Representing Synonymity in Causal Logic and in Logic Programming

Joohyung Lee[1], Yuliya Lierler[2], Vladimir Lifschitz[2], Fangkai Yang[2]
[1]School of Computing, Informatics and Decision Systems Engineering
Arizona State University
email:joolee@asu.edu
[2]Department of Computer Sciences
The University of Texas at Austin
email:{yuliya,vl,fkyang}@cs.utexas.edu

Abstract

We investigate the relationship between rules representing synonymity in nonmonotonic causal logic and in answer set programming. This question is of interest in connection with current work on modular languages for describing actions.

Introduction

This paper is about representing the important idea of synonymity in knowledge representation formalisms with nonmonotonic semantics. In classical logic, we can express that, under a certain condition, atom p is synonymous to atom q by a formula of the form

$$\mathit{Condition} \rightarrow (p \leftrightarrow q). \tag{1}$$

What are the counterparts of this formula in the language of logic programming under the answer set semantics (Gelfond and Lifschitz 1991) and in nonmonotonic causal logic (McCain and Turner 1997)?

This question is closely related to current research on modular action description languages. In STRIPS (Fikes and Nilsson 1971) and similar formalisms, actions are described in terms of their effects and preconditions. As observed in (Erdoğan and Lifschitz 2006), informal descriptions of actions that humans give are often strikingly different in their content. For instance, the dictionary defines pushing as *moving by steady pressure*. This phrase explains the meaning of the word *push* not by listing the effects of this action, but by referring to another action, *move*, that is supposed to be already familiar to the reader. It tells us that, under some conditions, *push* is synonymous to *move*. Relationships of this kind can be expressed in modular action description languages MAD (Lifschitz and Ren 2006) and $\mathcal{ALM}$ (Gelfond and Inclezan 2009). The semantics of MAD is defined in terms of a translation into nonmonotonic causal logic; the semantics of $\mathcal{ALM}$ is based on answer sets. Representing synonymity in these formalisms plays an important role in the definitions of MAD and $\mathcal{ALM}$.

A causal theory in the sense of (McCain and Turner 1997) is a set of rules $F \Leftarrow G$ ("there is a cause for F to hold if G holds"), where the head F and the body G are propositional formulas. Under the approach adopted in the semantics of MAD, the counterpart of (1) in this language is the rule

$$p \leftrightarrow q \Leftarrow \mathit{Condition}. \tag{2}$$

On the other hand, a (nondisjunctive) logic program in the sense of (Gelfond and Lifschitz 1991) is a set of rules

$$\mathit{Head} \leftarrow \mathit{Body},$$

where *Head* is a literal (an atom possibly preceded by the classical, or strong, negation symbol $\neg$), and *Body* is a list consisting of expressions of the forms

$$l, \quad \mathit{not}\ l,$$

where l is a literal. Under the $\mathcal{ALM}$ approach, the counterpart of (1) is the set of 4 rules:

$$\begin{array}{rl} p & \leftarrow q, \mathit{Condition}, \\ q & \leftarrow p, \mathit{Condition}, \\ \neg p & \leftarrow \neg q, \mathit{Condition}, \\ \neg q & \leftarrow \neg p, \mathit{Condition}. \end{array} \tag{3}$$

We are interested in the relationship between causal rule (2) and logic programming rules (3).

Any attempt to relate causal rules in the sense of (McCain and Turner 1997) to logic programming rules in the sense of (Gelfond and Lifschitz 1991) has to deal with two difficulties. One is syntactic: each of these languages uses some constructs that are not available in the other. In a causal rule, the head and the body can be arbitrary propositional formulas; on the other hand, that language does not distinguish between strong negation and negation as failure (and this distinction is a crucial element of the answer set semantics). In particular, restrictions on the syntactic form of *Condition* in (2) are not the same as in (3).

Second, there is a semantic difficulty: models of a causal theory and answer sets are objects of two different kinds. A model in the sense of causal logic is an interpretation, as this term is understood in classical propositional logic, that is, a truth assignment. An answer set, on the other hand, is a consistent set of literals.

The earliest result on the relationship between nonmonotonic causal logic and answer sets (McCain 1997, Proposition 6.7) handles these difficulties in the following way. It is limited to causal rules of the form

$$l_0 \Leftarrow l_1 \wedge \cdots \wedge l_n, \tag{4}$$

where each l_i is a literal. The logic programming counterpart of this rule, according to McCain, is

$$l_0 \leftarrow not\ \overline{l_1}, \ldots, not\ \overline{l_n}, \tag{5}$$

($\bar{l}$ stands for the literal complementary to l). Generally, (5) contains both strong negation and negation as failure. Furthermore, McCain's theorem asserts that the models of a causal theory are identical to the answer sets of the corresponding logic program that are *complete* (that is, contain one member of every complementary pair of literals), provided that we identify a complete set of literals with the corresponding truth assignment.

In this paper, McCain's theorem is extended to causal theories consisting of any number of rules of the form (4) and one "synonymity rule," which has an equivalence between two atoms (or, more generally, between two literals) in the head and a conjunction of literals in the body. In the corresponding logic program, the synonymity rule is represented by a group of rules similar to (3), with every conjunctive term l of its body rewritten as *not* $\bar{l}$, as in McCain's translation. We show that the models of the given causal theory are identical to the complete answer sets of the corresponding logic program. In this sense, expressing synonymity by causal rules of the form (2) has the same effect as expressing it by (3) in the language of answer set programming.

Clarifying the relationship between the MAD and $\mathcal{ALM}$ approaches to synonymity is one part of the motivation behind the work presented in this paper. The other part is related to implementing MAD. The existing implementation (Erdoğan 2008) translates MAD action descriptions into the input language of the Causal Calculator (CCALC)[1]. That system is a partial implementation of nonmonotonic causal logic, and it is limited to definite causal rules, that is, to rules $F \Leftarrow G$ such that F is a literal. Rules (4), for instance, are definite, but synonymity rules (2) are not. Erdoğan's translation eliminates synonymity rules in favor of definite rules, and it is only applicable when the synonymity rules are "unconditional"—have tautological bodies. We plan to design a more versatile implementation of MAD that will use an answer set solver for search (in the style of the COALA approach to action languages[2]), instead of CCALC. Research on representing synonymity rules, such as (2), by logic programs provides a theoretical foundation for this future work.

Our proof of the theorem on synonymity uses loop formulas, defined in (Lin and Zhao 2004) for logic programs and in (Lee 2004) for causal theories. The logic programs that we deal with contain both negation as failure and strong negation, and, as a preliminary step, we extend the Lin-Zhao theorem to logic programs with strong negation. (An alternative approach would be to eliminate strong negation in favor of additional atoms, but that would have made the proof more complicated.) Since our extension of the Lin-Zhao theorem can be of interest in its own right, we present it in a slightly more general form than strictly necessary for our present purposes: the formulation below covers disjunctive programs.[3]

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

[2]http://www.cs.uni-potsdam.de/~tgrote/al2asp/

Preliminaries

Logic Programs

We begin with a propositional signature—a finite set of *atoms*. A *literal* is an atom possibly preceded by the strong negation sign ¬. A *head expression* is an expression of the form

$$l_1; \ldots; l_n \tag{6}$$

($n \geq 0$), where each l_i is a literal. Sometimes we will identify a head expression (6) with the set $\{l_1, \ldots, l_n\}$. A *body expression* is an expression of the form

$$l_1, \ldots, l_m, not\ l_{m+1}, \ldots, not\ l_n \tag{7}$$

($n \geq m \geq 0$), where each l_i is a literal. A *rule* is an expression of the form $H \leftarrow B$, where H is a head expression and B is a body expression. A rule is called a *constraint* if its head is empty. A *logic program* is a finite set of rules.

Answer sets for programs of this type are defined in (Gelfond and Lifschitz 1991).

Causal Theories

A *causal theory* T is a finite set of *causal rules* of the form $F \Leftarrow G$, where F and G are propositional formulas (the *head* and the *body* of the rule). The semantics of causal theories defines when an interpretation I of the underlying signature (that is, a truth assignment) is a model of a causal theory T. The *reduct* of T with respect 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 unique interpretation satisfying the reduct of T with respect to I (McCain and Turner 1997).

The McCain Translation

If C is a conjunction of literals $l_1 \wedge \cdots \wedge l_n$ then the *McCain translation* of C, denoted by MC(C), is the body expression

$$not\ \overline{l_1}, \ldots, not\ \overline{l_n}.$$

Let T be a causal theory such that all its rules have the form (4). The *McCain translation* MC(T) of T is the logic program consisting of the rules $l \leftarrow$ MC(C) for

[3]The definitions of completion (Clark 1978) and loop formulas were extended to disjunctive programs (without strong negation) in (Lee and Lifschitz 2003).

all rules $l \Leftarrow C$ from T. In the statement of the theorem below, we identify an interpretation I of the underlying signature with the set of literals satisfied by I.

Theorem ((McCain 1997), Proposition 6.7). *An interpretation I is a model of T iff I is an answer set for $MC(T)$.*

Example. Consider the causal theory

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

It has one model:

$$I(p) = I(q) = \mathbf{t}. \tag{8}$$

The corresponding logic program

$$\begin{aligned} p &\leftarrow not\ \neg q, \\ q &\leftarrow not\ \neg q, \\ \neg q &\leftarrow not\ q \end{aligned} \tag{9}$$

has the answer sets $\{p, q\}$ and $\{\neg q\}$. The first of them is identical to I; the second is incomplete.

Theorem on Synonymity

A *synonymity rule* is a rule of the form

$$l_0 \leftrightarrow l_1 \Leftarrow Condition, \tag{10}$$

where l_0, l_1 are literals, and *Condition* is a conjunction of literals. We extend the McCain translation to synonymity rules as follows: if R is (10) then MC(R) is the logic program

$$\begin{aligned} l_1 &\leftarrow l_0, \mathrm{MC}(Condition), \\ l_0 &\leftarrow l_1, \mathrm{MC}(Condition), \\ \overline{l_1} &\leftarrow \overline{l_0}, \mathrm{MC}(Condition), \\ \overline{l_0} &\leftarrow \overline{l_1}, \mathrm{MC}(Condition). \end{aligned} \tag{11}$$

In the statement of the theorem below, T is a causal theory such that all its rules have the form (4), and R is a synonymity rule.

Theorem on Synonymity. *An interpretation I is a model of causal theory $T \cup \{R\}$ iff I is an answer set of logic program $MC(T) \cup MC(R)$.*

Example. Consider the causal theory

$$\begin{aligned} p &\Leftarrow q, \\ q &\Leftarrow q, \\ r &\Leftarrow \neg q, \\ r \leftrightarrow \neg p &\Leftarrow \top, \end{aligned} \tag{12}$$

where $\top$ is the empty conjunction. It has one model:

$$I(p) = I(q) = \mathbf{t},\ I(r) = \mathbf{f}.$$

The corresponding logic program is

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

The only answer set of this program is $\{p, q, \neg r\}$; it is identical to interpretation I.

We will now outline a proof of the theorem stated above. The first step is to extend the Lin-Zhao theorem on loop formulas to logic programs with strong negation.

Completion and Loop Formulas for Programs with Strong Negation

If B is a body expression (7) then by $pf(B)$ we denote the propositional formula

$$l_1 \wedge \ldots \wedge l_m \wedge \overline{l_{m+1}} \wedge \ldots \wedge \overline{l_n}.$$

The *literal completion*[4] of a logic program Π is the set of propositional formulas that includes

- for each literal l, the equivalence

$$l \leftrightarrow \bigvee_{\substack{H \leftarrow B\, \in\, \Pi \\ l\, \in\, H}} \left(pf(B) \wedge \bigwedge_{l' \in H \setminus \{l\}} \overline{l'} \right); \tag{13}$$

- the formula $\neg pf(B)$ for each constraint $\leftarrow B$ in Π.

If Π is nondisjunctive (at most one literal in the head of each rule) then (13) can be written as

$$l \leftrightarrow \bigvee_{\substack{H \leftarrow B\, \in\, \Pi \\ H\, =\, l}} pf(B).$$

Example. The literal completion of program (9) consists of the formulas

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

where $\bot$ is the empty disjunction. If we extend (9) by adding the rules

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

then the literal completion will become

$$\begin{aligned} p &\leftrightarrow q \vee (\neg q \wedge \neg r), \\ \neg p &\leftrightarrow \bot, \\ q &\leftrightarrow q, \\ \neg q &\leftrightarrow \neg q, \\ r &\leftrightarrow \neg q \wedge \neg p, \\ \neg r &\leftrightarrow \bot, \\ &\neg(p \wedge \neg r). \end{aligned}$$

If B is a body expression (7) then B^+ denotes the set $\{l_1, \ldots, l_m\}$. The *dependency graph* of a logic program Π is the directed graph that has arbitrary literals

[4]This term was introduced originally by McCain and Turner (1997) in the context of causal logic. Their definition is reviewed in the next section.

as its vertices and has an edge from each element of H to each element of B^+ for each rule $H \leftarrow B$ of Π. A nonempty set L of literals is called a *(nontrivial) loop* of Π if, for every pair l, l' of literals in L, there exists a path of non-zero length from l to l' in the dependency graph of Π such that all vertices in this path belong to L.[5] The *external support formula* for a loop L of Π, denoted by $ES_\Pi(L)$, is the disjunction of the formulas

$$pf(B) \wedge \bigwedge_{l \in H \setminus L} \bar{l}$$

for all rules $H \leftarrow B$ of Π such that

$$H \cap L \neq \emptyset \text{ and } B^+ \cap L = \emptyset. \tag{14}$$

Example. The only loop L of the program

$$\begin{array}{rl} \neg p & \leftarrow q, \\ q & \leftarrow \neg p, r, \\ q; r & \leftarrow \mathit{not}\ p, s \end{array} \tag{15}$$

is $\{\neg p, q\}$. The first of conditions (14) is satisfied for each of the rules (15), but the second is satisfied for the last rule only (the support for L provided by the other rules is "not external"). Consequently, the external support formula for this loop has only one disjunctive term $\neg p \wedge s \wedge \neg r$. If we replace *not* p with $\neg p$ in the body of the last rule of (15) then the external support formula for L will turn into the empty disjunction $\bot$. On the other hand, if we replace $\neg p$ with *not* p in the body of the second rule of (15) then $\{\neg p, q\}$ will not be a loop anymore.

In the definition of loop formulas below, $L^\vee$ stands for the disjunction of the elements of L. The *(disjunctive)*[6] *loop formula* for L, denoted by $LF(L, \Pi)$, is

$$L^\vee \rightarrow ES_\Pi(L).$$

Theorem on Loop Formulas. *A consistent complete set of literals is an answer set of a program* Π *iff it satisfies the literal completion of* Π *and the formulas* $LF(L, \Pi)$ *for all loops* L *of* Π.

The proof is based on the theorem from (Lee 2005, Section 2.2).

Review: Literal Completion of a Causal Theory

Our proof of the theorem on synonymity uses the concept of the literal completion of a definite causal theory, introduced in (McCain and Turner 1997). Recall that a causal theory is called *definite* if the head of each of its rules is a literal. The *literal completion* of a definite causal theory T is the set of formulas

$$l \leftrightarrow \bigvee_{\substack{H \Leftarrow B \,\in\, T \\ H = l}} B$$

for all literals l.

According to a theorem from (McCain and Turner 1997), the models of a definite causal theory can be equivalently described as the models of its completion.

Review: Loop Formulas of a Causal Theory

The proof of the theorem on synonymity uses also the concept of a loop formula for causal theories, introduced in (Lee 2004). Its definition, for the case when the head of every causal rule is a non-empty disjunction of distinct literals, is reproduced below. The *head dependency graph* of a causal theory T consisting of such rules is the directed graph that has arbitrary literals as its vertices and has an edge from l to $\overline{l'}$ for each pair of distinct literals l, l' from the head of the same rule of T. A nonempty set L of literals is called a *loop* of T if, for every pair l, l' of literals in L, there exists a path of non-zero length from l to l' in the head dependency graph of T such that all vertices in this path belong to L. Clearly, if L is a loop of T then the set $\overline{L}$ of literals complementary to the elements of L is a loop of T also.

The *(disjunctive) loop formula* of a loop L is

$$L^\vee \rightarrow \bigvee_{\substack{F \Leftarrow G \,\in\, T \\ F \cap L \neq \emptyset,\ F \cap \overline{L} = \emptyset}} \left(G \wedge \bigwedge_{l \in F \setminus L} \bar{l} \right). \tag{16}$$

According to Theorem 1 from (Lee 2004), an interpretation I is a model of T iff

- I is a model of the definite causal theory consisting of the rules
$$l \Leftarrow G \wedge \bigwedge_{l' \in F \setminus \{l\}} \overline{l'} \tag{17}$$
for all rules $F \Leftarrow G$ of T and all $l \in F$, and
- I satisfies the loop formulas of all loops of T.

Proof of the Theorem on Synonymity

In the proof, the characterization of the models of a causal theory quoted above is applied to the causal theory T_1 obtained from T by adding rule (10) rewritten as the pair of rules

$$\begin{array}{rl} \overline{l_0} \vee l_1 & \Leftarrow \mathit{Condition}, \\ l_0 \vee \overline{l_1} & \Leftarrow \mathit{Condition}. \end{array} \tag{18}$$

If l_0 and l_1 are equal to each other or complementary then the assertion of the theorem on synonymity easily follows from McCain's theorem. Otherwise the four literals in the heads of (18) are pairwise distinct. The

[5]The adjective "nontrivial" here reflects the fact that paths of length 0 are not allowed. Loops in a more general sense, without this limitation, are studied in (Lee 2005).

[6]The concept of a conjunctive loop formula, introduced in (Lee and Lifschitz 2003), can be extended to programs with strong negation in a similar way.

corresponding set of rules (17) consists of two parts: the rules of T and the rules

$$\begin{array}{ll} l_0 & \Leftarrow l_1 \wedge \mathit{Condition}, \\ l_1 & \Leftarrow l_0 \wedge \mathit{Condition}, \\ \overline{l_0} & \Leftarrow \overline{l_1} \wedge \mathit{Condition}, \\ \overline{l_1} & \Leftarrow \overline{l_0} \wedge \mathit{Condition}. \end{array} \tag{19}$$

We will show, first, that the literal completion of the causal theory consisting of the rules of T and rules (19) is identical to the literal completion of the logic program Π consisting of the rules of $\mathrm{MC}(T)$ and rules (11). Second, the set of loop formulas of all loops of causal theory T_1 is identical to the set of loop formulas of all nontrivial loops of Π. In view of the theorem on loop formulas stated above, the assertion of the theorem on synonymity follows from these two claims.

Take an arbitrary literal l, and assume first that l is different from l_0, l_1, $\overline{l_0}$, $\overline{l_1}$. The equivalence corresponding to l in the literal completion of Π is

$$l \leftrightarrow \bigvee_{C\,:\,l \Leftarrow C \in T} \mathit{pf}(\mathrm{MC}(C)).$$

Since

$$\mathit{pf}(\mathrm{MC}(C)) = C,$$

it can be written as

$$l \leftrightarrow \bigvee_{C\,:\,l \Leftarrow C \in T} C.$$

This is the equivalence corresponding to l in the literal completion of the theory consisting of the rules of T and rules (19).

If l is l_0 then the equivalence corresponding to l in the literal completion of Π is

$$l_0 \leftrightarrow \left(\bigvee_{C\,:\,l_0 \Leftarrow C \in T} \mathit{pf}(\mathrm{MC}(C)) \right) \vee \mathit{pf}(l_1 \wedge \mathrm{MC}(\mathit{Condition})),$$

which can be written as

$$l_0 \leftrightarrow \left(\bigvee_{C\,:\,l_0 \Leftarrow C \in T} C \right) \vee (l_1 \wedge \mathit{Condition}).$$

This is the equivalence corresponding to l_0 in the literal completion of the theory consisting of the rules of T and rules (19). When l is one of the literals l_1, $\overline{l_0}$, $\overline{l_1}$, the reasoning is similar.

To prove the second claim, note that the dependency graph of Π is identical to the head dependency graph of T_1: each of them has 4 edges, from l_0 to l_1 and back, and from $\overline{l_0}$ to $\overline{l_1}$ and back. Consequently, the nontrivial loops of Π are the same as the loops of T_1: $\{l_0, l_1\}$ and $\{\overline{l_0}, \overline{l_1}\}$. The loop formula of $\{l_0, l_1\}$ is

$$(l_0 \vee l_1) \rightarrow ES_\Pi(\{l_0, l_1\}).$$

Recall that Π consists of rules of two types: $l \leftarrow \mathrm{MC}(C)$ for all causal rules $l \Leftarrow C$ from T and rules (11). If $H \leftarrow B$ is $l \leftarrow \mathrm{MC}(C)$ then conditions (14) are satisfied whenever l is l_0 or l_1. If $H \leftarrow B$ is one of the first two rules (11) then the second of conditions (14) is violated. If $H \leftarrow B$ is one of the last two rules (11) then the first of conditions (14) is violated. Consequently, $ES_\Pi(\{l_0, l_1\})$ is the disjunction of the formulas $\mathit{pf}(\mathrm{MC}(C))$ for the bodies C of all rules of the forms $l_0 \Leftarrow C$ and $l_1 \Leftarrow C$ from T. Hence the loop formula of $\{l_0, l_1\}$ is

$$(l_0 \vee l_1) \rightarrow \bigvee_{\substack{l \Leftarrow C \in T \\ l \in \{l_0, l_1\}}} \mathit{pf}(\mathrm{MC}(C)),$$

or

$$(l_0 \vee l_1) \rightarrow \bigvee_{\substack{l \Leftarrow C \in T \\ l \in \{l_0, l_1\}}} C. \tag{20}$$

The loop formula (16), in the case when L is $\{l_0, l_1\}$ and T is T_1, can be simplified in a similar way. The conditions

$$F \cap \{l_0, l_1\} \neq \emptyset \text{ and } F \cap \{\overline{l_0}, \overline{l_1}\} = \emptyset$$

cannot be both satisfied if $F \Leftarrow G$ is one of the rules (18). On the other hand, if $F \Leftarrow G$ is $l \Leftarrow C$ then these conditions turn into $l \in \{l_0, l_1\}$, and $G \wedge \bigwedge_{l \in F \setminus L} \overline{l}$ turns into C. Consequently, (16) becomes (20). For the loop $\{\overline{l_0}, \overline{l_1}\}$ the reasoning is similar.

Related and Future Work

Ferraris (2007) showed how to translate an arbitrary causal theory into the language of nested logic programs in the sense of (Lifschitz *et al.* 1999), and his translation can be applied, in particular, to synonymity rules. For instance, the result of applying Ferraris's translation to

$$p \leftrightarrow q \Leftarrow r$$

is the pair of rules

$$\begin{array}{l} p; \neg q \leftarrow \mathit{not\ not}\ r, (\neg p; \mathit{not}\ \neg p), (q; \mathit{not}\ q), \\ q; \neg p \leftarrow \mathit{not\ not}\ r, (\neg q; \mathit{not}\ \neg q), (p; \mathit{not}\ p). \end{array}$$

It looks quite different from the translation proposed in this paper:

$$\begin{array}{rl} p & \leftarrow q, \mathit{not}\ \neg r, \\ q & \leftarrow p, \mathit{not}\ \neg r, \\ \neg p & \leftarrow \neg q, \mathit{not}\ \neg r, \\ \neg q & \leftarrow \neg p, \mathit{not}\ \neg r. \end{array}$$

However, the result of (Ferraris 2007) can be used as a basis for an alternative proof of our theorem on synonymity.[7]

In the future, we plan to extend the theorem on synonymity in several ways, with the eye on applying this line of research to re-implementing MAD, as discussed in Introduction. First, we would like to allow several synonymity rules, and not just one as in this paper.

[7]Paolo Ferraris, personal communication, July 18, 2009.

Furthermore, in connection with the semantics of variables in action descriptions proposed in (Lifschitz and Ren 2007) and (Ren 2009, Chapter 10), the theorem needs to be extended to first-order causal logic in the sense of (Lifschitz 1997) and to the version of the stable model semantics developed in (Ferraris *et al.* 2007) and (Ferraris *et al.* 2010).

We will also investigate the possibility of using the extension of the theory of loop formulas presented in this paper for improving the computational method of the answer set solver CMODELS when the goal is to generate complete answer sets.

Acknowledgements

We are grateful to Selim Erdoğan, Paolo Ferraris, and Michael Gelfond for comments on a draft of this note. This research was partially supported by the National Science Foundation under grants IIS-0712113 and IIS-0916116 and by the Office of the Director of National Intelligence (ODNI), Intelligence Advanced Research Projects Activity (IARPA), through US army. All statements of fact, opinion or conclusions contained herein are those of the authors and should not be construed as representing the official views or policies of IARPA, the ODNI or the U.S. Government.

References

Keith Clark. Negation as failure. In Herve Gallaire and Jack Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Press, New York, 1978.

Selim T. Erdoğan and Vladimir Lifschitz. Actions as special cases. In *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 377–387, 2006.

Selim T. Erdoğan. *A Library of General-Purpose Action Descriptions.*[8] PhD thesis, University of Texas at Austin, 2008.

François Fages. A fixpoint semantics for general logic programs compared with the well–supported and stable model semantics. *New Generation Computing*, 9:425–443, 1991.

Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. A new perspective on stable models. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 372–379, 2007.

Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. Stable models and circumscription.[9] *Artificial Intelligence*, 2010. To appear.

Paolo Ferraris. A logic program characterization of causal theories. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 366–371, 2007.

Richard Fikes and Nils Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4):189–208, 1971.

Michael Gelfond and Daniela Inclezan. Yet another modular action language. In *Proceedings of the Second International Workshop on Software Engineering for Answer Set Programming*[10], pages 64–78, 2009.

Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

Joohyung Lee and Vladimir Lifschitz. Loop formulas for disjunctive logic programs. In *Proceedings of International Conference on Logic Programming (ICLP)*, pages 451–465, 2003.

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

Joohyung Lee. A model-theoretic counterpart of loop formulas. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 503–508. Professional Book Center, 2005.

Vladimir Lifschitz and Wanwan Ren. A modular action description language. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 853–859, 2006.

Vladimir Lifschitz and Wanwan Ren. The semantics of variables in action descriptions. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, 2007.

Vladimir Lifschitz, Lappoon R. Tang, and Hudson Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25:369–389, 1999.

Vladimir Lifschitz. On the logic of causal explanation. *Artificial Intelligence*, 96:451–465, 1997.

Fangzhen Lin and Yuting Zhao. ASSAT: Computing answer sets of a logic program by SAT solvers. *Artificial Intelligence*, 157:115–137, 2004.

Norman McCain and Hudson Turner. Causal theories of action and change. In *Proceedings of National Conference on Artificial Intelligence (AAAI)*, pages 460–465, 1997.

Norman McCain. *Causality in Commonsense Reasoning about Actions.*[11] PhD thesis, University of Texas at Austin, 1997.

Wanwan Ren. *A Modular Language for Describing Actions.*[12] PhD thesis, University of Texas at Austin, 2009.

[8]http://www.cs.utexas.edu/users/tag/mad/erdogan-dissertation.pdf

[9]http://peace.eas.asu.edu/joolee/papers/smcirc.pdf

[10]http://www.sea09.cs.bath.ac.uk/downloads/sea09proceedings.pdf

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

[12]http://www.cs.utexas.edu/users/rww6/dissertation.pdf