Simplifying A Logic Program Using Its Consequences

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 Ziwei Huo and Zhenfeng Yuan
School of Software
Sun Yat-sen University
Guangzhou 510006, China
wanhai@mail.sysu.edu.cn

Abstract
A consequence of a logic program is a consistent set of literals that are satisfied by every answer set. The well-founded model is a consequence that can be used to simplify the logic program. In this paper, we extend the notion of well-founded models to consequences for simplifying disjunctive logic programs (DLPs) in a general manner. Specifically, we provide two main notions, strong reliable set and weak reliable set, and show that a DLP is strongly equivalent to the simplified program if and only if the consequence is a strong reliable set, and they have the same answer sets if and only if the consequence is a weak reliable set. Then we provide computational complexity on identifying both notions. In addition, we provide an algorithm to compute some strong reliable sets and show that the approach is an extension of the well-founded model in simplifying logic programs.

1 Introduction

Answer Set Programming (ASP) has been considered as one of the most popular nonmonotonic rule-based formalisms, mainly due to the availability of efficient ASP solvers such as smodels [Syrjänen and Niemelä, 2001], ASSAT [Lin and Zhao, 2004], cmodels [Lierler and Maratea, 2004], clasp [Gebser et al., 2007a], claspD [Drescher et al., 2008], and DLV [Leone et al., 2002].

All of these modern ASP solvers require a preprocessing state for simplifying a logic program by its consequences in their grounding engines, like lparse [Syrjänen, 2000], gringo [Gebser et al., 2007b], and the grounding engine for DLV [Leone et al., 2002]. In specific, a consequence is a consistent set of literals that are satisfied by every answer set of the program. Many consequences can be derived using efficient inference rules before computing the answer sets of the program. For instance, a consequence can be computed by applying unit propagation to the set of clauses obtained from Clark’s completion [Clark, 1978] of the program, applying the well-founded operator [Leone et al., 1997] to the program, using the loop formulas of loops with at most one external support rule [Chen et al., 2013], and applying the lookahead operator based on previous operations.

These consequences can help computing the answer sets of the program. The notion of consequences is also extended in [Eiter et al., 2004] to simplify logic programs under uniform and strong equivalence. However, not all consequences can be used to simplify the logic program, as simplified programs may have answer sets that are not answer sets of the original program. The best known consequences for simplifying programs are the well-founded models for normal logic programs [Van Gelder et al., 1991] and the results computed by the well-founded operator for disjunctive logic programs [Leone et al., 1997], the approximations of which have been used in grounding engines of most ASP solvers.

To extend the notion of well-founded models, we consider when a consequence can be used to simplify a logic program. In this paper, we provide two notions, called strong reliable set and weak reliable set, so that

- a logic program is strongly equivalent to the simplified program iff the consequence is a strong reliable set,
- they have the same answer sets iff the consequence is a weak reliable set.

We also explore computational complexity on identifying both sets. These notions provide a guideline to explore classes of consequences that could be used to simplify a logic program. As an example, we provide an algorithm to compute some strong reliable sets and show that the approach is an extension of the well-founded model in simplifying logic programs.

2 Preliminaries

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

$$a_1 \lor \cdots \lor a_k \leftarrow a_{k+1}, \ldots, a_m, \text{not } a_{m+1}, \ldots, \text{not } a_n,$$

where \(n \geq m \geq k \geq 0, n \geq 1\) and \(a_1, \ldots, a_n\) are atoms. If \(k \leq 1\), it is a normal rule; if \(m = n\), it is a positive rule; if \(n = m = k = 1\), it is a fact. In particular, a normal logic program (NLP) is a finite set of normal rules and a positive logic program is a finite set of positive rules.

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

$$\text{head}(r) \leftarrow \text{body}(r),$$

\(*\)Corresponding author.
where \( \text{head}(r) \) is \( a_1 \lor \cdots \lor a_k \), \( \text{body}(r) = \text{body}^+(r) \land \text{body}^-(r) \), \( \text{body}^+(r) \) is \( a_{k+1} \land \cdots \land a_m \), and \( \text{body}^-(r) \) is \( \neg a_{m+1} \land \cdots \land \neg a_n \), and we identify \( \text{head}(r) \), \( \text{body}^+(r) \), \( \text{body}^-(r) \) with their corresponding sets of atoms.

A set \( S \) of atoms satisfies a rule \( r \), if \( \text{body}^+(r) \subseteq S \) and \( \text{body}^-(r) \cap S = \emptyset \) implies \( \text{head}(r) \cap S \neq \emptyset \). \( S \) satisfies a program \( P \), if \( S \) satisfies every rules in \( P \). Let \( L \) be a set of literals and \( F \) a propositional formula, we write \( L \models F \) if \( L \) entails \( F \) in the sense of classical logic, \( L = \{ p \mid p \in L \} \lor \{ p \mid \neg p \in L \} \), \( L^+ = \{ p \mid p \in L \} \), \( L^- = \{ p \mid \neg p \in L \} \).

In the following we recall the basic notions about answer sets [Gelfond and Lifschitz, 1991], SE-models [Turner, 2003], external support rules and loop formulas [Lee and Lifschitz, 2003], the well-founded model [Van Gelder et al., 1991], and the well-founded operator [Leone et al., 1997].

### 2.1 Answer Set Semantics

Given a DLP \( P \) and a set \( S \) of atoms, the Gelfond-Lifschitz reduct of \( P \) on \( S \), written \( PS^S \), is obtained from \( P \) by deleting:

1. each rule that has a formula \( \text{not } p \) in its body with \( p \in S \),
2. all formulas of the form \( \text{not } p \) in the bodies of the remaining rules.

A set \( S \) of atoms is an answer set of \( P \) if \( S \) is a minimal set satisfying \( PS^S \).

A SE-interpretation is a pair \((X,Y)\) where \( X \) and \( Y \) are sets of atoms and \( X \subseteq Y \). A SE-interpretation \((X,Y)\) is a SE-model of a DLP \( P \) if \( X \) satisfies \( P \) and \( Y \) satisfies \( P^Y \). A SE-model \((Y,Y)\) of \( P \) is an equilibrium model of \( P \), if there does not exist another set \( X \) such that \( X \subseteq Y \) and \( (X,Y) \) is a SE-model of \( P \).

Two DLPs \( P_1 \) and \( P_2 \) are strongly equivalent, if for any DLP \( P' \), programs \( P_1 \cup P' \) and \( P_2 \cup P' \) have the same set of answer sets.

The SE-models and the answer sets of a DLP have the following proposition [Ferraris, 2005; Lifschitz et al., 2001].

**Proposition 1** Let \( P_1, P_2 \) be two DLPs and \( S \) a set of atoms.

- \( S \) is an answer set of \( P_1 \) iff \( (S,S) \) is an equilibrium model of \( P_1 \).
- \( P_1 \) is strongly equivalent to \( P_2 \) iff \( P_1 \) and \( P_2 \) have the same SE-models.

### 2.2 External Support Rules and Loop Formulas

Given a nonempty set \( E \) of atoms, a rule \( r \) is an external support of \( E \) if \( \text{head}(r) \cap E \neq \emptyset \) and \( E \cap \text{body}^+(r) = \emptyset \). Given a DLP \( P \), we use \( R^-(E,P) \) to denote the set of external support rules of \( E \) in \( P \).

The (conjunctive) loop formula of \( E \) under \( P \), written \( LF(E,P) \), is the following implication

\[
\bigwedge_{p \in E} \bigvee_{r \in R^-(E,P)} \left( \text{body}(r) \land \bigwedge_{q \in \text{head}(r) \setminus E} \neg q \right).
\]

A set \( S \) of atoms satisfies \( LF(E,P) \), if \( E \subseteq S \) implies that there exists a rule \( r \in R^-(E,P) \) such that \( \text{body}^+(r) \subseteq S \), \( \text{body}^-(r) \cap S = \emptyset \), and \( \text{head}(r) \setminus E \cap S = \emptyset \), i.e., \( S \cup \{-p \mid p \notin S \} \models LF(E,P) \).

### Proposition 2 (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 to each other:

- \( S \) is an answer set of \( P \);
- \( S \) satisfies \( LF(E,P) \) for every nonempty set \( E \) of atoms.

### 2.3 Well-Founded Semantics

Let \( P \) be a DLP and \( I \) a set of literals, a set \( X \) of atoms is an unfounded set of \( P \) w.r.t. \( I \) if for each atom \( p \in X \) and each rule \( r \in P \) such that \( p \in \text{head}(r) \), at least one of the following conditions holds:

- \( \text{body}(r) \cap I \neq \emptyset \)
- \( X \cap \text{body}^+(r) \neq \emptyset \), or
- \( (\text{head}(r) \setminus X) \cap I \neq \emptyset \).

Let \( I \) be a set of literals. If \( P \) is an NLP, the union of two unfounded sets is also an unfounded set. \( I \) is unfounded-free for a DLP \( P \) if \( I \cap X = \emptyset \) for each unfounded set \( X \) of \( P \) w.r.t. \( I \). If \( I \) is unfounded-free, then the union of two unfounded sets of \( P \) w.r.t. \( I \) is also an unfounded set, thus there exists the greatest unfounded set of \( P \) w.r.t. \( I \). We use \( U_P(I) \) to denote such greatest unfounded set, if exists.

We define two operators for a DLP \( P \) and a set \( I \) of literals:

\[
T_P(I) = \{ p \mid \text{there exists a rule } r \in P \text{ s.t. } p \in \text{head}(r) \text{ and } I \models \text{body}(r) \land \bigwedge_{q \in \text{head}(r) \setminus \{p\}} \neg q \};
\]

\[
W_P(I) = T_P(I) \cup U_P(I).
\]

\( T_P, U_P, W_P \) are monotonic operators. We use \( \text{WFM}(P) \) to denote the least fixed point of the operator \( W_P \). If \( P \) is an NLP, \( \text{WFM}(P) \) is the well-founded model of \( P \) as defined in [Van Gelder et al., 1991].

### 3 Simplifying A Logic Program

A consequence of a program is a consistent set of literals that are satisfied by every answer set of the program. A consequence is positive if it is a set of atoms. Some consequences can simplify the logic program in the manner that, the resulting program consists of two parts, a part without any atoms appearing in the consequence and the facts of the consequence. \( \text{WFM}(P) \) is such a consequence that is commonly used to simplify the program in the grounding engines, like lparses [Syrjänen, 2000], gingo [Gebser et al., 2007b], and the grounding engine for DLV [Leone et al., 2002], of modern ASP solvers. In this section, we consider how to extend the idea to other consequences of the program.

Let \( L \) be a consequence of a DLP \( P \), we define \( t_{\text{tr}}(P,L) \) to be the program obtained from \( P \) by

1. deleting each rule \( r \) that has an atom \( p \in \text{body}^+(r) \) with \( \neg p \in L \), and
2. replacing each rule \( r \) that has an atom \( p \in \text{head}(r) \) or \( p \in \text{body}^-(r) \) with \( \neg p \in L \) by a rule \( r' \) such that \( \text{head}(r') = \text{head}(r) \setminus L^- \), \( \text{body}^+(r') = \text{body}^+(r) \), and \( \text{body}^-(r') = \text{body}^-(r) \setminus L^- \).

We define \( t_{\text{tr}}(P,L) \) to be the program obtained from \( P \) by

1. deleting each rule \( r \) that has an atom \( p \in \text{head}(r) \) or \( p \in \text{body}^-(r) \) with \( p \in L \), and
2. replacing each rule \( r \) that has an atom \( p \in \text{body}^+(r) \) with \( L \) by a rule \( r' \) such that \( \text{head}(r') = \text{head}(r) \), \( \text{body}^+(r') = \text{body}^+(r) \setminus L^+ \), and \( \text{body}^-(r') = \text{body}^-(r) \).

Note that \( \text{tr}_n(P, L) \) (resp. \( \text{tr}_p(P, L) \)) does not contain any atoms in \( L^- \) (resp. \( L^+ \)) and \( \text{tr}_p(\text{tr}_n(P, L), L) \) does not contain any atoms occurring in \( L \).

The following property explains why \( WFM(P) \) can be used to simplify the program \( P \).

**Proposition 3** Let \( P \) be a DLP and \( L = WFM(P) \).

(i) \( P \) and \( \text{tr}_p(\text{tr}_n(P, L), L) \cup \{ p \leftarrow | p \in L \} \) have the same set of answer sets.

(ii) \( P \cup \{ p \leftarrow | \neg p \in L \} \) is strongly equivalent to \( \text{tr}_p(\text{tr}_n(P, L), L) \cup \{ p \leftarrow | p \in L \} \cup \{ \neg p | \neg p \in L \} \).

Both conditions in Proposition 3 would no longer be true, if \( L \) is larger than \( WFM(P) \).

**Example 1** Consider the logic program \( P_0 \):

\[
a \leftarrow b,
\quad b \leftarrow \neg c,
\quad c \leftarrow \neg b,
\quad \neg \neg a.
\]

\( WFM(P_0) = \emptyset \), \( L = \{ a \} \) is a consequence of \( P_0 \), and \( \text{tr}_p(\text{tr}_n(P_0, L), L) = \{ b \leftarrow \neg c, c \leftarrow \neg b \} \). Notice that, \( \{ a, c \} \) is an answer set of \( \text{tr}_p(\text{tr}_n(P_0, L), L) \cup \{ p \leftarrow | p \in L \} \), but not an answer set of \( \text{tr}_p(P, L) \).

**Proposition 4** Let \( L \) be a consequence of a DLP \( P \).

- \( P \cup \{ p \leftarrow | \neg p \in L \} \) have the same set of answer sets.
- \( P \cup \{ \neg p \leftarrow | \neg p \in L \} \) is strongly equivalent to \( \text{tr}_p(\text{tr}_n(P, L), L) \cup \{ p \leftarrow | \neg p \in L \} \).
- An answer set of \( P \) is always an answer set of \( \text{tr}_p(\text{tr}_n(P, L), L) \cup \{ p \leftarrow | p \in L \} \), but not vice versa in general.

**Proof Sketch:** For any nonempty set \( E \) of atoms, under \( L \), the loop formula \( LF(E, P) \) is equivalent to \( LF(E, \text{tr}_n(P, L)) \), and \( LF(E, P) \) implies \( LF(E, \text{tr}_p(P, L)) \cup \{ p \leftarrow | p \in L \} \).

Given a consequence \( L \) of a DLP \( P \), we can simplify the program by \( \text{tr}_n(P, L) \) which would no longer contain atoms appearing in \( L^- \). However, we cannot use \( \text{tr}_p(P, L) \) to simplify \( P \) in general.

**Example 2** Consider the logic program \( P_1 \):

\[
a \leftarrow b,
\quad b \leftarrow c,
\quad c \leftarrow d,
\quad a \leftarrow f,
\quad d \leftarrow \neg e,
\quad e \leftarrow \neg d,
\quad \neg a,
\quad f \leftarrow a.
\]

\( L = \{ a, f \} \) is a consequence of \( P_1 \), and \( \text{tr}_p(P_1, L) \) is:\n
\[c \leftarrow,\quad b \leftarrow c,\quad c \leftarrow d,\quad d \leftarrow \neg e,\quad e \leftarrow \neg d.\]

The only answer set of \( P_1 \) is \( \{ a, b, c, d, f \} \). However, \( \text{tr}_p(P_1, L) \) has two answer sets: \( \{ b, c, d \} \) and \( \{ b, c, e \} \).

In the next section, we consider when the conditions in Proposition 3 would be true for a positive consequence \( L \).

4 Strong and Weak Reliable Sets

In this section, we introduce notions of the strong reliable set and the weak reliable set, so that they specify sufficient and necessary conditions when a positive consequence \( L \) satisfies condition (ii) and (i) in Proposition 3 respectively. We also explore computational complexity on identifying a strong or a weak reliable set.

Given a DLP \( P \), a set \( U \) of atoms is a strong reliable set of \( P \), if for every nonempty subset \( E \) of \( U \) and every SE-model \( (X, Y) \) of \( P \), there exists a rule \( r \in R^-(E, P) \) such that \( \text{head}(r) \cap X \subseteq E \), \( \text{body}^+(r) \subseteq X \cap U \), and \( \text{body}^-(r) \cap (Y \cup U) = \emptyset \).

**Proposition 5** If \( U \) is a strong reliable set of a DLP \( P \), then for every SE-model \((X, Y)\) of \( P \), \( U \subseteq X \), and \( U \) is a consequence of \( P \).

**Proof Sketch:** From the definition, for every SE-model \((X, Y)\) of \( P \) and every nonempty subset \( E \) of \( U \), \( U \setminus E \) implies \( E \cap X \neq \emptyset \), \( P \) is finite, then \( U \subseteq X \).

**Theorem 1** Let \( P \) be a DLP and \( U \) a set of atoms. \( P \) is strongly equivalent to \( \text{tr}_p(P, U) \cup \{ p \leftarrow | p \in U \} \) if and only if \( U \) is a strong reliable set of \( P \).

**Proof Sketch:** \( \iff \): From Proposition 5, for every SE-model \((X, Y)\) of \( P \), \( U \subseteq X \). Then both programs have the same set of SE-models.

\( \Rightarrow \): Assume that \( U \) is not a strong reliable set of \( P \), then there exists a nonempty subset \( E \) of \( U \) and a SE-model \((X', Y)\) of \( P \) that prevent \( U \) to be a strong reliable set. From the definition, \( X' \setminus E \) satisfies \( P_Y' \), so \( (X' \setminus E, Y) \) is also a SE-model of \( P \), which conflicts to the fact that \((X, Y)\) is not a SE-model of \( tr_p(P, U) \cup \{ p \leftarrow | p \in U \} \).

So given a positive consequence \( L \) of a DLP \( P \), the condition (ii) in Proposition 3 is true if and only if \( L \) is a strong reliable set of \( P \).

Before defining the notion of the weak reliable set, we introduce a notation. Given a DLP \( P \) and a set \( U \) of atoms, a SE-model \((X, Y)\) of \( P \) is called a \( U \)-equilibrium model of \( P \), if \( U \subseteq Y \), \( Y \setminus U = X \setminus U \) and there does not exist another set \( X' \) such that \( X' \setminus U \subseteq X \setminus U \) and \((X' \setminus U, Y)\) is a SE-model of \( P \). We show that, a \( U \)-equilibrium model of \( P \) is related to an answer set of \( \text{tr}_p(P, U) \).

**Lemma 1** Let \( P \) be a DLP and \( U \) a set of atoms. A SE-interpretation \((X, Y)\) is a \( \text{tr}_p(P, U) \) if and only if \( U \) for every SE-model \((X', Y)\) of \( P \) such that \( X' \setminus U = Y \setminus U \).

**Proposition 6** Let \( P \) be a DLP and \( U \) a set of atoms. \( S \) is an answer set of \( \text{tr}_p(P, U) \) if and only if \( \text{tr}_p(S, U) \) is a SE-model of \( P \) and there does not exist a set \( S' \) such
that \( S' \subseteq S \) and \((S', S)\) is a SE-model of \( tr_p(U) \). From Lemma 1, \((S' \cup U, S \cup U)\) is a SE-model of \( P \). Then \((S \cup U, S \cup U)\) is a \( U \)-equilibrium model of \( P \).

Given a DLP \( P \), a set \( U \) of atoms is a weak reliable set of \( P \) if for every nonempty subset \( E \) of \( U \) and every \( U \)-equilibrium model \((X, Y)\) of \( P \), there exists a rule \( r \in R^+(E, P) \) such that \( \text{head}(r) \cap X \subseteq E \), \( \text{body}^+(r) \subseteq X \cup U \), and \( \text{body}^-(r) \cap (Y \cup U) = \emptyset \).

Similar to the proof of Proposition 5, we have the following proposition.

**Proposition 7** If \( U \) is a weak reliable set of a DLP \( P \), then for every \( U \)-equilibrium model \((X, Y)\) of \( P \), \( U \subseteq X \), and \( U \) is a consequence of \( P \).

**Theorem 2** Let \( P \) be a DLP and \( U \) a set of atoms. \( P \) and \( tr_p(P, U) \cup \{ p \leftarrow p \in U \} \) have the same set of answer sets if and only if \( U \) is a weak reliable set of \( P \).

**Proof Sketch:** From Proposition 7, for every \( U \)-equilibrium model \((X, Y)\) of \( P \), \( U \subseteq X \), and \( X = Y \). From Proposition 6, for every answer set \( X \) of \( S \cup U \), there exists a \( U \)-equilibrium model \((S \cup U, S \cup U)\) of \( P \), \( X \cup U \) is an answer set of \( P \). So \( P \) and \( tr_p(P, U) \cup \{ p \leftarrow p \in U \} \) have the same set of answer sets.

So given a positive consequence \( L \) of a DLP \( P \), the condition (i) in Proposition 3 is true if and only if \( L \) is a weak reliable set of \( P \).

**Proposition 8** Let \( P \) be a DLP and \( U \) a set of atoms. If \( U \) is a strong reliable set of \( P \), then \( U \) is a weak reliable set of \( P \).

**Proposition 9** If \( U_1 \) and \( U_2 \) are strong (resp. weak) reliable sets of a DLP \( P \), then \( U_1 \cup U_2 \) is also a strong (resp. weak) reliable set of \( P \).

Given a DLP \( P \), there exists the greatest strong (resp. weak) reliable set of \( P \), denoted by \( GSRS(P) \) (resp. \( GWRS(P) \)), i.e., the union of all possible strong (resp. weak) reliable sets.

**Proposition 10** Let \( P \) be a DLP and \( U \) a set of atoms.

- Deciding whether \( U \) is a strong reliable set of \( P \) is coNP-complete.
- Deciding whether \( U \) is a weak reliable set of \( P \) is coNP-hard.
- Deciding whether \( U \) is equivalent to \( GSRS(P) \) (resp. \( GWRS(P) \)) is coNP-hard.
- Deciding whether an atom \( p \) is in \( GSRS(P) \) (resp. \( GWRS(P) \)) is coNP-hard.

**Proof Sketch:** The first item is a coNP problem, as we can guess a nonempty subset \( E \) of \( U \) and a SE-model \((X, Y)\) of \( P \) which prevents \( U \) to be a strong reliable set.

The hardness is proved by converting the UNSAT problem to these problems. Let \( t \) and \( e \) be new atoms not appearing in a set \( C \) of clauses, and \( Atoms(C) \) the set of atoms in \( C \). We can construct a DLP \( P \) from \( C \) by:

- adding rules \( t \lor e \lor \bigvee_{p \in Atoms(C)} P \leftarrow t \sim e \),
- for each clause \( C \in C \), adding the rule \( e \lor \bigvee_{\sim q \in C} P \leftarrow \bigwedge_{q \in C} \neg q \), and
- for each atom \( p \in Atoms(C) \), adding rules \( t \sim p \) and \( p \sim e \).

It can be verified that \( U = Atoms(C) \cup \{ t, e \} \) is a strong reliable set of \( P \) if and only if \( C \) is not satisfiable. Moreover, \( U \) is a strong reliable set of \( P \) iff it is a weak reliable set of \( P \) iff \( U = GSRS(P) = GWRS(P) \) iff \( e \in GSRS(P) \).

Even when \( P \) is an NLP, it is still hard to recognize a strong or a weak reliable set.

**Proposition 11** Let \( P \) be an NLP and \( U \) a set of atoms.

- Deciding whether \( U \) is a strong (resp. weak) reliable set of \( P \) is coNP-complete.
- Deciding whether \( U \) is equivalent to \( GSRS(P) \) (resp. \( GWRS(P) \)) is coNP-hard.
- Deciding whether an atom \( p \) is in \( GSRS(P) \) (resp. \( GWRS(P) \)) is coNP-hard.

**Proof Sketch:** As \( P \) is an NLP, whether a SE-model \((X, Y)\) is a \( U \)-equilibrium model can be checked in polynomial time. So deciding whether \( U \) is a weak reliable set of an NLP \( P \) is a coNP-problem. The hardness is proved by converting the UNSAT problem to these problems.

Let \( e, t, \) and \( t' \) be new atoms not appearing in a set \( C \) of clauses, and \( Atoms(C) \) the set of atoms in \( C \). We can construct an NLP \( P \) from \( C \) by:

- adding rules \( e \sim t, t \sim \sim t', \sim t' \sim \sim t, \) and \( \sim t', \sim e \),
- for each atom \( p \in Atoms(C) \), adding the rule \( p \sim e \), and
- for each clause \( C \in C \), adding the rule \( e \sim \bigwedge_{p \in C} P \land \bigwedge_{q \in C} \neg q \).

It can be verified that \( U = Atoms(C) \cup \{ e, t \} \) is a strong reliable set of \( P \) if and only if \( C \) is not satisfiable. Moreover, \( U \) is a strong reliable set of \( P \) iff \( U = GSRS(P) = GWRS(P) \) iff \( e \in GSRS(P) \).

Now we provide an alternative definition of strong reliable sets, which implies an approach to recognize a strong reliable set of a DLP.

**Proposition 12** Let \( P \) be a DLP and \( U \) a set of atoms. \( U \) is a strong reliable set of \( P \) if and only if \( U \subseteq X \) for every SE-model \((X, Y)\) of \( P \).

**Proof Sketch:** From the definition, if \( U \subseteq X \) for every SE-model \((X, Y)\) of \( P \), then \( U \) is a strong reliable set of \( P \). Proposition 5 proves the other direction.

From [Pearce et al., 2001], a DLP \( P \) can be translated to a propositional formula \( \pi(P) \), such that \((X, Y)\) is a SE-model of \( P \) iff \( X \cup Y \) is a model of \( \pi(P) \), where \( Y \) is a set of new atoms for each atom in \( Y \). Based on this result, the following proposition provides an approach to identify a strong reliable set of a DLP.

**Proposition 13** Let \( P \) be a DLP and \( U \) a set of atoms. \( U \) is a strong reliable set of \( P \) if and only if \( \pi(P) \models \bigwedge \sim U \) in propositional logic.
Though, it is difficult to verify whether a positive consequence $U$ is a strong or a weak reliable set of a program $P$, there are some easy cases as follows.

**Proposition 14** If $U$ is a positive consequence of a positive logic program $P$, then $U$ is a strong reliable set of $P$.

From Proposition 13, given a DLP $P$, we can compute a consequence $L$ by using efficient inference rules, like unit propagation, on the CNF form of $\pi(P)$. Then $L^+$ is a strong reliable set of $P$ and $L$ can be used to simplify the program $P$.

The notions of the strong and the weak reliable sets also provide a guideline to explore classes of positive consequence that could be used to simplify the program. We provide such an example in the next section.

## 5 Reliable Sets Under a Consequence

Guided by the definition of strong reliable sets, we introduce a sufficient condition for a consequence to simplify a logic program. We also provide an algorithm to identify the class of consequences specified by this sufficient condition. We show that for some programs, the consequence in the class is larger than the well-founded model.

Given a DLP $P$ and a consistent set $L$ of literals, a set $U$ of atoms is a reliable set of $P$ under $L$ if for every nonempty subset $E$ of $U$, there exists a rule $r \in R_+(E, P)$ such that $\text{head}(r) \subseteq E$ and $U \cup (L \setminus L^+) \models \text{body}(r)$.

Let $L$ be a consequence of a DLP $P$. A reliable set of $P$ under $L$ is also a strong (resp. weak) reliable set of a DLP constructed from $P$ and $L$.

**Proposition 15** Let $P$ be a DLP, $U$ a set of atoms, and $L$ a consequence of $P$. If $U$ is a reliable set of $P$ under $L$, then $U$ is a strong and a weak reliable set of the program:

$$P \cup \{\leftarrow p \mid p \in E \} \cup \{\leftarrow \neg p \mid \neg p \in L\}.$$

**Proof Sketch:** $U$ is a reliable set of $P$ under $L$. From the definition, for any SE-model $(X, Y)$ of $P$ with $L^+ \subseteq Y$ and $L^+ \cap Y = \emptyset$ and any nonempty subset $E$ of $U$, if $U \setminus E \subseteq X$, then $E \cap X \neq \emptyset$. There is a finite number of atoms, then $U \subseteq X$. So there exists a rule $r \in R_+(E, P)$ such that $\text{head}(r) \cap X \subseteq E$, $\text{body}(r) \subseteq X \cup U$ and $\text{body}(r) \cap (Y \cup U) = \emptyset$. $U$ is a strong and a weak reliable set of the new program. ■

From Proposition 15, Theorem 1 and 2, the notion can be used to simplify a logic program.

**Corollary 3** Let $P$ be a DLP, $U$ a set of atoms, and $L$ a consequence of $P$. If $U$ is a reliable set of $P$ under $L$, then

(i) $P$ and $tr_p(U) \cup \{p \leftarrow p \in U\}$ have the same set of answer sets.

(ii) $P \cup \{\neg p \mid p \in L\} \cup \{\leftarrow p \mid \neg p \in L\}$ is strongly equivalent to

$$tr_p(P, U) \cup \{p \leftarrow p \in U\} \cup \{\neg p \mid p \in L\} \cup \{\leftarrow p \mid \neg p \in L\}.$$

The strongly equivalent relation mentioned in the previous corollary can be further specified in the following theorem.

**Theorem 4** Let $P$ be a DLP, $U$ a set of atoms, and $L$ a consistent set of literals.

$$\{r \mid r \in P, \text{head}(r) \setminus L^+ \subseteq U \cup L^+ \} \cup \{p \mid \neg p \in L\} \cup \{\neg p \mid p \in L\}$$

is strongly equivalent to

$$\{p \mid p \in U\} \cup \{\leftarrow p \mid \neg p \in L\} \cup \{\neg p \mid p \in L\}$$

if and only if $U$ is a reliable set of $P$ under $L$.

**Proof Sketch:** Let $P_1, P_2$ stand for these programs respectively.

$U$: $U$ is a reliable set of $P$ under $L$ implies $U$ is a strong reliable set of $P_1$ under $L$. The direction can be proved from Corollary 3.

$\Rightarrow$: Assume that $U$ is not a reliable set of $P$ under $L$, then there exists a nonempty subset $E$ of $U$ which prevents $U$ to be a strong reliable set of $P$ under $L$. It can be verified that $(U \cup L^+) \setminus E$ satisfies $P_1^{U \cup L^+}$, so $((U \cup L^+) \setminus E, U \cup L^+)$ is a SE-model of $P_1$, which conflicts to the fact it is not a SE-model of $P_2$. ■

**Proposition 16** If $U_1$ and $U_2$ are reliable sets of a DLP $P$ under a consistent set $L$ of literals, then $U_1 \cup U_2$ is also a reliable set of $P$ under $L$.

Given a DLP $P$ and a consistent set $L$ of literals, there exists the greatest reliable set of $P$ under $L$. We denote it by $\text{GRS}(P, L)$, i.e., the union of all possible reliable sets of $P$ under $L$.

**Example 3** (Example 2 continued) Recall that $L = \{a, f\}$ is a consequence of $P_1$. From the definition, $\text{GRS}(P_1, L) = \emptyset$. $L' = \{\neg e\}$ is another consequence of $P_1$, and $\text{GRS}(P_1, L') = \{a, b, c, d, f\}$.

**Proposition 17** Let $P$ be a DLP, $U$ a set of atoms, and $L$ a consequence of $P$.

- Deciding whether $U$ is a reliable set of $P$ under $L$ is coNP-complete.
- Deciding whether $U$ is equivalent to $\text{GRS}(P, L)$ is coNP-hard.
- Deciding whether an atom $p$ is in $\text{GRS}(P, L)$ is coNP-hard.

**Proof Sketch:** The first item is a coNP problem, as we can guess a corresponding set $E$ which prevents $U$ to be a reliable set of $P$ under $L$.

In the proof of Proposition 10, the set $U = \text{Atoms}(C) \cup \{t, e\}$ is a strong reliable set of $P$ iff $U$ is a reliable set of $P$ under $\emptyset$ iff $U = \text{GRS}(P, \emptyset)$ iff $\emptyset \in \text{GRS}(P, \emptyset)$. ■

When $P$ is an NLP, $\text{GRS}(P, L)$ can be computed efficiently. In the following, we provide such a polynomial time algorithm, which can also be used to compute a subset of $\text{GRS}(P, L)$ for a DLP $P$.

Let $P$ be a DLP, $X$ a set of atoms and $L$ a set of literals. $T_{P,L}(X) = \{p \mid p \in \text{head}(r), (L \setminus L^+) \cup X \models \text{body}(r) \land \bigwedge_{q \in \text{head}(r) \setminus \{p\}} \neg q\}$. $T_{P,L}$ is a monotonic operator. We use $T(P, L)$ to denote the least fixed point of $T_{P,L}$.
Proposition 18  Let P be an NLP and L a consistent set of literals. $GRS(P, L) = T(P, L)$.

Proof Sketch: (1) If U is a reliable set of P under L, then $T_{P, L}(U)$ is also a reliable set of P under L. So $T(P, L) \subseteq GRS(P, L)$.

(2) If there is an atom $p \in U$, U is a reliable set of P under L, and $p \notin T(P, L)$, then $U \setminus \{p\} \not\subseteq T(P, L)$. So $GRS(P, L) \not\subseteq T(P, L)$. ■

Note that, $T(P, L)$ can be computed in polynomial time, then $GRS(P, L)$ can be computed efficiently for NLPs.

Proposition 19  Let P be a DLP and L a consistent set of literals. $T(P, L) \subseteq GRS(P, L)$.

We show that $GRS(P, L)$ also extends the well-founded operator.

Proposition 20  Let P be a DLP, $L_1$ and $L_2$ be consistent sets of literals with $L_1 \subseteq L_2$. The following hold:

- $WFM(P)^+ \subseteq GRS(P, ¬WFM(P)^-)$. 
- $GRS(P, L_1) \subseteq GRS(P, L_2)$.

Example 4  Consider the NLP $P_2$:

\[
a \leftarrow \neg b, \quad b \leftarrow \neg a, \quad \leftarrow a.
\]

$WFM(P_2) = \emptyset$, $L = \{-a, b\}$ is a consequence of $P_2$, and $T(P_2, L) = GRS(P_2, L) = \{\emptyset\}$.

Example 5  Consider the DLP $P_3$:

\[
a \lor b \leftarrow c, \quad a \leftarrow b, \quad b \leftarrow a.
\]

$GRS(P_3, \emptyset) = \{a, b\}$, $T(P_3, \emptyset) = \emptyset$, and $WFM(P_3) = \emptyset$.

We also provide an algorithm for computing $GRS(P, L)$, when P is a DLP. Given a DLP P, a set X of atoms, and a set L of literals, the operator $RS_{P, L}(X)$ is defined in Algorithm 1. Note that, $RS_{P, L}(X)$ is monotonic. We use $RS(P, L)$ to denote the least fixed points of the corresponding operators.

Algorithm 1: $RS_{P, L}(X)$

1. $A := X$;
2. $H := \{head(r) \setminus L^+ \mid r \in P, head(r) \cap body^+(r) = \emptyset\}$;
3. $X \cap head(r) = \emptyset$, and $X \cup (L \setminus L^+) \models body(r)$;
4. for each $C \in H$ do
5. \quad if $|C| = 1$ or
6. \quad for each $p \in C, C \subseteq RS_{P, L}(X \cup \{p\})$ then
7. \quad $A := A \cup C$;
8. return A

Proposition 21  Let P be a DLP and L a consistent set of literals. $GRS(P, L) = RS(P, L)$.

Proof Sketch: The proof is similar to Proposition 18. Additionally with the fact that, if $C \not\subseteq RS_{P, L}(X \cup \{p\})$ for some $p \in C$, then for the set $E = C \setminus \{p\}$, there does not exist a corresponding rule $r \in R^-(E, P)$, which prevents $C \cup X$ to be a reliable set of P under L. ■

Now we compare the reliable set under a consequence of a program with the results computed from the approximation of the well-founded operator. Given a logic program P, we can preprocess P by the grounding engine gringo [Gebser et al., 2007b] (or lparse) and receive the simplified program $P'$. In specific, $P'$ is consisted from two parts $P^*$ and $R$, where no atoms in $WFM^+(P)$ occurring in $P^*$, $R = \{p \leftarrow p \in WFM^+(P)\}$, and $WFM^+(P)$ is an approximation of $WFM(P)$ by gringo. We can compute a consequence of $P^*$ by applying unit propagation to the set of clauses obtained from the rules in $P^*$. For example, let $P^* = \{p \lor q \leftarrow q \leftarrow \neg r, \neg r\}$, the corresponding set of clauses is $\{p \lor q, q \lor r, \neg r\}$, and the consequence computed by using unit propagation is $\{\neg r, q\}$. Notice that both gringo and lparse do not compute the full well-founded model, unit propagation would produce useless consequences for programs that have no answer sets. Then we also characterize the benchmark by the number of programs that have no answer sets. Based on such consequence $L_i$, we can compute $T(P^*, L_i)$. From Proposition 18 and 19, it is equal to $GRS(P^*, L)$ if $P^*$ is an NLP; it is a subset of $GRS(P^*, L)$ if $P^*$ is a DLP. In the following, for the program $P'$ grounded by gringo, we use $GRS^*$ to denote such $T(P^*, L_i)$.

We have implemented a program to compute $GRS^*$ for programs grounded by gringo (version 4.4.0). Table 1 contains average sizes of consequents and $GRS^*$ of $P^*$ for different instances from 3 classes of NLPs and 2 classes of DLPs, and average times for computing these notions.

These benchmarks were frequently used to compare the performance of ASP solvers [Denecker et al., 2009; Gebser et al., 2013]. If a class’s name contains “(N)” (resp. “(D)”) then instances in the class are NLPs (resp. DLPs). In the table, a number under “#ins” denotes the number of instances in the corresponding class, a number under “#NAS” denotes the number of instances which have no answer sets in the corresponding class and a number under “#WFM” denotes the average number of facts in grounded programs, which is also the size of positive consequence in $WFM^+(P^*)$.

The result shows that for some programs in benchmarks, $GRS^*$ is larger than the results computed from the approximation of the well-founded operator. The performance could be improved when a larger consequence is considered. We have also implemented a program for Algorithm 1 to compute $GRS(P, L)$ for DLPs. However the program requires a long running time and returns nothing more than $GRS^*$ for benchmarks.

6 Conclusions

In this paper, we consider when a consequence of a logic program can be used to simplify a logic program so that the resulting program would no longer contain atoms appearing in

\[1 \text{http://ss.sysu.edu.cn/~ewh/simplifying.html} \]
the consequence. We introduce notions of the strong and the weak reliable set of a program. We show that a logic program is strongly equivalent to the simplified program if and only if the consequence is a strong reliable set, and they have the same answer sets if and only if the consequence is a weak reliable set. We explore computational complexity on identifying a strong or a weak reliable set and provide an approach to verify a strong reliable set. These notions can provide a guideline to explore classes of consequences that could be used to simplify a logic program. As an example, we introduce the notion of the reliable set under a consequence of a program, which extends the well-founded model and provides a sufficient condition for a consequence to simplify a logic program. We also provide an algorithm to compute the notion. We plan to extend the idea to other notions of equivalence, such as relativized, uniform equivalence, and hyperequivalence, and find out corresponding applications in future.

7 Acknowledgments
We are grateful to Fangzhen Lin for many helpful and informative discussions. 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 National Natural Science Foundation of China under grant 61175057, the National Natural Science Foundation for the Youth of China under grant 61403539, as well as the USTC Key Direction Project and the USTC 985 Project. Hai Wan thanks Natural Science Foundation of Guangdong Province of China (No. S2012101009836), and Guangzhou Science and Technology Project (No. 2013J410058) for the support of this research.

References


