom or $\top$ or $\bot$.

Finally, we define T' to be the causal theory consisting of

$$d_F \Leftarrow G \tag{13}$$

for all rules $F \Leftarrow G$ in T, plus

$$d_F \equiv \mathit{def}(F) \Leftarrow \top \tag{14}$$

for all formulas $F \in \mathit{form}(T)$.

Theorem 2. *For any causal theory T over σ, $I \mapsto I|_\sigma$ is a 1–1 correspondence between the models of T' and the models of T.*

The clasification process described at the beginning of the section is actually linear in size when applied to causal theories of the form T'. Indeed, it doesn't modify rules of the form (13) since they are already in clausal form. About rules of the form (14), each head can be rewritten as a formula in conjunctive normal form that is only linearly larger than the head of (14), because it contains only three atoms. Finally, the transformation from rule (11) to rules (12) is normally quadratic, but in this case it is linear because the body of each rule (14) — and then of the corresponding rule (11) — has constant size: it is $\top$.

Finally, note also that

– the translation $T \mapsto \Pi_T$ of the previous section is linear,
– the transformation $T \mapsto T'$ is almost linear, and
– the translation of logic programs defined by [Pearce *et al.*, 2002] is almost linear.

As a result, for an arbitrary causal theory T, we can get a logic program consisting of rules of the form (6) — such that its answer sets are the models of T extended by auxiliary atoms — that is not much larger than T.

6 Proofs

6.1 Proof of Theorem 1

The following lemma is easily verifiable by structural induction.

Lemma 1. *For any interpretation I and any propositional formula F, I is a model of F iff $I \models F^{ne}$.*

Lemma 2. *For any two interpretations I and J and any causal theory T in clausal form, $I \cap J \models (\Pi_T)^I$ iff J is a model of T^I.*

Proof. It is sufficient to prove the claim for the case when T is a single rule (5). Then Π_T is rule (9). **Case 1:** I is not a model for G. Then $T^I = \emptyset$, and, in view of Lemma 1, $I \not\models G^{ne}$. Consequently, the term *not not* G^{ne} in the body of (9) is replaced by $\bot$ in the reduct $(\Pi_T)^I$. The claim in this case immediately follows. **Case 2:** I is a model for G. Then T^I is

$$l_1 \vee \cdots \vee l_n.$$

Considering the reduct of (9) relative to I, we notice that *not not* G^{ne} is replaced by $\top$ in view of Lemma 1, as in the previous case. Also, the terms of the form $(\bar{l}_i; \mathit{not}\ \bar{l}_i)$ of (9) can be rewritten, in the reduct, as $\bar{l}_i$ if $\bar{l}_i \in I$, and dropped otherwise. Consequently, we can write $(\Pi_T)^I$ as

$$l_1; \dots; l_n \leftarrow \mathop{,}_{l \in \{l_1,\dots,l_n\} : \bar{l} \in I} \bar{l}$$

where the "big comma" is similar to $\bigwedge$ (in particular, it is $\top$ if there are no conjunctive terms). Consequently, since interpretations are complete sets of literals, (l ranges over $l_1, \dots, l_n$)

$$\begin{aligned} I \cap J \models (\Pi_T)^I \ &\text{iff } l \in I \cap J \text{ for some } l \text{ whenever } \bar{l} \in I \cap J \text{ for all } \bar{l} \in I \\ &\text{iff } l \in J \text{ for some } l \in I \text{ or } \bar{l} \notin I \cap J \text{ for some } \bar{l} \in I \\ &\text{iff } l \in J \text{ for some } l \in I \text{ or } \bar{l} \notin J \text{ for some } \bar{l} \in I \\ &\text{iff } l \in J \text{ for some } l \in I \text{ or } l \in J \text{ for some } l \notin I \\ &\text{iff } l \in J \text{ for some } l \\ &\text{iff } J \text{ is a model of } T^I. \end{aligned}$$

Theorem 1. *For any causal theory T in clausal form, an interpretation I is a model of T iff I is an answer set for Π_T.*

Proof. The condition

$$I \text{ is a model of } T$$

means that

$$\text{for every interpretation } J,\ J \text{ is a model of } T^I \text{ iff } J = I.$$

In view of Lemma 2, this is equivalent to the condition

$$\text{for every interpretation } J,\ I \cap J \models (\Pi_T)^I \text{ iff } J = I.$$

This is further equivalent to the assertion

$$I \text{ is the only subset of } I \text{ that satisfies } (\Pi_T)^I,$$

because $J \mapsto I \cap J$ is a 1–1 correspondence between the set of all interpretations and the set of all the subsets of I. This means that I is an answer set for Π_T.

6.2 Proof of Proposition 1

We will use the following characterization of strong equivalence, which is basically a rephrasing of the characterization from [Turner, 2003]: two logic programs Π_1 and Π_2 are strongly equivalent iff for every consistent set X of literals,

- $X \not\models (\Pi_1)^X$ and $X \not\models (\Pi_2)^X$, or
- $(\Pi_1)^X$ and $(\Pi_2)^X$ are satisfied by the same subsets of X.

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

$$l \leftarrow F, (\bar{l}; \mathit{not}\ \bar{l}) \tag{15}$$

is strongly equivalent to

$$l \leftarrow F. \tag{16}$$

Proof. Let Π_1 be (15), and Π_2 be (16). Take any consistent set X of literals. Then $(\Pi_2)^X$ is

$$l \leftarrow F^X,$$

where F^X stands for F with all maximal subexpressions of the form *not* G replaced by $\top$ if $X \models \mathit{not}\ G$, and by $\bot$ otherwise.

Case 1: $\bar{l} \notin X$. Then $(\Pi_1)^X$ is

$$l \leftarrow F^X, (\bar{l}; \top)$$

which is clearly satisfied by the same subsets of X that satisfy $(\Pi_2)^X$.

Case 2: $\bar{l} \in X$. Then $(\Pi_1)^X$ is

$$l \leftarrow F^X, (\bar{l}; \bot)$$

Case 2a: $X \models F^X$. Since $l \notin X$, we conclude that $X \not\models (\Pi_1)^X$ and $X \not\models (\Pi_2)^X$. **Case 2b:** $X \not\models F^X$. Since F^X doesn't contain negation *not*, no subset of X satisfies F^X (easily provable by induction). It easily follows that all subsets of X satisfy both $(\Pi_1)^X$ and $(\Pi_2)^X$.

6.3 Proof of Theorem 2

Let $\sigma' = \sigma \cup \{d_F : F \in \mathit{form}(T)\}$ be the signature of T'. Let Δ be the propositional theory $\{d_F \equiv \mathit{def}(F) : F \in \mathit{form}(T)\}$, and Δ' be $\{d_F \equiv F : F \in \mathit{form}(T)\}$. The following lemma is provable by induction.

Lemma 3. *Δ is equivalent to Δ'.*

Lemma 4. *For any interpretation I over σ', $M \mapsto M|_\sigma$ is a 1–1 correspondence between the models of $(T')^I$ and the models of $T^{I|_\sigma}$.*

Proof. Note that $(T')^I$ is

$$\{d_F : F \Leftarrow G \in T, I \models G\} \cup \Delta.$$

In view of Lemma 3 and the fact that the body of each rule of T is over σ, $(T')^I$ is equivalent to

$$\{d_F : F \Leftarrow G \in T, I|_\sigma \models G\} \cup \Delta'$$

and then to

$$T^{I|_\sigma} \cup \Delta'.$$

It remains to notice that Δ' only defines each atom of $\sigma' \setminus \sigma$, which do not occur in $T^{I|_\sigma}$, in terms of atoms from σ.

Theorem 2. *For any causal theory T over σ, $I \mapsto I|_\sigma$ is a 1–1 correspondence between the models of T' and the models of T.*

Proof. If I is a model of T' then I is the only model of $(T')^I$. In view of Lemma 4, $I|_\sigma$ is the only model of $T^{I|_\sigma}$, and then a model of T. Now take any model J of T. It remains to show that, among all the interpretations I of the signature of T' such that $J = I|_\sigma$, exactly one is a model of T'. First we notice that, since T and the bodies of the rules of T' are over σ,

(i) each $T^{I|_\sigma}$ is identical to T^J, and
(ii) all $(T')^I$ are identical to each other.

From (i) we get that J is the only model of every $T^{I|_\sigma}$. Consequently, by Lemma 4, $(T')^I$ has a unique model M with $M|_\sigma = J$; this M is the same for all $I's$ by (ii). Note that since $M|_\sigma = J$, M is one of the interpretations I that we are considering, so that M is the only model of $(T')^M$ and consequently a model of T'. No I different from M is a model of T' because M is a model of $(T')^I$.

7 Related work and conclusions

Our translation is similar to McCain's translation [1997] in the sense that they are both modular, linear, and they share the same soundness property: the complete answer sets of the translations are the models of the original causal theory. On the other hand, McCain's translation is only applicable to rules of the form

$$l \Leftarrow l_1 \wedge \cdots \wedge l_n \tag{17}$$

($l_1, \ldots, l_n$ are literals and l is a literal or $\bot$), and its output is different from ours applied to (17). McCain's translation turns (17) into

$$l \leftarrow \mathit{not}\ \overline{l_1}, \ldots, \mathit{not}\ \overline{l_n} \tag{18}$$

while the transformation of Section 4, simplified using Proposition 1, turns it into

$$l \leftarrow \mathit{not}\ \mathit{not}\ ((l_1)^{ne}, \ldots, (l_n)^{ne}). \tag{19}$$

It can be shown that (19) is c-strongly equivalent to (18), so that our translation and McCain's translation indeed have the same complete answer sets. Note that their incomplete answer sets may be different: McCain's translation of $p \Leftarrow p$ is

$$p \leftarrow not\ \neg p$$

which has answer set $\{p\}$ only. Our translation gives

$$p \leftarrow not\ not\ p$$

which has the answer set $\emptyset$ as well.

As a generalization of McCain's translation, [Doğandağ *et al.*, 2004] proposed a translation from causal theories called "almost definite" into logic programs that is linear and modular also. Generally, almost definite causal theories can contain rules that are not in clausal form. On the other hand, whether or not a causal theory is almost definite depends on which rules of the form $l \Leftarrow l$ occur in it. For instance,

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

is almost definite, but the first rule alone is not. We saw that every causal theory can be converted to clausal form, at the cost of introducing additional atoms, in a modular way, and with an increase in size close to linear. It is not clear if similar simple reductions to almost definite causal theories exist. In this sense, our translation is more general than the one of [Doğandağ *et al.*, 2004].

Even though rules of the form $l \Leftarrow l$ play no special role in our translation, they can be used to reduce the size of the output. Indeed,

$$\Pi_{\{l \Leftarrow l\}} \cup \{l; F \leftarrow \left(\bar{l}; not\ \bar{l}\right), G\}$$

is c-strongly equivalent to

$$\Pi_{\{l \Leftarrow l\}} \cup \{F \leftarrow \bar{l}, G\},$$

and

$$\Pi_{\{\bar{l} \Leftarrow \bar{l}\}} \cup \{l; F \leftarrow \left(\bar{l}; not\ \bar{l}\right), G\}$$

is c-strongly equivalent to

$$\Pi_{\{\bar{l} \Leftarrow \bar{l}\}} \cup \{l; F \leftarrow G\}.$$

In some cases, these simplifications allow us to translate nondefinite causal theories into nondisjunctive logic programs. This happens, for instance, with two examples of almost definite causal theories — that are essentially in clausal form — from Section 5 of [Doğandağ *et al.*, 2004]: the description of the transitive closure of a relation, and the formalization of the two-gear domain. The application of the translation from [Doğandağ *et al.*, 2004] to those examples produces nondisjunctive programs as well. In fact, the translation from that paper and the two c-strong equivalences above are related to each other.

Another way of translating an arbitrary causal theory into a logic program is by first making it definite, as described by Lee [2004]. We can then use McCain's translation to get a logic program. However, that process is not modular, and it may lead to an exponential increase in size.

The translation from this paper can be extended to a large subset of the logic of universal causation [Turner, 1999].

Acknowledgments

We thank Selim Erdoğan and Hudson Turner for comments on this topic. We are grateful to Vladimir Lifschitz for many discussions and suggestions. This research was partially supported by the National Science Foundation under Grant IIS-0412907.

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.

[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, 2005] Joohyung Lee and Vladimir Lifschitz. A knowledge module: buying and selling [6] Unpublished draft, 2005.

[Lee, 2004] Joohyung Lee. Nondefinite vs. definite causal theories. In *Proc. 7th Int'l Conference on Logic Programming and Nonmonotonic Reasoning*, pages 141–153, 2004.

[Lee, 2005] Joohyung Lee. *Automated Reasoning about Actions.*[7] PhD thesis, University of Texas at Austin, 2005.

[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, 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.

[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.*[8] 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.

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

[Turner, 2003] Hudson Turner. Strong equivalence made easy: nested expressions and weight constraints. *Theory and Practice of Logic Programming*, 3(4,5):609–622, 2003.

[6] `http://www.cs.utexas.edu/users/vl/papers/buy.ps` .

[7] `http://www.cs.utexas.edu/users/appsmurf/papers/dissertation.ps` .

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