Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

On Elementary Loops and Proper Loops for Disjunctive Logic Programs

Jianmin Ji
School of Computer Science and Technology
University of Science and Technology of China
Hefei 230027, China
jianmin@ustc.edu.cn

Hai Wan* and **Peng Xiao**
School of Software
Sun Yat-sen University
Guangzhou 510006, China
wanhai@mail.sysu.edu.cn

Abstract

This paper proposes an alternative definition of *elementary loops* and extends the notion of *proper loops* for disjunctive logic programs. Different from normal logic programs, the computational complexities of recognizing elementary loops and proper loops for disjunctive programs are coNP-complete. To address this problem, we introduce weaker versions of both elementary loops and proper loops and provide polynomial time algorithms for identifying them respectively. On the other hand, based on the notion of elementary loops, the class of Head-Elementary-loop-Free (HEF) programs was presented, which can be turned into equivalent normal logic programs by shifting head atoms into bodies. However, the problem of recognizing an HEF program is coNP-complete. Then we present a subclass of HEF programs which generalizes the class of Head-Cycle-Free programs and provide a polynomial time algorithm to identify them. At last, some experiments show that both elementary loops and proper loops could be replaced by their weak versions in practice.

Introduction

The notions of loops and loop formulas were first proposed by Lin and Zhao (2004) for Normal Logic Programs (NLPs). They showed that a set of atoms is an answer set of a program iff it satisfies both the loop formulas and the program. Later, the notions and the result were extended to Disjunctive Logic Programs (DLPs) (Lee and Lifschitz 2003). These results guarantee the correctness and completeness of SAT approach based answer set solvers for both NLPs and DLPs, like ASSAT (Lin and Zhao 2004), cmodels (Giunchiglia, Lierler, and Maratea 2006), clasp (Gebser et al. 2007), and claspD (Drescher et al. 2008).

In general there may be an exponential number of loops (Lifschitz and Razborov 2006). Gebser and Schaub (2005) showed that not all loops are necessary for selecting the answer sets among the models of a NLP. They introduced the subclass *elementary loops* and refined the Lin-Zhao theory by considering elementary loops only. Ji et al. (2014) further showed that some elementary loops could also be disregarded in the answer set computation. They introduced a subclass *proper loops* of elementary loops and refined the Lin-Zhao theory by considering a special form of loop formulas for proper loops only.

Gebser, Lee, and Lierler (2011) extended the notion of elementary loops to DLPs and showed that these elementary loops are sufficient for selecting the answer sets among the models of a DLP. Based on elementary loops, they presented the class of Head-Elementary-loop-Free (HEF) programs, which strictly generalizes the class of Head-Cycle-Free (HCF) programs in (Ben-Eliyahu and Dechter 1994). Like an HCF program, an HEF program can be turned into an equivalent NLP in polynomial time by "shifting" head atoms into the body.

In this paper, we extend the notion of proper loops to DLPs, show that they are a subclass of elementary loops, and refine the Lin-Zhao theory to DLPs by considering a special form of loop formulas for proper loops only. Based on proper loops, we introduce the class of Head-Proper-loop-Free (HPF) programs, which strictly generalizes the class of HEF programs, and show that an HPF program can be turned into an equivalent NLP by "shifting". Although properties of elementary loops and proper loops could benefit the answer set computation, the computational complexities of recognizing them are coNP-complete. To address this problem, we introduce weaker versions of elementary loops and proper loops and provide polynomial time algorithms for identifying these classes of loops respectively. Moreover, we present a subclass of HEF programs which still generalizes HCF programs and provide a polynomial time algorithm to identify them. At last, we provide experiments to illustrate how these notions of loops work out in the benchmark suite. The result implies that both elementary loops and proper loops could be replaced by their weak versions in practice.

Preliminaries

Disjunctive Logic Programs

In this paper, we consider only fully grounded finite logic programs. A *(disjunctive) logic program* (DLP) is a finite set of (disjunctive) rules of the form

$$a_1 \vee \cdots \vee a_k \leftarrow a_{k+1}, \ldots, a_m, not\, a_{m+1}, \ldots, not\, a_n, \quad (1)$$

where $n \geq m \geq k \geq 1$ and $a_1, \ldots, a_n$ are atoms. If $k = 1$, it is a *normal rule*. In particular, a *normal logic program*

*Corresponding author

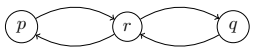

Figure 1: The positive dependency graph of program P_1

(NLP) is a finite set of normal rules.

We will also write rule r of form (1) as

$$head(r) \leftarrow body(r),$$

where $head(r)$ is $a_1 \vee \cdots \vee a_k$, $body(r) = body^+(r) \wedge body^-(r)$, $body^+(r)$ is $a_{k+1} \wedge \cdots \wedge a_m$, and $body^-(r)$ is $\neg a_{m+1} \wedge \cdots \wedge \neg a_n$, and we identify $head(r)$, $body^+(r)$, $body^-(r)$ with their corresponding sets of atoms. Let R be a set of rules, we denote $head(R) = \bigcup_{r \in R} head(r)$.

The *answer sets* of a DLP are defined in (Gelfond and Lifschitz 1991). Given a DLP P and a set S of atoms, the GL transformation of P on S, written P^S, is obtained from P by deleting:

1. each rule that has $not\, p$ in its body with $p \in S$, and
2. all $not\, p$ in the bodies of the remaining rules.

For any S, P^S has a set of minimal models, denoted $\Gamma(P^S)$. Now a set S of atoms is an *answer set* of P iff $S \in \Gamma(P^S)$.

Gelfond et al. (1991) provided a mapping from a DLP to a NLP by "shifting" head atoms into the bodies. We denote $sh(P)$ be the NLP obtained from a DLP P by substituting every rule of form (1) with the k rules

$$a_i \leftarrow not\, a_1, \ldots, not\, a_{i-1}, not\, a_{i+1}, \ldots, not\, a_k, \\ a_{k+1}, \ldots, a_m, not\, a_{m+1}, \ldots, not\, a_n. \ (1 \leq i \leq k)$$

It is known that every answer set of $sh(P)$ is also an answer set of P, but the converse is not true in general. Ben-Eliyahu and Dechter (1994) identified a class of DLP, called "Head-Cycle-Free" (HCF), and showed that the converse is true if P is an HCF program. In particular, a DLP P is called *HCF* if $|head(r) \cap L| \leq 1$ for every rule r of P and every loop L of P, and the answer sets of an HCF program P coincide with the answer sets of the NLP $sh(P)$.

Loops and Loop Formulas

Lee and Lifschitz (2003) extend the notion of loops and loop formulas to DLPs. Given a DLP P, the *positive dependency graph* of P, written G_P, is the directed graph whose vertices are atoms in P, and there is an arc from p to q if there is a rule $r \in P$ *s.t.* $p \in head(r)$ and $q \in body^+(r)$. A set L of atoms is said to be a *loop* of P if the L-induced subgraph of G_P is strongly connected. Note that, every singleton whose atom occurs in P is also a loop of P.

Example 1 *Consider the logic program P_1:*

$$p \vee q \leftarrow r. \qquad r \leftarrow p. \qquad r \leftarrow q. \qquad p \leftarrow .$$

Figure 1 shows the positive dependency graph of P_1. P_1 has six loops: $\{p\}$, $\{r\}$, $\{q\}$, $\{p, r\}$, $\{r, q\}$, and $\{p, r, q\}$.

Let L be a loop of a program, a rule r is an *external support* of L if $head(r) \cap L \neq \emptyset$ and $L \cap body^+(r) = \emptyset$. Let $R^-(L)$ be the set of external support rules of L. Note that, the function R^- can be defined for any set C of atoms, *i.e.*,

$$R^-(C) = \left\{ r \in P \mid head(r) \cap C \neq \emptyset, C \cap body^+(r) = \emptyset \right\}.$$

The *disjunctive loop formula* of L under P, written $DLF(L, P)$, is the following implication

$$\bigvee_{p \in L} p \supset \bigvee_{r \in R^-(L)} \left(body(r) \wedge \bigwedge_{q \in head(r) \setminus L} \neg q \right). \qquad (2)$$

An alternative definition of a loop formula (Lee and Lifschitz 2003) replaces $\bigvee$ in the antecedent of (2) with $\bigwedge$, called the *conjunctive loop formula* and written $CLF(L, P)$:

$$\bigwedge_{p \in L} p \supset \bigvee_{r \in R^-(L)} \left(body(r) \wedge \bigwedge_{q \in head(r) \setminus L} \neg q \right). \qquad (3)$$

Moveover, we can replace the antecedent of (2) or (3) with a propositional formula that entails $\bigvee_{p \in L} p$ and is entailed by $\bigwedge_{p \in L} p$. For instances, for any loop L, let F_L be a formula formed from atoms in L using conjunctions and disjunctions. Thus let $LF(L, P)$ denote a formula of the form:

$$F_L \supset \bigvee_{r \in R^-(L)} \left(body(r) \wedge \bigwedge_{q \in head(r) \setminus L} \neg q \right).$$

Theorem 1 (Theorem 1 in (Lee and Lifschitz 2003)) *Let P be a DLP and S a set of atoms. If S satisfies P, then following conditions are equivalent:*

1. *S is an answer set of P;*
2. *S satisfies $DLF(L, P)$ for all loops L of P;*
3. *S satisfies $CLF(L, P)$ for all loops L of P;*
4. *S satisfies $LF(L, P)$ for all loops L of P.*

Elementary Loops and HEF Programs

Gebser and Schaub (2005) introduced a subclass *elementary loops* of loops for NLPs. Later, Gebser, Lee, and Lierler (2011) extend the notion to DLPs.

Let X be a set of atoms and Y a subset of X, we say that Y is *outbound* in X for a DLP P if there is a rule r in P *s.t.*

- $head(r) \cap Y \neq \emptyset$,
- $body^+(r) \cap (X \setminus Y) \neq \emptyset$,
- $head(r) \cap (X \setminus Y) = \emptyset$, and
- $body^+(r) \cap Y = \emptyset$.

A loop L is an *elementary loop* of P if all nonempty proper subsets of L are outbound in L for P.

Example 1 (Continued) *Program P_1 has five elementary loops: $\{p\}$, $\{r\}$, $\{q\}$, $\{p, r\}$, $\{r, q\}$. $\{p, r, q\}$ is not an elementary loop as $\{p\}$ is not outbound in $\{p, r, q\}$.*

Theorem 2 (Theorem$^{\mathbf{d}}$ 1(d) in (Gebser et al. 2011)) *Each of following conditions is equivalent to each of conditions in Theorem 1:*

5. *S satisfies $CLF(L, P)$ for all elementary loops L of P;*
6. *S satisfies $DLF(L, P)$ for all elementary loops L of P;*
7. *S satisfies $LF(L, P)$ for all elementary loops L of P.*

Gebser and Schaub (2005) proved that the problem of recognizing an elementary loop for a NLP is tractable. However, the problem for a DLP is coNP-complete (Gebser, Lee, and Lierler 2011).

Based on the notion of elementary loops, Gebser, Lee, and Lierler (2011) presented the class of Head-Elementary-loop-Free (HEF) programs, which generalizes the class of HCF programs. A DLP P is called *HEF* if $|head(r) \cap L| \leq 1$ for every rule r of P and every elementary loop L of P. They proved that the answer sets of an HEF program P coincide with the answer sets of $sh(P)$. Moveover, the problem of recognizing an elementary loop for an HEF program is tractable. However, the problem of identifying HEF programs is coNP-complete (Fassetti and Palopoli 2010).

An Alternative Definition of Elementary Loops

Ji et al. (2014) provided an alternative definition of elementary loops for NLPs. Here we extend the idea to DLPs. The problem of identifying an elementary loop for DLPs is coNP-complete, then based on the new definition, we provide a tractable approximate algorithm for the problem.

Let X, Y be sets of atoms in a DLP P, we denote

$$R^-_X(Y) = \{r \mid r \in R^-(Y) \text{ and } head(r) \cap (X \setminus Y) = \emptyset\}.$$

Note that, if P is a NLP, then $R^-_X(Y) = R^-(Y)$.

Proposition 1 *Let P be a DLP and L_1, L_2 be loops of P. If $L_1 \subseteq L_2$ and $R^-_{L_2}(L_1) \subseteq R^-(L_2)$, then $CLF(L_1, P) \supset CLF(L_2, P)$.*

From Proposition 1, the conjunctive loop formula of such a loop L_2 could be ignored for the answer set computation.

A loop L is an elementary loop if all nonempty proper subsets of L are outbound in L. A set C is outbound in L iff there exists a rule r *s.t.* $r \in R^-_L(C)$ and $r \notin R^-(L)$ iff $R^-_L(C) \not\subseteq R^-(L)$. Then we have the following proposition.

Proposition 2 *Let P be a DLP and L a loop of P. L is an elementary loop of P iff there does not exist a nonempty proper subset C of L s.t. $R^-_L(C) \subseteq R^-(L)$.*

Gebser, Lee, and Lierler (2011) proved that the problem of recognizing an elementary loop for a DLP is coNP-complete. From the new definition, we provide Algorithm 1 running in polynomial time to decide whether a loop L is in $\text{EL}^*(P)$, a superset of all elementary loops of a DLP P.

Algorithm 1: $\text{EL}^*(L, P)$

```
for each atom a ∈ L do
    G* := the L \ {a} induced subgraph of G_P;
    SCC* := the set of SCCs of G*;
    for each C ∈ SCC* do
        if R^-_L(C) ⊆ R^-(L) then
            return C
        else
            G_C := the C \ head(R^-_L(C) \ R^-(L))
                   induced subgraph of G*;
            SCC_C := the set of SCCs of G_C;
            append new elements from SCC_C to SCC*;
return L
```

Intuitively, $\text{EL}^*(L, P)$ considers sub-loops of L one by one in a top-down process. L' is a sub-loop of L iff L' is a *Strongly Connected Component (SCC)* of a subgraph of the L induced subgraph of G_P. For each C of such SCCs, there are two cases: either $R^-_L(C) \subseteq R^-(L)$ or not. If $R^-_L(C) \subseteq R^-(L)$, then we already find a loop C preventing L to be an elementary loop. In the latter case, for any rule $r \in R^-_L(C) \setminus R^-(L)$, if $|head(r) \cap C| = 1$, then $head(r) \cap C$ cannot be in a loop $L' \subseteq C$ not outbound in L, otherwise r must be in $R^-_L(L')$, a contradiction with $R^-_L(L') \subseteq R^-(L)$. If $|head(r) \cap C| > 1$, there could be a loop $L' \subseteq C$ *s.t.* $r \in R^-(L')$ and $head(r) \cap (L \setminus L') \neq \emptyset$, then $L' \cap (head(r) \cap C) \neq \emptyset$. So if $|head(r) \cap C| = 1$, then we can remove $head(r)$ from the subgraph and continue the procedure, else we cannot remove $head(r)$ in general. However, Algorithm 1 removes $head(R^-_L(C) \setminus R^-(L))$ from the subgraph, then for some loop L that is not an elementary loop, $\text{EL}^*(L, P)$ would still return L.

Algorithm 1 removes at least one atom in the subgraph at one time. In the worst case, the process runs n^2 times where n is the number atoms in L. As the set of SCCs of a subgraph can be computed in linear time, the time complexity of the algorithm is $O(n^2)$.

Proposition 3 *For any DLP P and any loop L of P, the function $EL^*(L, P)$ returns either L or a set C of atoms s.t. C is a loop of P, $C \subset L$, and $R^-_L(C) \subseteq R^-(L)$ in $O(n^2)$, where n is the number of atoms in L. If $EL^*(L, P)$ returns C s.t. $C \neq L$, then L is not an elementary loop of P.*

$\text{EL}^*(L, P)$ may return L while L is not an elementary loop.

Example 2 *Consider a loop $L = \{p, q, r\}$ of a program P_2:*

$$p \vee q \leftarrow r. \qquad p \vee r \leftarrow q. \qquad q \vee r \leftarrow p. \qquad r \leftarrow .$$

$EL^*(L, P)$ returns L, while L is not an elementary loop as $R^-_L(\{p, q\}) = \emptyset$.

We use $\text{EL}(P)$ to denote the set of all elementary loops of a DLP P. With a slight abuse of the notion, we denote

$$\text{EL}^*(P) = \{L \mid L \text{ is a loop of } P \text{ and } \text{EL}^*(L, P) \text{ returns } L\}.$$

For any DLP P, $\text{EL}(P) \subseteq \text{EL}^*(P)$. If P is a NLP, then $\text{EL}(P) = \text{EL}^*(P)$. The result is also true for HEF programs.

Proposition 4 *For any HEF program P, $EL(P) = EL^*(P)$.*

Weak Elementary Loops and HWEF Programs

Although properties of elementary loops and HEF programs could benefit the answer set computation, the computational complexities of recognizing them are coNP-complete. In this section, we present a superclass of elementary loops and a subclass of HEF programs and provide polynomial time algorithms to recognize them respectively.

For any loops L_1, L_2, $R^-(L_1) \subseteq R^-(L_2)$ implies $R^-_{L_2}(L_1) \subseteq R^-(L_2)$. Proposition 1 implies the corollary.

Corollary 3 *Let P be a DLP and L_1, L_2 loops of P. If $L_1 \subseteq L_2$ and $R^-(L_1) \subseteq R^-(L_2)$, then $CLF(L_1, P) \supset CLF(L_2, P)$.*

Then we can define a weaker version of elementary loops. Let P be a DLP and L a loop of P, we say that L is a *weak elementary loop* of P if there does not exist a proper subset C of L *s.t.* $R^-(C) \subseteq R^-(L)$.

Proposition 5 *For any DLP P and any loop L of P, L is an elementary loop of P implies L is a weak elementary loop, but not vice versa in general.*

Example 1 (Continued) *$\{p, q, r\}$ is a weak elementary loop of P_1 but it is not an elementary loop.*

The problem of deciding whether a loop is a weak elementary loop for a DLP is tractable. We can construct a polynomial time algorithm by replacing each occurrence of the formula $R^-_L(C)$ with $R^-(C)$ in Algorithm 1. In specific, we use WEL(L, P) to denote the resulting function.

Proposition 6 *For any DLP P and any loop L of P, the function WEL(L, P) returns either L or a set C of atoms s.t. C is a loop of P, $C \subset L$, and $R^-(C) \subseteq R^-(L)$ in $O(n^2)$, where n is the number of atoms in L. WEL(L, P) returns L iff L is a weak elementary loop.*

We use WEL(P) to denote the set of all weak elementary loops of P. For any DLP P, EL$(P) \subseteq$ EL$^*(P) \subseteq$ WEL(P). If P is a NLP, then WEL$(P) =$ EL$^*(P) =$ EL(P).

Proposition 7 *For any HEF program P, EL$(P) =$ EL$^*(P) =$ WEL(P).*

Based on the notion, we define a class of DLP, called "Head-Weak-Elementary-loop-Free" (HWEF). A DLP P is called *HWEF* if $|head(r) \cap L| \leq 1$ for every rule r of P and every weak elementary loop L of P. An HWEF program is an HEF program, so the answer sets of an HWEF program P coincides with the answer sets of $sh(P)$. However it is still coNP-hard to identify HWEF programs.

Proposition 8 *Let P be a DLP. Deciding whether P is an HWEF program is coNP-complete.*

Weak elementary loops have some interesting properties, that could help us to construct a polynomial time algorithm to identify a subclass of HWEF programs.

Proposition 9 *Let P be a DLP, L a weak elementary loop of P and E a nonempty proper subset of L. Then there exists a rule $r \in P$ s.t. $body^+(r) \cap E \neq \emptyset$, $body^+(r) \cap (L \setminus E) = \emptyset$ and $head(r) \cap (L \setminus E) \neq \emptyset$.*

In the proposition, if such a rule r does not exist, then $L \setminus E \subset L$ and $R^-(L \setminus E) \subseteq R^-(L)$, a contradiction with the condition that L is a weak elementary loop. On the other hand, if there do not exist such a loop L and a rule r, then every proper superset of E is not a weak elementary loop of P.

For any DLP P and any nonempty set E of atoms, we provide Algorithm 2 to decide whether there exists a loop L and a rule r *s.t.* $E \subseteq L$, $body^+(r) \cap E \neq \emptyset$, $body^+(r) \cap (L \setminus E) = \emptyset$, and $head(r) \cap (L \setminus E) \neq \emptyset$, *i.e.*, L has the possibility to be a weak elementary loop of P.

Proposition 10 *For any DLP P and any nonempty set E of atoms, the function EWEL(P, E) returns either $true$ or $false$ in $O(m)$, where m is the number of rules in P. If EWEL(P, E) returns $false$, then there does not exist a weak elementary loop L of P s.t. $E \subset L$.*

We provide Algorithm 3 running in polynomial time to decide whether a DLP is in a subclass of HWEF programs.

Algorithm 2: EWEL(P, E)

```
G := the positive dependency graph of P;
if there does not exist a SCC C of G s.t. E ⊆ C then
    return false
C := the SCC of G s.t. E ⊆ C;
R_E := {r | r ∈ P and body+(r) ∩ E ≠ ∅};
for each r ∈ R_E do
    G_r := the C\(body+(r)\E) induced subgraph of G;
    if there exists a SCC C_r of G_r s.t. E ⊆ C_r and
       head(r) ∩ (C_r \ E) ≠ ∅ then
        return true
return false
```

Algorithm 3: HWEF$^*(P)$

```
E := {{a, b} | there is a rule r ∈ P s.t. {a, b} ⊆ head(r)};
for each E ∈ E do
    if E is a weak elementary loop of P then
        return false
    if EWEL(P, E) returns true then
        return false
return true
```

Proposition 11 *For any DLP P, the function HWEF$^*(P)$ returns either $true$ or $false$ in $O(mn^2)$ where m is the number of rules in P and n is the number of atoms in P. If HWEF$^*(P)$ return $true$, then P is an HWEF program.*

With a slight abuse of the notion, we call a DLP P to be an HWEF* program, if HWEF$^*(P)$ returns $true$. An HWEF* program is an HWEF program, but not vice versa in general.

Example 3 *Consider the logic program P_3:*

$$p \vee q \leftarrow r. \qquad r \leftarrow p, q. \qquad p \leftarrow .$$

P_3 has five weak elementary loops: $\{p\}$, $\{q\}$, $\{r\}$, $\{p, r\}$, $\{r, q\}$. The loop $\{p, r, q\}$ is not a weak elementary loop as $R^-(\{p, r, q\}) = \{p \leftarrow .\}$ and $R^-(\{r, q\}) = \emptyset$. So P_3 is an HWEF program, however EWEL$(P_3, \{p, q\})$ returns $true$, then HWEF$^*(P_3)$ returns $false$.

The class of HWEF* programs strictly generalizes the class of HCF programs and the problem of recognizing them is tractable. We found that 32% DLPs in "qbf.cgs", a class of benchmark for DLPs which are not HCF (Gebser, Kaufmann, and Schaub 2012), are HWEF* programs.[1]

Proper Loops and HPF Programs

Following the idea proposed in (Ji et al. 2014), we show that not all elementary loops are needed for the answer set computation. We identify a subclass *proper loops*, and show that, by applying a special form of their loop formulas, they are sufficient for selecting answer sets from models of a DLP. Moreover, we characterize a superclass of HEF programs and show that these programs can be turned into equivalent NLPs by "shifting".

[1] http://ss.sysu.edu.cn/%7ewh/properloopdlp.html

Let P be a DLP and L a loop of P, we use $RLF(L,P)$ to denote the implication:

$$\bigwedge_{p\in head(R^-(L))\cap L} p \supset \bigvee_{r\in R^-(L)} \left(body(r) \wedge \bigwedge_{q\in head(r)\setminus L} \neg q \right)$$

if $R^-(L) \neq \emptyset$, otherwise

$$\bigwedge_{p\in L} p \supset \bot.$$

Clearly, $RLF(L,P)$ is a special case of $LF(L,P)$.

Proposition 12 *Let P be a DLP and L_1, L_2 loops of P. If $R^-(L_1) \neq \emptyset$, $R^-(L_2) \neq \emptyset$, $head(R^-(L_1)) \cap (L_1 \cup L_2) \subseteq head(R^-(L_2)) \cap L_2$, and $R^-_{L_2}(L_1) \subseteq R^-(L_2)$, then $RLF(L_1,P) \supset RLF(L_2,P)$.*

A loop L is a *proper loop* of a DLP P, if

- L is an elementary loop of P, and
- there does not exist another elementary loop L' of P *s.t.* $R^-(L') \neq \emptyset$, $head(R^-(L')) \cap (L' \cup L) \subseteq head(R^-(L)) \cap L$ and $R^-_L(L') \subset R^-(L)$.

Theorem 4 *Each of following conditions is equivalent to each of conditions in Theorem 1:*

8. S satisfies $RLF(L,P)$ for all proper loops L of P;
9. S satisfies $DLF(L,P)$ for all proper loops L of P.

When loop formulas are in the form of *RLF*, more redundant loops can be removed from elementary loops.

Example 1 (Continued) *Program P_1 has four proper loops: $\{p\}$, $\{q\}$, $\{p,r\}$, and $\{r,q\}$. $\{p,r,q\}$ is not a proper loop as it is not an elementary loop. $\{r\}$ is not a proper loop as $R^-_{\{r\}}(\{r,q\}) = \{r \leftarrow p.\}$ and $R^-(\{r\}) = \{r \leftarrow p.\, r \leftarrow q.\}$.*

The definition of proper loops can be simplified for programs that do not have loops without external support rules. In particular, a DLP P is called *simplified* if there does not exist a loop L of P *s.t.* $R^-(L) = \emptyset$. Note that, every DLP P can be turned into an equivalent DLP $simp(P)$ by deleting:

1. each rule r that $body^+(r) \cap L \neq \emptyset$ for some loop L of P *s.t.* $R^-(L) = \emptyset$, and
2. all atoms p in the heads and all formulas of the form $not\, p$ in the bodies of the remaining rules that $p \in L$ for some loop L of P *s.t.* $R^-(L) = \emptyset$.

$simp(P)$ can be turned into $simp(simp(P))$, and so on. The process continues until resulting a simplified program. Any DLP P can be turned into an equivalent simplified program.

Proposition 13 *A loop L is a proper loop of a simplified program P, iff*

- *there does not exist a nonempty proper subset C of L s.t. $R^-_L(C) \subseteq R^-(L)$, and*
- *there does not exist a nonempty set C of atoms s.t. $head(R^-(C)) \cap (C \cup L) \subseteq head(R^-(L)) \cap L$ and $R^-_L(C) \subset R^-(L)$.*

Proposition 14 *For any DLP P and loop L of P, the problem of deciding whether L is a proper loop of P is coNP-complete.*

Algorithm 4: PL$^*(L,P)$

```
SCC := the set of SCCs of G_P;
for each C ∈ SCC do
    if C ⊂ L and R^-_L(C) ⊆ R^-(L) then return C;
    else if head(R^-(C))∩(C ∪ L) ⊆ head(R^-(L)) ∩ L
                and R^-_L(C) ⊂ R^-(L) then
        return C
    else if R^-_L(C) = R^-(L) then
        for each atom a ∈ C do
            G* := the C \ {a} induced subgraph of G_P;
            SCC* := the set of SCCs of G*;
            append new elements from SCC* to SCC;
    else if head(R^-(C)) ∩ (C ∪ L) ⊈ head(R^-(L)) ∩ L
                and C ⊈ L then
        C' := C \ ((head(R^-(C)) ∩ C) \ (head(R^-(L)) ∩ L));
        G' := the C' induced subgraph of G_P;
        SCC' := the set of SCCs of G';
        append new elements from SCC' to SCC;
    else
        G_C := the C \ head(R^-_L(C) \ R^-(L)) induced
                    subgraph of G_P;
        SCC_C := the set of SCCs of G_C;
        append new elements from SCC_C to SCC;
return L
```

Similar to Algorithm 1, we provide Algorithm 4 running in polynomial time to decide whether a loop L is in a superclass of all proper loops of a simplified program P.

Proposition 15 *For any simplified program P and any loop L of P, PL$^*(L,P)$ returns either L or a loop C of P s.t.*

- *$C \subset L$ and $R^-_L(C) \subseteq R^-(L)$, or*
- *$head(R^-(C)) \cap (C \cup L) \subseteq head(R^-(L)) \cap L$ and $R^-_L(C) \subset R^-(L)$,*

in $O(n^2)$, where n is the number of atoms in P. If PL$^*(L,P)$ returns C s.t. $C \neq L$, then L is not a proper loop of P.

In the following we use PL(P) to denote the set of all proper loops of a DLP P. Similarly, we denote

$$\text{PL}^*(P) = \{\, L \mid L \text{ is a loop of } P \text{ and PL}^*(L,P) \text{ returns } L \,\}.$$

For any simplified program P, PL$(P) \subseteq$ PL$^*(P)$. If P is a simplified NLP, then PL$(P) =$ PL$^*(P)$.

Based on the notion of proper loops, we define a class of DLPs, called "Head-Proper-loop-Free" (HPF). A DLP P is called *HPF* if $|head(r) \cap L| \leq 1$ for every rule r of P and every proper loop L of P. We show that, every HPF program can also be turned into an equivalent NLP by "shifting".

Proposition 16 *For any HPF program P, a set S is an answer set of P iff S is an answer set of $sh(P)$.*

Proposition 17 *A DLP P is HEF implies P is HPF.*

Proposition 18 *The problem of deciding whether a DLP P is HPF is coNP-complete.*

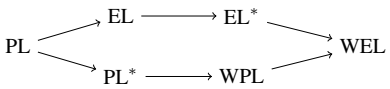

Figure 2: Summary of relations between classes of loops

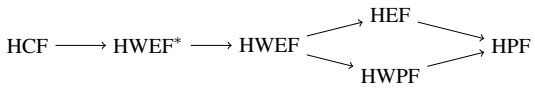

Figure 3: Summary of relations between classes of DLPs

Weak Proper Loops and HWPF Programs

The computational complexities of recognizing proper loops and HPF programs are coNP-complete respectively. In this section, we characterize a superclass of proper loops which can be recognized in polynomial time and present a subclass of HPF programs.

Form Proposition 12, we have the following corollary.

Corollary 5 *Let P be a DLP and L_1, L_2 loops of P. If $R^-(L_1) \neq \emptyset$, $head(R^-(L_1)) \cap L_1 \subseteq head(R^-(L_2)) \cap L_2$ and $R^-(L_1) \subseteq R^-(L_2)$, then $RLF(L_1, P) \supset RLF(L_2, P)$.*

Then we can define a weaker version of proper loops. A loop L is a *weak proper loop* of a DLP P if

- L is a weak elementary loop of P, and
- there does not exist another weak elementary loop L' of P *s.t.* $R^-(L') \neq \emptyset$, $head(R^-(L')) \cap L' \subseteq head(R^-(L)) \cap L$ and $R^-(L') \subset R^-(L)$.

Example 1 (Continued) *Program P_1 has five weak proper loops: $\{p\}$, $\{q\}$, $\{p, r\}$, $\{r, q\}$, $\{p, r, q\}$. $\{r\}$ is not a proper loop as $R^-(\{r, q\}) = \{r \leftarrow p.\}$ and $R^-(\{r\}) = \{r \leftarrow p.\, r \leftarrow q.\}$.*

Theorem 6 *Each of following conditions is equivalent to each of conditions in Theorem 1:*

10. S satisfies $RLF(L, P)$ for all weak proper loops L of P;
11. S satisfies $DLF(L, P)$ for all weak proper loops L of P.

Proposition 19 *For any DLP P and any loop L of P, L is a proper loop of P implies L is a weak proper loop of P, but not vice versa in general.*

If P is a NLP, L is a proper loop iff L is a weak proper loop. Figure 2 summarizes the relations between classes of loops, where $\rightarrow$ indicates the subset relation (also in Figure 3).

For simplified programs, the definition could be easier.

Proposition 20 *A loop L is a weak proper loop of a simplified program P, iff*

- *there does not exist a nonempty proper subset C of L s.t. $R^-(C) \subseteq R^-(L)$, and*
- *there does not exist a nonempty set C s.t. $head(R^-(C)) \cap C \subseteq head(R^-(L)) \cap L$ and $R^-(C) \subset R^-(L)$.*

Different from proper loops, the problem of recognizing a weak proper loop for a simplified program is tractable. We can construct a polynomial time algorithm by replacing each occurrence of the formula $R^-_L(C)$ with $R^-(C)$ and each occurrence of the formula $head(R^-(C)) \cap (C \cup L)$ with $head(R^-(C)) \cap C$ in Algorithm 4. In specific, we use $\mathrm{WPL}(L, P)$ to denote the resulting function.

Proposition 21 *For any simplified program P and loop L of P, $WPL(L, P)$ returns either L or a loop C of P s.t.*

- *$C \subset L$ and $R^-(C) \subseteq R^-(L)$, or*
- *$head(R^-(C)) \cap C \subseteq head(R^-(L)) \cap L$ and $R^-(C) \subset R^-(L)$,*

in $O(n^2)$, where n is the number of atoms in P. $WPL(L, P)$ returns L iff L is a weak proper loop of P.

A DLP P is called "Head-Weak-Proper-loop-Free" (HWPF) if $|head(r) \cap L| \leq 1$ for every rule r of P and every weak proper loop L of P. An HWPF program is an HPF program and an HWEF program is an HWPF program. Figure 3 summarizes the relations between classes of DLPs.

Proposition 22 *The problem of deciding whether a DLP P is HWPF is coNP-complete.*

Experiments

In this section, to compare various notions of loops in practice, *i.e.*, elementary loops (EL), $\mathrm{EL}^*(P)$ (EL^*), weak elementary loops (WEL), proper loops (PL), $\mathrm{PL}^*(P)$ (PL^*), and weak proper loops (WPL), we count numbers of loops in these classes on DLPs in the benchmark. The experiments were run on a Linux machine with AMD A10-5800K (3.8GHz) CPU and 3.3GB RAM, limiting each run to 1 hour. Table 1[2] shows average numbers of loops for different notions on 63 DLPs from 7 classes[3], which were frequently used to compare the performance of ASP solvers for DLPs (Denecker et al. 2009; Gebser, Kaufmann, and Schaub 2012; 2013). DLPs in the classes of "Sokoban" and "SCore-disjunctiveloop" are HCF programs, while others are not HCF. When it was "timeout" for finding all loops of a DLP, the numbers of loops for the DLP were counted only for loops that had been found. The last row displays average numbers of classes of loops over all DLPs.

The result shows that, on average for these DLPs P, $|\mathrm{PL}(P)|$ is smaller than $|\mathrm{EL}(P)|$, $|\mathrm{EL}^*(P)|$ and $|\mathrm{WEL}(P)|$ are similar to $|\mathrm{EL}(P)|$, and $|\mathrm{PL}^*(P)|$ and $|\mathrm{WPL}(P)|$ are similar to $|\mathrm{PL}(P)|$. Even though finding all loops is a blow up job, in the experiment of large programs, we found that in the process of calculating, the number of certain class of loops increases in a similar ratio, which implies that we can replace elementary loops and proper loops by weak elementary loops and weak proper loops respectively in practice.

Conclusion

We provide an alternative definition of elementary loops for DLPs, which results an approximate algorithm running in polynomial time for recognizing elementary loops. Then we extend the notion of proper loops to DLPs and provide a further refinement of the Lin-Zhao theorem for DLPs. We also provide an approximate algorithm running in polynomial

[2] http://ss.sysu.edu.cn/%7ewh/properloopdlp.html
[3] http://www.cs.uni-potsdam.de/claspD

Table 1: Numbers of loops of different types

Benchmark	number	timeout	Loops	EL	EL*	WEL	PL	PL*	WPL
Sokoban (HCF)	11	0	2546	2525	2525	2525	642	642	642
SCore-disjunctiveloop (HCF)	2	1	5251	5251	5251	5251	1751	1751	1751
qbf.cgs	23	3	15095	1407	1407	1441	1407	1407	1441
qbf.gw	18	7	10751	4153	4162	5302	4150	4153	5302
SCore-Mutex	2	2	9250	2445	2562	3371	2445	2448	3371
SCore-RandomQBF	6	6	6417	6388	6391	6392	6388	6391	6392
SCore-StrategicCompanies	1	1	8000	6954	6983	6959	6983	6985	6992
Average Number	63	20	8188	4165	4180	4463	3396	3398	3698

time for recognizing proper loops. The computational complexities of recognizing elementary loops and proper loops for DLPs are coNP-complete. Then we introduce weaker versions, *i.e.*, weak elementary loops and weak proper loops, and provide polynomial time algorithms for recognizing them respectively.

The class of HEF programs which strictly generalizes the class HCF programs are defined based on the notion of elementary loops. Similarly, we define the class of HPF programs, HWEF programs, and HWPF programs based on notions of proper loops, weak elementary loops, and weak proper loops respectively. However, recognizing these classes of DLPs are coNP-complete. We discuss relationships between these classes of loops and DLPs respectively. Moreover, we present the class of HWEF* programs which generalizes HCF programs, show that an HWEF* program can be turned into an equivalent NLP by shifting head atoms into the body, and provide a polynomial time algorithm to identify them. At last, some experiments show that both elementary loops and proper loops could be replaced by weak elementary loop and weak proper loops in practice, which could improve the performance of ASP solvers for DLPs.

Acknowledgments

We thank the reviewers for their comments and suggestions for improving the paper. We are grateful to Fangzhen Lin for many helpful and informative discussions. We are also grateful to Yan Zhang for his useful suggestions. We would also like to thank Xiaoping Chen and his research group for their useful discussions. Jianmin Ji's research was partially supported by the Fundamental Research Funds for the Central Universities under grant WK0110000035, the National Natural Science Foundation of China under grant 61175057, the National Natural Science Foundation for the Youth of China under grant 61403359, as well as the USTC Key Direction Project and the USTC 985 Project. Hai Wan thanks Research Fund for the Doctoral Program of Higher Education of China (No. 20110171120041), Natural Science Foundation of Guangdong Province of China (No. S2012010009836), and Guangzhou Science and Technology Project (No. 2013J4100058) for the support of this research. We also thank supports from the National Natural Science Foundation of China under grant 61370161.

References

Ben-Eliyahu, R., and Dechter, R. 1994. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial intelligence* 12(1-2):53–87.

Denecker, M.; Vennekens, J.; Bond, S.; Gebser, M.; and Truszczyński, M. 2009. The second answer set programming competition. In *Proceedgins of the 10th International Converence on Logic Programming and Nonmonotonic Reasoning (LPNMR-09)*, 637–654. Springer.

Drescher, C.; Gebser, M.; Grote, T.; Kaufmann, B.; König, A.; Ostrowski, M.; and Schaub, T. 2008. Conflict-driven disjunctive answer set solving. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR-08)*, 422–432.

Fassetti, F., and Palopoli, L. 2010. On the complexity of identifying head-elementary-set-free programs. *Theory and Practice of Logic Programming* 10(01):113–123.

Gebser, M., and Schaub, T. 2005. Loops: relevant or redundant? In *Proceedings of 8th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-05)*, 53–65.

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T. 2007. Conflict-driven answer set solving. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 386–392.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2012. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence* 187:52–89.

Gebser, M.; Kaufmann, B.; and Schaub, T. 2013. Advanced conflict-driven disjunctive answer set solving. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-13)*, 912–918. AAAI Press.

Gebser, M.; Lee, J.; and Lierler, Y. 2011. On elementary loops of logic programs. *Theory and Practice of Logic Programming* 11(6):953–988.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New generation computing* 9(3-4):365–385.

Gelfond, M.; Lifschitz, V.; Przymusinska, H.; and Truszczynski, M. 1991. Disjunctive defaults. In *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR-91)*, 230–237.

Giunchiglia, E.; Lierler, Y.; and Maratea, M. 2006. Answer set programming based on propositional satisfiability. *Journal of Automated Reasoning* 36(4):345–377.

Ji, J.; Wan, H.; Xiao, P.; Huo, Z.; and Xiao, Z. 2014. Elementary loops revisited. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI-14)*, 1063–1069.

Lee, J., and Lifschitz, V. 2003. Loop formulas for disjunctive logic programs. In *Proceedings of the 19th International Conference on Logic Programming (ICLP-03)*, 451–465.

Lifschitz, V., and Razborov, A. 2006. Why are there so many loop formulas? *ACM Transactions on Computational Logic* 7(2):261–268.

Lin, F., and Zhao, Y. 2004. ASSAT: computing answer sets of a logic program by SAT solvers. *Artificial Intelligence* 157(1-2):115–137.