Elementary Sets for Logic Programs

Martin Gebser
Institut für Informatik
Universität Potsdam, Germany

Joohyung Lee
Computer Science and Engineering
Arizona State University, USA

Yuliya Lierler
Department of Computer Science
Universität Erlangen-Nürnberg, Germany

Abstract

By introducing the concepts of a loop and a loop formula, Lin and Zhao showed that the answer sets of a nondisjunctive logic program are exactly the models of its Clark's completion that satisfy the loop formulas of all loops. Recently, Gebser and Schaub showed that the Lin-Zhao theorem remains correct even if we restrict loop formulas to a special class of loops called "elementary loops." In this paper, we simplify and generalize the notion of an elementary loop, and clarify its role. We propose the notion of an elementary set, which is almost equivalent to the notion of an elementary loop for nondisjunctive programs, but is simpler, and, unlike elementary loops, can be extended to disjunctive programs without producing unintuitive results. We show that the maximal unfounded elementary sets for the "relevant" part of a program are exactly the minimal sets among the nonempty unfounded sets. We also present a graph-theoretic characterization of elementary sets for nondisjunctive programs, which is simpler than the one proposed by Gebser and Schaub. Unlike in the case of nondisjunctive programs, we show that, for disjunctive programs, the problem of deciding whether a set is elementary is coNP-complete.

Introduction

By introducing the concepts of a loop and a loop formula, Lin and Zhao [2004] showed that the answer sets (a.k.a. stable models) of a nondisjunctive logic program are exactly the models of its Clark's completion (Clark 1978) that satisfy the loop formulas $LF(L)$ of all loops L of the program. This important result has shed new light on the relationship between answer sets and completion, and allowed us to compute answer sets using SAT solvers, which led to the design of answer set solvers ASSAT[1] (Lin and Zhao 2004) and CMODELS[2] (Giunchiglia *et al.* 2004).

The concepts of a loop and a loop formula were further clarified in (Lee 2005). By slightly modifying the definition of a loop, Lee observed that adding loop formulas can be viewed as a generalization of completion, which allows us to characterize the stability of a model in terms of loop formulas: A model is stable iff it satisfies the loop formulas of all loops. He also observed that the mapping *LF*, which turns loops into loop formulas, can be applied to arbitrary sets of atoms, not only to loops: Adding $LF(Y)$ where Y is a non-loop does not affect the models of the theory because $LF(Y)$ is always entailed by $LF(L)$ for some loop L. Though this reformulation of the Lin-Zhao theorem, in which *LF* is not restricted to loops, is less economical, it is interesting to note that this is essentially a theorem on assumption sets (Saccá and Zaniolo 1990), or unfounded sets (Van Gelder *et al.* 1991; Leone *et al.* 1997), which has been known for many years. In this sense, one can say that the most original contribution of (Lin and Zhao 2004) was not the mapping that turns loops into loop formulas, but the definition of a loop, which yields a relatively small class of sets of atoms for the mapping *LF*.

However, for nondisjunctive programs, even the definition of a loop turned out still "too generous." Gebser and Schaub [2005] showed that restricting the mapping even more to a special class of loops called "elementary loops," yields a valid modification of the Lin-Zhao theorem (or the Saccá-Zaniolo theorem). That is, some loops are identified as redundant, just as all non-loops are redundant. They noted that the notion of a positive dependency graph, which is used for defining a loop, is not expressive enough to allow us to distinguish between elementary and non-elementary loops, and proposed an alternative graph-theoretic characterization, based on the notion of a so-called "body-head dependency graph."

Our work is motivated by the desire to understand the role of an elementary loop further and to extend the results to disjunctive programs. For nondisjunctive programs, we propose a simpler notion corresponding to elementary loops, which we call "elementary sets," and provide a further enhancement of the Lin-Zhao theorem based on it. Unlike elementary loops, elementary sets can be extended to disjunctive programs without producing unintuitive results. We show that a special class of unfounded elementary sets coincides with the minimal sets among nonempty unfounded sets. Instead of relying on the complicated notion of a "body-head dependency graph," we present a simpler graph-theoretic characterization of elementary sets, based on a subgraph of the positive dependency graph.

Nondisjunctive Programs

Review of Loop Formulas: Nondisjunctive Case

A *nondisjunctive rule* is an expression of the form

$$a_1 \leftarrow a_2, \ldots, a_m, \mathit{not}\ a_{m+1}, \ldots, \mathit{not}\ a_n \qquad (1)$$

where $n \geq m \geq 1$ and $a_1, \ldots, a_n$ are propositional atoms. A *nondisjunctive program* is a finite set of nondisjunctive rules.

[1] http://assat.cs.ust.hk/

[2] http://www.cs.utexas.edu/users/tag/cmodels/

We will identify a nondisjunctive rule (1) with the propositional formula

$$(a_2 \wedge \cdots \wedge a_m \wedge \neg a_{m+1} \wedge \cdots \wedge \neg a_n) \rightarrow a_1,$$

and will often write (1) as

$$a_1 \leftarrow B, F \tag{2}$$

where B is $a_2, \ldots, a_m$ and F is *not* $a_{m+1}, \ldots,$ *not* a_n. We will sometimes identify B with its corresponding set.

For the definition of a stable model of a nondisjunctive program, we refer the reader to (Lee 2005, Section 2.1).

Let Π be a nondisjunctive program. The *(positive) dependency graph* of Π is the directed graph such that its vertices are the atoms occurring in Π, and its edges go from a_1 to $a_2, \ldots, a_m$ for all rules (1) of Π. A nonempty set L of atoms is called a *loop* of Π if, for every pair p, q of atoms in L, there exists a path (possibly of length 0) from p to q in the dependency graph of Π such that all vertices in this path belong to L. In other words, L is a loop of Π iff the subgraph of the dependency graph of Π induced by L is strongly connected. It is clear that any set consisting of a single atom is a loop.

For example, consider the following program Π_1:

$$p \leftarrow \textit{not } s \qquad p \leftarrow r \qquad q \leftarrow r \qquad r \leftarrow p, q\ .$$

Figure 1 shows the dependency graph of Π_1; the program has seven loops: $\{p\}$, $\{q\}$, $\{r\}$, $\{s\}$, $\{p, r\}$, $\{q, r\}$, $\{p, q, r\}$.

For any set Y of atoms, the *external support formula* of Y, denoted by $ES_\Pi(Y)$, is the disjunction of conjunctions $B \wedge F$ for all rules (2) of Π such that $a_1 \in Y$ and $B \cap Y = \emptyset$. The first condition expresses that the atom "supported" by (2) is an element of Y. The second condition ensures that this support is "external": The atoms in B that it relies on do not belong to Y. Thus Y is called *externally supported* by Π w.r.t. a set X of atoms if $X \models ES_\Pi(Y)$.

For any set Y of atoms, by $LF_\Pi(Y)$ we denote the following formula:

$$\bigwedge_{a \in Y} a \rightarrow ES_\Pi(Y)\ . \tag{3}$$

Formula (3) is called the *(conjunctive) loop formula* of Y for Π. Note that we still call (3) a loop formula even when Y is not a loop.

The following reformulation of the Lin-Zhao theorem, which characterizes the stability of a model in terms of loop formulas, is a part of the main theorem from (Lee 2005) for the nondisjunctive case.

Theorem 1 *(Lee 2005) Let Π be a nondisjunctive program, and X a set of atoms occurring in Π. If X satisfies Π, then the following conditions are equivalent:* [3]

(a) X is stable;

(b) X satisfies $LF_\Pi(Y)$ for all nonempty sets Y of atoms that occur in Π;

(c) X satisfies $LF_\Pi(Y)$ for all loops Y of Π.

According to the equivalence between conditions (a) and (b) in Theorem 1, a model of Π_1 is stable iff it satisfies

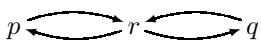

Figure 1: The dependency graph of Program Π_1

the loop formulas of all 15 nonempty sets of atoms occurring in Π_1. On the other hand, condition (c) tells us that it is sufficient to restrict attention to the following 7 loop formulas:

$$\begin{array}{lll} p \rightarrow \neg s \vee r & r \rightarrow p \wedge q & p \wedge r \rightarrow \neg s \\ q \rightarrow r & s \rightarrow \bot & q \wedge r \rightarrow \bot \\ & & p \wedge q \wedge r \rightarrow \neg s\ . \end{array} \tag{4}$$

Program Π_1 has six models: $\{p\}$, $\{s\}$, $\{p, s\}$, $\{q, s\}$, $\{p, q, r\}$, and $\{p, q, r, s\}$. Among them, $\{p\}$ is the only stable model, which is also the only model that satisfies all loop formulas in (4). In the next section, we will see that in fact the last loop formula can be disregarded also, if we take elementary sets into account.

As noted in (Lee 2005), the equivalence between conditions (a) and (c) is a reformulation of the Lin-Zhao theorem; the equivalence between conditions (a) and (b) is a reformulation of Corollary 2 of (Saccá and Zaniolo 1990), and Theorem 4.6 of (Leone *et al.* 1997) (in the nondisjunctive case), which characterizes the stability of a model in terms of *unfounded sets*. For sets X, Y of atoms, we say that Y is *unfounded* by Π w.r.t. X if Y is not externally supported by Π w.r.t. X. Condition (b) can be stated in terms of unfounded sets as follows:

(b′) X contains no nonempty unfounded subsets for Π w.r.t. X.

Elementary Sets for Nondisjunctive Programs

As mentioned in the introduction, Gebser and Schaub [2005] showed that *LF* in Theorem 1 can be further restricted to "elementary loops." In this section, we present a simpler reformulation of their results. We will compare our reformulation with the original definition from (Gebser and Schaub 2005) later in this paper.

Let Π be a nondisjunctive program. The following proposition tells us that a loop can be defined even without referring to a dependency graph.

Proposition 1 *For any nondisjunctive program Π and for any nonempty set Y of atoms that occur in Π, Y is a loop of Π iff, for every nonempty proper subset Z of Y, there is a rule (2) in Π such that $a_1 \in Z$ and $B \cap (Y \setminus Z) \neq \emptyset$.*

For any set Y of atoms and any subset Z of Y, we say that Z is *outbound* in Y for Π if there is a rule (2) in Π such that $a_1 \in Z$, $B \cap (Y \setminus Z) \neq \emptyset$, and $B \cap Z = \emptyset$.

For any nonempty set Y of atoms that occur in Π, we say that Y is *elementary* for Π if all nonempty proper subsets of Y are outbound in Y for Π.

As with loops, it is clear from the definition that every singleton set of atoms occurring in Π is an elementary set for Π. It is also clear that every elementary set for Π is a loop of Π: The conditions for being an elementary set are stronger than the conditions for being a loop, as given in Proposition 1. On the other hand, a loop is not necessarily an elementary set. For instance, one can check that, for Π_1, $\{p, q, r\}$ is not elementary since $\{p, r\}$ (or $\{q, r\}$) is not outbound in $\{p, q, r\}$. All the other loops of Π_1 are elementary. Note that an elementary set may be a proper subset of another elementary set (both $\{p\}$ and $\{p, r\}$ are elementary sets for Π_1).

[3] We identify an interpretation with the set of atoms that are true in it. Recall that we identify the rule (1) with the corresponding propositional formula.

The following program replaces the last rule of Π_1 by two rules:

$$\begin{array}{lll} p \leftarrow \mathit{not}\ s & q \leftarrow r & r \leftarrow q. \\ p \leftarrow r & r \leftarrow p & \end{array} \quad (5)$$

Program (5) has the same dependency graph as program Π_1 and hence has the same set of loops. However its elementary sets are different: All its loops are elementary.

Consider another program:

$$\begin{array}{lll} p \leftarrow r & q \leftarrow q, r & r \leftarrow p, q \\ p \leftarrow \mathit{not}\ s & q \leftarrow s & s \leftarrow q, s\ . \end{array} \quad (6)$$

Program (6) has 10 loops: $\{p\}$, $\{q\}$, $\{r\}$, $\{s\}$, $\{p, r\}$, $\{q, r\}$, $\{q, s\}$, $\{p, q, r\}$, $\{q, r, s\}$, $\{p, q, r, s\}$. Among them, only the first five sets are elementary.

From the definition of an elementary set above, we get an alternative, equivalent definition by requiring that only the loops contained in Y be outbound, instead of requiring that all nonempty proper subsets of Y be outbound.

Proposition 2 *For any nondisjunctive program Π and for any nonempty set Y of atoms that occur in Π, Y is an elementary set for Π iff all loops Z of Π such that $Z \subset Y$ are outbound in Y for Π.*

Note that a subset of an elementary set, even if that subset is a loop, is not necessarily elementary. For instance, in the following program

$$\begin{array}{lll} p \leftarrow p, q & p \leftarrow r & r \leftarrow p \\ q \leftarrow p, q & q \leftarrow r & r \leftarrow q, \end{array}$$

set $\{p, q, r\}$ is elementary, but $\{p, q\}$ is not.

The following proposition describes a relationship between loop formulas of elementary sets and loop formulas of arbitrary sets.

Proposition 3 *Let Π be a nondisjunctive program, X a set of atoms, and Y a nonempty set of atoms. If $X \models LF_\Pi(Z)$ for all elementary sets Z of Π that are contained in Y, then $X \models LF_\Pi(Y)$.*

In view of Proposition 3, the following theorem, which is a reformulation of Theorem 3 from (Gebser and Schaub 2005) in terms of an elementary set, tells us that in Theorem 1 above it is sufficient to consider only the loop formulas of elementary sets.

Theorem 1(d) *The following condition is equivalent to conditions (a)–(c) of Theorem 1.*

(d) X satisfies $LF_\Pi(Y)$ for all elementary sets Y of Π.

Theorem 1(d) tells us that a model of Π_1 is stable iff it satisfies the first six formulas in (4); the loop formula of non-elementary set $\{p, q, r\}$ (the last one in (4)) can be disregarded.

Elementarily Unfounded Sets for Nondisjunctive Programs

If we modify condition (c) of Theorem 1 by replacing "loops" in its statement with "maximal loops," the condition becomes weaker, and the modified statement of Theorem 1 does not hold. For instance, program Π_1 has only two maximal loops, $\{p, q, r\}$ and $\{s\}$, and their loop formulas are satisfied by the non-stable model $\{p, q, r\}$. In fact, maximal loop $\{p, q, r\}$ is not even an elementary set for Π_1.

This is also the case with maximal elementary sets: Theorem 1(d) does not hold if "elementary sets" in its statement is replaced with "maximal elementary sets" as the following program shows:

$$p \leftarrow q, \mathit{not}\ p \qquad q \leftarrow p, \mathit{not}\ p \qquad p\ . \quad (7)$$

Program (7) has two models, $\{p\}$ and $\{p, q\}$, but the latter is not stable. The loop formula of the only maximal elementary set $\{p, q\}$ for (7) ($p \wedge q \rightarrow \top$) is satisfied by both models.

However, in the following we show that if we consider the "relevant" part of the program w.r.t. a given interpretation, it is sufficient to restrict attention to maximal elementary sets.

Given a nondisjunctive program Π and a set X of atoms, by Π_X we denote the set of rules (2) of Π such that $X \models B, F$. The following proposition tells us that all nonempty proper subset of an elementary set for Π_X are externally supported w.r.t. X.

Proposition 4 *For any nondisjunctive program Π, any set X of atoms, and any elementary set Y for Π_X, X satisfies $ES_\Pi(Z)$ for all nonempty proper subsets Z of Y.*

From Proposition 4, it follows that every unfounded elementary set Y for Π_X w.r.t. X is maximal among the elementary sets for Π_X. If Y is a nonempty unfounded set for Π w.r.t. X, but does not contain a maximal elementary set for Π_X, then Y is a singleton whose atom does not occur in Π_X. From this, we obtain the following result.

Theorem 1(e) *The following condition is equivalent to conditions (a)–(c) of Theorem 1.*

(e) X satisfies $LF_\Pi(Y)$ for every set Y of atoms such that Y is a maximal elementary set for Π_X or Y is a singleton whose atom occurs in Π.

We say that a set Y of atoms that occur in Π is *elementarily unfounded* by Π w.r.t. X if Y is an elementary set for Π_X that is unfounded by Π w.r.t. X or if Y is a singleton that is unfounded by Π w.r.t. X.[4] By Proposition 4, it follows that every non-singleton elementarily unfounded set for Π w.r.t. X is a maximal elementary set for Π_X.

It is clear from the definition that every elementarily unfounded set for Π w.r.t. X is an elementary set for Π and that it is also an unfounded set for Π w.r.t. X. However, a set that is both elementary for Π and unfounded by Π w.r.t. X is not necessarily an elementarily unfounded set for Π w.r.t. X. For example, consider the following program.

$$p \leftarrow q, \mathit{not}\ r \qquad q \leftarrow p, \mathit{not}\ r\ . \quad (8)$$

Set $\{p, q\}$ is both elementary for program (8), and unfounded by the program w.r.t. $\{p, q, r\}$, but it is not an elementarily unfounded set w.r.t. $\{p, q, r\}$.

The following corollary, which follows from Proposition 4, tells us that all nonempty proper subsets of an elementarily unfounded set are externally supported. It is essentially a reformulation of Theorem 5 from (Gebser and Schaub 2005).

Corollary 1 *Let Π be a nondisjunctive program, X a set of atoms, and Y an elementarily unfounded set for Π w.r.t. X. Then X does not satisfy $ES_\Pi(Y)$, but X satisfies $ES_\Pi(Z)$ for all nonempty proper subsets Z of Y.*

[4] Elementarily unfounded sets are closely related to "active elementary loops" in (Gebser and Schaub 2005).

Corollary 1 tells us that elementarily unfounded sets form an "anti-chain": One of them cannot be a proper subset of another. [5] In combination with Proposition 4, this tells us that elementarily unfounded sets are minimal among nonempty unfounded sets. Interestingly, the converse also holds.

Proposition 5 *For any nondisjunctive program* Π *and any sets* X, Y *of atoms,* Y *is an elementarily unfounded set for* Π *w.r.t.* X *iff* Y *is minimal among the nonempty sets of atoms occurring in* Π *that are unfounded by* Π *w.r.t.* X.

Similarly to Theorem 1(b′), Theorem 1(e) can be stated in terms of elementarily unfounded sets, thereby restricting attention to minimal unfounded sets.

Theorem 1(e′) *The following condition is equivalent to conditions (a)–(c) of Theorem 1.*

(e′) X *contains no elementarily unfounded subsets for* Π *w.r.t.* X.

The notion of an elementarily unfounded set may help improve computation performed by SAT-based answer set solvers. Since there are exponentially many loops in the worst case, SAT-based answer set solvers do not add all loop formulas at once. Instead, they check whether a model returned by a SAT solver is an answer set. If not, a loop formula that is not satisfied by the current model is added, and the SAT solver is invoked again.[6] This process is repeated until an answer set is found or the search space is exhausted.

In view of Theorem 1(e′), it is sufficient to restrict attention to elementarily unfounded sets during the computation. This guarantees that loop formulas considered are only those of elementary sets. Since every elementary set is a loop, but not vice versa, the computation may involve fewer loop formulas overall than the case when arbitrary loops are considered. In view of Proposition 3 and Corollary 1, considering elementarily unfounded sets provides reasonably the most economical way to eliminate unfounded models.

Deciding Elementary Sets: Nondisjunctive Case

The definition of an elementary set above involves all its nonempty proper subsets (or at least all loops that are subsets of that set). This seems to imply that deciding whether a set is elementary is a computationally hard problem. But in fact, Gebser and Schaub [2005] showed that, for nondisjunctive programs, deciding whether a set is elementary can be done efficiently. They noted that positive dependency graphs are not expressive enough to allow us to distinguish between elementary and non-elementary loops, and introduced the rather complicated notion of a *body-head dependency graph* to identify elementary loops. In this section, we simplify this result by still referring to positive dependency graphs. We show that removing some "unnecessary" edges from a dependency graph is just enough to allow us to distinguish elementary sets from non-elementary sets.

For any nondisjunctive program Π and any set Y of atoms:

Figure 2: The elementary subgraph of $\{p, q, r\}$ for Π_1

$$
\begin{aligned}
EC^0_\Pi(Y) &= \emptyset \\
EC^{i+1}_\Pi(Y) &= EC^i_\Pi(Y) \cup \\
&\quad \{(a_1, b) \mid \text{there is a rule (2) in } \Pi \text{ such that} \\
&\quad b \in B \text{ and the graph } (Y, EC^i_\Pi(Y)) \text{ has a} \\
&\quad \text{strongly connected subgraph containing all} \\
&\quad \text{atoms in } B \cap Y\} \\
EC_\Pi(Y) &= \textstyle\bigcup_{i \geq 0} EC^i_\Pi(Y)\,.
\end{aligned}
$$

Note that this is a "bottom-up" construction. We call the graph $(Y, EC_\Pi(Y))$ the *elementary subgraph* of Y for Π. It is clear that an elementary subgraph is a subgraph of a dependency graph and that it is not necessarily the same as the subgraph of the dependency graph induced by Y. Figure 2 shows the elementary subgraph of $\{p, q, r\}$ for Π_1, which is not strongly connected.

The following theorem is similar to (Gebser and Schaub 2005, Theorem 10), but instead of referring to the notion of a body-head dependency graph, it refers to an elementary subgraph as defined above.

Theorem 2 *For any nondisjunctive program* Π *and any set* Y *of atoms occurring in* Π, Y *is an elementary set for* Π *iff the elementary subgraph of* Y *for* Π *is strongly connected.*

Clearly, constructing an elementary subgraph and checking whether it is strongly connected can be done in polynomial time. Therefore, the problem of deciding whether a given set of atoms is elementary is tractable.

Disjunctive Programs

Review of Loop Formulas: Disjunctive Case

A *disjunctive rule* is an expression of the form

$$
\begin{aligned}
a_1; \ldots; a_k \;\leftarrow\; & a_{k+1}, \ldots, a_l, \mathit{not}\ a_{l+1}, \ldots, \mathit{not}\ a_m, \\
& \mathit{not\ not}\ a_{m+1}, \ldots, \mathit{not\ not}\ a_n
\end{aligned}
\tag{9}
$$

where $n \geq m \geq l \geq k \geq 0$ and $a_1, \ldots, a_n$ are propositional atoms. A *disjunctive program* is a finite set of disjunctive rules.

We will identify a disjunctive rule (9) with the propositional formula

$$
\begin{aligned}
&(a_{k+1} \wedge \cdots \wedge a_l \wedge \neg a_{l+1} \wedge \cdots \wedge \neg a_m \wedge \\
&\qquad \neg\neg a_{m+1} \wedge \cdots \wedge \neg\neg a_n) \rightarrow (a_1 \vee \cdots \vee a_k)\,,
\end{aligned}
$$

and will often write (9) as

$$
A \leftarrow B, F \tag{10}
$$

where A is $a_1, \ldots, a_k$, B is $a_{k+1}, \ldots, a_l$, and F is

$\mathit{not}\ a_{l+1}, \ldots, \mathit{not}\ a_m, \mathit{not\ not}\ a_{m+1}, \ldots, \mathit{not\ not}\ a_n$.

We will sometimes identify A and B with their corresponding sets.

For the definition of a stable model of a disjunctive program, we refer the reader to (Lee 2005, Section 2.2).

The definition of a dependency graph is extended to disjunctive programs in a straightforward way: The vertices of the graph are the atoms occurring in the program, and its edges go from the elements of A to the elements of B for all rules (10) of the program. The definition of a loop in terms of the dependency graph remains the same as in the case of nondisjunctive programs.

[5] Recall that the anti-chain property does not hold for elementary sets for Π: An elementary set may contain another elementary set as its proper subset.

[6] To be precise, CMODELS adds "conflict clauses."

For any set Y of atoms that occur in a disjunctive program Π, the *external support formula* of Y, denoted by $ES_\Pi(Y)$, is the disjunction of conjunctions

$$B \wedge F \wedge \bigwedge_{a \in A \setminus Y} \neg a$$

for all rules (10) of Π such that $A \cap Y \neq \emptyset$ and $B \cap Y = \emptyset$. When Π is a nondisjunctive program, this definition reduces to the definition of ES_Π for nondisjunctive programs given earlier.

The notion of LF_Π and the term *(conjunctive) loop formula* similarly apply to formulas (3) when Π is a disjunctive program. As shown in (Lee 2005), Theorem 1 remains correct after replacing "nondisjunctive program" in its statement with "disjunctive program."

Elementary Sets for Disjunctive Programs

In this section, we generalize the definition of an elementary set to disjunctive programs.

Note that a loop of a disjunctive program can be defined without referring to a dependency graph: Proposition 1 remains correct after replacing "nondisjunctive" in its statement with "disjunctive" and "$a_1 \in Z$" with "$A \cap Z \neq \emptyset$."

Let Π be a disjunctive program. For any set Y of atoms, we say that a subset Z of Y is *outbound* in Y for Π if there is a rule (10) in Π such that $A \cap Z \neq \emptyset$, $B \cap (Y \setminus Z) \neq \emptyset$, $A \cap (Y \setminus Z) = \emptyset$, and $B \cap Z = \emptyset$. Note that, for a nondisjunctive program Π, this definition reduces to the corresponding definition given before.

As with nondisjunctive programs, for any nonempty set Y of atoms occurring in Π, we say that Y is *elementary* for a disjunctive program Π if all nonempty proper subsets of Y are outbound in Y for Π. Similarly, every singleton set of atoms occurring in Π is an elementary set for Π, and every elementary set for Π is a loop of Π. The definition of an elementary set for a disjunctive program is stronger than the alternative definition of a loop provided in Proposition 1 for the disjunctive case: It requires that the rules satisfy two additional conditions, $A \cap (Y \setminus Z) = \emptyset$ and $B \cap Z = \emptyset$."

With these extended definitions, Propositions 2 and 3 remain correct after replacing "nondisjunctive program" in their statements with "disjunctive program." Theorem 1(d) holds even when Π is disjunctive.

To illustrate the definition, consider the following program: $p\ ;\ q \leftarrow p \qquad p \leftarrow q \qquad p \leftarrow not\ r$

Among the four loops of the program, $\{p\}$, $\{q\}$, $\{r\}$, and $\{p, q\}$, the last one is not an elementary set because $\{q\}$ is not outbound in $\{p, q\}$: the first rule contains q in the head and p in the body, but it also contains $\{p, q\} \cap (\{p, q\} \setminus \{q\}) = \{p\}$ in the head. According to the extension of Theorem 1(d) to disjunctive programs, the loop formula for $\{p, q\}$ can be disregarded.

Elementarily Unfounded Sets for Disjunctive Programs

Let Π be a disjunctive program. For any sets X, Y of atoms, by $\Pi_{X,Y}$ we denote the set of all rules (10) of Π such that $X \models B, F$ and $X \cap (A \setminus Y) = \emptyset$. Program $\Pi_{X,Y}$ contains all rules of Π that can provide supports for Y w.r.t. X. When Π is nondisjunctive and every atom in Y has a rule (2) in Π such that $a_1 \in Y$ and $X \models B, F$, set Y is elementary for $\Pi_{X,Y}$ iff it is elementary for Π_X.

We extend the definition of an elementarily unfounded set to disjunctive programs by replacing "Π_X" with "$\Pi_{X,Y}$" and by identifying Π as a disjunctive program. It is clear from the definition that every elementarily unfounded set for Π w.r.t. X is an elementary set for Π and that it is also an unfounded set for Π w.r.t. X.

Propositions 4, 5, Corollary 1, and Theorems 1(e), 1(e′) remain correct after replacing "nondisjunctive program" in their statements with "disjunctive program" and "Π_X" with "$\Pi_{X,Y}$." For preserving the intended meaning of Theorem 1(e), "Y is a maximal elementary set for Π_X" can be alternatively replaced with "Y is maximal among all sets Z of atoms that are elementary for $\Pi_{X,Z}$"

Deciding Elementary Sets: Disjunctive Case

Although deciding an elementary set can be done efficiently for nondisjunctive programs, it turns out that the corresponding problem in the case of (arbitrary) disjunctive programs is intractable.

Proposition 6 *For any disjunctive program Π and any set Y of atoms, deciding whether Y is elementary for Π is* coNP*-complete.*

This result can be explained by the close relationship to the problem of deciding whether a set of atoms is *unfounded-free* (Leone *et al.* 1997), which means that the set contains no nonempty unfounded subsets. In fact, the reduction from deciding unfounded-freeness to deciding elementariness is straightforward from Proposition 6.11 in (Leone *et al.* 1997).

However, for the class of disjunctive programs called head-cycle-free (Ben-Eliyahu and Dechter 1994), deciding whether a set is elementary is tractable. A disjunctive program Π is called *head-cycle-free* if, for every rule (10) in Π, there is no loop L of Π such that $|A \cap L| > 1$.

The definition of an elementary subgraph for a nondisjunctive program above can be extended to head-cycle-free programs by modifying the equation for EC^{i+1}_Π by replacing "(2)" with "(10)" and "$b \in B$" with "$a_1 \in A,\ b \in B$." With this extended definition of an elementary subgraph, Theorem 2 remains correct after replacing "nondisjunctive program" in its statement with "head-cycle-free program."

Comparison with the Gebser-Schaub Definition

In this section, we compare our reformulation of elementary loops with the original definition given in (Gebser and Schaub 2005) for nondisjunctive programs.

Let Π be a nondisjunctive program. A loop of Π is called *trivial* if it consists of a single atom such that the dependency graph of Π does not contain an edge from the atom to itself. Non-trivial loops were called simply loops in (Lin and Zhao 2004; Gebser and Schaub 2005). For a non-trivial loop L,

$$R^-_\Pi(L) = \{(2) \in \Pi \mid a_1 \in L,\ B \cap L = \emptyset\},$$

$$R^+_\Pi(L) = \{(2) \in \Pi \mid a_1 \in L,\ B \cap L \neq \emptyset\}.$$

Definition 1 *(Gebser and Schaub 2005, Definition 1) Given a nondisjunctive program Π and a non-trivial loop L of Π, L is called a GS-elementary loop for Π if, for each non-trivial loop L' such that $L' \subset L$, $R^-_\Pi(L') \cap R^+_\Pi(L) \neq \emptyset$.*[7]

[7] A GS-elementary loop was called an "elementary loop" in

Proposition 7 *For any nondisjunctive program* Π *and any non-trivial loop* L *of* Π, L *is a GS-elementary loop for* Π *iff* L *is an elementary set for* Π.

There are a few differences between Definition 1 and our definition of an elementary set. First, the definition of an elementary set does not assume a priori that the set is a loop. Rather, the fact that an elementary set is a loop is a consequence of our definition. Furthermore, our definition is simpler because it does not refer to a dependency graph.

Second, the two definitions do not agree on trivial loops: A trivial loop is an elementary set, but not a GS-elementary loop. This originates from the difference between the definition of a loop adopted in (Lin and Zhao 2004) and its reformulation given in (Lee 2005). As shown in the main theorem of (Lee 2005), identifying a trivial loop as a loop provides a simpler reformulation of the Lin-Zhao theorem by allowing us not to refer to completion. Furthermore, in the case of elementary sets, this reformulation also enables us to see a close relationship between maximal elementary sets (elementarily unfounded sets) and minimal nonempty unfounded sets. It also allows us to extend the notion of an elementary set to disjunctive programs without producing unintuitive results, unlike with GS-elementary loops. For instance, consider the following program:

$$p\ ;\ q \leftarrow r \qquad p\ ;\ r \leftarrow q \qquad q\ ;\ r \leftarrow p\ . \tag{11}$$

The non-trivial loops of this program are $\{p, q\}$, $\{p, r\}$, $\{q, r\}$, and $\{p, q, r\}$, but not singletons $\{p\}$, $\{q\}$, and $\{r\}$. If we were to extend GS-elementary loops to disjunctive programs, a reasonable extension would say that $\{p, q, r\}$ is a GS-elementary loop for program (11) because all its nontrivial proper subloops are "outbound" in $\{p, q, r\}$. Note that $\{p, q, r\}$ is unfounded w.r.t. $\{p, q, r\}$; moreover, every singleton is also unfounded w.r.t $\{p, q, r\}$, which is in contrast with our Proposition 4, according to which all nonempty proper subsets of an elementary set for program (11) w.r.t. $\{p, q, r\}$ are externally supported w.r.t. $\{p, q, r\}$. This anomaly does not happen with our definition of an elementary set: $\{p, q, r\}$ is not elementary for (11). More generally, an elementary set is potentially elementarily unfounded w.r.t. some model, which does not hold for GS-elementary loops extended to disjunctive programs.

Conclusion

We have proposed the notion of an elementary set and, based on it, provided a further refinement of the Lin-Zhao theorem, which simplifies the Gebser-Schaub theorem and extends it to disjunctive programs.

We have shown properties of elementary sets that allow us to disregard redundant loop formulas. One property is that, if all elementary subsets of a given set of atoms are externally supported, the set is externally supported as well. Another property is that, for a maximal set that is elementary for the relevant part of the program w.r.t. some interpretation, all its nonempty proper subsets are externally supported w.r.t. the same interpretation. Taking both properties together, elementary sets are the smallest class of atom sets whose external support must be considered in the characterization of answer sets. Related to this, we have proposed the concept of elementarily unfounded sets, which turn out to be precisely the minimal sets among nonempty unfounded sets.

Unlike elementary loops proposed by Gebser and Schaub, elementary sets and the related results are extended to disjunctive programs in a straightforward way. For nondisjunctive and head-cycle-free programs, we have provided a graph-theoretic characterization of elementary sets, which is simpler than the one by Gebser and Schaub. For arbitrary disjunctive programs, we have shown that deciding elementariness is coNP-complete, which can be explained by the close relationship to deciding unfounded-freeness of a given interpretation.

Elementary sets allow us to find more relevant unfounded sets than what loops allow. An apparent application is to consider elementarily unfounded sets in place of arbitrary unfounded loops as considered in the current SAT-based answer set solvers, at least for the tractable cases. For nondisjunctive programs, an efficient algorithm for computing elementarily unfounded sets is described in (Anger *et al.* 2006), which can be easily applied to head-cycle-free programs as well. Based on the theoretical foundations provided in this paper, we plan to integrate elementarily unfounded set computation in a SAT-based answer set solver for empirical evaluation.

(Gebser and Schaub 2005). Here we put "GS-" in the name, to distinguish it from a loop that is elementary under our definition.

References

Christian Anger, Martin Gebser, and Torsten Schaub. Approaching the core of unfounded sets. Submitted for publication, 2006.

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

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

Martin Gebser and Torsten Schaub. Loops: Relevant or redundant? In *Proc. LPNMR-05*, pages 53–65, 2005.

Enrico Giunchiglia, Yuliya Lierler, and Marco Maratea. SAT-based answer set programming. In *Proc. AAAI-04*, pages 61–66, 2004.

Joohyung Lee and Vladimir Lifschitz. Loop formulas for disjunctive logic programs. In *Proc. ICLP-03*, pages 451–465, 2003.

Joohyung Lee. A model-theoretic counterpart of loop formulas. In *Proc. IJCAI-05*, pages 503–508, 2005.

Nicola Leone, Pasquale Rullo, and Francesco Scarcello. Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation. *Information and Computation*, 135(2):69–112, 1997.

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

Domenico Saccá and Carlo Zaniolo. Stable models and non-determinism in logic programs with negation. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 205–217, 1990.

Allen Van Gelder, Kenneth Ross, and John Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, 38(3):620–650, 1991.