Almost Definite Causal Theories

Semra Doğandağ[1], Paolo Ferraris[2], and Vladimir Lifschitz[2]

[1] Computer Engineering Department, Middle East Technical University, 06531 Ankara, Turkey
semra@ceng.metu.edu.tr
[2] Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712-1188, USA
{otto, vl}@cs.utexas.edu

Abstract. The language of nonmonotonic causal theories, defined by Norman McCain and Hudson Turner, is an important formalism for representing properties of actions. For causal theories of a special kind, called definite, a simple translation into the language of logic programs under the answer set semantics is available. In this paper we define a similar translation for causal theories of a more general form, called almost definite. Such theories can be used, for instance, to characterize the transitive closure of a binary relation. The new translation leads to an implementation of a subclass of almost definite causal theories that employs the answer set solver SMODELS as the search engine.

1 Introduction

The language of nonmonotonic causal theories defined in [McCain and Turner, 1997] is an important formalism for representing properties of actions. The postulates of a causal theory are *causal rules*—expressions of the form

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

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

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

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 $t+1$.

Many useful causal theories contain rules of the form

$$l \Leftarrow l \tag{2}$$

where l is a literal ("if l is true then there is a cause for this"). In the semantics of causal theories, this rule expresses, intuitively, that l is true by default. For instance, the assumption that a Boolean fluent p is normally true can be expressed by the rules $p_t \Leftarrow p_t$. The frame problem [Shanahan, 1997] is solved in causal logic using rules similar to (2):

$$\begin{array}{l} p_{t+1} \Leftarrow p_{t+1} \wedge p_t \\ \neg p_{t+1} \Leftarrow \neg p_{t+1} \wedge \neg p_t . \end{array} \tag{3}$$

These rules express that fluent p obeys the commonsense law of inertia: if p holds at time t then it is assumed by default to hold at time $t+1$ also, and similarly for $\neg p$.

A causal theory is *definite* if the head G of each of its causal rules (1) is a literal or the 0-place connective $\bot$. A finite definite causal theory can be easily turned into an equivalent set of propositional formulas using the "literal completion" procedure defined in [McCain and Turner, 1997], and then queries about it can be answered by invoking 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 challenge problems in the theory of commonsense reasoning [Lifschitz *et al.*, 2000], [Lifschitz, 2000], [Akman *et al.*, 2003], [Campbell and Lifschitz, 2003].

An alternative computational procedure for answering questions about causal theories is to translate them into logic programs under the answer set semantics as in Proposition 6.7 from [McCain, 1997], and then invoke an answer set solver, such as smodels[2] [Doğandağ *et al.*, 2001]. McCain's translation, like literal completion, is limited to definite theories.

In this paper we extend McCain's translation to the class of "almost definite" causal theories, which includes all definite theories and covers some other interesting cases. One kind of interesting almost definite theories has to do with the idea of transitive closure, or reachability in a directed graph, which plays an important role in formal commonsense reasoning. We have used the new translation to extend the implementation of definite causal theories based on smodels to a class of almost definite causal theories.

After a review of causal theories and logic programs in the next two sections, we define the concept of an almost definite causal theory and the new translation in Section 4. Examples of almost definite theories that formalize commmonsense knowledge and their translations are discussed in Section 5, and the implementation in Section 6. In Section 7 we relate this paper to other work on causal logic and answer sets and discuss the computational complexity of almost definite causal theories. The proofs are relegated to the appendix.

2 Causal Theories

As defined in the introduction, a *causal theory* is a set of *causal rules* of form (1), where F and G are propositional formulas.[3] These formulas are called the *body*

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

[2] `http://www.tcs.hut.fi/Software/smodels/` .

[3] In [Giunchiglia *et al.*, 2003] the syntax of causal rules allows F and G to be "multi-valued propositional formulas." Causal theories in this more general sense can be

and the *head* of the rule. The semantics of causal theories [McCain and Turner, 1997] defines when an interpretation I of the underlying signature (that is, a function from atoms 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.

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

$$\begin{array}{l} p \Leftarrow q \\ q \Leftarrow q \\ \neg q \Leftarrow \neg q. \end{array} \tag{4}$$

The interpretation I defined by $I(p) = I(q) = \mathbf{t}$ is a model of T. Indeed, I satisfies the bodies of the first two rules of T, so that the reduct T^I consists of the heads p, q of these rules; I is the only interpretation satisfying these formulas. The other 3 interpretations of $\{p, q\}$ are not models of T. Indeed, if $I(q) = \mathbf{f}$ then T^I is the set $\{\neg q\}$, which is satisfied by two interpretations; if $I(p) = \mathbf{f}$ and $I(q) = \mathbf{t}$ then the reduct is the set $\{p\}$, which is not satisfied by I.

The semantics of causal theories expresses the idea that a formula can be true only if the given causal rules guarantee that there is a cause for this. For instance, the rules in example (4) tell us that, under some conditions, there is a cause for p to be true, and similarly for q and $\neg q$, but these rules do not assert the existence of a cause for $\neg p$. Consequently, $\neg p$ cannot be true, which implies that p has to be true. Since the first rule of (4) is the only rule expressing the existence of a cause for p, the body q of that rule has to be true also. This informal reasoning explains why (4) has no models other than the interpretation that makes both p and q true.

Note that without the second rule, causal theory (4) would be inconsistent: q would have to be true, but there would have been no cause for this.

3 Logic Programs

The review of the answer set semantics in this section follows [Lifschitz *et al.*, 1999]. The original definition of this semantics in [Gelfond and Lifschitz, 1988] and [Gelfond and Lifschitz, 1991] applied to programs of a more special form, without nested expressions.

Elementary expressions are literals and the symbols $\bot$ ("false") and $\top$ ("true"). *Nested expressions* are built from elementary expressions using the unary connective *not* (negation as failure) and the binary conncectives , (conjunction) and ; (disjunction). A *logic program* is a set of *program rules* of the form

$$Head \leftarrow Body \tag{5}$$

where both *Head* and *Body* are nested expressions.

reduced to causal theories in the sense of [McCain and Turner, 1997] that are studied in this note.

The answer set semantics defines when a consistent set X of literals is an answer set for a program Π. As a preliminary step, we define when X *satisfies* a nested expression F (symbolically, $X \models F$), as follows:

- $X \models l$ if $l \in X$ for any literal l,
- $X \models \top$,
- $X \not\models \bot$,
- $X \models F, G$ if $X \models F$ and $X \models G$,
- $X \models F; G$ if $X \models F$ or $X \models G$,
- $X \models not\ F$ if $X \not\models F$.

We say that X satisfies a program Π (symbolically, $X \models \Pi$), if for each rule (5) in Π, if $X \models Body$ then $X \models Head$.

The *reduct* Π^X of Π relative to X is obtained from Π by substituting each outermost expression of the form $not\ F$ in Π with $\bot$ if $X \models F$, and with $\top$ otherwise. We say that X is an *answer set* for Π if X is a minimal set satisfying Π^X.

For instance, it is easy to check that the program

$$\begin{array}{l} p \leftarrow not\ \neg q \\ q \leftarrow not\ \neg q \\ \neg q \leftarrow not\ q \end{array} \tag{6}$$

has two answer sets:

$$\{p, q\},\ \{\neg q\}. \tag{7}$$

4 Translation

Recall that in a definite causal theory, the head of every rule is a literal or $\bot$.[4] In the definition of almost definite causal theories below, the rules of the theory have the form

$$G \supset H \Leftarrow F. \tag{8}$$

When G is $\top$, we will identify the head $G \supset H$ of this rule with H, so that definite rules can be viewed as a special case of (8).

A propositional formula is *in standard form* if it is formed from literals (called its *component* literals) using the connectives $\wedge$, $\vee$, $\top$ and $\bot$.

Let T be a causal theory whose rules have the form (8), where F, G and H are in standard form. We say that a literal l is *default false* if T contains the rule

$$\bar{l} \Leftarrow \bar{l} \tag{9}$$

($\bar{l}$ stands for the literal complementary to l). We say that T is *almost definite* if, in each of its rules (8),

[4] The definition of a definite causal theory in [Giunchiglia *et al.*, 2003, Section 2.6] includes also a finiteness assumption. That assumption is essential for the concept of completion discussed in that paper, but it is not needed for our present purposes.

– the component literals of G are default false, and
– the component literals of H are default false, or H is a conjunction of literals.

Every definite theory (with the bodies of rules written in standard form) is almost definite: in a definite rule, G is $\top$, and H is a literal or $\bot$. The pair of rules

$$\begin{array}{l} \neg q \supset p \Leftarrow \top \\ q \Leftarrow q \end{array} \tag{10}$$

is an example of an almost definite theory that is not definite.

Now we will describe a translation that turns any almost definite causal theory T into a logic program Π_T. We will identify $\wedge$ with the comma and $\vee$ with the semicolon. Under this convention, a standard formula can be viewed as a nested expression that does not contain negation as failure. For any standard formula F, by F_{not} we denote the nested expression obtained from F by replacing each component literal l with $not\ \overline{l}$. For instance,

$$(\neg p \wedge q)_{not} = not\ p, not\ \neg q.$$

For any almost definite causal theory T, the logic program Π_T is defined as the set of the program rules

$$H \leftarrow G, F_{not} \tag{11}$$

for all causal rules (8) in T.

For instance, the translation of (4) is the program

$$\begin{array}{l} p \leftarrow \top, not\ \neg q \\ q \leftarrow \top, not\ \neg q \\ \neg q \leftarrow \top, not\ q \end{array}$$

which is equivalent to (6). (On equivalent transformations of logic programs, see [Lifschitz *et al.*, 1999, Section 4]). The translation of (10) can be similarly written as

$$\begin{array}{l} p \leftarrow \neg q \\ q \leftarrow not\ \neg q. \end{array} \tag{12}$$

The theorem below expresses the soundness of this translation. We identify each interpretation with the set of literals that are satisfied by it. Clearly this set is complete: for each atom a, it contains either a or $\neg a$.

Theorem 1. *An interpretation is a model of an almost definite causal theory T iff it is an answer set for Π_T.*

Note that this theorem does not say anything about the answer sets for the translation Π_T that are not complete. For instance, the first of the answer sets (7) for program (6) is complete, and the second is not; in accordance with Theorem 1, the first answer set is identical to the only model of the corresponding causal theory (4). The only answer set for the translation (12) of causal theory (10) is $\{q\}$; it is incomplete, and accordingly (10) has no models.[5]

[5] The incomplete answer sets of a logic program can be eliminated by adding the constraints $\leftarrow not\ a, not\ \neg a$ for all atoms a.

Rules of Π_T can be more complex than allowed by the preprocessor LPARSE of the system SMODELS. (The syntax of LPARSE does not permit disjunctions in the bodies of rules, as well as conjunctions and non-exclusive disjunctions in the heads of rules.) If, however, the formulas F and G in a rule (8) are conjunctions of literals, and H is a literal or $\bot$, then the translation (11) of that rule can be processed by LPARSE. These conditions are satisfied in many interesting cases, including the examples of almost definite causal theories discussed in the next section.

5 Examples

5.1 Transitive Closure

The transitive closure of a binary relation P on a set A can be described by an almost definite causal theory as follows. For any $x, y \in A$ such that xPy, the theory includes the causal rule

$$p(x, y) \Leftarrow \top. \tag{13}$$

In the remaining causal rules, x, y and z stand for arbitrary elements of A. By default, P does not hold:

$$\neg p(x, y) \Leftarrow \neg p(x, y). \tag{14}$$

If xPy then there is a cause for x and y to satisfy the transitive closure of P:

$$tc(x, y) \Leftarrow p(x, y), \tag{15}$$

and there is a cause for the implication $tc(y, z) \supset tc(x, z)$ to hold:

$$tc(y, z) \supset tc(x, z) \Leftarrow p(x, y). \tag{16}$$

Finally, by default, the transitive closure relation does not hold:

$$\neg tc(x, y) \Leftarrow \neg tc(x, y). \tag{17}$$

The following theorem expresses that this representation of transitive closure in causal logic is adequate. By P^* we denote the transitive closure of P.

Theorem 2. *Causal theory (13)–(17) has a unique model. In this model M, an atom is true iff it has the form $p(x, y)$ where xPy, or the form $tc(x, y)$ where xP^*y.*

If we attempt to make the almost definite causal theory (13)–(17) definite by replacing (16) with the definite rule

$$tc(x, z) \Leftarrow p(x, y) \wedge tc(y, z)$$

$$\begin{array}{rll} p(x,y) & \leftarrow \top & \text{if } xPy \\ \neg p(x,y) & \leftarrow \mathit{not}\ p(x,y) & \\ tc(x,y) & \leftarrow \mathit{not}\ \neg p(x,y) & \\ tc(x,z) & \leftarrow tc(y,z), \mathit{not}\ \neg p(x,y) & \\ \neg tc(x,y) & \leftarrow \mathit{not}\ tc(x,y) & \end{array}$$

Fig. 1. Translation of causal theory (13)–(17).

then the assertion of Theorem 2 will become incorrect [Giunchiglia *et al.*, 2003, Section 7.2].

The logic program corresponding to causal theory (13)–(17) is shown in Figure 1.

Rules (15) and (16) above can be replaced with

$$p(x,y) \supset tc(x,y) \Leftarrow \top$$

and

$$p(x,y) \wedge tc(y,z) \supset tc(x,z) \Leftarrow \top.$$

The assertion of Theorem 2 holds for the modified theory also. Since the atoms $p(x,y)$ are default false, the modified theory is almost definite, so that Theorem 1 can be used to turn it into a logic program. Its translation differs from the one shown in Figure 1 in that it does not have the combination *not* $\neg$ in the third and forth lines.

The fact that the atoms $p(x,y)$ are assumed to be defined by rules of the forms (13) and (14) is not essential for the validity of Theorem 2, or for the claim that the theory is almost definite; the relation P can be characterized by any definite causal theory with a unique model. But the modification described above would not be almost definite in the absence of rules (14).

Transitive closure often arises in work on formalizing commonsense reasoning. For instance, Erik Sandewall's description of the Zoo World[6] says about the neighbor relation among positions that it "is symmetric, of course, and the transitive closure of the neighbor relation reaches all positions." Because of the difficulty with expressing transitive closure in the definite fragment of causal logic, this part of the specification of the Zoo World is disregarded in the paper by Akman *et al.* [2003] where the Zoo World is formalized in the input of language of CCalc.

Here is another example of this kind. The apartment where Robby the Robot lives consists of several rooms connected by doors, and Robby is capable of moving around and of locking and unlocking the doors. This is a typical action domain of the kind that are easily described by definite causal theories. But the assignment given to Robby today is to unlock enough doors to make any room accessible from any other (Figure 2). To express this goal in the language of causal logic we need, for any time instant t, the transitive closure of the relation

[6] `http://www.ida.liu.se/ext/etai/lmw/` .

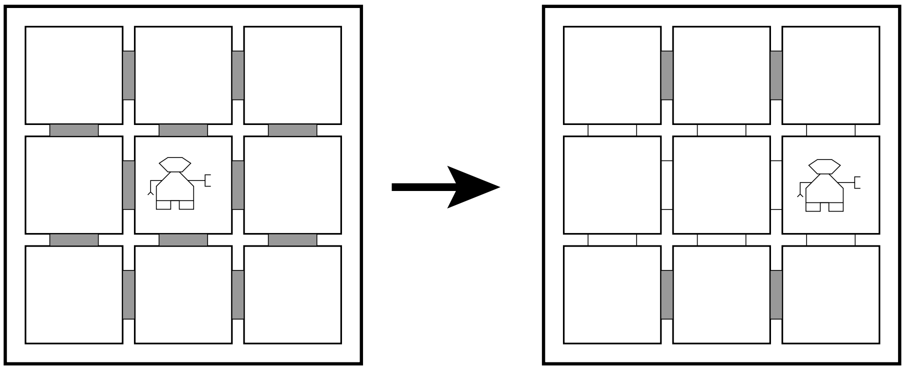

Fig. 2. Robby's apartment is a 3×3 grid, with a door between every pair of adjacent rooms. Initially Robby is in the middle, and all doors are locked. The goal of making every room accessible from every other can be achieved by unlocking 8 doors, and the robot will have to move to other rooms in the process.

"there is currently an unlocked door connecting Room i with Room j." In the spirit of the representation discussed above, the transitive closure can be defined by the causal rules

$$\begin{array}{r} \mathit{Accessible}(i,j)_t \Leftarrow \mathit{Unlocked}(i,j)_t \\ \mathit{Accessible}(j,k)_t \supset \mathit{Accessible}(i,k)_t \Leftarrow \mathit{Unlocked}(i,j)_t \\ \neg \mathit{Accessible}(i,j)_t \Leftarrow \neg \mathit{Accessible}(i,j.)_t \end{array} \tag{18}$$

5.2 Two Gears

This domain, invented by Marc Denecker, is described in [McCain, 1997, Section 7.5.5] as follows:

> Imagine that there are two gears, each powered by a separate motor. There are switches that toggle the motors on and off, and 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 representation of this domain as a causal theory is shown in Figure 3. The first 4 lines describe the direct effects of actions. The next 4 lines have the form (3) and express the commonsense law of inertia. The 8 lines that follow say that the initial values of fluents and the execution of actions are "exogenous." The last 3 lines express that a gear's motor being on causes it to turn, that the gears being connected causes them to turn (and not turn) together, and that by default the gears are assumed not to turn.

$$
\begin{aligned}
\neg MotorOn(G(i))_{t+1} &\Leftarrow Toggle(S(i))_t \wedge MotorOn(G(i))_t\\
MotorOn(G(i))_{t+1} &\Leftarrow Toggle(S(i))_t \wedge \neg MotorOn(G(i))_t\\
\neg Connected_{t+1} &\Leftarrow Push_t \wedge Connected_t\\
Connected_{t+1} &\Leftarrow Push_t \wedge \neg Connected_t\\
MotorOn(G(i))_{t+1} &\Leftarrow MotorOn(G(i))_{t+1} \wedge MotorOn(G(i))_t\\
\neg MotorOn(G(i))_{t+1} &\Leftarrow \neg MotorOn(G(i))_{t+1} \wedge \neg MotorOn(G(i))_t\\
Connected_{t+1} &\Leftarrow Connected_{t+1} \wedge Connected_t\\
\neg Connected_{t+1} &\Leftarrow \neg Connected_{t+1} \wedge \neg Connected_t\\
MotorOn(G(i))_0 &\Leftarrow MotorOn(G(i))_0\\
\neg MotorOn(G(i))_0 &\Leftarrow \neg MotorOn(G(i))_0\\
Connected_0 &\Leftarrow Connected_0\\
\neg Connected_0 &\Leftarrow \neg Connected_0\\
Toggle(S(i))_t &\Leftarrow Toggle(S(i))_t\\
\neg Toggle(S(i))_t &\Leftarrow \neg Toggle(S(i))_t\\
Push_t &\Leftarrow Push_t\\
\neg Push_t &\Leftarrow \neg Push_t
\end{aligned}
$$

$(i = 1, 2;\ t = 0, \dots, n-1)$;

$$
\begin{aligned}
Turning(G(i))_t &\Leftarrow MotorOn(G(i))_t\\
Turning(G(1))_t \equiv Turning(G(2))_t &\Leftarrow Connected_t\\
\neg Turning(G(i))_t &\Leftarrow \neg Turning(G(i))_t
\end{aligned}
$$

$(i = 1, 2;\ t = 0, \dots, n)$.

Fig. 3. Two gears domain.

Because of the second line from the end, this theory is not definite. But we can make it almost definite by replacing that line with

$$
\begin{aligned}
Turning(G(1))_t \supset Turning(G(2))_t &\Leftarrow Connected_t\\
Turning(G(2))_t \supset Turning(G(1))_t &\Leftarrow Connected_t.
\end{aligned}
$$

By Proposition 4(i) from [Giunchiglia *et al.*, 2003], this transformation does not change the set of models.

The corresponding logic program is shown in Figure 4. Many occurrences of the combination *not* $\neg$ in this program can be dropped without changing the answer sets.

6 Implementation

The system for answering queries about definite causal theories described in [Doğandağ *et al.*, 2001] has the same input language as CCALC but uses a different computational mechanism: it turns the given theory into a logic program and invokes SMODELS to find the program's answer sets.

$$
\begin{aligned}
\neg MotorOn(G(i))_{t+1} &\leftarrow not\ \neg Toggle(S(i))_t, not\ \neg MotorOn(G(i))_t\\
MotorOn(G(i))_{t+1} &\leftarrow not\ \neg Toggle(S(i))_t, not\ MotorOn(G(i))_t\\
Connected_{t+1} &\leftarrow not\ \neg Push_t, not\ Connected_t\\
\neg Connected_{t+1} &\leftarrow not\ \neg Push_t, not\ \neg Connected_t\\
MotorOn(G(i))_{t+1} &\leftarrow not\ \neg MotorOn(G(i))_{t+1}, not\ \neg MotorOn(G(i))_t\\
\neg MotorOn(G(i))_{t+1} &\leftarrow not\ MotorOn(G(i))_{t+1}, not\ MotorOn(G(i))_t\\
Connected_{t+1} &\leftarrow not\ \neg Connected_{t+1}, not\ \neg Connected_t\\
\neg Connected_{t+1} &\leftarrow not\ Connected_{t+1}, not\ Connected_t\\
MotorOn(G(i))_0 &\leftarrow not\ \neg MotorOn(G(i))_0\\
\neg MotorOn(G(i))_0 &\leftarrow not\ MotorOn(G(i))_0\\
Connected_0 &\leftarrow not\ \neg Connected_0\\
\neg Connected_0 &\leftarrow not\ Connected_0\\
Toggle(S(i))_t &\leftarrow not\ \neg Toggle(S(i))_t\\
\neg Toggle(S(i))_t &\leftarrow not\ Toggle(S(i))_t\\
Push_t &\leftarrow not\ \neg Push_t\\
\neg Push_t &\leftarrow not\ Push_t
\end{aligned}
$$

$(i = 1, 2;\ t = 0, \ldots, n-1)$,

$$
\begin{aligned}
Turning(G(i))_t &\leftarrow not\ \neg MotorOn(G(i))_t\\
Turning(G(2))_t &\leftarrow Turning(G(1))_t, not\ \neg Connected_t\\
Turning(G(1))_t &\leftarrow Turning(G(2))_t, not\ \neg Connected_t\\
\neg Turning(G(i))_t &\leftarrow not\ Turning(G(i))_t
\end{aligned}
$$

$(i = 1, 2;\ t = 0, \ldots, n)$.

Fig. 4. Translation of the two gears domain.

We have extended this system to a class of theories that are almost definite (or can be made almost definite by simple equivalent transformations), for which the logic program obtained by the translation from Section 4 is nondisjunctive.[7]

At the beginning of the translation process, the system finds all rules of the form $l \Leftarrow l$ in the given causal theory, and thus identifies all default false literals. (To be more precise, the input consists of *schematic rules*, or rules with variables. The system finds all schematic rules of the form $l \Leftarrow l$ and adds the ground instances of $\bar{l}$ to the list of default false literals.) The head of each rule that contains more than one atom is converted to conjunctive normal form, and then the rule is replaced by a group of rules whose heads are disjunctions of literals. Then the systems attempts to rewrite each of these disjunctions in the form $G \supset H$ where G is a conjunction of default false literals and H is a literal or $\bot$. (A similar transformation was applied to an equivalence in Section 5.2 above.) If this can be done for the head of every rule then the corresponding logic program Π_T is formed, the rules from Footnote ([5]) are added to it, and SMODELS is called to solve the given problem.

[7] The need for this restriction is related to the fact that SMODELS is not applicable to general disjunctive programs. Using the answer set solver DLV (`http://www.dbai.tuwien.ac.at/proj/dlv/`) instead of SMODELS would be a way to overcome this limitation.

To test the system, we have used the examples from Section 5 expressed in the "action language" notation [Giunchiglia and Lifschitz, 1998], [Giunchiglia *et al.*, 2003, Section 4] that eliminates the need to mention time explicitly. For instance, the assumption that executing the action $Go(i)$ makes the fluent $At(i)$ true can be expressed in this notation by the "causal law"

$$Go(i) \textbf{ causes } At(i)$$

that is viewed as an abbreviation for

$$At(i)_{t+1} \Leftarrow Go(i)_t.$$

Similarly, the causal law

$$\textbf{nonexecutable } Go(i) \textbf{ if } At(j) \wedge \neg Unlocked(i,j)$$

(the robot cannot go to Room i if there is no unlocked door connecting Room i with the room where the robot is) stands for

$$\bot \Leftarrow Go(i)_t \wedge At(j)_t \wedge \neg Unlocked(i,j)_t.$$

Causal laws like these are used to describe the effects of actions—going to other rooms and unlocking doors—and restrictions on their executability. In addition, the formalization contains the definition of accessibility by causal laws that are expanded into causal rules (18):

$$\begin{array}{l}\textbf{caused } Accessible(i,j) \textbf{ if } Unlocked(i,j)\\ \textbf{caused } Accessible(j,k) \supset Accessible(i,k) \textbf{ if } Unlocked(i,j)\\ \textbf{default } \neg Accessible(i,j).\end{array}$$

The solution shown in Figure 2 corresponds to one of the plans generated by this system. The plan consist of 11 steps: 3 *Go* actions and 8 *Unlock* actions.

7 Conclusion

The main theorem of this paper extends Proposition 6.7 from [McCain, 1997] from definite to almost definite causal theories. Another special case of the main theorem that was known before is the case when

- every atom in the language of T is default false, that is to say, T contains the causal rule $\neg a \Leftarrow \neg a$ for every atom a, and
- in every other rule (8) of T, F is a conjunction of negative literals, G is a conjunction of atoms, and H is a disjunction of atoms.

This special case is covered essentially by the lemma from [Giunchiglia *et al.*, 2003, Section 7.3]. What is interesting about this case is that by forming the translation Π_T of a causal theory T satisfying the conditions above we can get an *arbitrary* set of rules of the form

$$a_1; \dots; a_m \leftarrow b_1, \dots, b_n, not\ c_1, \dots, not\ c_p \tag{19}$$

where $a_1, \dots, a_m, b_1, \dots, b_n, c_1, \dots, c_p$ are atoms, plus the "closed world assumption" rules

$$\neg a \leftarrow not\ a$$

for all atoms a. Since the problem of existence of an answer set for a finite set of rules of the form (19) is Σ_2^P-hard [Eiter and Gottlob, 1993, Corollary 3.8], it follows that the problem of existence of a model for an almost definite causal theory is Σ_2^P-hard also. This fact shows that from the complexity point of view almost definite causal theories are as general as arbitrary causal theories.

On the other hand, if the formula H in every rule (8) of an almost definite causal theory T is a literal or $\bot$ then the corresponding logic program Π_T is nondisjunctive. Consequently, the problem of existence of a model for the almost definite causal theories satisfying this condition is in class NP, just as for definite causal theories. This condition is satisfied, for instance, for both examples discussed in Section 5.

If a causal theory T is definite then the corresponding logic program Π_T is tight in the sense of [Erdem and Lifschitz, 2003]. The answer sets for a finite tight program can be computed by eliminating classical negation from it in favor of additional atoms and then generating the models of the program's completion [Babovich *et al.*, 2000]. This process is essentially identical to the use of literal completion mentioned in the introduction. If T is almost definite but not definite then the program Π_T, generally, is not tight.

An earlier version of CCalc had a mechanism for answering queries about arbitrary causal theories, not necessarily definite or even almost definite, by calling a satisfiability solver several times. That mechanism was based on an algorithm that is, in principle, complete, but very inefficient.

Our future plans include extending the implementation discussed in Section 6 to causal theories with multi-valued formulas defined in [Giunchiglia *et al.*, 2003].

Acknowledgements

We are grateful to Joohyung Lee and Hudson Turner for comments on a draft of this paper. This work was partially supported by the Texas Higher Education Coordinating Board under Grant 003658-0322-2001, by the National Science Foundation under Grant INT-0004433, and by the Scientific and Technical Research Council of Turkey under Grant 101E024.

References

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

[Babovich *et al.*, 2000] Yuliya Babovich, Esra Erdem, and Vladimir Lifschitz. Fages' theorem and answer set programming.[8] In *Proc. Eighth Int'l Workshop on Non-Monotonic Reasoning*, 2000.

[8] `http://arxiv.org/abs/cs.ai/0003042` .

[Campbell and Lifschitz, 2003] Jonathan Campbell and Vladimir Lifschitz. Reinforcing a claim in commonsense reasoning.[9] 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.

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

[Erdem and Lifschitz, 2003] Esra Erdem and Vladimir Lifschitz. Tight logic programs. *Theory and Practice of Logic Programming*, 2003. To appear.

[Gelfond and Lifschitz, 1988] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In Robert Kowalski and Kenneth Bowen, editors, *Logic Programming: Proc. Fifth Int'l Conf. and Symp.*, 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 and Lifschitz, 1998] Enrico Giunchiglia and Vladimir Lifschitz. An action language based on causal explanation: Preliminary report. In *Proc. AAAI-98*, pages 623–630. AAAI Press, 1998.

[Giunchiglia *et al.*, 2003] Enrico Giunchiglia, Joohyung Lee, Vladimir Lifschitz, Norman McCain, and Hudson Turner. Nonmonotonic causal theories.[10] *Artificial Intelligence*, 2003. To appear.

[Lifschitz and Turner, 1994] Vladimir Lifschitz and Hudson Turner. Splitting a logic program. In Pascal Van Hentenryck, editor, *Proc. Eleventh Int'l Conf. on Logic Programming*, pages 23–37, 1994.

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

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

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

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

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

Appendix: Proofs of Theorems

We begin with a comment about notation. The semantics of propositional logic defines when an interpretation satisfies a propositional formula; on the other hand, we defined in Section 3 when a set of literals satisfies a nested expression. Since we have agreed to identify any interpretation with a set of literals, and any standard formula with a nested expression, these two definitions of satisfaction overlap when applied to an interpretation (a complete set of literals) and a standard formula (a nested expression without negation as failure). It is easy to see, however, that the two definitions are equivalent to each other in this special case, so that we can safely use the same symbol $\models$ for both relations.

Proof of Theorem 1

For any nested expression F we denote by $lit(F)$ the set of literals that have regular occurrences in F. (An occurrence is *regular* if it is not an occurence of an atom within a negative literal [Lifschitz *et al.*, 1999],) In particular, if F is a standard formula then $lit(F)$ is the set of all component literals of F. The following fact is easy to check by structural induction:

Fact 1. *For any nested expression F and any consistent set X of literals,*

$$X \models F \textit{ iff } X \cap lit(F) \models F.$$

Consider an almost definite causal theory T. For any interpretation I, the reduct T^I consists of implications $G \supset H$ where G and H are standard. We will denote by $\Delta(I)$ the set of program rules

$$H \leftarrow G \tag{20}$$

for all implications $G \supset H$ in T^I. It is clear that for any interpretation J,

$$J \models \Delta(I) \text{ iff } J \models T^I. \tag{21}$$

By D we denote the set of default false literals of T. Since T is almost definite, for any rule (20) in $\Delta(I)$

$$lit(G) \subseteq D \tag{22}$$

and

$$lit(H) \subseteq D \text{ or } H \text{ is a conjunction of literals.} \tag{23}$$

Lemma 1. *For any interpretations I, J, if $I \models \Delta(I)$ then*

$$J \models \Delta(I) \textit{ iff } I \cap J \models \Delta(I).$$

In the proof we use the following fact, which is easy to prove by structural induction.

Fact 2. *Let F be a nested expression without negation as failure, and let X, Y be consistent sets of literals. If $X \models F$ and $X \subseteq Y$ then $Y \models F$.*

Proof of Lemma 1. Consider two cases.

Case 1: $J \cap D \not\subseteq I$. Take a literal l such that $l \in J \cap D$ and $l \notin I$. Since $l \in D$, T contains the rule $\bar{l} \Leftarrow \bar{l}$. Since $l \notin I$, it follows that the formula $\bar{l}$ belongs to T^I, so that $\Delta(I)$ contains the program rule $\bar{l} \leftarrow \top$. Consequently any set of literals that satisfies $\Delta(I)$ contains $\bar{l}$. But $l \in J$, so that $\bar{l} \notin J$, which implies that neither J nor $I \cap J$ satisfies $\Delta(I)$.

Case 2: $J \cap D \subseteq I$.

Left to right. Assume that I and J satisfy $\Delta(I)$, and take any rule (20) in $\Delta(I)$ such that $I \cap J \models G$. We need to check that $I \cap J \models H$. By Fact 2, $I \models G$ and $J \models G$. Since I and J satisfy $\Delta(I)$, it follows that $I \models H$ and $J \models H$. According to (23), there are two possibilities. One is that $lit(H) \subseteq D$. Then, by the assumption of Case 2,

$$J \cap lit(H) \subseteq J \cap D \subseteq I \cap J. \tag{24}$$

By Fact 1, from $J \models H$ we can conclude that $J \cap lit(H) \models H$. By (24) and Fact 2, it follows that $I \cap J \models H$. The other possibility is that H is a conjunction of literals $l_1, \dots, l_n$. Since I and J both satisfy H, each of these literals belongs both to I and to J, so that $l_1, \dots, l_n \in I \cap J$, and we arrive at the same conclusion $I \cap J \models H$.

Right to left. Assume that $I \cap J \models \Delta(I)$, and take any rule (20) in $\Delta(I)$ such that $J \models G$. We need to check that $J \models H$. By Fact 1, from $J \models G$ can conclude that $J \cap lit(G) \models G$. On the other hand, by (22) and the assumption of Case 2,

$$J \cap lit(G) \subseteq J \cap D \subseteq I \cap J.$$

By Fact 2, it follows that $I \cap J \models G$. Since $I \cap J \models \Delta(I)$, we can conclude that $I \cap J \models H$. By Fact 2, it follows that $J \models H$.

Lemma 2. *For any consistent set X of literals and any interpretation I,*

$$X \models \Delta(I) \text{ iff } X \models (\Pi_T)^I.$$

In the proof we use the following fact, which is easy to prove by structural induction.

Fact 3. *For any standard formula F, any interpretation I and any consistent set X of literals,*

$$X \models (F_{not})^I \text{ iff } I \models F.$$

Proof of Lemma 2. The condition $X \models (\Pi_T)^I$ means that X satisfies the reduct with respect to I of the translation (11) of every rule (8) of T. That reduct can be written as

$$H \leftarrow G, (F_{not})^I \tag{25}$$

(the reduct of a formula is defined in the same way as the reduct of a program in Section 3). Consequently, by Fact 3, the condition $X \models (\Pi_T)^I$ can be expressed by saying that, for every rule (8) of T, if $I \models F$ and $X \models G$ then $X \models H$. This is equivalent to $X \models \Delta(I)$.

Theorem 1. *An interpretation is a model of an almost definite causal theory T iff it is an answer set for Π_T.*

Proof. Let I be an interpretation. The condition

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

means that

$$I \models T^I \text{ and, for any interpretation } J, \text{ if } J \models T^I \text{ then } J = I.$$

In view of (21), this is equivalent to the condition

$$I \models \Delta(I) \text{ and, for any interpretation } J, \text{ if } J \models \Delta(I) \text{ then } J = I$$

and then, by Lemma 1, to

$$I \models \Delta(I) \text{ and, for any interpretation } J, \text{ if } I \cap J \models \Delta(I) \text{ then } J = I.$$

The last condition can be expressed by saying that

$$\text{for any interpretation } J,\ I \cap J \models \Delta(I) \text{ iff } J = I.$$

This is further equivalent to the assertion

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

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. By Lemma 2, this assertion is equivalent to the claim that I is minimal among the sets satisfying $(\Pi_T)^I$, which means that I is an answer set for Π_T.

Proof of Theorem 2

Let P be a binary relation on a set A, P^* its transitive closure, and T the causal theory (13)–(17). Define the interpretation M by

$$\begin{aligned} M = &\{p(x,y) : xPy\} \cup \{\neg p(x,y) : \text{ not } xPy\} \\ &\cup \{tc(x,y) : xP^*y\} \cup \{\neg tc(x,y) : \text{ not } xP^*y\}. \end{aligned}$$

In this notation, the theorem to be proved can be stated as follows:

Theorem 2. *M is the only model of T.*

Lemma 3. *Let I be an interpretation such that T^I is consistent. For any $x, y \in A$, if xP^*y then $T^I \models tc(x,y)$.*

Proof. Observe that

$$\begin{aligned} T^I = {} & \{p(x,y) \,:\, xPy\} \\ & \cup \{\neg p(x,y) \,:\, I \not\models p(x,y)\} \\ & \cup \{tc(x,y) \,:\, I \models p(x,y)\} \cup \{tc(y,z) \supset tc(x,z) \,:\, I \models p(x,y)\} \\ & \cup \{\neg tc(x,y) \,:\, I \not\models tc(x,y)\}. \end{aligned} \tag{26}$$

Since T^I is consistent, from lines 1 and 2 of (26) we see that $I \models p(x,y)$ whenever xPy. Consequently, T^I contains

$$\{tc(x,y) \,:\, xPy)\} \cup \{tc(y,z) \supset tc(x,z) \,:\, xPy\}.$$

It remains to notice that these formulas entail $tc(x,y)$ for all x, y such that xP^*y.

Lemma 4. *For any interpretation I, if T^I is consistent and complete then $M \models T^I$.*

Proof. According to (26), the set of formulas in T^I that contain the atoms $p(x,y)$ is

$$\{p(x,y) \,:\, xPy\} \cup \{\neg p(x,y) \,:\, I \not\models p(x,y)\}.$$

Since T^I is consistent and complete, for any $x, y \in A$

$$xPy \text{ iff } I \models p(x,y).$$

It follows that (26) can be rewritten as

$$\begin{aligned} T^I = {} & \{p(x,y) \,:\, xPy\} \\ & \cup \{\neg p(x,y) \,:\, \text{ not } xPy\} \\ & \cup \{tc(x,y) \,:\, xPy\} \cup \{tc(y,z) \supset tc(x,z) \,:\, xPy\} \\ & \cup \{\neg tc(x,y) \,:\, I \not\models tc(x,y)\}. \end{aligned} \tag{27}$$

M clearly satisfies the formulas in the first three lines of (27). To prove that M satisfies the formulas in line 4, take any x, y such that $I \not\models tc(x,y)$; we need to check that $M \not\models tc(x,y)$, or, in other words, that xP^*y doesn't hold. Assume xP^*y. Then, by Lemma 3, $T^I \models tc(x,y)$; since T^I contains the formula $\neg tc(x,y)$, this contradicts the consistency of T^I.

Proof of Theorem 2. First we need to check that M is the only interpretation satisfying T^M. By formula (26) applied to $I = M$,

$$\begin{aligned} T^M = {} & \{p(x,y) \,:\, xPy\} \\ & \cup \{\neg p(x,y) \,:\, \text{ not } xPy\} \\ & \cup \{tc(x,y) \,:\, xPy\} \cup \{tc(y,z) \supset tc(x,z) \,:\, xPy\} \\ & \cup \{\neg tc(x,y) \,:\, \text{ not } xP^*y)\}. \end{aligned} \tag{28}$$

It is clear that M satisfies all these formulas. Take any interpretation I satisfying T^M and any $x, y \in A$. From the first two lines of (28) we see that

$$I \models p(x,y) \text{ iff } xPy \text{ iff } M \models p(x,y).$$

If xP^*y then, by Lemma 3 applied to $I = M$, $T^M \models tc(x, y)$, and consequently $I \models tc(x, y)$. Otherwise, from line 4 of (28) we see that $I \models \neg tc(x, y)$. Thus $I = M$.

To show that an interpretation I different from M cannot be a model of T, notice that for such I, by Lemma 4, T^I either is inconsistent, or is incomplete, or is satisfied by an interpretation different from I. In each of these cases, I cannot be the only interpretation satisfying T^I.