From Causal Theories to Logic Programs (Sometimes)

Fangzhen Lin and Kewen Wang

Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon
email: {flin, kwang}@cs.ust.hk

Abstract. In this paper, we shall provide a translation of a class of causal theories in (Lin [3]) to Gelfond and Lifschitz's disjunctive logic programs with classical negation [1]. We found this translation interesting for at least the following two reasons: it provides a basis on which a wide class of causal theories in ([3]) can be computed; and it sheds some new lights on the nature of the causal theories in [3]. Our translation is in many ways similar to the one given in [9, 2]. Our main result is a theorem that shows how action precondition and fully instantiated successor state axioms can be computed from the answer sets of the translated logic program.

Keywords: disjunctive logic programs with classical negation; causal theories of actions; situation calculus.

1 Introduction

The ramification problem in reasoning about action concerns with the implications that domain constraints have on action effects. For instance, in the blocks world, given that $stack(A, B)$ causes A to be on B and the constraint that a block can be only at one location at any given time, we can deduce that after the action is performed, A will not be on any block other than B.

Current work on the ramification problem centers on the idea that those domain constraints that can entail additional action effects should be encoded as causal constraints. There are currently several closely related theories of causal constraints. In this paper, we shall pursue the approach developed in (Lin [3]), and provide a translation of a class of causal theories in (Lin [3]) to Gelfond and Lifschitz's disjunctive logic programs with classical negation [1]. We found this translation interesting for at least the following two reasons: it provides a basis on which a wide class of causal theories in ([3]) can be computed; and it sheds some new lights on the nature of the causal theories in [3]. For instance, the causal theories in [3] use a two-stage minimization policy that first minimizes causation and then maximizes action precondition. In contrast, the intended meaning of the translated logic programs is captured using Gelfond and Lifschitz's answer set semantics alone.

This paper is organized as follows. In section 2, we define our language and introduce some notations. In section 3, we reformulate in this language causal theories of [3, 4]. In section 4, we briefly review Gelfond and Lifschitz's disjunctive logic programs with classical negation. In section 5, we provide a translation from a class of causal theories to logic programs and prove the correctness of this translation. In section 6, we illustrate our translation with some examples. Finally, we conclude in section 7.

2 The language

Given an action, and a starting situation, our goal in this paper is to determine whether the action is possible in the starting situation, and if so what the effects of this action are. We assume that the action and the starting situation are fixed and are implicit.

We shall assume a propositional language since logic programs are essentially propositional when only Herbrand models are considered. We assume a set of fluents $\mathcal{F}$. Fluents, i.e. elements of $\mathcal{F}$ are not propositions in the language. They are used to generate propositions in the following ways:

- if $f \in \mathcal{F}$, then $init(f)$, meaning that fluent f holds in the initial situation, is a proposition.
- if $f \in \mathcal{F}$, then $succ(f)$, meaning that fluent f holds in the successor situation of performing the action in the initial situation, is a proposition.
- if $f \in \mathcal{F}$, then $ctinit(f)$, meaning that fluent f is caused to be true in the initial situation, is a proposition.
- if $f \in \mathcal{F}$, then $cfinit(f)$, meaning that fluent f is caused to be false in the initial situation, is a proposition.
- if $f \in \mathcal{F}$, then $ctsucc(f)$, meaning that fluent f is caused to be true in the successor situation, is a proposition.
- if $f \in \mathcal{F}$, then $cfsucc(f)$, meaning that fluent f is caused to be false in the successor situation, is a proposition.

In the following, the last four types of propositions are called *causal propositions*. In addition, we assume that $poss$ is a proposition meaning that the action is possible in the starting situation.[1]

In the following, we take a *model* to be a set of propositions, that is, the set of propositions that are true in the model.

3 Causal theories

With respect to this language, a causal theory according to (Lin [3, 4]) is a set of sentences of the following form:

[1] As one of the anonymous referees pointed out, we do not really need $poss$ in this paper because we assume a fixed action and, as we shall see later in Theorems 1 and 2, we consider only models in which $poss$ holds. We however keep it here as one of the main tasks of this paper is to compute an explicit $poss$ axiom from answer sets.

– an explicit action precondition axiom of the form:

$$poss \supset \varphi_0$$

where φ_0 is a formula consisting only of propositions of the form $init(f)$ for $f \in \mathcal{F}$. (In the situation calculus, this corresponds to $\forall s.Poss(A, s) \supset \varphi$, where A is the assumed action and φ is a situation calculus formula obtained from φ_0 by replacing in it each occurrence of $init(f)$ by $Holds(f, s)$.)

– a set of effect axioms of the form

$$poss \wedge \varphi_0 \supset l_1 \vee \cdots \vee l_k$$

where φ_0 is a formula consisting of propositions of the form $init(f)$ for $f \in \mathcal{F}$, and each l_i, $1 \leq i \leq k$, is either $ctsucc(f)$ or $cfsucc(f)$ for some $f \in \mathcal{F}$.

– a set of causal constraints of the form:

$$\varphi_0 \wedge \varphi_1,$$

where φ_0 is a formula consisting of propositions of the form $init(f)$, $ctinit(f)$, or $cfinit(f)$, for $f \in \mathcal{F}$, and φ_1 is obtained from φ_0 by replacing $init(f)$ by $succ(f)$, $ctinit(f)$ by $ctsucc(f)$, and $cfinit(f)$ by $cfsucc(f)$. (In the situation calculus, this corresponds to the causal constraint $\forall s.\varphi(s)$, where $\varphi(s)$ is obtained from φ_0 by replacing $init(f)$ by $Holds(f, s)$, $ctinit(f)$ by $Caused(f, true, s)$, and $cfinit(f)$ by $Caused(f, false, s)$.) Notice that when φ_0 does not mention causal propositions, then $\varphi_0 \wedge \varphi_1$ is an acausal state constraint.

Example 1. Consider the suitcase domain in (Lin [3]). In this domain, there is a suitcase with two latches such that the suitcase will open up when both of the two latches are in the up position. We have the following fluents: $up1$ (the first latch is up), $up2$ (the second latch is up), and $open$ (the suitcase is open). Now consider the action $press1$ that will turn up the first latch if it was not up initially. The causal theory for this action is the following set of axioms:

– explicit action precondition:

$$poss \supset \neg init(up1).$$

– action effect axiom:

$$poss \supset ctsucc(up1).$$

– causal constraints:

$$init(up1) \wedge init(up2) \supset ctinit(open),$$
$$succ(up1) \wedge succ(up2) \supset ctsucc(open).$$

Given a causal theory T, its semantics is given by its completion, $comp(T)$, which is the circumscription of $\neg poss$ in $precom(T)$ with the propositions in $\mathcal{I} = \{init(f) \mid f \in \mathcal{F}\}$ fixed, where $precom(T)$ is the set of following axioms:

- The circumscription of all causal propositions:

$$\{ctinit(f), cfinit(f), ctsucc(f), cfsucc(f) \ \mid f \text{ is a fluent}\}$$

simultaneously with all other propositions fixed.
- For each fluent f, the following *basic causation axioms* which say that if a proposition is caused to have certain truth value, then it must have that truth value:

$$\begin{aligned}
&ctinit(f) \supset init(f),\\
&cfinit(f) \supset \neg init(f),\\
&ctsucc(f) \supset succ(f),\\
&cfsucc(f) \supset \neg succ(f).
\end{aligned}$$

- For each fluent f, the following generic successor state axiom which says that a fluent is true in a successor situation if and only if it either is caused to be true or was true and is not caused to be false:

$$poss \supset [succ(f) \equiv ctsucc(f) \vee (init(f) \wedge \neg cfsucc(f))].$$

For instance, for our suitcase domain in Example 1, the completion of that causal theory will yield the following axioms:

$$\begin{aligned}
&poss \equiv \neg init(up1),\\
&poss \supset [succ(up1) \equiv true],\\
&poss \supset [succ(up2) \equiv init(up2)],\\
&poss \supset [succ(open) \equiv init(up2) \vee init(open)].
\end{aligned}$$

4 Extended disjunctive logic programs

According to Gelfond and Lifschitz [1], an extended disjunctive logic program is one that can have disjunction in the head of a rule, and can have two kinds of negations: the so-called classical negation and negation-as-failure. More precisely, an extended disjunctive logic program P is a set of rules of the form

$$l_1 \vee \cdots \vee l_k \leftarrow l_{k+1}, \ldots, l_m, not\ l_{m+1}, \ldots, not\ l_n,$$

where each l_i, $1 \leq i \leq n$, is a *literal* which is either a proposition p or the (classical) negation of a proposition $\neg p$. In the following, we let Lit to be the set of all literals.

The semantics of an extended disjunctive logic program P is given by its *answer sets* as follows:

1. For any set S of literals, define P^S to be the GL-transformation of P obtained from P in the following ways:
   (a) if a rule has some $not\ l$ in its body such that $l \in S$, then delete this rule;

   (b) if for some l, *not* l occurs in the body of a rule and $l \notin S$, then delete *not* l from this rule.

   It is clear that P^S so obtained is a program without mentioning '*not*'.
2. A set S of literals is an *answer set* of P, if S is a minimal consequence set of P^S, i. e. S is a minimal set of literals satisfying the following properties:
   (a) if $l_1 \vee \cdots \vee l_k \leftarrow l_{k+1}, \ldots, l_m$ is a rule in P^S, and $l_{k+1}, \ldots, l_m \in S$, then $l_i \in S$ for some $i \in \{1, \ldots, k\}$;
   (b) if S contains a pair of complementary literals p and $\neg p$, then $S = Lit$.

An answer set E is consistent if it does not contain two contradictory literals, i.e. $E \neq Lit$.

5 From causal theories to logic programs (sometimes)

We shall now provide a translation of causal theories of certain form to Gelfond and Lifschitz's extended disjunctive logic programs.

Definition 1. *A causal theory is a* clausal *one if it satisfies the following conditions:*

- *The φ_0 in the explicit action precondition axiom and all effect axioms are of the form: $l_1 \wedge \cdots \wedge l_n$, where each l_i, $1 \leq i \leq n$, is either $init(f)$ or $\neg init(f)$ for some fluent f.*
- *The φ_0 in each causal constraint has the following form:*

$$l_1 \wedge \cdots \wedge l_k \supset c_1 \vee \cdots \vee c_t,$$

  where each l_i is either $init(f)$ or $\neg init(f)$, and each c_i is either $ctinit(f)$ or $cfinit(f)$, for some fluent f.

In other words, a clausal causal theory is a set of following sentences:

- a set of explicit action precondition axioms of the following form:

$$poss \supset l_1 \wedge \cdots \wedge l_n, \tag{1}$$

  where each l_i is either $init(f)$ or $\neg init(f)$.
- a set of action effect axioms of the following form:

$$poss \wedge l_1 \wedge \cdots \wedge l_n \supset \tilde{c}_1 \vee \cdots \vee \tilde{c}_m, \tag{2}$$

  where each l_i is either $init(f)$ or $\neg init(f)$, and each $\tilde{c}_i$ is either $ctsucc(f)$ or $cfsucc(f)$.
- a set of causal constraints of the following form:

$$l_1 \wedge \cdots \wedge l_k \supset c_1 \vee \cdots \vee c_t, \tag{3}$$

$$\tilde{l}_1 \wedge \cdots \wedge \tilde{l}_k \supset \tilde{c}_1 \vee \cdots \vee \tilde{c}_t, \tag{4}$$

  where each l_i is either $init(f)$ or $\neg init(f)$, each c_i is either $ctinit(f)$ or $cfinit(f)$, $\tilde{l}_i$ is obtained from l_i by replacing $succ(f)$ for $init(f)$, and $\tilde{c}_i$ is obtained from c_i by replacing $ctsucc(f)$ for $ctinit(f)$ and $cfsucc(f)$ for $cfinit(f)$.

For instance, the causal theory given in Example 1 is a clausal causal theory. We now proceed to translate a clausal causal theory T into an extended disjunctive logic program πT.

First of all, independent of any causal theory, we have the following rules for each fluent f:

$$
\begin{aligned}
init(f) &\leftarrow ctinit(f) && (5)\\
\neg init(f) &\leftarrow cfinit(f) && (6)\\
succ(f) &\leftarrow ctsucc(f) && (7)\\
\neg succ(f) &\leftarrow cfsucc(f) && (8)\\
succ(f) &\leftarrow init(f), not\ cfsucc(f), poss && (9)\\
\neg succ(f) &\leftarrow \neg init(f), not\ ctsucc(f), poss && (10)\\
poss &\leftarrow not\ \neg poss && (11)\\
init(f) &\leftarrow not\ \neg init(f) && (12)\\
\neg init(f) &\leftarrow not\ init(f) && (13)
\end{aligned}
$$

It is clear that the first four rules correspond to basic causation axioms. Rules (9) and (10) correspond to generic successor state axioms, that is, the inertia axioms. Rule (11) maximizes *poss*. Rules (12) and (13) assume a complete knowledge about the initial situation, and were adapted from (Turner [8]). These two rules are crucial for our translation.

In addition to the above domain independent rules, πT contains the following domain specific rules obtained from T:

- if T contains an explicit action precondition axiom (1), then πT contains the following rules:
$$\neg poss \leftarrow not\ l_i, \quad i = 1, \ldots, n. \qquad (14)$$
- if T contains an action effect axiom (2), then πT contains the following rule:
$$\tilde{c}_1 \vee \cdots \vee \tilde{c}_m \leftarrow l_1, \ldots, l_n, poss \qquad (15)$$
- if T contains causal constraints (3) and (4), then πT contains the following rules:
$$
\begin{aligned}
c_1 \vee \cdots \vee c_t &\leftarrow l_1, \ldots, l_k && (16)\\
\tilde{c}_1 \vee \cdots \vee \tilde{c}_t &\leftarrow not\ \hat{l}_1, \cdots, not\ \hat{l}_k && (17)
\end{aligned}
$$
where $\hat{l}_i$, $1 \leq i \leq k$, is the complement of $\tilde{l}_i$, i.e. if $\tilde{l}_i$ is atom p, then $\hat{l}_i$ is $\neg p$, and if $\tilde{l}_i$ is negated atom $\neg p$, then $\hat{l}_i$ is p.

Notice that in the last rule, $not\ \hat{l}_i$ cannot be replaced by $\tilde{l}_i$. Otherwise, the rule would be too weak, as the following example illustrates:

Example 2. Consider the following causal theory T consisting of a single cyclic causal rule which says that if q is true, then q is caused to be true:

$$init(q) \supset ctinit(q),$$
$$succ(q) \supset ctsucc(q).$$

These two sentences are translated to the following two rules:

$$ctinit(q) \leftarrow init(q)$$
$$ctsucc(q) \leftarrow not\ \neg succ(q)$$

The logic program consisting of these two rules and the domain independent ones has three consistent answer sets:

$$S_1 = \{poss, \neg init(q), \neg succ(q)\},$$
$$S_2 = \{poss, \neg init(q), succ(q), ctsucc(q)\},$$
$$S_3 = \{poss, init(q), succ(q), ctinit(q), ctsucc(q)\}.$$

From these answer sets,[2] we see that the action is always possible, and that if q is true initially, then q is still true afterward, but if q is not true initially, then the action is indeterminate about q afterward. This is the same conclusion entailed by $comp(T)$.

However, if we replace $not\ \neg succ(q)$ by $succ(q)$ in the second rule:

$$ctinit(q) \leftarrow init(q)$$
$$ctsucc(q) \leftarrow succ(q)$$

Then we would get a different result: S_2 above is no longer an answer set of the new logic program.

This translation is provably correct, in the sense that there is a one to one correspondence between models of the completion of a causal theory and answer sets of its corresponding logic program, provided the models and the answer sets satisfy $poss$:

Theorem 1. *Let T be a clausal causal theory and πT its corresponding extended disjunctive logic program. We have:*

1. *If N is a model of $comp(T)$ and $poss \in N$, then $S_N = N \cup N^{\neg}$ is a consistent answer set of πT, where*[3]

$$N^{\neg} = \{\neg init(f) \mid init(f) \notin N\} \cup \{\neg succ(f) \mid succ(f) \notin N\}.$$

2. *If E is a consistent answer set of πT with $poss \in E$, then $M_E = B_T \cap E$ is a model of $comp(T)$, where B_T is the set of atoms in T.*

[2] See Theorem 2 below.

[3] We thank one of the anonymous referees for suggesting this definition of $N^{\neg}$ which is simpler and more intuitive than our previous one.

Proof. See Appendix.

We can use this theorem to compute action precondition and successor state axioms for clausal causal theories as follows. For any set E of literals. Let $init(E)$ be the conjunction of all literals in E about the initial situation:

$$init(E) = \bigwedge_{init(f)\in E} init(f) \wedge \bigwedge_{\neg init(f)\in E} \neg init(f).$$

The following consequence of Theorem 1 shows how action precondition and successor state axioms of a clausal causal theory can be recovered from the answer sets of the corresponding logic program:

Theorem 2. *Let T and πT be as in Theorem 1. The following are consequences of $comp(T)$:*

$$poss \equiv \bigvee_{E\in\mathcal{E}_{poss}} init(E), \tag{18}$$

$$poss \wedge succ(f) \supset \bigvee_{E\in\mathcal{E}_{f}} init(E), \tag{19}$$

$$poss \wedge \neg succ(f) \supset \bigvee_{E\in\mathcal{E}_{\neg f}} init(E), \tag{20}$$

where f is a fluent, and

$$\mathcal{E}_{poss} = \{E \mid E \textit{ is a consistent answer set of } \pi T \textit{ and } poss \in E\},$$
$$\mathcal{E}_{f} = \{E \mid E \textit{ is a consistent answer set of } \pi T,\ succ(f) \in E, \textit{ and } poss \in E\},$$
$$\mathcal{E}_{\neg f} = \{E \mid E \textit{ is a consistent answer set of } \pi T, \neg succ(f) \in E, \textit{ and } poss \in E\}.$$

Furthermore, for any fluent f, there is a sentence φ about the initial situation such that

$$comp(T) \models poss \supset succ(f) \equiv \varphi \tag{21}$$

if and only if

$$\models \bigvee_{E\in\mathcal{E}_{f}} init(E) \supset \neg \bigvee_{E\in\mathcal{E}_{\neg f}} init(E). \tag{22}$$

Proof. We prove (18) first. Let N be a model of $comp(T)$. If $N \models poss$, then by Theorem 1, there is an answer set S_N of πT such that $poss \in S_N$. By the definition of S_N, it is easy to see $N \models init(S_N)$. Thus N satisfies the right side of (18) as well.

Conversely, if N satisfies the right side, then there is an answer set $E \in \mathcal{E}_{poss}$ such that $N \models init(E)$. By Theorem 1, M_E is a model of $comp(T)$ and $poss \in M_E$. Recall that $comp(T)$ is the result of minimizing $\neg poss$ in $precomp(T)$ with propositions in $\mathcal{I} = \{init(f) \mid f \in \mathcal{F}\}$ fixed. Because $N \models init(E)$, N and M_E must agree on propositions in $\mathcal{I}$. So $poss$ must be in N as well.

Similarly, (19) and (20) can be proven.

It remains to prove the last part of this corollary:

(22)⇒(21): this follows trivially from (19) and (20) by letting φ be $\bigvee_{E\in\mathcal{E}_f} init(E)$.

(21)⇒(22): Suppose (21) holds. If (22) does not hold, then there is an interpretation N such that $N \models \bigvee_{E\in\mathcal{E}_f} init(E) \wedge \bigvee_{E\in\mathcal{E}_{\neg f}} init(E)$. Thus there are two answer sets $E \in \mathcal{E}_f$ and $E' \in \mathcal{E}_{\neg f}$ such that $N \models init(E) \wedge init(E')$. This implies that $init(E) = init(E')$. By Theorem 1, both M_E and $M_{E'}$ are models of $comp(T)$ and $poss$. So by (21), both of them satisfy $succ(f) \equiv \varphi$. But this is impossible because $M_E \models succ(f)$, $M_{E'} \models \neg succ(f)$, and since $init(E) = init(E')$, $M_E \models \varphi$ iff $M_{E'} \models \varphi$. Therefore, (22) must hold as well.

Except for the fact that we uses causal propositions, the translation to logic programs given in this section is very similar to the one given in [9, 2] for causal logic of (Turner [9]). In fact, Turner [9] shows that there is a mapping from our definite clausal causal theories to his causal theories. However, the clausal causal theories here can have disjunctions in the head.

6 Some examples

Using Theorem 2, we can compute action preconditions and fully instantiated successor state axioms[4] using systems like Smodel [6] and dlv [5]. In fact, all examples in this paper have been checked using Smodel for those that do not have disjunctions.

Example 3. Consider again the suitcase domain in Example 1. The logic program rules corresponding to the causal theory are as follows:

$$\begin{aligned}
\neg poss &\leftarrow not\ \neg init(up1)\\
ctsucc(up1) &\leftarrow poss\\
ctinit(open) &\leftarrow init(up1), init(up2)\\
ctsucc(open) &\leftarrow not\ \neg succ(up1), not\ \neg succ(up2)
\end{aligned}$$

The logic program consisting of these rules and the domain independent ones have the following four answer sets that contain $poss$:

$$E_1 = \{poss, \neg init(up1), \neg init(up2), init(open), succ(up1), succ(open), ctsucc(up1)\},$$
$$E_2 = \{poss, \neg init(up1), \neg init(up2), \neg init(open), succ(up1), \neg succ(up2), \neg succ(open), ctsucc(up1)\},$$
$$E_3 = \{poss, \neg init(up1), init(up2), \neg init(open), succ(up1), succ(up2), succ(open), ctsucc(up1), ctsucc(open)\},$$
$$E_4 = \{poss, \neg init(up1), init(up2), init(open), succ(up1), succ(up2), succ(open), ctsucc(up1), ctsucc(open)\}.$$

[4] Successor state axioms in [7] quantify over actions. Here we deal with actions one at a time. So what we are interesting in here are successor state axioms with action variables fully instantiated.

Since

$$\begin{aligned}
init(E_1) &= \neg init(up1) \wedge \neg init(up2) \wedge init(open),\\
init(E_2) &= \neg init(up1) \wedge \neg init(up2) \wedge \neg init(open),\\
init(E_3) &= \neg init(up1) \wedge init(up2) \wedge \neg init(open),\\
init(E_4) &= \neg init(up1) \wedge init(up2) \wedge init(open).
\end{aligned}$$

Thus by Theorem 2, we have

$$poss \equiv init(E_1) \vee init(E_2) \vee init(E_3) \vee init(E_4),$$

which is equivalent to $poss \equiv \neg init(up1)$. We can similarly use the theorem to compute successor state axioms. We consider the fluent *open*. Others are similar. Notice that $\mathcal{E}_{open}$, the set of the answer sets containing $succ(open)$, is $\{E_1, E_3, E_4\}$. After some simplifications, $init(E_1) \vee init(E_3) \vee init(E_4)$ is

$$\neg init(up1) \wedge (init(up2) \vee init(open)). \tag{23}$$

Now $\mathcal{E}_{\neg open}$ is $\{E_2\}$, and $init(E_2)$ is

$$\neg init(up1) \wedge \neg init(up2) \wedge \neg init(open). \tag{24}$$

It can be seen that $\models (23) \supset (24)$, so we have the following successor state axiom for *open*:

$$poss \supset succ(open) \equiv \neg init(up1) \wedge (init(up2) \vee init(open)),$$

which is equivalent to

$$poss \supset succ(open) \equiv (init(up2) \vee init(open))$$

because $poss \equiv \neg init(up1)$.

The following example, adapted from [3], shows a causal rule that can entail implicit action preconditions.

Example 4. In this domain, we have two fluents: w (the agent is walking) and a (the agent is alive), and a causal rule which says that if the agent is not alive, then it is not walking:

$$\begin{aligned}
&\neg init(a) \supset \mathit{cfinit}(w),\\
&\neg succ(a) \supset \mathit{cfsucc}(w).
\end{aligned}$$

Now consider the action *start-walk* that initiates walking:

$$\begin{aligned}
&poss \supset \neg init(w),\\
&poss \supset \mathit{ctsucc}(w),
\end{aligned}$$

where the first one says that for this action to be possible, the agent should not already be walking, and the second axiom says that this action, if possible, causes the agent to be walking.

The logic program rules that correspond to the above axioms are as follows:

$$\begin{aligned} cfinit(w) &\leftarrow \neg init(a) \\ cfsucc(w) &\leftarrow not\ succ(a) \\ \neg poss &\leftarrow not\ init(w) \\ ctsucc(w) &\leftarrow poss \end{aligned}$$

When put together with the domain independent rules, this logic program yields a unique answer set that is consistent and containing *poss*:

$$S = \{poss, init(a), \neg init(w), succ(a), succ(w), ctsucc(w)\}.$$

So according to Theorem 2, we have the following action precondition axiom:

$$poss \equiv init(a) \wedge \neg init(w),$$

where the condition $init(a)$ is the implicit action precondition entailed by the causal rule.

The following example, adapted from [4], illustrates our translation for an indeterminate action.

Example 5. **(Reiter's Dropping Pin Example)** In this domain, an agent can drop a pin on a checkerboard and this causes the pin to land either inside a white square, inside a black square, or touching both.

To axiomatize this domain, we introduce two fluents: *white* (all or part of the pin is in a white square), *black* (all or part of the pin is in a black square). Now consider action *drop* that drops the pin on the checkerboard. It's effect is given by the following effect axiom:

$$\begin{aligned} poss \supset\ & ctsucc(white) \wedge cfsucc(black) \vee \\ & cfsucc(white) \wedge ctsucc(black) \vee \\ & ctsucc(white) \wedge ctsucc(black), \end{aligned}$$

which is logically equivalent to the following three effect axioms:

$$\begin{aligned} poss &\supset ctsucc(white) \vee ctsucc(black), \\ poss &\supset ctsucc(white) \vee cfsucc(black), \\ poss &\supset cfsucc(white) \vee ctsucc(black). \end{aligned}$$

For the causal theory consisting only of the above three axioms, the corresponding logic program will consist of the following rules together with the domain independent rules about fluents *white* and *black*:

$$\begin{aligned} ctsucc(white) \vee ctsucc(black) &\leftarrow poss \\ ctsucc(white) \vee cfsucc(black) &\leftarrow poss \\ cfsucc(white) \vee ctsucc(black) &\leftarrow poss \end{aligned}$$

This program has 12 answer sets that contain $poss$. They are all possible unions of any of the following sets:

$$\{poss, init(white), \neg init(black)\},$$
$$\{poss, init(white), init(black)\},$$
$$\{poss, \neg init(white), init(black)\},$$
$$\{poss, \neg init(white), \neg init(black)\}$$

with any of the following sets:

$$\{succ(white), succ(black), ctsucc(white), ctsucc(black)\},$$
$$\{succ(white), \neg succ(black), ctsucc(white), cfsucc(black)\},$$
$$\{\neg succ(white), succ(black), cfsucc(white), ctsucc(black)\}.$$

Using Theorem 2, we can then compute that $poss \equiv true$, which means the action is always possible.

For fluent $white$, both $\bigvee_{E \in \mathcal{E}_{white}} init(E)$ and $\bigvee_{E \in \mathcal{E}_{\neg white}} init(E)$ are equivalent to $true$, so condition (21) in Theorem 2 does not hold, and that we do not have a successor state axiom for $white$. All we have are the two axioms (19) and (20):

$$poss \wedge succ(white) \supset true,$$
$$poss \wedge \neg succ(white) \supset true$$

which are tautologies. This means that the action is always indeterminate about fluent $white$, and symmetrically, about fluent $black$ as well.

7 Concluding remarks

We have provided a translation of so-called clausal causal theories of [3] to Gelfond and Lifschitz's logic programs with classical negation and disjunction. Our translation is in many ways similar to the one given in [9, 2]. Our key technical result is Theorem 2 that shows how action precondition and fully instantiated successor state axioms can be computed from the answer sets of the translated logic program. However, the main computational impediment in using this theorem to large causal theories is that the number of answer sets grows exponentially in terms of the number of fluents in the domain. For instance, if we add n irrelevant fluents to the suitcase domain, then the number of answer sets would be in the order of 2^n. There is probably nothing that we can do in the worst case. However, we are working on strategies that would handle cases where the number of "relevant" answer sets is much smaller.

Acknowledgments

We would like to thank the anonymous referees for their useful comments. This work was supported in part by the Research Grants Council of Hong Kong under Competitive Earmarked Research Grants HKUST6091/97E and HKUST6145/98E.

References

1. M. Gelfond, V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365-385, 1991.
2. V. Lifschitz. Action languages, answer sets and planning. In: *The Logic Programming Paradigm: a 25-Year Perspective*, Springer Verlag, 1999.
3. F. Lin. Embracing causality in specifying the indirect effects of actions. In: *Proc. IJCAI'95*, pages 1985-1991, 1995.
4. F. Lin. Embracing causality in specifying the indeterminate effects of actions. In: *Proc. AAAI'96*, pages 670-676, 1996.
5. N. Leone, G. Pfeifer and W. Faber. The dlv project: A disjunctive datalog system, http://www.dbai.tuwien.ac.at/proj/dlv/
6. I. Niemelä, P. Simons. Efficient implementation of the well-founded model and stable model semantics. In: *Proc. JICLP'96*, pages 289-303, 1996.
7. R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 418–420. Academic Press, San Diego, CA, 1991.
8. H. Turner. Representing actions in logic programs and default theories: A situation calculus approach. *Journal of Logic Programming*, 31(1-3): 245-298, 1997.
9. H. Turner. A logic of universal causation. *Artificial Intelligence*, 1998.

Appendix Proof of Theorem 1

Theorem 1. *Let T be a clausal causal theory and πT its corresponding extended disjunctive logic program. We have:*

1. *If N is a model of $comp(T)$ and $poss \in N$, then $S_N = N \cup N^{\neg}$ is a consistent answer set of πT, where*

$$N^{\neg} = \{\neg init(f) \mid init(f) \notin N\} \cup \{\neg succ(f) \mid succ(f) \notin N\}.$$

2. *If E is a consistent answer set of πT with $poss \in E$, then $M_E = B_T \cap E$ is a model of $comp(T)$, where B_T is the set of atoms in T.*

Proof. For any rule R, the rule obtained from R by deleting all default literals in its body is written as R^+.

1. Assume that $N \models comp(T)$, then by the construction of S_N, S_N does not contain any complementary literals. We can directly verify that S_N is a model of πT^{S_N} from $N \models comp(T)$: for instance, the rule $(9)^+$: $succ(f) \leftarrow init(f), poss$ is in πT^{S_N} if $cfsucc(f)$ is not in S_N. If $init(f) \in S_N$, then $init(f) \in N$ and $cfsucc(f) \notin N$ since S_N and N have the same set of $\mathcal{I}$-atoms and $\mathcal{S}$-atoms. Since N satisfies the inertia axioms, we have $succ(f) \in N$. Thus, $succ(f) \in S_N$. This implies that the rule $(9)^+$: $succ(f) \leftarrow init(f), poss$ in πT^{S_N} is satisfied by S_N ($poss$ is in S_N). The satisfiability of other rules in πT^{S_N} can be verified similarly.

To show that S_N is an answer set of πT, it is enough to show that S_N is a minimal consequence set of πT^{S_N}. Let E_0 is a minimal consequence set of πT^{S_N} such that $E_0 \subseteq S_N$.

It suffices to show that $S_N \subseteq E_0$:

Consider eight possible cases:

Case 1. $init(f) \in S_N$: then $\neg init(f) \notin S_N$ and thus the rule $(12)^+$: $init(f) \leftarrow$ is in πT^{S_N}. Hence, $init(f) \in E_0$.

Case 2. $\neg init(f) \in S_N$, then $init(f) \notin S_N$ and thus the rule $(13)^+$: $\neg init(f) \leftarrow$ is in πT^{S_N}. Hence, $\neg init(f) \in E_0$.

Case 3. $ctinit(f) \in S_N$: then $ctinit(f) \in N$. Since $N \models precom(T)$, there exists an axiom of form (3): $l_1 \wedge \cdots \wedge l_k \wedge c_{t+1} \wedge \ldots \wedge c_n \supset ctinit(f) \vee c_1 \vee \cdots \vee c_t$ such that $l_1, \ldots, l_k, c_{t+1}, \ldots, c_n \in N$ but $c_1, \ldots, c_t \notin N$. Let this axiom of form (3) be translated into the rule R in πT. Then the rule $R^+ = R$ is in πT^{S_N}. This implies one of literals in $ctinit(f) \vee c_1 \vee \cdots \vee c_t$ should be in E_0. Since $c_1, \ldots, c_t \notin S_N$ and $E_0 \subseteq S_N$, $c_1, \ldots, c_t \notin E_0$. Thus, it forces $ctinit(f) \in E_0$.

Case 4. $cfinit(f) \in S_N$: similar to the Case 3, we can prove $cfinit(f) \in E_0$.

Case 5. $ctsucc(f) \in S_N$: then $ctsucc(f) \in N$. Since N is a model of $precomp(T)$, there is at least one axiom in T such that (1) the left side of the axiom is true, (2) $ctsucc(f)$ appears in the right side of the axiom, (3) the other disjuncts except for $ctsucc(f)$ are not in N. Therefore, there are two possible subcases:

Subcase 1. there exists an axiom of form (2): $poss \wedge l_1 \wedge \cdots \wedge l_n \supset ctsucc(f) \vee \tilde{c}_1 \vee \cdots \vee \tilde{c}_m$ such that $poss, l_1, \ldots, l_n \in N$ and $\tilde{c}_1, \ldots, \tilde{c}_m \notin N$. This means the rule $ctsucc(f) \vee \tilde{c}_1 \vee \cdots \vee \tilde{c}_m \leftarrow l_1, \ldots, l_n, poss$ is in πT^{S_N} with $l_1, \ldots, l_n \in S_N$ but $\tilde{c}_1, \ldots, \tilde{c}_m \notin E_0$. Thus it forces $ctsucc(f) \in E_0$.

Subcase 2. there exists an axiom of form (4): $\tilde{l}_1 \wedge \cdots \wedge \tilde{l}_k \wedge \tilde{c}_{t+1} \wedge \cdots \wedge \tilde{c}_n \supset ctsucc(f) \vee \tilde{c}_1 \vee \cdots \vee \tilde{c}_t$ such that $\tilde{l}_1, \ldots, \tilde{l}_k, \tilde{c}_{t+1}, \ldots, \tilde{c}_n \in N$ and $\tilde{c}_1, \ldots, \tilde{c}_t \notin N$. Then $\neg\tilde{l}_1, \ldots, \neg\tilde{l}_k \notin S_N$. Thus, we know that the rule $ctsucc(f) \vee \tilde{c}_1 \vee \cdots \vee \tilde{c}_t \leftarrow \tilde{c}_{t+1}, \ldots, \tilde{c}_n$ is in πT^{S_N}. Thus we also have $ctsucc(f) \in E_0$.

Case 6. $cfsucc(f) \in S_N$: similar to the Case 5.

Case 7. $succ(f) \in S_N$: there are again two subcases:

Subcase 1. $ctsucc(f) \in S_N$: by Case 5, $ctsucc(f) \in E_0$. Since the rule $(7)^+ =(7)$: $succ(f) \leftarrow ctsucc(f)$ is in πT^{S_N}. Thus, $succ(f) \in E_0$.

Subcase 2. $ctsucc(f) \notin S_N$: then $ctsucc(f) \notin N$. since $poss \in N$, we have $init(f) \in N$ and $cfsucc(f) \notin N$. Hence, $init(f) \in S_N$ and $cfsucc(f) \notin S_N$. The last fact means that the rule $(16)^+$: $succ(f) \leftarrow init(f)$ is in πT^{S_N}. By Case 1, $init(f) \in E_0$. Therefore, $succ(f) \in E_0$.

Case 8. $\neg succ(f) \in S_N$: according to the definition of S_N, $succ(f) \notin N$. There are two possible subcases:

Subcase 1. $init(f) \notin N$ and $ctsucc(f) \notin N$: then $\neg init(f) \in S_N$ and $ctsucc(f) \notin S_N$. This means the rule $(10)^+$ is in πT^{S_N}. Therefore, $\neg succ(f) \in E_0$.

Subcase 2. $cfsucc(f) \in N$: since the rule $(8)^+ = (8)$: $\neg succ(f) \leftarrow$ is in πT^{S_N}. Thus, $\neg succ(f) \in E_0$.

Combining the eight cases above, we have $E_0 = S_N$. That is, S_N is an answer sets of πT.

2. If E is an answer set of πT, we want to show $M_E = B_T \cap E$ is a model of $comp(T)$.

It can be directly verified that M_E satisfies T and the caused-axioms (see the end of Section 3). Now we remain to prove M_E satisfies the inertial axiom for *succ*, that is,

$$M_E \models ctsucc(f) \vee (init(f) \wedge \neg cfsucc(f)) \equiv succ(f).$$

By (7) and (9), $M_E \models ctsucc(f) \vee (init(f) \wedge \neg cfsucc(f)) \supset succ(f)$. For the opposite implication, if $succ(f) \in E$, $cfsucc(f)$ could not be in E. Thus, if $succ(f) \in M_E$ and $ctsucc(f) \notin M_E$, then the rule $(9)^+$: $succ(f) \leftarrow init(f), poss$ is in πT^E and $succ(f)$ can only be supported by this rule in πT^E. That is, $init(f)$ must be in M_E. This implies $M_E \models succ(f) \supset ctsucc(f) \vee (init(f) \wedge \neg cfsucc(f))$.

Now we prove $M_E \models precomp(T)$, where $precomp(T) = Circ(T; \mathcal{C})$.

Suppose N is a model of $precomp(T)$ such that $N \subseteq M_E$. Then the following conditions are satisfied:

(1) N and M_E have the same sets of $\mathcal{I}$-atoms and $\mathcal{S}$-atoms,

(2) every $\mathcal{C}$-atom in N is also in M_E and

(3) $poss \in N$.

We claim the following two facts:

Fact 1. S_N is also a model of πT^E and

Fact 2. $S_N \subseteq E$.

Fact 1 follows directly from the conditions (1)-(3) above. To convince Fact 2, first note that $S_N = N \cup N^\neg$. It is obvious that $N \subseteq M_E \subseteq E$. It needs to show that $N^\neg \subseteq E$. If fact, for any $L \in N^\neg$, then there are two possible cases by the definition of $N^\neg$:

Case 1. $L = \neg init(f)$ such that $init(f) \notin N$: By the condition (1) above, $init(f) \notin M_E$. Thus, $init(f) \notin E$. This implies $\neg init(f) \in E$ since πT^E should contain the rule $(11)^+$.

Case 2. $L = \neg succ(f)$ such that $succ(f) \notin N$. This case can again be divided into two subcases:

Subcase 1. $L = \neg succ(f)$ such that $init(f) \notin N$ and $ctsucc(f) \notin N$: Similar to Case 1, $\neg init(f) \in E$. If we have shown that $ctsucc(f) \notin E$, then the rule $(10)^+$: $\neg succ(f) \leftarrow \neg init(f), poss$ is in πT^E. Thus, $\neg succ(f) \in E$.

It remains to show that $ctsucc(f) \notin E$ under the assumption of the Subcase 1: On the contrary, suppose that $ctsucc(f) \in E$ for some fluent f. Then $succ(f) \in E$ by (7). This implies $succ(f) \in N$. On the other hand, the left sides of both $(7)^+$ and $(9)^+$ are not satisfied by N. Thus, $succ(f) \notin N$, a contradiction.

Subcase 2. $L = \neg succ(f)$ such that $cfsucc(f) \in N$: then $cfsucc(f) \in M_E$ and thus, $cfsucc(f) \in E$. By (8), $\neg succ(f) \in E$.

Combining the three cases above, we know that the Fact 2 is convinced.

By the minimality of E, we have $S_N = E$. Therefore, $N = M_E$. This means that M_E is a model of $precomp(T)$. Note that $poss \in M_E$, we have thus shown that M_E is a model of $comp(T)$.