Nondefinite vs. Definite Causal Theories

Joohyung Lee

Department of Computer Sciences
University of Texas, Austin, TX, USA
appsmurf@cs.utexas.edu

Abstract. Nonmonotonic causal logic can be used to represent properties of actions, including actions with conditional and indirect effects, nondeterministic actions, and concurrently executed actions. The definite fragment of causal logic can be mapped to propositional logic by the process of completion, and this idea has led to the development of the Causal Calculator. In this note, we show how to turn arbitrary causal theories into definite theories without changing the sets of models. The translation consists of two parts: one is a set of definite rules which is obtained from the given theory by translating each rule one by one, in a modular way, and the other is a set of constraints similar to loop formulas for logic programs. Our result characterizes the semantics of causal logic in terms of propositional logic and tells us that an essential difference between the semantics of causal logic and the answer set semantics is related to the definition of a loop in each.

1 Introduction

The nonmonotonic causal logic defined in [McCain and Turner, 1997; Giunchiglia *et al.*, 2003] can be used to represent properties of actions, including actions with conditional and indirect effects, nondeterministic actions, and concurrently executed actions. Action language $\mathcal{C}+$ [Giunchiglia *et al.*, 2003], a high level notation for causal logic, is a useful formalism for describing transition systems—directed graphs whose vertices correspond to states and whose edges correspond to the executions of actions.

For the "definite" fragment of causal logic and $\mathcal{C}+$, the problem of determining the existence of models can be reduced to the satisfiability problem for propositional logic by the process of completion [McCain and Turner, 1997]—a translation similar to Clark's completion [Clark, 1978] familiar from logic programming. The Causal Calculator (CCALC)[1] is an implementation of the definite fragment of causal logic based on this idea. After turning a definite causal theory into a classical propositional theory, CCALC finds the models of the latter by invoking a satisfiability solver, such as CHAFF[2], SATO[3] and RELSAT[4].

[1] http://www.cs.utexas.edu/users/tag/ccalc/ .
[2] http://www.ee.princeton.edu/~chaff/ .
[3] http://www.cs.uiowa.edu/~hzhang/sato.html .
[4] http://www.almaden.ibm.com/cs/people/bayardo/resources.html .

Although the limitation of definiteness may look too restrictive, CCALC has been successfully applied to several challenge problems in the theory of commonsense knowledge [Lifschitz, 2000], [Lifschitz *et al.*, 2000], [Campbell and Lifschitz, 2003], [Lee and Lifschitz, 2003a], [Akman *et al.*, 2003] and to the formalization of multi-agent computational systems [Artikis *et al.*, 2003a; Artikis *et al.*, 2003b; Chopra and Singh, 2003].

In this note we show that the limitation is indeed not essential: we show how to turn *arbitrary* causal theories into definite theories without changing the sets of models. The translation consists of two parts: one is a set of definite rules which is obtained from the given theory by translating each rule one by one, in a modular way, and the other is a set of constraints. The main idea of the translation is related to the concept of a loop formula introduced in [Lin and Zhao, 2002], and our main theorem is similar to Theorem 1 from [Lee and Lifschitz, 2003b], which generalizes the result of [Lin and Zhao, 2002] to disjunctive logic programs. The theorem in this note tells us the relationship between nondefinite and definite theories and characterizes the semantics of causal logic in terms of propositional logic. Surprisingly, this result, together with Theorem 1 from [Lee and Lifschitz, 2003b] that characterizes the semantics of logic programs also in terms of propositional logic, tells us that an essential difference between the semantics of causal logic and the answer set semantics is related to the definition of a loop in each. This idea can guide us in translating a representation in one formalism to the other. The proposed translation from causal logic to propositional logic is also interesting from a computational point of view because it can be used to make CCALC understand arbitrary causal theories, not necessarily definite.

The problem of determining that a finite causal theory is consistent is Σ_2^P-hard [Giunchiglia *et al.*, 2003, Proposition 3], while the same problem for a definite causal theory is in NP. Not surprisingly, our reduction of arbitrary causal theories to definite theories introduces an exponential number of loop formulas in the worst case.

In the next section, we review the definition of causal logic and give an example of a nondefinite theory that formalizes an action domain. The translation from arbitrary causal theories to definite theories is described in Section 3; the relationship between causal logic and logic programs is discussed in Section 4.

2 Review of Causal Logic

2.1 Definition

Begin with a propositional signature σ.[5] By a *(causal) rule* we mean an expression of the form

$$F \Leftarrow G$$

[5] Formulas defined in [Giunchiglia *et al.*, 2003] are slightly more general than formulas of the usual propositional logic. In this note, we restrict attention to propositional signatures only.

("F is caused if G holds"), where F, G are formulas in propositional logic of the signature σ. F is called the *head* and G is called the *body* of the rule. Rules with the head $\bot$ are called *constraints*.

A *causal theory* is a finite set of causal rules. A causal theory is called *definite* if the head of every rule in it is either a literal or $\bot$.

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.

2.2 Examples

In Section 2.3 of [Giunchiglia *et al.*, 2003], the definition of causal logic is illustrated by the following example. Let T_1 be the following causal theory whose signature is $\{p, q\}$:

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

Consider, one by one, all interpretations of that signature (we identify an interpretation with the set of literals that are true in it):

- $I_1 = \{p, q\}$. The reduct consists of the heads of the first two rules of T_1: $T_1^{I_1} = \{p, q\}$. Since I_1 is the unique model of $T_1^{I_1}$, it is a model of T_1.
- $I_2 = \{\neg p, q\}$. 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_1.
- $I_3 = \{p, \neg q\}$. The only element of the reduct is the head of the third rule of T_1: $T_1^{I_3} = \{\neg q\}$. It has two models. Consequently, I_3 is not a model of T_1.
- $I_4 = \{\neg p, \neg q\}$. The reduct is the same as above, so that I_4 is not a model of T_1 either.

We see that I_1 is the only model of T_1.

Consider another example T_2 whose signature is $\{p, q\}$:

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

T_2^I is equal to the set of the heads of the rules in T_2 regardless of interpretation I; T_2^I has two models, $\{p, q\}$ and $\{\neg p, \neg q\}$, so that T_2 has no models.

T_3 is the following theory of the same signature that adds one rule to T_2:

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

Similarly to the previous example, T_3^I is equal to the set of the heads of the rules in T_3 regardless of interpretation I. Now $\{\neg p, \neg q\}$ is not a model of T_3^I, so that T_3 has one model: $\{p, q\}$.

Let T_4 be the following causal theory whose signature is $\{p, q, r\}$:

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

We see that T_4 has only one model: $\{p, q, \neg r\}$.

In the case when the theory is definite, it is known that its models can be computed by the process of completion. Consider a definite causal theory T of a signature σ. For each literal l, the *completion formula* for l is the formula

$$l \equiv G_1 \vee \cdots \vee G_n$$

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

For example, the completion of T_1 is

$$\begin{aligned} p &\equiv q \\ \neg p &\equiv \bot \\ q &\equiv q \\ \neg q &\equiv \neg q, \end{aligned}$$

and its only model is $\{p, q\}$, which is exactly the model found above using the definition of causal logic. However, the method of completion is not applicable to nondefinite theories, such as T_2, T_3 and T_4. In Section 3.3, we show how completion can be extended to cover nondefinite theories.

We understand the term *clause* to mean a disjunction of distinct literals $l_1 \vee \cdots \vee l_n$ $(n \geq 0)$, and identify a clause with the corresponding set $\{l_1, \ldots, l_n\}$. Using proposition 4 of [Giunchiglia *et al.*, 2003], any causal theory can be rewritten as a theory of the "normal form" without changing the set of models:

Fact 1 *Any rule of a causal theory T can be replaced with a finite set of rules of the form*

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

where F is a clause and G is a formula, without changing the set of models of T.

2.3 An Example of a Nondefinite Causal Theory for Action Domains

There are some cases where nondefinite theories look natural to formalize certain action domains. An action domain of this kind, due to Marc Denecker, is discussed in [McCain, 1997, Section 7.5]:

Imagine that there are two gears, each powered by a separate motor. There are switches that toggle the motors on and off, and there is a button that moves the gears so as to connect or disconnect them from one another. The motors turn the gears in opposite (i.e., compatible) directions. A gear is caused to turn if either its motor is on or it is connected to a gear that is turning.

McCain's formalization of the example in causal logic is reproduced in Figure 1. The signature consists of symbols

$$MotorOn(x),\ Connected,\ Turning(x)$$

prefixed by time stamp i : where $i \in \{0, \ldots, m\}$ and symbols

$$Toggle(x),\ Push$$

prefixed by time stamp i : where $i \in \{0, \ldots, m-1\}$; x ranges over $\{1, 2\}$. In Figure 1, the expression of the form $i{:}\neg\, c$ where c is an atom stands for a literal $\neg\, i{:}c$. The presence of the last rule makes the theory nondefinite.

$$
\begin{array}{rcl}
i{+}1{:}\,MotorOn(x) & \Leftarrow & i{:}\,Toggle(x) \wedge\ i{:}\neg MotorOn(x) \\
i{+}1{:}\neg MotorOn(x) & \Leftarrow & i{:}\,Toggle(x) \wedge\ i{:}\,MotorOn(x) \\
i{+}1{:}\,Connected & \Leftarrow & i{:}\,Push \wedge\ i{:}\neg Connected \\
i{+}1{:}\neg Connected & \Leftarrow & i{:}\,Push \wedge\ i{:}\,Connected \\
i+1{:}\,MotorOn(x) & \Leftarrow & i+1{:}\,MotorOn(x) \wedge\ i{:}\,MotorOn(x) \\
i+1{:}\neg MotorOn(x) & \Leftarrow & i+1{:}\neg MotorOn(x) \wedge\ i{:}\neg MotorOn(x) \\
i+1{:}\,Connected(x) & \Leftarrow & i+1{:}\,Connected(x) \wedge\ i{:}\,Connected(x) \\
i+1{:}\neg Connected(x) & \Leftarrow & i+1{:}\neg Connected(x) \wedge\ i{:}\neg Connected(x) \\
0{:}\,MotorOn & \Leftarrow & 0{:}\,MotorOn \\
0{:}\neg MotorOn & \Leftarrow & 0{:}\neg MotorOn \\
0{:}\,Connected & \Leftarrow & 0{:}\,Connected \\
0{:}\neg Connected & \Leftarrow & 0{:}\neg Connected \\
i{:}\,Toggle(x) & \Leftarrow & i{:}\,Toggle(x) \\
i{:}\neg Toggle(x) & \Leftarrow & i{:}\neg Toggle(x) \\
i{:}\,Push & \Leftarrow & i{:}\,Push \\
i{:}\neg Push & \Leftarrow & i{:}\neg Push \\
(i = 0, \ldots, m-1) & & \\
 & & \\
i{:}\,Turning(x) & \Leftarrow & i{:}\,MotorOn(x) \\
i{:}\neg Turning(x) & \Leftarrow & i{:}\neg Turning(x) \\
i{:}\,Turning(1) \equiv\ i{:}\,Turning(2) & \Leftarrow & i{:}\,Connected \\
(i = 0, \ldots, m) & &
\end{array}
$$

Fig. 1. Two Gears

3 Translation to Definite Theories

3.1 Translating the Rules

Let T be a causal theory of a signature σ whose rules have the form (1). For every rule $F \Leftarrow G$ of T, the corresponding definite rule $DR(F \Leftarrow G)$ is defined as follows:

$$DR(F \Leftarrow G) = \left\{ l \Leftarrow G \wedge \bigwedge_{l' \in F \setminus \{l\}} \overline{l'} \ : \ l \in F \right\}^6$$

if $|F| > 1$; $DR(F \Leftarrow G) = \{F \Leftarrow G\}$ otherwise.

A main part of the translation, the set of definite rules corresponding to T, is the union of $DR(r)$ for all rules r in T:

$$DR(T) = \bigcup_{r \in T} DR(r).$$

Note that $DR(T) = T$ when T is definite.

For example, $DR(T_3)$ is

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

and its only model is $\{p, q\}$, which is the only model of T_3 also.

The following proposition describes a relationship between the two causal theories.

Proposition 1 *For any causal theory T whose rules have the form (1), every model of T is a model of $DR(T)$.*

However, the converse of Proposition 1 does not hold as shown in the following examples. $DR(T_2)$ is

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

and it has two models, $\{p, q\}$ and $\{\neg p, \neg q\}$, while T_2 has no models.

Similarly, $DR(T_4)$ is

$$\begin{aligned} p &\Leftarrow q \\ \neg q &\Leftarrow \neg p \\ \neg p &\Leftarrow \neg q \\ q &\Leftarrow p \\ p &\Leftarrow \neg r \\ r &\Leftarrow \neg p \\ \neg r &\Leftarrow \neg r, \end{aligned}$$

[6] $\overline{l}$ denotes the literal complementary to literal l.

and it has two models, $\{p, q, \neg r\}$ and $\{\neg p, \neg q, r\}$, while the first is the only model of T_4.

In the next section we show how to strengthen the corresponding definite theory so that its models are exactly the models of the original theory.

3.2 Loops and Loop Formulas

Let T be a causal theory of a signature σ whose rules have the form (1). The *head dependency graph* of T is the directed graph G such that

- the vertices of G are the literals of σ, and
- for every rule $l_1 \vee \cdots \vee l_n \Leftarrow F$ of T, G has an edge from each l_i to each $\overline{l_j}$ where $j \neq i$, $1 \leq i, j \leq n$.

Notice that the head dependency graph is similar to the positive dependency graph defined in [Lee and Lifschitz, 2003b].

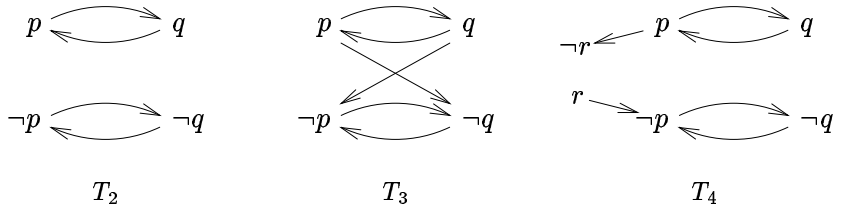

Fig. 2. Head dependency graphs for T_2, T_3, T_4

A nonempty set L of literals is called a *loop* of T if, for every pair l_1, l_2 of literals in L, there exists a path of non-zero length from l_1 to l_2 in the head dependency graph of T such that all vertices in this path belong to L. It is clear that every definite theory, including T_1, has no loops. Each of T_2, T_3 and T_4 has two loops: $\{p, q\}$ and $\{\neg p, \neg q\}$.

Given a set X of literals, $\overline{X}$ is the set of literals complementary to literals in X. The following fact easily follows from the definition.

Fact 2 *For any causal theory T whose rules have the form (1), if a set L of literals is a loop of T, then $\overline{L}$ is a loop of T also.*

For any loop L of T, by $R(L)$ we denote the set of formulas

$$G \wedge \bigwedge_{l \in F \setminus L} \bar{l}$$

for all rules (1) in T such that $F \cap L \neq \emptyset$ and $F \cap \overline{L} = \emptyset$. By $CLF(L)$ we denote the *conjunctive loop formula* of L:

$$CLF(L) = \bigwedge L \supset \bigvee R(L).^{7} \tag{2}$$

By $CLC(T)$ we denote the set of all constraints that express the conjunctive loop formulas for T:

$$CLC(T) = \{\bot \Leftarrow \neg CLF(L) : L \textit{ is a loop of } T\}.$$

For instance,

$$CLC(T_2) = \{\bot \Leftarrow \neg(p \wedge q \supset \bot),\ \bot \Leftarrow \neg(\neg p \wedge \neg q \supset \bot)\},$$
$$CLC(T_3) = \{\bot \Leftarrow \neg(p \wedge q \supset \top),\ \bot \Leftarrow \neg(\neg p \wedge \neg q \supset \bot)\},$$
$$CLC(T_4) = \{\bot \Leftarrow \neg(p \wedge q \supset \neg r),\ \bot \Leftarrow \neg(\neg p \wedge \neg q \supset \bot)\}.$$

In loop formulas of another kind, $\bigwedge$ in the antecedent of (2) is replaced with $\bigvee$. Let's call this formula $DLF(L)$ (*disjunctive loop formula*):

$$DLF(L) = \bigvee L \supset \bigvee R(L). \tag{3}$$

$DLC(T)$ is the set of all constraints that express the disjunctive loop formulas for T.

3.3 Main Theorem

Theorem 1 *For any causal theory T whose rules have the form (1) and any set X of literals, the following conditions are equivalent:*

(a) X is a model of T,
(b) X is a model of $DR(T) \cup CLC(T)$,
(c) X is a model of $DR(T) \cup DLC(T)$.

The theorem tells us that the models of any causal theory T whose rules have the form (1) can be found by computing the models of the completion of the corresponding definite theory $DR(T) \cup CLC(T)$ (or $DR(T) \cup DLC(T)$). Recall that by Fact 1, an arbitrary causal theory can be rewritten as a causal theory whose rules have the form (1) without changing the set of models.

According to Proposition 5 from [Giunchiglia *et al.*, 2003], adding constraints to a causal theory T does not introduce new models, but simply eliminates the models of T that violate at least one of these constraints. That is, adding $\bot \Leftarrow \neg F$ eliminates the models that do not satisfy F. Since $CLC(T)$ (or $DLC(T)$) is a set of constraints, the models of $DR(T) \cup CLC(T)$ are exactly the models of $DR(T)$ that satisfy every loop formula for T.

[7] The expression $\bigwedge L$ in the antecedent stands for the conjunction of all elements of L. The expression in the consequent has a similar meaning.

For instance, each of the models $\{p, q\}$ and $\{\neg p, \neg q\}$ of $DR(T_2)$ does not satisfy one of the loop formulas $p \wedge q \supset \bot$ and $\neg p \wedge \neg q \supset \bot$, so that T_2 has no models. T_3 has one model because the only model $\{p, q\}$ of $DR(T_3)$ satisfies both loop formulas $p \wedge q \supset \top$ and $\neg p \wedge \neg q \supset \bot$. One of the models $\{p, q, \neg r\}$ of $DR(T_4)$ is also the model of T_4 because it satisfies both loop formulas for T_4; the other model $\{\neg p, \neg q, r\}$ of $DR(T_4)$ is not a model of T_4 because it does not satisfy the loop formula $\neg p \wedge \neg q \supset \bot$.

From the fact that (b) and (c) are equivalent to each other we can infer that the antecedent of a loop formula can be more general. For any loop L, let F_L be a formula formed from literals in L using conjunctions and disjunctions. For instance, F_L can be any single literal that belongs to L or a conjunction or disjunction of several literals from L. It is clear that F_L entails $\bigvee L$ and is entailed by $\bigwedge L$. We conclude:

Corollary 1 *For any causal theory T whose rules have the form (1) and any set X of literals, the condition*

(d) X is a model of $DR(T) \cup \{\bot \Leftarrow \neg(F_L \supset \bigvee R(L)) \ : \ L$ is a loop of $T\}$

is equivalent to each of conditions (a)–(c) from Theorem 1.

The last rule of the description in Figure 1 can be replaced by the following rule without changing the set of models of the theory:

$$\neg(i\colon Turning(x)) \vee i\colon Turning(x_1) \Leftarrow i\colon Connected \qquad (x \neq x_1). \tag{4}$$

(x, x_1 range over $\{1, 2\}$.)

According to Theorem 1, the description in Figure 1 can be turned into a definite theory with the same set of models by replacing the rule (4) with the following rules [8]:

$$\begin{array}{rcl} i\colon Turning(x) & \Leftarrow & i\colon Turning(x_1) \wedge i\colon Connected \qquad (x \neq x_1) \\ \bot & \Leftarrow & \neg\big(\bigwedge_x i\colon Turning(x) \supset \bigvee_x i\colon MotorOn(x)\big). \end{array} \tag{5}$$

For any causal theory T whose rules have the form (1), the completion of $DR(T)$ is equivalent to a set of implications of the following two kinds:

$$G \supset F \tag{6}$$

for every rule (1) in T, and

$$l \ \supset \bigvee_{\substack{F \Leftarrow G \ \in \ T \\ l \in F}} \left(G \wedge \bigwedge_{l' \in F \setminus \{l\}} \overline{l'} \right) \tag{7}$$

for each literal l. Accordingly, the completion of the definite theory

$$DR(T) \cup \Big\{ \bot \Leftarrow \neg\Big(F_L \supset \bigvee R(L)\Big) \ : \ L \textit{ is a loop of } T \Big\}$$

from Corollary 1 is equivalent to the union of sets of formulas (6), (7) and $\{F_L \supset \bigvee R(L) \ : \ L \textit{ is a loop of } T\}$.

[8] The other rules obtained from translating (4) can be dropped without changing the set of models.

4 The Relationship between Causal Logic and Logic Programs

We discuss that the semantics of causal logic is closely related to the answer set semantics. This section is not self-contained and readers are assumed to be familiar with [Lee and Lifschitz, 2003b], which defines completion and loop formulas for disjunctive logic programs. In that paper we showed that the answer sets for a disjunctive logic program can be characterized as the models of its completion that satisfy its loop formulas. It is interesting to note that formulas (6), (7) in this paper are very similar to the completion of a disjunctive logic program defined in that paper, and that a loop formula for a causal theory has the same form with a loop formula for a logic program. Comparing Theorem 1 in this note with Theorem 1 from [Lee and Lifschitz, 2003b] tells us that an essential difference between the two formalisms is related to the definition of a loop in each. In the answer set semantics, an edge of the positive dependency graph goes from an atom in the head to a positive atom in the body, while in causal logic an edge of the head dependency graph goes from a literal in the head to the literal complementary to another literal in the head.[9]

4.1 From Logic Programs to Causal Logic

The idea above can be used to prove theorems about the relationship between logic programs and causal logic. For instance, it can be used to get an alternative proof of the lemma from [Giunchiglia *et al.*, 2003, Section 8.3], which shows how to embed disjunctive logic programs into causal logic. Proposition 2 below is slightly more general than the lemma[10] and is a special case of Theorem 1 from [Doğandağ *et al.*, 2003].

We follow the definition of the answer set semantics reviewed in Section 2 of [Lee and Lifschitz, 2003b], which applies to finite programs without classical negation. Fact 3 from [Lee and Lifschitz, 2003b] tells us that any program whose rules may have nested expressions [Lifschitz *et al.*, 1999] both in the heads and the bodies of rules is strongly equivalent to a program whose rules have the form

$$A \leftarrow B, F \tag{8}$$

where A is a disjunction of distinct atoms, B is a conjunction of distinct atoms, and F is a formula in which every occurrence of each atom is in the scope of negation as failure. In the following, we identify an interpretation with the set of atoms that are true in it.

[9] Another difference is that logic programs defined in [Lee and Lifschitz, 2003b] do not allow classical negation but allow negation as failure, while in causal logic it is opposite.

[10] The lemma from [Giunchiglia *et al.*, 2003, Section 8.3] allows infinite causal theories also.

Proposition 2 *Let Π be a (finite) program of a signature σ (without classical negation) whose rules have the form (8). A set X of atoms is an answer set for Π iff X is a model of the causal theory*

$$\{B \supset A \Leftarrow F \ : \ A \leftarrow B, F\} \ \cup \ \{\neg c \Leftarrow \neg c \ : \ c \in \sigma\}.^{11}$$

Proof Let T_Π be the corresponding causal theory for Π. It is easy to check that the completion of Π and the completion of $DR(T_\Pi)$ are equivalent to each other. As for the loop formulas, observe first that every loop L of Π is a loop of T_Π. It is also easy to check that for every loop L of Π, $CLF(L)$ for Π is equivalent to $CLF(L)$ for T_Π. Note that every loop that is in T_Π but not in Π contains at least one negative literal. $CLF(L')$ of these extra loops L' are tautologies due to the presence of $\{\neg c \Leftarrow \neg c \ : \ c \in \sigma\}$: the antecedent of the implication $CLF(L')$ is a conjunction of literals that contains $\neg c \in L'$ and one of the disjunctive terms in the consequent is $\neg c$. By Theorem 1 in this note and by Theorem 1 from [Lee and Lifschitz, 2003b] the proposition follows. ∎

Using the fact that the completion of Π and the completion of $DR(T_\Pi)$ are equivalent to each other, if a logic program Π is tight as defined in [Lee and Lifschitz, 2003b], we get a causal theory whose models are exactly the program's answer sets by simply replacing $\leftarrow$ in the logic program with $\Leftarrow$ (assuming the signature of the causal theory contains only atoms used in the logic program) and by adding the closed world assumption for atoms, i.e., $\neg c \Leftarrow \neg c$ for all atoms c.

4.2 Transitive Closure

The difference on the definition of a loop in each formalism can guide us in translating a representation in one formalism to the other. In logic programming the following describes the transitive closure tc of a binary relation p on a set A:

$$\begin{array}{rcll} p(x,y) & & \textit{for any pair } x,y \in A \textit{ such that } p(x,y) \textit{ holds} \\ tc(x,y) & \leftarrow & p(x,y) & \\ tc(x,z) & \leftarrow & p(x,y), tc(y,z). & \end{array} \tag{9}$$

One might be tempted to write the corresponding representation in causal logic as follows:

$$\begin{array}{rcll} p(x,y) & \Leftarrow & \top \qquad \textit{for any pair } x,y \in A \textit{ such that } p(x,y) \textit{ holds} \\ tc(x,y) & \Leftarrow & p(x,y) & \\ tc(x,z) & \Leftarrow & p(x,y) \wedge tc(y,z) & \\ \neg p(x,y) & \Leftarrow & \neg p(x,y) & \\ \neg tc(x,y) & \Leftarrow & \neg tc(x,y). & \end{array} \tag{10}$$

[11] We agree to identify '*not*' in logic programs with '$\neg$' in causal theories, ',' with '$\wedge$', and ';' with '$\vee$'.

Note that the completion of (9) is equivalent to the completion of (10). If p is acyclic, then tc in (10) describes the transitive closure correctly. Otherwise, the representation may allow spurious models that do not correspond to the transitive closure.

The presence of spurious models is related to the cyclic causality in the third rule. The loop formulas for (10) are not equivalent to the loop formulas for (9). In (9) the third rule tells us that the positive dependency graph has edges that go from $tc(x,z)$ to $tc(y,z)$, while in (10) the corresponding rule does not contribute to edges of the head dependency graph. Indeed, there are no loops for (10).

The problem can be corrected by moving $tc(y,z)$ in the third rule from the body to the head, so that the head dependency graph contains the corresponding edges:

$$tc(y,z) \supset tc(x,z) \Leftarrow p(x,y).^{12}$$

The modified causal theory may have more loops than (9), but the loop formulas for these extra loops are tautologies because each of the loops contains at least one negative literal (same reason as in the proof of Proposition 2). Thus it is easy to see that the loop formulas for the modified causal theory are equivalent to the loop formulas for (9). The translation of the causal logic representation of transitive closure to the corresponding logic program provides an alternative proof of Theorem 2 from [Doğandağ *et al.*, 2003], which shows the correctness of the modified casual theory for representing transitive closure. According to Theorem 1, tc can be also described by definite theories using the translation from Section 3.

4.3 Almost Definite Theories

[Doğandağ *et al.*, 2003] defines a class of causal theories called *almost definite*, which is more general than the class of definite theories. The paper describes a translation that turns any almost definite causal theory into a logic program. Theorem 1 from that paper shows the correctness of the translation: an interpretation is a model of an almost definite causal theory iff it is an answer set for the corresponding logic program.

An alternative (easier) proof of that theorem can be given by using Theorem 1 in this note and Theorem 1 from [Lee and Lifschitz, 2003b]: first translate from each of the formalisms to propositional logic and then show that the translations are equivalent to each other.

Acknowledgements

We are grateful to Selim Erdoğan, Paolo Ferraris, Vladimir Lifschitz and Hudson Turner for useful discussions related to the subject of this paper. This work was partially supported by the Texas Higher Education Coordinating Board under Grant 003658-0322-2001.

[12] According to Proposition 2, we can also write $p(x,y) \wedge tc(y,z) \supset tc(x,z) \Leftarrow \top$. Since $p(x,y)$ does not contribute to any loops, moving $p(x,y)$ from the head to the body does not change loop formulas. The case is similar with the second rule of (10).

Appendix: Proofs

Lemma 1. *If X is the unique model of a set Γ of clauses, then for each $l \in X$ there exists a clause F in Γ such that*

$$X \cap F = \{l\}$$

Proof If this is not the case, then there exists a literal $l \in X$ such that every clause in Γ that contains l also includes another literal that is in X. It follows that $(X \setminus \{l\}) \cup \{\bar{l}\}$ also satisfies Γ, which contradicts the uniqueness of X. ∎

Proposition 1 *For any causal theory T whose rules have the form (1), every model of T is a model of $DR(T)$.*

Proof Let X be a model of T. By Lemma 1, for each $l \in X$ there exists a clause F in T^X such that $X \cap F = \{l\}$. By the definition of T^X, there exists a rule $F \Leftarrow G$ in T such that $X \models G$. One can see that $DR(F \Leftarrow G)^X = \{l\}$ and furthermore $DR(T)^X = X$. ∎

Theorem 1 *For any causal theory T whose rules have the form (1) and any set X of literals, the following conditions are equivalent.*

(a) X is a model of T,
(b) X is a model of $DR(T) \cup CLC(T)$,
(c) X is a model of $DR(T) \cup DLC(T)$.

Proof From (c) to (b): use the fact that $DLF(L) \models CLF(L)$ for every loop L of T.

From (a) to (c): Let X be a model of T. Then X is a model of $DR(T)$ by Proposition 1. It remains to show that X satisfies every formula $DLF(L)$ where L is a loop of T. Assume that X satisfies the antecedent $\bigvee L$ of a disjunctive loop formula (3); we want to show that X satisfies the consequent of this formula. Since X is the unique model of T^X and $X \cap L \neq \emptyset$, another interpretation $Y = (X \setminus L) \cup \overline{L} \neq X$ does not satisfy T^X. Consequently, there is a rule $F \Leftarrow G$ in T such that X satisfies G, but Y does not satisfy F, that is, $F \cap Y = \emptyset$. Since

$$F \cap Y = F \cap ((X \setminus L) \cup \overline{L}) = (F \cap (X \setminus L)) \cup (F \cap \overline{L})$$

we conclude that

$$F \cap (X \setminus L) = \emptyset \tag{11}$$

and

$$F \cap \overline{L} = \emptyset. \tag{12}$$

On the other hand, since X satisfies F,

$$X \cap F \neq \emptyset. \tag{13}$$

From (11) and (13), we conclude that

$$F \cap L \neq \emptyset$$

From this and (12) we can conclude that the formula

$$G \wedge \bigwedge_{l \in F \setminus L} \overline{l}$$

belongs to $R(L)$, and is one of the disjunctive terms in the consequent of (3). We will now check that X satisfies this formula. By the choice of the rule, X satisfies G. From

$$F \cap (X \setminus L) = X \cap (F \setminus L) = \emptyset,$$

we see that X satisfies $\bigwedge_{l \in F \setminus L} \overline{l}$.

From (b) *to* (a): Let X be a model of $DR(T) \cup CLC(T)$. First, we will prove that X satisfies T^X. Take any clause F in T^X such that a corresponding rule $F \Leftarrow G$ is in T and $X \models G$. We see that $X \cap F \neq \emptyset$. Otherwise $DR(F \Leftarrow G)^X = F$, which is not satisfied by X, which in turn contradicts that X satisfies $DR(T)^X$.

Next, we will prove that X is the unique model of T^X. Suppose that this is not the case, and consider another model Y of T^X. We will show that there exists an infinite sequence $l_1, l_2, \ldots$ of elements of $X \setminus Y$ with the following two properties for every $k > 0$:

(i) the head dependency graph of T contains a path of nonzero length from l_k to l_{k+1} such that all literals in the path belong to $\{l_1, \ldots, l_{k+1}\}$;
(ii) if L is a loop of T such that

$$l_k \in L \subseteq \{l_1, \ldots, l_k\}, \tag{14}$$

then $l_{k+1} \notin L$.

The sequence $l_1, l_2, \ldots$ with these properties is defined recursively, as follows. Since $Y \neq X$, there exists a literal in $X \setminus Y$. Take l_1 to be any element of $X \setminus Y$. To select l_{k+1} given the sequence $l_1, \ldots, l_k$ such that $l_1, \ldots, l_k \in X \setminus Y$, consider two cases.

Case 1: T has a loop satisfying condition (14). Take the union L_{max} of all such loops L. It is clear that L_{max} is a loop, and

$$l_k \in L_{max} \subseteq \{l_1, \ldots, l_k\}. \tag{15}$$

Since $l_1, \ldots, l_k \in X \setminus Y$, X satisfies $\bigwedge L_{max}$. On the other hand, by the choice of X, X satisfies all conjunctive loop constraints of T, including $CLF(L_{max})$. Consequently, X satisfies one of the disjunctive terms

$$G \wedge \bigwedge_{l \in F \setminus L_{max}} \overline{l} \tag{16}$$

of $R(L_{max})$. For the corresponding rule (1) of T,

$$F \cap L_{max} \neq \emptyset \tag{17}$$

and

$$F \cap \overline{L_{max}} = \emptyset. \tag{18}$$

Since X satisfies (16), $X \models G$ and

$$X \cap (F \setminus L_{max}) = \emptyset. \tag{19}$$

From (15),

$$Y \cap L_{max} \subseteq Y \cap \{l_1, \ldots, l_k\} \subseteq Y \cap (X \setminus Y) = \emptyset,$$

so that $Y \cap L_{max} = \emptyset$. By the choice of Y, Y satisfies every formula in T^X, including F. Consequently,

$$Y \cap (F \setminus L_{max}) \neq \emptyset,$$

so that there is a literal $l \in Y$ that belongs to $F \setminus L_{max}$. Take $\overline{l}$ to be l_{k+1}. It follows from (19) that $X \not\models l$, i.e., $X \models \overline{l}$, so that l_{k+1} is indeed an element of $X \setminus Y$.

Case 2: T does not have a loop satisfying condition (14). Since X is the unique model of $DR(T)^X$ and $X \models l_k$, $l_k \in DR(T)^X$. Consequently, there exists a rule

$$l_k \Leftarrow G \wedge \bigwedge_{l \in F \setminus \{l_k\}} \overline{l}$$

in $DR(T)$ such that X satisfies the body of the rule. Consider a corresponding rule $F \Leftarrow G$ in T. Since $X \models G$, $F \in T^X$. Since $Y \not\models l_k$ and $Y \models F$, there is a literal l that is different from l_k among F that is satisfied by Y. Take $\overline{l}$ to be l_{k+1}. Since $l \in F \setminus \{l_k\}$ and $X \models \bigwedge_{l \in F \setminus \{l_k\}} \overline{l}$, $X \not\models l$. Consequently, X satisfies l_{k+1}, so that l_{k+1} is indeed an element of $X \setminus Y$.

Let us check that the sequence $l_1, l_2, \ldots$ satisfies property (i) above. Assume that l_{k+1} was chosen according to Case 1. By (17), there is a literal l that belongs both to the head F of (1) and to L_{max}. Since $\overline{l_{k+1}} \in F$, the head dependency graph of T contains an edge from l to l_{k+1}. Since both l_k and l belong to the loop L_{max}, this graph contains a path from l_k to l such that all literals in the path belong to $\{l_1, \ldots, l_k\}$. Consequently, the graph contains a path of nonzero length from l_k to l_{k+1} such that all literals in the path belong to $\{l_1, \ldots, l_{k+1}\}$.

Assume now that l_{k+1} was chosen according to Case 2. It is easy to see that there is an edge from l_k to l_{k+1} in the head dependency graph of T.

Next we will prove that the sequence satisfies property (ii) above. If a loop L of T satisfies condition (14), then l_{k+1} was chosen according to Case 1. Since $\overline{l_{k+1}} \in F$, from (18) we conclude that $\overline{l_{k+1}} \notin \overline{L_{max}}$, that is, $l_{k+1} \notin L_{max}$. Since $L \subseteq L_{max}$, it follows that $l_{k+1} \notin L$.

Notice that a literal may belong to X only if it occurs in the head of one of the rules of T. Indeed, if there is a literal that does not occur in the head of a

rule of T, then there is no unique model of T^X. Since T has finitely many rules, it follows that X is finite. Since $l_1, l_2, \ldots \in X \setminus Y$, this sequence must contain repetitions. Assume that $l_j = l_{k+1}$ for $j \leq k$. From property (i) we see that the head dependency graph of T contains a path from l_j to l_k whose vertices belong to $\{l_1, \ldots, l_{k+1}\}$. This property implies also that there is such a path of nonzero length from l_k to l_{k+1}. Since $l_j = l_{k+1}$, it follows that this graph contains a loop L such that

$$l_k, l_{k+1} \in L \subseteq \{l_1, \ldots, l_{k+1}\} = \{l_1, \ldots, l_k\},$$

which contradicts property (ii). ∎

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.[13] *Artificial Intelligence*, 2003. To appear.

[Artikis *et al.*, 2003a] A. Artikis, M. Sergot, and J. Pitt. An executable specification of an argumentation protocol. In *Proceedings of Conference on Artificial Intelligence and Law (ICAIL)*, pages 1–11. ACM Press, 2003.

[Artikis *et al.*, 2003b] A. Artikis, M. Sergot, and J. Pitt. Specifying electronic societies with the Causal Calculator. In F. Giunchiglia, J. Odell, and G. Weiss, editors, *Proceedings of Workshop on Agent-Oriented Software Engineering III (AOSE)*, LNCS 2585. Springer, 2003.

[Campbell and Lifschitz, 2003] Jonathan Campbell and Vladimir Lifschitz. Reinforcing a claim in commonsense reasoning.[14] In *Working Notes of the AAAI Spring Symposium on Logical Formalizations of Commonsense Reasoning*, 2003.

[Chopra and Singh, 2003] Amit Chopra and Munindar Singh. Nonmonotonic commitment machines. In *Agent Communication Languages and Conversation Policies AAMAS 2003 Workshop*, 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.

[Doğandağ *et al.*, 2003] Semra Doğandağ, Paolo Ferraris, and Vladimir Lifschitz. Almost definite causal theories. In this volume.

[Giunchiglia *et al.*, 2003] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner. Nonmonotonic causal theories.[15] *Artificial Intelligence*, 2003. To appear.

[Lee and Lifschitz, 2003a] Joohyung Lee and Vladimir Lifschitz. Describing additive fluents in action language $\mathcal{C}+$[16] In *Proc. IJCAI-03*, pages 1079–1084, 2003.

[Lee and Lifschitz, 2003b] Joohyung Lee and Vladimir Lifschitz. Loop formulas for disjunctive logic programs[17] In *Proc. ICLP-03*, 2003. To appear.

[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.

[13] `http://www.cs.utexas.edu/users/appsmurf/papers/zt.ps` .

[14] `http://www.cs.utexas.edu/users/vl/papers/sams.ps` .

[15] `http://www.cs.utexas.edu/users/vl/papers/nmct.ps` .

[16] `http://www.cs.utexas.edu/users/appsmurf/papers/additive-ijcai.ps` .

[17] `http://www.cs.utexas.edu/users/appsmurf/papers/disjunctive.ps` .

[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.

[Lin and Zhao, 2002] Fangzhen Lin and Yuting Zhao. ASSAT: Computing answer sets of a logic program by SAT solvers. In *Proc. AAAI-02*, 2002.

[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.*[18] PhD thesis, University of Texas at Austin, 1997.

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