Causal Theories of Actions Revisited

Fangzhen Lin
Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clear Water Bay,
Kowloon, Hong Kong.
Email: flin@cs.ust.hk

Mikhail Soutchanski
Department of Computer Science
Ryerson University
245 Church Street, ENG281
Toronto, ON, M5B 2K3, Canada
Email: mes@scs.ryerson.ca

Abstract

It has been argued that causal rules are necessary for representing both implicit side-effects of actions and action qualifications, and there have been a number different approaches for representing causal rules in the area of formal theories of actions. These different approaches in general agree on rules without cycles. However, they differ on causal rules with mutual cyclic dependencies, both in terms of how these rules are supposed to be represented and their semantics. In this paper we show that by adding one more minimization to Lin's circumscriptive causal theory in the situation calculus, we can have a uniform representation of causal rules including those with cyclic dependencies. We also demonstrate that sometimes causal rules can be compiled into logically equivalent (under a proposed semantics) successor state axioms even in the presence of cyclical dependencies between fluents.

Introduction

Reiter (2001) argues that to solve many reasoning problems about actions, it is convenient to work with the precondition axioms and the successor state axioms. For each *action* function $A(\vec{x})$, where $\vec{x}$ is a tuple of distinct variables, a precondition axiom (PA) has a syntactic form

$$Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s), \qquad (1)$$

which says that an action $A(\vec{x})$ is possible in situation s if and only if $\Pi_A(\vec{x}, s)$ holds in s, where $\Pi_A(\vec{x}, s)$ is a formula with free variables among $\vec{x}$ and s.[1]

For each fluent $F(\vec{x}, s)$, a successor state axiom (SSA) has a syntactic form

$$F(\vec{x}, do(a, s)) \equiv [\exists \vec{y}_i](a = PosAct_i(\vec{t}_i) \wedge \phi_i^+(\vec{x}, \vec{y}_i, s)) \vee$$
$$F(\vec{x}, s) \wedge \neg[\exists \vec{z}_j](a = NegAct_j(\vec{t'}_j) \wedge \phi_j^-(\vec{x}, \vec{z}_j, s)), \quad (2)$$

where $\vec{t}_i$ is a tuple of terms whose variables are in $\vec{x} \cup \vec{y}_i$ and $\vec{t'}_j$ a tuple of terms whose variables are in $\vec{x} \cup \vec{z}_j$. This axiom says that a fluent F is true in a successor situation $do(a, s)$ iff either a has a positive effect on it ($PosAct_i$ holds) under a context ϕ_i^+ and the context holds in the current situation s or it is true in s and a is not an action that has a negative effect on it under a context that is true in s.

If for each action we have an action precondition axiom, and for each fluent a successor state axiom, then we have provided a complete axiomatization of the effects of the actions in the domain. To date, much of the work on cognitive robotics (e.g. (Levesque et al. 1997; Reiter 2001)) has assumed that a dynamic domain is axiomatized this way.

However, it has been observed that sometimes, an axiomatizer has to start not with PAs and SSAs, but with a different set of axioms representing (domain) state constraints (Finger 1986). For example, (McIlraith 2000) argues that it is more convenient for an axiomatizer to start with state constraints that characterize a complex technical or software system. She also demonstrates when a syntactically restricted set of situation calculus constraints and effect axioms can be compiled into a set of SSAs. From another perspective, (Baader et al. 2005) investigate how reasoning about actions can be carried out in description logics. The authors embed reasoning problems from the general situation calculus into a description logic setting. For the sake of simplicity, they consider only a special case of domain constraints that correspond to a set of acyclic definitions between concept-like fluents. In description logics, this set of axioms is called an *acyclic TBox*. Axioms in a TBox express general knowledge about a domain and may include both terminological definitions and constraints that should hold after execution of arbitrary actions. It is important to observe that in description logics, it is the set of state constraints in TBox that is a primary concern of an axiomatizer.

State (or domain) constraints are traditionally facts that are true in every possible state. In the situation calculus, they are normally represented as first-order sentences with universal quantifiers over situations. For instance, the fact that no object x can be at two different locations l, l' in the same situation s can be represented as:

$$at(x, l, s) \wedge at(x, l', s) \supset l = l'.$$

However, surprisingly, not all state constraints are created equal. (Ginsberg and Smith 1988) (see also (Lin and Reiter 1994)) first point out that while some of them contribute to indirect effects of actions (called *ramification state constraints* in (Lin and Reiter 1994)), others serve as implicit qualifications on actions (called *qualification state constraints* in (Lin and Reiter 1994)). For instance, consider

[1]Here and subsequently, all free variables (typically written in lower case letters) including *object* variable $\vec{x}$, *situation* variable s, and a variable of sort *action* a are implicitly $\forall$-quantified at front of formulas.

the action $move(x,l)$ that moves the object x to the location l

$$Poss(move(x,l),s) \supset at(x,l,do(move(x,l),s)).$$

(If the action $move(x,l)$ is possible (executable), then after it is performed the object x will be at location l.) Then the above state constraint about uniqueness of a location is a ramification one, for it should be used to imply the following indirect effect of $move(x,l)$:

$$Poss(move(x,l),s) \supset \\ l' \neq l \supset \neg at(x,l',do(move(x,l),s)).$$

Now suppose that each location can have just one object:

$$at(x,l,s) \wedge at(y,l,s) \supset x = y.$$

Then this constraint about uniqueness of an object occupying l should be a qualification one, for it should be used to derive the following qualification on the action:

$$Poss(move(x,l),s) \supset \neg(\exists y) y \neq x \wedge at(y,l,s).$$

(One cannot move an object to a location which is already occupied by another object, for otherwise, the last state constraint will be violated.)

What is disturbing here is that although our intuitions about how the two state constraints should be used are different, they are represented in the same way. It seems clear that these two kind of state constraints are fundamentally different, and should be represented in fundamentally different ways.

Moreover, several researchers argued that the indirect effects of actions should be represented differently (e.g. (Baral 1995; Lin 1995; McCain and Turner 1995; Sandewall 1994; Thielscher 1995)). In particular, Lin (1995) argued that the indirect effects of actions cannot be faithfully described using ramification state constraints alone, and proposed to use *causal rules* to specify the constraints. The method proposed in (Lin 1995) is illustrated with several examples of how causal rules together with direct effect axioms can be successfully compiled into PAs and SSAs, and later implemented (Lin 2003). Once this compilation step has been completed, the resulting BAT can be subsequently used for reasoning about actions. However, the general results about applicability of this approach are stated only for acyclic (stratified) sets of causal rules. As soon as there are fluents with mutual causal dependencies, the approach proposed in (Lin 1995) is no longer applicable. The goal of this paper is to elaborate the approach proposed in (Lin 1995), so that an arbitrary finite set of causal rules can be handled as well. Ultimately, we are looking for computational mechanisms that can take an arbitrary finite set of causal rules and a finite set of direct effect axioms on the input and can compile them into a set of PAs and a set of SSAs. This paper can be considered a first step in this direction. Subsequently, we concentrate on solving the ramification problem only, and do not consider explicit action qualification axioms.

The paper is organized as follows. Section 2 briefly reviews the situation calculus. Section 3 discusses our approach in more details. Section 4 contains several simple examples to illustrate our method. Section 5 includes discussions and comparisons with previously proposed solutions to the ramification problem. Section 6 concludes this paper and points to some future work.

The Situation Calculus

The situation calculus was invented by McCarthty (1963; 1969). The version of the situation calculus that we use here follows (Reiter 2001; Lin 2008).

Briefly, the situation calculus is a many-sorted logical theory for reasoning about actions and their effects. We assume sorts for *situation*, *action*, *fluent*, *truth-value*, and *object*. We use the binary predicate $Holds(f,s)$ to say that a fluent f holds in s. Notice that in the introduction, we wrote, for instance, $at(x,l,s)$ instead of $Holds(at(x,l),s)$. We consider the former to be a shorthand for the latter. We shall continue to do so in an effort to improve the readability of our formulas. Formally, if F is a fluent name of arity $object^n \rightarrow fluent$, then we define the expression $F(t_1,...,t_n,s)$ to be a shorthand for the formula $Holds(F(t_1,...,t_n),s)$, where $t_1,\cdots,t_n$ are terms of sort $object$, and s is a term of sort $situation$.

A distinguished constant S_0 is used to denote the *initial situation*, and function $do(a,s)$ denotes the situation that results from performing action a in situation s. Under Reiter's foundational axioms for the situation calculus, every situation corresponds uniquely to the sequence of actions that is performed in S_0 to reach this situation.

In this paper, we also use a ternary predicate $Caused(f,v,s)$, where f is of sort *fluent*, v of *truth-value*, and s of *situation*. Informally, it means that f is caused to have the truth value v in s. We assume two constants $\mathcal{T}$ and $\mathcal{F}$ of sort *truth-value*. Later, we'll have axioms saying that these truth values are distinct and the only ones.

In the following, we call a formula $\varphi(s)$ a *state formula* or interchangeably *uniform* in situation argument s if it does not mention the predicates $Poss$ or $Caused$, it does not quantify over variables of sort situation, and it has no occurrences of situation terms other than the variable s (see (Reiter 2001)).

In the action precondition axiom (1) above, $\Pi_A(\vec{x},s)$ must be a formula that is uniform in situation argument s. Similarly, we require the context conditions $\phi_i^+(\vec{x},\vec{y}_i,s)$ and $\phi_j^-(\vec{x},\vec{z}_j,s)$ in the SSA (2) be formulas uniform in s.

The Method

We consider causal theories of the following form:

- A set of direct action effect axioms of the form:
$$\Phi(s) \supset Caused(F(\vec{x}),v,do(A(\vec{y}),s)), \quad (3)$$
where Φ is a formula uniform in s, $F(\vec{x})$ is a fluent, $A(\vec{y})$ is an action, and v is a variable of sort *truth value*. Compared to (Lin 1995), we omit the predicate $Poss(A(\vec{y}),s)$ as a precondition of a direct effect, following (Reiter 2001).
- Causal rules of the form:
$$\Phi(s) \supset Caused(F(\vec{x}),v,s), \quad (4)$$
where $\Phi(s)$ is a formula uniform in s, and F a fluent. Compared to (Lin 1995), we do not allow the predicate $Caused$ in the premises, but allow only arbitrary state formulas. We believe this simplifies the task of a knowledge engineer who is responsible for writing causal rules.

If both arbitrary state formulas and causation statements would be allowed in premises of causal rules, but there is no recipe which of them should be used when, then this permissiveness could create uncertainty for a knowledge engineer.

As in (Lin 1995), there are general axioms about $Caused$:

$$\mathcal{T} \neq \mathcal{F} \wedge \forall v. v = \mathcal{T} \vee v = \mathcal{F}, \quad (5)$$

$$Caused(f, \mathcal{T}, s) \supset Holds(f, s), \quad (6)$$

$$Caused(f, \mathcal{F}, s) \supset \neg Holds(f, s), \quad (7)$$

where (5) is the domain closure axiom for sort *truth-value*.

For each fluent $F(\vec{x})$, the generic frame axiom, called *pseudo-successor state* axiom, is

$$Holds(F(\vec{x}), do(a, s)) \equiv Caused(F(\vec{x}), \mathcal{T}, do(a, s)) \vee Holds(F(\vec{x}), s) \wedge \neg Caused(F(\vec{x}), \mathcal{F}, do(a, s)). \quad (8)$$

From this axiom, we see that to get a real SSA as in (Reiter 2001) for each fluent, we need to derive some definitions of $Caused(F(\vec{x}), \mathcal{T}, do(a, s))$ and $Caused(F(\vec{x}), \mathcal{F}, do(a, s))$ in terms of two state formulas on s, respectively. To achieve this, Lin (1995) proposed to circumscribe the $Caused$ predicate in a theory consisting of the above direct effect axioms (3) and causal rules (4), but not the pseudo-successor state (8) and general axioms (5)-(7). While this approach works for acyclic causal rules such as those in the suitcase example from (Lin 1995), it does not work when there are cycles as we will see from the examples in the next section.

We propose here to add a second minimization, and show that this solves the problem of cyclic causal rules. To present our approach, we first make precise Lin's approach.

Given a set T_0 of the direct effect axioms and causal rules of the forms (3) and (4), respectively, Lin's causal theory, written $\mathcal{C}_l(T_0)$ below, consists of the general axioms (5) - (7) about $Caused$, the pseudo-successor state axioms (8), and $CIRC(T_0; Caused)$, the circumscription of $Caused$ in T_0 with all other predicates fixed. (See (McCarthy 1986; Lifschitz 1985; 1994; Doherty, Łukaszewicz, and Szałas 1997) for details about circumscription.)

Since the formulas (3) and (4) in T_0 are Horn in the $Caused$ predicate, $CIRC(T_0; Caused)$ can be computed by a simple Clark predicate completion (Clark 1978; Reiter 1982) to yield the following formulas, two for each fluent F:

$$Caused(F(\vec{x}), v, S_0) \equiv \Phi_0(S_0), \quad (9)$$

$$Caused(F(\vec{x}), v, do(a, s)) \equiv \Phi_1(do(a, s)), \quad (10)$$

where Φ_0 and Φ_1 are computed as follows. Let the following be the list of direct effect axioms about F:

$$\phi_1(s) \supset Caused(F(\vec{x}), v, do(A_1(\vec{y_1}), s)),$$
$$\ldots$$
$$\phi_k(s) \supset Caused(F(\vec{x}), v, do(A_k(\vec{y_k}), s))$$

and the following the list of causal rules about F:

$$\psi_1(s) \supset Caused(F(\vec{x}), v, s),$$
$$\ldots$$
$$\psi_m(s) \supset Caused(F(\vec{x}), v, s).$$

Then $\Phi_0(S_0)$ is

$$\psi_1(S_0) \vee \cdots \vee \psi_m(S_0),$$

and $\Phi_1(do(a, s))$ is

$$[\phi_1(s) \wedge a = A_1(\vec{y_1})] \vee \cdots \vee [\phi_k(s) \wedge a = A_k(\vec{y_k})] \vee \psi_1(do(a, s)) \vee \cdots \vee \psi_m(do(a, s)).$$

Notice that if $m = 0$ (meaning no causal rules about F), then Φ_0 is $\perp$, where $\perp$ stands for $L \wedge \neg L$. If both $m = 0$ and $k = 0$, then Φ_1 is $\perp$.

In the following, given a set T_0 of direct effect axioms and causal rules of the forms (3) and (4), respectively, we denote by T_1 the set of equivalences (9) and (10).

We can now state our method as below:

1. Let T_1' be the result of replacing each atom of the form $Holds(F(\vec{t}), do(a, s))$ in T_1 by the right hand side of (8), i.e., with
$Caused(F(\vec{t}), \mathcal{T}, do(a, s)) \vee$
$Holds(F(\vec{t}), s) \wedge \neg Caused(F(\vec{t}), \mathcal{F}, do(a, s))$.
2. Our second minimization is then to circumscribe $Caused$ in T_1' with all the other predicates fixed, $CIRC(T_1'; Caused)$.
3. Our final causal action theory $\mathcal{CAT}\ (T_0)$ will then consist of the general axioms (5) - (7) about $Caused$, the pseudo-successor state axioms (8), and $CIRC(T_1'; Caused)$.

The following result says that our new causal theory is stronger than the one in (Lin 1995).

Theorem 1 $\mathcal{CAT}\ (T_0) \models \mathcal{C}_l(T_0)$

Proof: This follows from the following entailments:

$$CIRC(T_1'; Caused) \models T_1',$$
$$\{(8) \mid F \text{ is a fluent}\} \models T_1 \equiv T_1',$$
$$\models CIRC(T_0; Caused) \equiv T_1.$$

■

Thus, if the method of (Lin 1995) yields a successor state axiom for each fluent, as when there are no cycles in causal rules, so will our new method. In this sense, our new approach indeed extends the one in (Lin 1995).

Examples

In this section, we would like to consider a few examples explaining our proposal. First of all, as mentioned above, for the suitcase example from (Lin 1995), our method yields exactly the same SSAs as in (Lin 1995).

Similarly, one can verify that for the complex electric circuit[2] from Figure 2.2 in (Thielscher 2000), our method also yields a SSA for each fluent.

Subsequently, we concentrate on examples of $\mathcal{CAT}$ where our new approach can produce SSAs, but the method from (Lin 1995) is not strong enough to do that.

[2]This circuit consists of a battery connected to a separate switch sw_0 that controls n parallel sub-circuits. Each sub-circuit contains its own switch sw_i connected to a light bulb l_i. If sw_0 is not up, then there is no light in any of the bulbs no matter what are the positions of their switches, but if sw_0 is up, then the fluent l_i is true if and only if sw_i is up.

A Chain Reaction

A chain reaction is any self-sustaining physical or chemical process such that its by-products cause the process to continue (with or without acceleration). There are many examples, but one of the simplest is an example of a fire started in a large pile of matches. Once a match inside a pile has been lit, it causes other surrounding matches to burn, and so on. To reason about fire in a pile, let x vary over the whole piles of matches, and let $fire(x,s)$ be a fluent that can become true after executing an action $ignite(x)$, but if it is true, then it becomes false after doing $extinguish(x)$ action. For the purposes of this example, we do not quantify over individual matches. In this example, a theory T_0 includes two direct effect axioms

$\neg fire(x,s) \supset Caused(fire(x),\mathcal{T},do(ignite(x),s))$,
$fire(x,s) \supset Caused(fire(x),\mathcal{F},do(extinguish(x),s))$,

a single causal rule with a cycle (fluent depends on itself):

$fire(x,s) \supset Caused(fire(x),\mathcal{T},s)$.

It is easy to see that in this case $CIRC(T_0; Caused)$ yields the following:

$$\begin{array}{l} Caused(fire(x),v,S_0) \equiv\ v{=}\mathcal{T} \wedge fire(x,S_0), \\ Caused(fire(x),v,do(a,s)) \equiv \\ \qquad a{=}extinguish(x) \wedge v{=}\mathcal{F} \wedge fire(x,s) \vee \\ \qquad a{=}ignite(x) \wedge v{=}\mathcal{T} \wedge \neg fire(x,s) \vee \\ \qquad v{=}\mathcal{T} \wedge fire(x,do(a,s)). \end{array}$$

According with Step 2 of our method, we have to replace $fire(x,do(a,s))$ in the last formula with

$$\begin{array}{l} Caused(fire(x),\mathcal{T},do(a,s)) \vee \\ fire(x,s) \wedge \neg Caused(fire(x),\mathcal{F},do(a,s)). \end{array}$$

But this yields a theory T_1' with the predicate $Caused$ defined in terms of itself. Consequently, the single minimization $CIRC(T_0; Caused)$ is not strong enough to produce a SSA for the fluent $fire(x)$. However, the second minimization $CIRC(T_1'; Caused)$ yields the formulas

$$\begin{array}{l} Caused(fire(x),\mathcal{T},do(a,s)) \equiv \\ \qquad\qquad a{=}ignite(x) \wedge \neg fire(x,s), \\ Caused(fire(x),\mathcal{F},do(a,s)) \equiv \\ \qquad\qquad a{=}extinguish(x) \wedge fire(x,s). \end{array}$$

Using these definitions, we can easily obtain a SSA for the fluent $fire(x)$ from the pseudo-successor state axiom (8).

Two Gear Wheels

In this well-known example by Denecker *et al.* (Belleghem, Denecker, and Dupré 1998), there are two interlocked gear wheels. We characterize each with a fluent $gw(n)$ meaning that the n-th gear wheel is turning. There are actions to initiate/halt rotation of wheels: $turn(n)$ and $block(n)$, respectively.

$Caused(gw(n),\mathcal{T},do(turn(n),s))$,
$Caused(gw(n),\mathcal{F},do(block(n),s))$,

Since the gear wheels are interlocked, rotation of one of the gear wheels causes another one to rotate too, but if one of them halts, the second one must halt too.

$gw(1,s) \supset Caused(gw(2),\mathcal{T},s)$,
$gw(2,s) \supset Caused(gw(1),\mathcal{T},s)$,
$\neg gw(1,s) \supset Caused(gw(2),\mathcal{F},s)$,
$\neg gw(2,s) \supset Caused(gw(1),\mathcal{F},s)$.

Let T_0 be a conjunction of these six axioms. Then, skipping axioms (9) related to S_0, $CIRC(T_0; Caused)$ yields

$$\begin{array}{l} Caused(gw(1),v,do(a,s)) \equiv \\ \quad a{=}turn(1) \wedge v{=}\mathcal{T} \vee gw(2,do(a,s)) \wedge v{=}\mathcal{T} \vee \\ \quad a{=}block(1) \wedge v{=}\mathcal{F} \vee \neg gw(2,do(a,s)) \wedge v{=}\mathcal{F} , \\ Caused(gw(2),v,do(a,s)) \equiv \\ \quad a{=}turn(2) \wedge v{=}\mathcal{T} \vee gw(1,do(a,s)) \wedge v{=}\mathcal{T} \vee \\ \quad a{=}block(2) \wedge v{=}\mathcal{F} \vee \neg gw(1,do(a,s)) \wedge v{=}\mathcal{F} . \end{array}$$

As in the previous example, we observe that the first minimization does not allow us to compile direct effects and causal rules into SSAs. In Step 2, we replace $gw(1,do(a,s))$ and $gw(2,do(a,s))$ with the right hand sides of (8), and do some FO simplifications using general axioms about $Caused$. In the result, again skipping axioms (9) related to S_0, we get a theory T_1' that includes

$$\begin{array}{l} Caused(gw(1),v,do(a,s)) \equiv \\ \ v{=}\mathcal{T} \wedge \big(a{=}turn(1) \vee a{=}turn(2) \vee \\ \qquad Caused(gw(1),\mathcal{T},do(a,s)) \vee \\ \qquad gw(1,s) \wedge \neg Caused(gw(1),\mathcal{F},do(a,s)) \vee \\ \qquad gw(2,s) \wedge \neg Caused(gw(2),\mathcal{F},do(a,s)) \big) \vee \\ \ v{=}\mathcal{F} \wedge \big(a{=}block(1) \vee a{=}block(2) \vee \\ \qquad Caused(gw(1),\mathcal{F},do(a,s)) \vee \\ \qquad \neg gw(1,s) \wedge \neg Caused(gw(1),\mathcal{T},do(a,s)) \vee \\ \qquad \neg gw(2,s) \wedge \neg Caused(gw(2),\mathcal{T},do(a,s)) \big). \end{array}$$

and a similar axiom for $Caused(gw(2),v,do(a,s))$. In Step 3, we compute $CIRC(T_1'; Caused)$. This yields desirable definitions:

$$\begin{array}{l} Caused(gw(1),\mathcal{T},do(a,s)) \equiv \\ \quad a{=}turn(1) \vee a{=}turn(2) \vee \\ \ (gw(1,s) \vee gw(2,s)) \wedge \neg(a{=}block(1) \vee a{=}block(2)), \\ Caused(gw(1),\mathcal{F},do(a,s)) \equiv \\ \quad a{=}block(1) \vee a{=}block(2) \vee \\ (\neg gw(1,s) \vee \neg gw(2,s)) \wedge \neg(a{=}turn(1) \vee a{=}turn(2)). \end{array}$$

Again, to save space, we omit a similar axiom for $Caused(gw(2),v,do(a,s))$. Thus, in the two gear wheels example, we also computed successfully Reiter's SSAs.

Related Work

Much work has been done on incorporating causal rules into action theories. Virtually every major action formalism has been extended with causal rules. Some of these previous works are discussed in details in (Giordano and Schwind 2004). Since our focus in this paper is on cyclic causal rules, we'll just consider the work that can deal with cyclic causal rules.

Recall that under our proposal, a causal rule has the form

$$\Phi(s) \supset Caused(F(\vec{x}),v,s),$$

where $\Phi(s)$ is a state formula uniform in s, meaning that it mentions only $Holds(f,s)$ but not the $Caused$ predicate. Thus, the causal rule that *open* is caused to be true when the two switches are up in the suitcase example is represented as:

$$up_1(s) \wedge up_2(s) \supset Caused(open,\mathcal{T},s),$$

and that the two gears are interlocked and one turning causes the other to turn is represented as

$gw(1,s) \supset Caused(gw(2),\mathcal{T},s)$,
$gw(2,s) \supset Caused(gw(1),\mathcal{T},s)$,
$\neg gw(1,s) \supset Caused(gw(2),\mathcal{F},s)$,
$\neg gw(2,s) \supset Caused(gw(1),\mathcal{F},s)$.

Notice that these formulas all have the same form. Intuitively, the one in the suitcase example is acyclic because there is no rule connecting $open$ to up_1 or up_2. The ones in the gears example are cyclic. Formally, we can treat a set of causal rules of the form

$$l_1(s) \wedge \cdots \wedge l_n(s) \supset Caused(p, v, s) \tag{11}$$

as a logic program, where p is a fluent atom, and l_i's are fluent literals, and define its dependency graph and loops, similar to what Lee did for McCain and Turner's causal theories (Lee 2004).

In McCain and Turner's causal logic (1997), the rule from the suitcase example would be represented as

$$up_1 \wedge up_2 \Rightarrow open,$$

and rules from the two gears example as

$$\top \Rightarrow gw(1) \equiv gw(2), \tag{12}$$

where $\top$ stands for $(L \vee \neg L)$ and represents propositional tautology. Compared to our representation, we see that cyclic and acyclic causal rules are represented differently in McCain and Turner's formalism. The same can be said about various action languages (Giunchiglia et al. 2004) as they are all based on McCain and Turner's causal logic.

Denecker *et al.* (1998) proposed a approach based on inductive definitions. In their formalism, the causal rule in the suitcase example is represented as

$$caus(open) \leftarrow init(up_1) \wedge holds(up_2) \wedge \neg init(\neg up_2),$$
$$caus(open) \leftarrow init(up_2) \wedge holds(up_1) \wedge \neg init(\neg up_1),$$
$$caus(open) \leftarrow init(up_1) \wedge init(up_2),$$

and for the two gears example, the following rules:

$$caus(gw(1)) \leftarrow caus(gw(2)),$$
$$caus(gw(2)) \leftarrow caus(gw(1)),$$
$$caus(\neg gw(1)) \leftarrow caus(\neg gw(2)),$$
$$caus(\neg gw(2)) \leftarrow caus(\neg gw(1)).$$

Again we see that in this formalism, cyclic and acyclic causal rules need to be represented differently.

More recently, Strass and Thielscher (2010) considered a restricted causal language in the style of McCain and Turner's causal logic, but provided a different semantics in the style of Clark's completion (1978) and loop formulas (Lin and Zhao 2004). The causal rules from the two gears example are written as

$$\top : gw(1) \Rrightarrow gw(2),$$
$$\top : gw(2) \Rrightarrow gw(1),$$
$$\top : \neg gw(1) \Rrightarrow \neg gw(2),$$
$$\top : \neg gw(2) \Rrightarrow \neg gw(1).$$

In general, a causal rule in this formalism is of the form

$$\Phi : l_1 \Rrightarrow l_2,$$

where Φ can be an arbitrary fluent formula, but l_1 and l_2 must be fluent literals. As mentioned, the semantics of these rules are defined using Clark completion and loop formulas.

It is interesting to note that Lee (2004) did something similar by providing a translation from a subset of McCain and Turner's causal logic to logic program. When Lee's translation is applied to (12), it yields a set of rules very similar to the rules above.

In comparison, our proposal here allows for more general form of causal rules, and uses minimization instead of Clark's completion and loop formulas.

Concluding Remarks and Future Work

We have proposed to add a second minimization to the circumscriptive action theories in (Lin 1995). Intuitively, the original minimization in (Lin 1995) yields a closed-form solution for $Caused$ in terms of $Holds$. However when $Holds(f, do(a, s))$ is replaced by its pseudo-successor state axioms, the closed-form solution for $Caused$ may have some cycles which will then be eliminated by the proposed new minimization.

We have shown that our method is stronger than the original method in (Lin 1995) so that if the method in (Lin 1995) produces a set of successor state axioms, so will our new approach.

The main advantage of our method as compared to others that can handle cyclic causal rules is that we have used a uniform representation for both acyclic causal rules such as those in the suitcase example and cyclic ones such as those in the two gears example.

We plan to consider the following future work:

1. Show that when a set of causal rules of the form (11) has cycles, then the result of two minimizations can be captured by loop formulas as done in (Lee 2004) and (Strass and Thielscher 2010).

2. While for the two gears example, the various different approaches outlined above all yield the same results, it is worthwhile proving a result that formally relate, e.g. causal theories here and causal theories by McCain and Turner.

3. Implement a system similar to (Lin 2003) that can compile a causal theory into STRIPS-like systems and successor state axioms.

4. Define *actual cause* within our framework to capture properly concepts of causation and demonstrate that it conforms to intuition on examples from (Hopkins and Pearl 2007; Pearl 2009; Halpern and Pearl 2005).

Acknowledgments

We would like to thank Joohyung Lee and Vladimir Lifschitz for pointing to us some related work on cyclic causal rules under McCain and Turner's framework for causal action theories. The first author thanks the support from HK RGC under GRF 616208 and 616909. The second author thanks the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Department of Computer Science of the Ryerson University for providing partial financial support of this research.

References

Baader, F.; Lutz, C.; Miliĉić, M.; Sattler, U.; and Wolter, F. 2005. Integrating Description Logics and Action Formalisms: First Results. In *Proc. of AAAI-05*, 572–577.

Baral, C. 1995. Reasoning about Actions: Nondeterministic Effects, Constraints, and Qualification. In *Proc. IJCAI-95*, 2017–2023.

Belleghem, K. V.; Denecker, M.; and Dupré, D. T. 1998. A Constructive Approach to the Ramification Problem. In *Workshop on Reasoning about Actions at 10th ESSLLI-98*.

Clark, K. L. 1978. Negation as Failure. In Gallaire, H., and Minker, J., eds., *Logics and Databases*. New York: Plenum Press. 293–322.

Denecker, M.; Dupré, D. T.; and Belleghem, K. V. 1998. An Inductive Definition Approach to Ramifications. *Electron. Trans. Artif. Intell.* 2:25–67.

Doherty, P.; Łukaszewicz, W.; and Szałas, A. 1997. Computing Circumscription Revisited: A Reduction Algorithm. *J. Autom. Reason.* 18(3):297–336.

Finger, J. 1986. *Exploiting Constraints in Design Synthesis*. Ph.D. Dissertation, Stanford University, Palo Alto, CA.

Ginsberg, M. L., and Smith, D. E. 1988. Reasoning about Action ii: The Qualification Problem. *Artif. Intell.* 35(3):311–342.

Giordano, L., and Schwind, C. 2004. Conditional Logic of Actions and Causation. *Artif. Intell.* 157(1-2):239–279.

Giunchiglia, E.; Lee, J.; Lifschitz, V.; McCain, N.; and Turner, H. 2004. Nonmonotonic Causal Theories. *Artif. Intell.* 153(1-2):49–104.

Halpern, J. Y., and Pearl, J. 2005. Causes and Explanations: A Structural-model Approach. *British Journal of Philosophy of Science* 56:843–911. Available as UCLA Cognitive Systems Laboratory Technical Reports R-266-BJPS1 and R-266-BJPS2.

Hopkins, M., and Pearl, J. 2007. Causality and Counterfactuals in the Situation Calculus. *J. of Logic and Computation* 17(5):939–953. Preliminary version appeared in Proceedings of the 7th International Symposium on Logical Formalizations of Commonsense Reasoning (2005).

Lee, J. 2004. Nondefinite vs. Definite Causal Theories. In *Proc. 7th Int'l Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR-04)*, 141–153.

Levesque, H.; Reiter, R.; Lespérance, Y.; Lin, F.; and Scherl, R. 1997. GOLOG: A logic programming language for dynamic domains. *Journal of Logic Programming, Special issue on Reasoning about Action and Change* 31:59–84.

Lifschitz, V. 1985. Computing Circumscription. In *IJCAI*, 121–127.

Lifschitz, V. 1994. Circumscription. In *Handbook of logic in artificial intelligence and logic programming (vol. 3): Nonmonotonic Reasoning and Uncertain Reasoning*. Oxford University Press. 297–352.

Lin, F., and Reiter, R. 1994. State Constraints Revisited. *J. of Logic and Computation* 4(5):655–678.

Lin, F., and Zhao, Y. 2004. ASSAT: Computing Answer Sets of a Logic Program by SAT Solvers. *Artificial Intelligence* 157(1-2):115–137.

Lin, F. 1995. Embracing Causality in Specifying the Indirect Effects of Actions. In *IJCAI-95*, 1985–1993.

Lin, F. 2003. Compiling Causal Theories to Successor State Axioms and STRIPS-Like Systems. *J. Artif. Intell. Res. (JAIR)* 19:279–314.

Lin, F. 2008. Chapter 16: Situation Calculus. In Frank van Harmelen, V. L., and Porter, B., eds., *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*. Elsevier. 649 – 669.

McCain, N., and Turner, H. 1995. A Causal Theory of Ramifications and Qualifications. In *Proceedings of the 14th IJCAI*, 1978–1984.

McCain, N., and Turner, H. 1997. Causal Theories of Action and Change. In *AAAI/IAAI*, 460–465.

McCarthy, J., and Hayes, P. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence*, volume 4. Edinburgh University Press, Reprinted in (McCarthy 1990). 463–502.

McCarthy, J. 1963. Situations, actions and causal laws. Technical Report Technical Report Memo 2, Stanford University Artificial Intelligence Laboratory, Stanford, CA. Reprinted in Marvin Minsky, editor, S̈emantic information processing,, MIT Press, 1968.

McCarthy, J. 1986. Applications of Circumscription to Formalizing Common Sense Knowledge. *Artificial Intelligence* 28:89–116.

McCarthy, J. 1990. *Formalization of Common Sense: Papers by John McCarthy edited by V. Lifschitz*. Norwood, N.J.: Ablex.

McIlraith, S. A. 2000. Integrating Actions and State Constraints: A Closed-form Solution to the Ramification Problem (sometimes). *Artificial Intelligence* 116(1-2):87–121.

Pearl, J. 2009. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition edition.

Reiter, R. 1982. Circumscription Implies Predicate Completion (sometimes). In *AAAI*, 418–420.

Reiter, R. 2001. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. MIT Press.

Sandewall, E. 1994. *Features and Fluents. A Systematic Approach to the Representation of Knowledge about Dynamical Systems. Volume I.* Oxford University Press.

Strass, H., and Thielscher, M. 2010. A General First-Order Solution to the Ramification Problem. In Booth, R., and Gabaldon, A., eds., *Proceedings of the International Workshop on Nonmonotonic Reasoning (NMR): Reasoning About Actions Track*.

Thielscher, M. 1995. Computing Ramifications by Postprocessing. In *Proc. of IJCAI-95*, 1994–2000.

Thielscher, M. 2000. *Challenges for Action Theories*. Berlin, Heidelberg: Springer-Verlag.