

On Odd and Even Cycles in Normal Logic Programs*

Fangzhen Lin

Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
flin@cs.hst.hk

Xishun Zhao

Institute of Logic and Cognition
Sun Yat-Sen University
510275, Guangzhou, P. R. China
hsdp08@zsu.edu.cn

Abstract

An odd cycle of a logic program is a simple cycle that has an odd number of negative edges in the dependency graph of the program. Similarly, an even cycle is one that has an even number of negative edges. For a normal logic program that has no odd cycles, while it is known that such a program always has a stable model, and such a stable model can be computed in polynomial time, we show in this paper that checking whether an atom is in a stable model is NP-complete, and checking whether an atom is in all stable models is co-NP complete, both are the same as in the general case for normal logic programs. Furthermore, we show that if a normal logic program has exactly one odd cycle, then checking whether it has a stable model is NP-complete, again the same as in the general case. For normal logic programs with a fixed number of even cycles, we show that there is a polynomial time algorithm for computing all stable models. Furthermore, this polynomial time algorithm can be improved significantly if the number of odd cycles is also fixed.

Introduction

One of the significant events in non-monotonic reasoning in recent years has been the emergence of answer set programming (see, for example (Niemelä 1999)), a constraint programming paradigm based on Gelfond and Lifschitz's stable model and answer set semantics of logic programs (Gelfond&Lifschitz 1988; 1991).

In general, most reasoning problems in ASP are NP-hard. For instance, checking if there exists an answer set of a normal logic program is an NP-complete problem.

Recently, Gottlob *et al* (2002) considered the following decision and search problems in logic programs:

1. Consistency: Determining whether a program P has a stable model.

*We thank the reviewers for comments to this paper. Much of this work was done during the visit of the second author to HKUST. This work has been supported in part by HK RGC under CERG HKUST6205/02E, and by the NSFC under its major research program "Basic Theory and Core Techniques of Non-Canonical Knowledge".

Copyright © 2004, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

2. Brave Reasoning: Check whether an atom appears in a stable model of a program P .
3. Cautious Reasoning: Check whether an atom appears in all stable models of a program P .
4. SM Computation: Find an arbitrary stable model of a program P .
5. SM Enumeration: Compute all stable models of a program P .

They defined the feedback width of a logic program to be the size of a smallest set of atoms such that when these atoms are deleted from the program's dependency graph, the resulting *un-directed* version of the graph has no cycles with a negative edge, and showed that the above tasks can all be solved in $O(2^k n)$, where n is the size of the program and k the feedback width. So for the class of logic programs that have a bounded feedback width, all of the reasoning problems can be solved in linear time on the size of the given input logic program.

One problem with this result is that there are a lot more cycles in the un-directed version than in the directed version of the dependency graph of a logic program. So in this paper, we consider cycles in the directed version of the dependency graph of a logic program. We distinguish two types of cycles. An *even* (*odd*, respectively) cycle is a simple cycle with a non-zero even (*odd*, respectively) number of negative edges.

Programs that have no odd cycles are also called call-consistent (Kunen ; Sato 1987). An early result by Papadimitriou and Yannakakis (1992), done in the context of deductive databases more than 10 years ago, showed that if a normal logic program is call-consistent, then there is a polynomial algorithm that can return a stable model of the logic program. This would suggest that perhaps other reasoning tasks, such as checking if there is a stable model containing a given atom, would be tractable as well. In this paper we show that this is not the case. In particular, checking if there is a stable model containing a given atom is an NP-complete problem, just as hard as in the general case.

It has also been shown that if a program has no even cycle, then it has at most one stable model (You&Yuan 1994): if the well-founded model of the program is also its stable model, then this is the only one; otherwise, it has no stable model (Zhao 2002). In this paper we extend this result and

show that a logic program with at most k even cycles has at most 2^k stable models.

In summary, the main results of this paper are as follows:

1. For call-consistent programs we show that the brave reasoning problem is NP-complete, and the cautious reasoning problem is co-NP-complete,
2. If a program has exactly one odd cycle (it could have any number of even cycles), then the consistency problem is NP-complete.
3. If a program has k even cycles (it could have any number of odd cycles), then it has at most 2^k stable models. Furthermore, the set of all stable models can be computed in $2^{2k}O(n^{k+2})$ time, where n is the size of the given program.
4. If the number of the total odd and even cycles of a logic program is at most k , then we show that the set of all stable models of the program can be computed in $2^{2k}O(n^2)$ time, where n is the size of the program.

As we shall see, besides of theoretical interests, these results suggest that for computing answer sets of a logic program, a good heuristic is to branch on the variable that would break the most number of even cycles.

This paper is organized as follows. In section 2, we review some basic concepts of logic programming. In section 3, we prove our results for call-consistent programs. We discuss programs with exactly one odd cycle in section 4, programs with a fixed number of even cycles in section 5, and programs with a fixed number of odd and even cycles in Section 6. We conclude this paper in section 7.

Preliminaries

We consider only normal propositional logic programs, which are finite sets of rules of the following form

$$r : \quad a \leftarrow b_1, \dots, b_m, \text{not } c_1, \dots, \text{not } c_n,$$

where r is the name of the rule, $a, b_i, 1 \leq i \leq m$, and $c_j, 1 \leq j \leq n$ are atoms, and $m + n \geq 1$. In the following, we let $\text{Head}(r)=a$ (the head of r), $\text{Pos}(r)=\{b_1, \dots, b_m\}$ (the positive body of r), and $\text{Neg}(r)=\{c_1, \dots, c_n\}$ (the negative body of r). The body of r is $\text{Pos}(r) \cup \text{Neg}(r)$.

In the following, given a (normal) logic program P , we denote by $\text{At}(P)$ the set of atoms occurring in P . We shall also call atoms positive literals. A negative literal is an expression of the form $\text{not } a$, where a is an atom. A literal is either a positive or a negative literal.

Given a logic program P , $S \subseteq \text{At}(P)$ is said to be a stable model (Gelfond&Lifschitz 1988) of P if $S = \text{Cons}(P_S)$, where P_S is obtained from P and S by the following Gelfond-Lifschitz transformation:

- (1) for each atom p and any rule $r \in P$, if $p \in \text{Neg}(r)$ and $p \notin S$, then remove $\text{not } p$ from r , and
- (2) in the set of remaining rules, delete those rules that still contain a negative literal in their bodies,

and $\text{Cons}(P_S)$ is the set of atoms derivable from the definite logic program P_S .

A logic program may have zero, one, or more than one stable models. In comparison, the well-founded model semantics (Van Gelder *et al* 1991) produces a unique 3-valued model for a normal logic program. Formally, a 3-valued interpretation I is a pair of two disjoint sets of atoms, (I^+, I^-) . Atoms in I^+ are assigned *true* and those in I^- *false*. Atoms occurring in neither I^+ nor I^- are *undefined*.

Given a logic program P , the operation Γ_P defined by $\Gamma_P(S) := \text{Cons}(P_S)$, is anti-monotone, i.e., if $S_1 \subseteq S_2$ then $\Gamma_P(S_2) \subseteq \Gamma_P(S_1)$. Thus the operation Γ_P^2 , defined as $\Gamma_P^2(S) := \Gamma_P(\Gamma_P(S))$, is monotone. Therefore, Γ_P^2 has the least fixed-point, say S , and the greatest fixed-point, say T . The well-founded model of P is $(S, \text{At}(P) - T)$.

The well-founded model can be computed in quadratic time (Van Gelder *et al* 1991), and has been used to simplify a logic program in almost all of the current answer set programming systems.

In the following, we denote by $\text{WFM}(P) = (\text{WFM}(P)^+, \text{WFM}(P)^-)$ the well-founded model of P . We use $\text{WFM}(P)$ to simplify P as follows:

- (1) Delete all rules $r \in P$ such that either $\text{Head}(r) \in \text{WFM}^+(P)$, $\text{Pos}(r) \cap \text{WFM}^-(P) \neq \emptyset$, or $\text{Neg}(r) \cap \text{WFM}^+(P) \neq \emptyset$.
- (2) From the bodies of the remaining rules, remove all positive literals in $\text{WFM}^+(P)$ and all negative literals $\text{not } q$ with $q \in \text{WFM}^-(P)$.

The resulting program is written as $P \setminus \text{WFM}(P)$.

The *dependency graph* (Apt *et al* 1988) of a normal logic program P , denoted by G_P , is a directed graph with signed edges. The vertices of G_P are atoms in $\text{At}(P)$. There is a positive (negative, respectively) edge from a vertex p to a vertex q if there is a rule $r \in P$ such that $\text{Head}(r) = q$ and $p \in \text{Pos}(r)$ ($p \in \text{Neg}(r)$, respectively). Informally, a positive (negative, respectively) edge from p to q means that q depends positively (negatively, respectively) on p .

If a simple cycle of G_P has an odd (a non-zero even, respectively) number of negative edges then we call it an *odd (even, respectively) cycle*. In the following, we denote by $\text{NOC}(P)$ ($\text{NEC}(P)$, respectively) the number of odd (even, respectively) cycles in the dependency graph of P .

Call-Consistent Programs

As mentioned earlier, a call-consistent program is a normal logic program that has no odd cycles. It has been shown that a call-consistent program always has a stable model, and such a stable model can be computed in polynomial time. Despite this, checking if an atom is in a stable model of a call-consistent program is still NP-complete, and checking if an atom is in all the stable models of a call-consistent program is still co-NP complete.

Theorem 1

- (a) *Brave reasoning for call-consistent programs is NP-complete.*
- (b) *Cautious reasoning for call-consistent programs is co-NP-complete.*

Proof. (Sketched) Memberships in NP for (a), and co-NP for (b) is obvious. For hardness, consider a set φ of clauses with variables x_1, \dots, x_n . Let y_1, \dots, y_n, a and b be new variables. Let $P(\varphi)$ be the program consisting of rules in the following groups.

(I) For each x_i ($1 \leq i \leq m$), the rules:

$$y_i \leftarrow \text{not } x_i, \quad x_i \leftarrow \text{not } y_i.$$

(II) For each C in φ , the rule: $a \leftarrow \overline{C}$, where for a clause C , if $\neg x \in C$, then $x \in \overline{C}$, and if an atom $x \in C$, then $\text{not } x \in \overline{C}$.

(III) The single rule: $b \leftarrow \text{not } a$.

The only cycles of $P(\varphi)$ are generated by rules in group (I), and they are all even cycles. Thus the program is a call-consistent.

We show that φ is satisfiable iff b is in at least one stable model of $P(\varphi)$, and φ is unsatisfiable iff a is in every stable model of $P(\varphi)$. From these, the NP-hardness of the brave reasoning, and the co-NP-hardness of the cautious reasoning follow.

Suppose φ is satisfiable. We can then generate a truth assignment satisfying it using the rules in (I). Under this truth assignment, rules in group (II) are not applicable. Hence we can not use them to get a . In this case, we get b by using the rule in (III). Therefore, we get a stable model of $P(\varphi)$ containing b . Suppose φ is not satisfiable. Then any truth assignment makes some clause C in φ false, and hence the rule in (II) is applicable. Thus, a is in every stable model of $P(\varphi)$. ■

Notice that a well-known technique for finding a stable model containing an atom a is to add the constraint “ $\leftarrow \text{not } a$ ” to the original logic program. According to our theorem, and the fact that there is a polynomial time algorithm that can return a stable model of a call-consistent logic program, one cannot replace this constraint by a polynomial number of rules without introducing some odd cycles unless the polynomial hierarchy collapses.

It is worth mentioning here that in disjunctive logic programs, the so-called head-cycle free programs (Ben-Eliyahu-Zohary&Paloponi 1997; Ben-Eliyahu-Zohary *et al* 2000) appear to exhibit similar properties like call-consistent programs. In particular, finding one answer set is tractable for these disjunctive logic programs, but brave reasoning and cautious reasoning are hard.

Programs with Exactly One Odd Cycle

Programs with odd cycles may or may not have stable models. For instance, consider the following programs

$$\begin{aligned} P_0 &= \{b \leftarrow \text{not } a, \quad c \leftarrow \text{not } b, \quad a \leftarrow \text{not } c\}, \\ P_1 &= P_0 \cup \{a \leftarrow\}. \end{aligned}$$

Both have exactly one odd loop, but P_0 has no stable model while P_1 has one $\{a, c\}$.

In general, even cycles generate models while odd cycles eliminate models. For instance, the following set of rules

$$\begin{aligned} \{x_1 \leftarrow \text{not } y_1, y_1 \leftarrow \text{not } x_1, \dots, x_n \leftarrow \text{not } y_n, \\ y_n \leftarrow \text{not } x_n\} \end{aligned} \quad (1)$$

has n even cycles and 2^n stable models. Now if we add an odd cycle such as $x_1 \leftarrow \text{not } x_1$ to it, then in principle, we may have to go through all these 2^n models before we are sure that there is no stable model of the new program.

This is basically the reason why the consistency problem for programs with exactly one odd cycle is intractable.

Theorem 2 *The consistency problem for programs whose dependency graphs have exactly one odd cycle is NP-complete.*

Proof. Membership in NP is obvious. The proof of hardness is similar to that of Theorem 1. Consider again a set φ of clauses with variables x_1, \dots, x_n . Let y_1, \dots, y_n, a and b be new variables. Let $P(\varphi)$ be as in the proof of Theorem 1 but with the rule (III) replaced by the following rule:

(III') $b \leftarrow a, \text{not } b$.

Clearly, $P(\varphi)$ has exactly one odd cycle formed by rule (III'), and it has a stable model if and only if φ is satisfiable. ■

Programs with a Fixed Number of Even Cycles

From the result in the last section, we see that restricting the number of odd cycles to a fixed positive number cannot reduce the complexity. As we mentioned above, the reason is that even cycles can generate an exponential number of models. We now consider what happens if we restrict the number of even cycles.

As mentioned in Introduction, if a program has no even cycle, then it has at most one stable model (You&Yuan 1994). Furthermore, if the well-founded model of the program is also its stable model, then this is the only one; otherwise, it has no stable model (Zhao 2002). To study the general case, we first introduce the following definition, which was first introduced in (Cholewinski&Truszczyński 1999).

Definition 1 *Let P be a program, and a an atom in $At(P)$. Define P_a^+ to be the program obtained from P by*

1. deleting all rules whose bodies contain $\text{not } a$, and
2. removing a from the bodies of remaining rules.

Similarly define P_a^- to be the program obtained from P by

1. deleting all rules whose bodies contain a , and
2. removing $\text{not } a$ from the remaining rules.

Notice that $G_{P_a^+}, G_{P_a^-}$ are subgraphs of G_P , and that in $G_{P_a^+}, G_{P_a^-}$ there are no edges going out from a .

Lemma 1 (Cholewinski&Truszczyński 1999) *Let P be a program, $a \in At(P)$. Suppose S is a stable model of P . Then*

1. *If $a \in S$ then S is a stable model of P_a^+ .*
2. *If $a \notin S$ then S is a stable model of P_a^- .*

Cholewinski and Truszczyński (1999) also proved that a program with n rules has at most $O(3^{n/3})$ stable models. Our following result gives the upper bound of the number of stable models according to the number of even cycles.

Recall that for a program P , $NEC(P)$ denotes the number of even cycles in P , and $NOC(P)$ the number of odd cycles in P .

Lemma 2 Let P be a program such that $NEC(P) \leq k$. Then P has at most 2^k stable models.

Proof. We show the theorem by induction on k , here k is $NEC(P)$.

If $k = 0$ then the lemma follows from the aforementioned result by You and Yuan (1994). Suppose $NEC(P) \geq 1$. Then pick an atom $a \in At(P)$ such that a occurs in an even cycle. Please note that $G_{P_a^+}, G_{P_a^-}$ are subgraphs of G_P , and that in $G_{P_a^+}, G_{P_a^-}$ there are no edges going out from a . Thus $NEC(P_a^+) \leq k-1$ and $NEC(P_a^-) \leq k-1$. By the induction hypothesis, P_a^+ (P_a^- , respectively) has at most 2^{k-1} stable models. By the above lemma, we know that P has at most $2^{k-1} + 2^{k-1} = 2^k$ stable models. ■

From these two lemmas, and the fact that when $NEC(P)$ is zero, either P has no stable model or its well-founded model is its stable model, a naive way to compute the stable models of a program with $NEC(P) \leq k$ is to choose an atom a that lies in an even cycle, split the current program P into P_a^+ and P_a^- , and then inductively compute the stable models of P_a^+ and P_a^- , which have fewer numbers of even cycles than P . Unfortunately, finding an atom that lies in an even loop of a program is an intractable problem:

Lemma 3 Given a program P and an atom a . The problem of determining whether a lies in an even cycle of G_P is NP-complete.

Proof. (Sketched) The proof of this lemma is similar to the proof of Theorem 16 of (Gottlob *et al* 2002). The membership in NP is obvious. To prove the hardness we reduce the node-disjoint path problem for directed graphs, which is a NP-complete problem (Garey&Johnson 1979), to our problem.

Let $G = (N, E)$ be a directed graph, and x_1, y_1 and x_2, y_2 two pairs of nodes of G .

The problem is deciding whether there are two node-disjoint paths linking x_1 to x_2 and y_1 to y_2 . From G , we construct a logic program $P(G)$ containing a rule $t \leftarrow s$ for each edge $(s, t) \in E$, plus two additional rules $x_1 \leftarrow \text{not } y_2$, and $y_1 \leftarrow \text{not } x_2$.

We can show that x_1 lies in an even cycle in P_G if and only if there are two node-disjoint paths linking x_1 to x_2 and y_1 to y_2 in G . ■

So instead of looking for an atom that lies in an even cycle of a program, and split the program on the atom, we simply split the program on every atom of the program. This is the basic idea behind our algorithm for computing $Ans(P, k)$ in Figure 1.

Lemma 4 Let k be a fixed natural number, P a program with $NEC(P) \leq k$. Then every stable model of P appears in $Ans(P, k)$.

Proof. We prove the assertion by induction on k . When $k = 0$, the assertion follows from the known result. Suppose $k > 0$. If the dependency graph G_P has an even cycle, then the cycle will be broken in $G_{P_a^+}$ and $G_{P_a^-}$ for some a . That is, $NEC(P_a^+) \leq k-1$ and $NEC(P_a^-) \leq k-1$. By the induction hypothesis, all stable models of P_a^+ (P_a^-), respectively) appear in $Ans(P_a^+, k-1)$ ($Ans(P_a^-, k-1)$), respectively). Now the lemma follows from Lemma 2. ■

Function $Ans(P, k)$

Input A normal program P , and a natural number k .

Output A set of stable models of P

begin

if $k = 0$ **then**

if $WFM^+(P)$ is a stable model of P **then return** $\{WFM^+(P)\}$ **else return** \emptyset .

$\Pi := \emptyset$.

for each atom $a \in At(P)$

for each $S \in Ans(P_a^+, k-1) \cup Ans(P_a^-, k-1)$

if S is a stable model of P **then** $\Pi := \Pi \cup \{S\}$.

return Π

end

Figure 1: An algorithm for computing the stable models of a logic program with at most k even cycles

Lemma 5 Let P be a program, and k a fixed natural number. Then

1. $Ans(P, k)$ contains at most $2^k n^k$ sets, where n is the size of P .
2. $Ans(P, k)$ can be computed in $2^{2k} O(n^{k+2})$ time.

Proof. (1) We shall proceed by induction on k . The assertion clearly holds when $k = 0$. Suppose $k > 0$. By the induction hypothesis, $Ans(P_a^+, k-1)$ ($Ans(P_a^-, k-1)$, respectively) contains at most $2^{k-1} n^{k-1}$ sets for each atom a . Thus, $Ans(P, k)$ has at most $2n(2^{k-1} n^{k-1}) = 2^k n^k$ sets.

(2) We shall prove the claim by induction on k . When $k = 0$ the claim is obviously valid because the well-founded model can be computed in quadratic time and to check whether it is a stable model costs not more than quadratic time. Suppose $k > 0$. By the induction hypothesis, to compute $Ans(P_a^+, k-1)$ and $Ans(P_a^-, k-1)$ for all a will cost time not more than

$$2n(2^{2(k-1)} n^{k+1}) = 2^{2k-1} n^{k+2}.$$

From (1), to check whether the sets in all $Ans(P_a^+, k-1) \cup Ans(P_a^-, k-1)$ are stable models needs time not more than

$$2n(2^{k-1} n^{k-1}) O(n^2) = 2^k O(n^{k+2}).$$

Altogether, the total running time is

$$2^{2k-1} n^{k+2} + 2^k n^{k+2} \leq 2^{2k} n^{k+2}. \quad \blacksquare$$

From these two lemmas we obtain the following theorem.

Theorem 3 For an arbitrary fixed natural number k , the set of stable models of a program with at most k even cycles can be computed in $2^{2k} O(n^{k+2})$ time, where n is the size of P .

Notice that in the theorem, the time complexity is bounded by n^{k+2} which is a polynomial whose degree depends on the parameter k . It remains an open question whether there is an algorithm computing all stable models in time bounded by a polynomial whose degree does not depend on k . In the next section, we show that this is true for logic programs with at most k total number of odd and even cycles.

Function All-Ans(P)**Input:** A normal program P **Output:** The set of stable models of P **begin****if** $\text{WFM}^+(P)$ is a stable model of P **then return** $\{\text{WFM}^+(P)\}$. $Q := P \setminus \text{WFM}(P)$ (now Q is non-empty).compute a bottom H of Q .pick an atom $a \in H$ such that there is another atom b in H ,
and a negative edge a to b in G_Q . $\Pi := \emptyset$.**for each** $S \in \text{All-Ans}(P_a^+) \cup \text{All-Ans}(P_a^-)$ **if** S is a stable model of P **then** $\Pi := \Pi \cup \{S\}$ **return** Π **end**

Figure 2: An algorithm for computing the stable models of a logic program

Programs with a Fixed Number of Cycles

In this section we show that for any given k , there is an algorithm that computes in quadratic time all stable models of any given program P with $\text{NEC}(P) + \text{NOC}(P) \leq k$.

Recall that a strongly connected component of a graph is a set of nodes such that for any two nodes in the set, there is a path from one node to the other, and it is not a proper subset of any such sets. There is a linear time algorithm that can compute all strongly connected components of a graph (Tarjan 1972).

Given a normal program P , we call a set B of atoms a *bottom* if it is a strongly connected component of G_P , and there is no other strongly connected component B' such that there is a path from an atom in B' to another atom in B .

Proposition 1 *Let P be a normal logic program. If $Q = P \setminus \text{WFM}(P)$ is not empty, then every bottom of Q must have a pair of nodes a and b such that there is a negative edge from a to b .*

Proof. We prove by contradiction. Suppose Q is not empty, H is a bottom of Q , and all paths in H are positive.

Let $R(H) = \{r \in Q \mid \text{Head}(r) \in H\}$. Since H is a bottom of Q , and Q is the result of simplifying P by its well-founded model, there cannot be any atom p in Q such that $p \notin H$, and there is a path from p to an atom in H . For otherwise, since there is no more cycles below p , its truth value would have been determined in the well-founded model of P , thus eliminated when P is simplified using its well-founded model. Thus the bodies of all rules in $R(H)$ must all be in H . As a consequence, no rule in $R(H)$ can have a negative literal in its body. Moreover, since $Q = P \setminus \text{WFM}(P)$, every rule in Q has to have a non-empty body. Then according to the procedure for calculating well-founded models, all atoms in H are false in the well-founded model of P , and hence they cannot occur in Q . This contradicts the assumption that H is a strongly connected component of Q . ■

Lemma 6 *Let P be a normal logic program. Then All-Ans(P) defined in Figure 2 contains all stable models of P .*

Proof. We show the theorem by induction on k , the number of cycles of G_P .

When $k = 0$, i.e., $\text{NEC}(P) = \text{NOC}(P) = 0$, the theorem follows from the fact that P has exact one stable model which is the well-founded model of P .

Suppose $k > 0$, and $\text{NEC}(P) + \text{NOC}(P) = k$. If $\text{WFM}(P)$ is a stable model of P , then it is the unique stable model of P , and the theorem follows. So, we assume that $\text{WFM}(P)$ is not a stable model of P . Thus $Q := P \setminus \text{WFM}(P)$ is not empty. Suppose H is a bottom of Q , let a and b be two atoms in H such that there is a negative edge from a to b . Since H is a connected component, there must a path from b to a . Thus, there is at least one cycle in H that goes through the negative edge from a to b . Then $\text{NEC}(P_a^+) + \text{NOC}(P_a^+) \leq k - 1$, and $\text{NEC}(P_a^-) + \text{NOC}(P_a^-) \leq k - 1$. We know that every stable model of P is a stable model of either P_a^+ or P_a^- . By the inductive assumption, every stable model of P_a^+ (P_a^- , respectively) appears in All-Ans(P_a^+) (All-Ans(P_a^-), respectively). Therefore, every stable model of P appears in All-Ans(P). ■

Lemma 7 *For each fixed natural number k , the function in Figure 2 runs in time $2^{2k}O(n^2)$ for any given program P such that $\text{NEC}(P) + \text{NOC}(P) \leq k$, where n is the size of P .*

Proof. We proceed by induction on k . When $k = 0$, the lemma follows from the fact that P has exact one stable model, i.e., the well-founded model, which can be computed in quadratic time and that to check whether the well-founded model is a stable model costs no more than quadratic time.

Suppose $k > 0$. To compute the well-founded model of a program costs quadratic time, to compute a bottom takes linear time, and the atom a and P_a^+, P_a^- can be computed in linear time. By the inductive assumption, to compute All-Ans(P_a^+) and All-Ans(P_a^-) costs not more than

$$2^{2(k-1)}O((n-1)^2) + 2^{2(k-1)}O((n-1)^2) \leq 2^{2k-1}O(n^2)$$

time. From Lemma 2, we know that All-Ans(P_a^+) \cup All-Ans(P_a^-) has at most 2^k elements. Thus to check whether they are stable models of P needs no more than time $2^kO(n^2)$.

Altogether the algorithm costs no more than

$$O(n^2) + O(n) + 2^{2k-1}O(n^2) + 2^kO(n^2) \leq 2^{2k}O(n^2)$$

By Lemmas 6 and 7 we obtain the following theorem. ■

Theorem 4 *The set of all stable models of a program with a fixed number of even and odd cycles can be computed in quadratic time $2^{2k}O(n^2)$, where k is the number of even and odd loops, and n the size of the given program.*

We want to emphasize that this result assumes a fixed k , and the co-efficient 2^{2k} means that this number should not be too big for the algorithm in Figure 2 to be practical. Of course, this is the worst case complexity. Problems normally encountered in practice could be much easier even for large k .

Conclusions

We have performed a detailed complexity analysis of normal logic programs with a limited number of odd and/or even cycles. To recast, for a normal logic program without any odd cycles, others have shown that it always has stable models (Dung 1992; You&Yuan 1994), and one of them can be computed in polynomial time (Papadimitriou&Yannakakis 1992). What we have shown here is that despite these positive results, given an atom, checking whether it is in one of its stable models is an NP-complete problem, and checking whether it is in all of its stable models is a coNP-complete problem. We have also shown that if we relax the condition a little bit, allowing just one odd cycle, then whether there is a stable model becomes an NP-complete program, the same as in the general case.

For a normal logic program without even cycles, others have shown that it has at most one stable model (You&Yuan 1994), and if it has a stable model, then the stable model must also be its well-founded model (Zhao 2002). We have extended these results, and shown that if a normal logic program has at most k even cycles, then it has at most 2^k stable models, and all of them can be computed in polynomial time for the fixed parameter k .

It is clear that there is a disparity between the results for programs with a limited number of odd cycles and those for programs with a limited even cycles. The reason is that these two types of cycles play very different roles. The even cycles are stable model generators, and the odd cycles are stable model eliminators.

We believe that our results are not just of theoretical interest. The algorithms in Figures 1 and 2, suggest that for computing answer sets of general logic programs, a good heuristic is to choose the next variable that would break the most number of even cycles. The key of course is how to efficiently compute the variables that would break a large number of even cycles. One future work for us is to address this problem and to experiment with this heuristic.

References

Apt, K.R., Blair, H.A., and Walker, A. 1988. Towards a Theory of Declarative Knowledge. In *Foundation of Deductive Database and Logic Programming* (ed. J. Minker), 293–322, Morgan Kaufmann, Los Altos, 1988.

Apt, K.R., and Bol, R.N., 1994. Logic Programming and Negation: A Survey. *Journal of Logic Programming* 19/20:9–71.

Ben-Eliyahu-Zohary, R. and Paloponi, L. 1997. Reasoning with minimal models: efficient algorithms and applications. *Artificial Intelligence* 96(2):421–449.

Ben-Eliyahu-Zohary, R., Paloponi, L., and Zemlyanker, V. 2000. More on tractable disjunctive datalog. *Journal of Logic Programming* 46(1-2):61–101.

Bidoit, N., and Froidevaux, C. 1991. Negation by Default and Unstratifiable Logic Programs. *Theoretical Computer Science* 78:85–112.

Cholewinski, P., and Truszczyński, M. 1999. Extremal Problems in Logic Programming and Stable Model Computation. *Journal of Logic Programming* 38:219–242.

Dung, P.M. 1992. On the Relation between Stable and Well-Founded Semantics of Logic Programs. *Theoretical Computer*

Science: Logic, Semantics and Theory of Programming 105:7–26.

Eiter, T., Leone, N., Mateis, C., Pfeifer, G., and Scarnello, F. 1998. The KR-System DLV: Progress Report, Comparison and Benchmarks. In *Proceedings of the 6th International Conference on Principles of Knowledge Representation and Reasoning*, 406–417. Morgan Kaufmann Publishers.

Garey, M.R., and Johnson, D.S. 1979. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, New York.

Gelfond, M., and Lifschitz, V. 1988. The Stable Model Semantics for Logic Programming. In *Proceedings of the 5th International Conference on Logic Programming*, 1070–1080. The MIT Press.

Gelfond, M., Lifschitz, V., 1991. Classical Negation in Logic Programs and Disjunctive Database. *New Generation Computing* 9: 365–385.

Gottlob, G., and Scarnello, F., and Sideri, M. 2002. Fixed-Parameter Complexity in AI and Nonmonotonic Reasoning. *Artificial Intelligence* 138:55–86.

Kleine Büning, H., and Lettmann, T. 1999. *Propositional Logic: Deduction and Algorithms*. Cambridge University Press.

Kunen, K. 1989. Signed Dada Dependencies in Logic Programs. *Journal of Logic Programming*, 7:231–245.

Lifschitz, V., and Turner, H. 1994. Splitting a Logic Program. In *Proceedings of International Conference on Logic Programming*, 23–37.

Lin, F., and You, J.H. 2002. Abduction in Logic Programming: A New Definition and an Abductive Procedure Based on Rewriting. *Artificial Intelligence*, 140(1/2):175–205.

Lin, F., and Zhao, Y. 2002. ASSAT: Computing Answer Sets of a Logic Program by SAT Solvers. In *Proceedings of AAAI-2002*, 112–117.

Niemelä, I. 1999. Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. *Annals of mathematics and Artificial Intelligence* 25:241–273.

Papadimitriou, C.H., and Yannakakis, M. 1992. Tie-Breaking Semantics and Structural Totality. In *Proceedings of 11th ACM conference on Principles of Database Systems (PODS 1992)*, 16–22.

Sato, T. 1987. On the Consistency of First-Order Logic Programs, ETL Technique Report, TR-87-12.

Tarjan, R.E. 1972. Depth-First Search and Linear Graph Algorithms, *SIAM Journal of on Computing* 1:146–160.

Van Gelder, A., Ross, K.R., and Schlipf, J.S. 1991. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM* 38:620–650.

You, J.H., and Yuan, L. 1994. A Three-Valued Semantics for Deductive Database and Logic Programs. *Journal of Computer and System Science* 49:334–361.

Zhao, J. 2002. A Study of Answer set Programming, MPhil Thesis, Dept. of Computer Science, The Hong Kong University of Science and Technology.