Simplifying a Logic Program Using Its Consequences

ID: 223
KRR: Logic Programming

Abstract

A consequence of a logic program is a consistent set of literals that are satisfied by every answer set of the program. Some consequences can be used to simplify the logic program so that the resulting program would no longer contain variables in the consequence and the answer sets can be computed from the answer sets of the resulting program. The well-founded model is such a consequence that is commonly used in the preprocessing state of answer set programming (ASP) solvers for simplifying logic programs. In this paper, we extend the notion of well-founded models and provide a sufficient and necessary condition for a consequence that could be used to simplify the logic program. We also provide a sufficient and necessary condition for a consequence so that the simplified logic program with rules constructed from the consequence is strongly equivalent to the original program. We explore the computational complexities on checking both conditions. As an application, we introduce a weak version of the conditions, which results a sufficient condition for a consequence to simplify a logic program. We provide an algorithm to identify the class of consequences specified by the sufficient condition. We show that for some programs, the consequence in the class is larger than the well-founded model.

1 Introduction

Nowadays *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. However, only a few of them can be used to simplify the logic program so that the resulting program would no longer contain variables in the consequence and the answer sets can be constructed from the answer sets of the resulting program. The best known examples 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], which have been used in grounding engines of ASP solvers.

To extend the notion of the well-founded model, we consider when a consequence can be used to simplify a logic program. The main contribution of the paper is that, we propose two conditions (a) and (b), so that

- a consequence satisfies the condition (a) iff the answer sets of the program can be constructed from the answer sets of the program simplified by the consequence,
- a consequence satisfies the condition (b) iff the original program is strongly equivalent to the simplified program with rules constructed from the consequence.

We also explore the computational complexities on checking both conditions. These conditions provide a guideline to explore classes of consequences that could be used to simplify a logic program. As an application, we introduce a weak version of the conditions, which provides a sufficient condition for a consequence to simplify a logic program. We 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.

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

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

where $n \geq m \geq k \geq 0$, $n \geq 1$ 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

$$head(r) \leftarrow body(r), \quad (2)$$

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.

A set S of atoms *satisfies* a rule r, if $body^+(r) \subseteq S$ and $body^-(r) \cap S = \emptyset$ implies $head(r) \cap S \neq \emptyset$. 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, $\overline{L} = \{\neg p \mid p \in L\} \cup \{p \mid \neg p \in L\}$, $L^+ = \{p \mid p \in L\}$, and $L^- = \{p \mid \neg p \in L\}$.

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 P^S, is obtained from P by deleting:

1. each rule that has a formula $not\ p$ in its body with $p \in S$,
2. all formulas of the form $not\ p$ 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 P^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 Y satisfies P and X 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 $X \subset Y$ 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 $head(r) \cap E \neq \emptyset$ and $E \cap body^+(r) = \emptyset$. 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

$$\bigwedge_{p \in E} p \supset \bigvee_{r \in R^-(E,P)} \Big(body(r) \wedge \bigwedge_{q \in head(r) \setminus E} \neg q \Big).$$

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 $body^+(r) \subseteq S$, $body^-(r) \cap S = \emptyset$, and $(head(r) \setminus E) \cap S = \emptyset$, *i.e.*, $S \cup \{\neg p \mid p \notin S\} \models LF(E, P)$.

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 $p \in head(r)$, at least one of the following conditions holds:

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

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:

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

$$W_P(I) = T_P(I) \cup \overline{U_P(I)}.$$

T_P, U_P, and W_P are monotonic operators. We use $WFM(P)$ to denote the least fixed point of the operator W_P. If P is an NLP, $WFM(P)$ 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 be used to simplify the logic program so that the resulting program would no longer contain variables in the consequence and the answer sets of the program can be computed from the answer sets of the resulting program. $WFM(P)$ is such a consequence that is commonly used to simplify the program in the grounding engines, like lparse [Syrjänen, 2000], gringo [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 $tr_n(P, L)$ to be the program obtained from P by

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

We define $tr_p(P, L)$ to be the program obtained from P by

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

Note that $tr_n(P, L)$ (resp. $tr_p(P, L)$) does not contain any atoms in L^- (resp. L^+) and $tr_p(tr_n(P, L), L)$ 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 $tr_p(tr_n(P, L), L) \cup \{p \leftarrow \mid p \in L\}$ have the same set of answer sets.

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

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

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

- *P and $tr_n(P, L)$ have the same set of answer sets.*
- *$P \cup \{\leftarrow p \mid \neg p \in L\}$ is strongly equivalent to $tr_n(P, L) \cup \{\leftarrow p \mid \neg p \in L\}$.*
- *An answer set of P is always an answer set of $tr_p(P, L) \cup \{p \leftarrow \mid p \in L\}$, 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 $LF(E, tr_n(P, L))$, and $LF(E, P)$ implies $LF(E, tr_p(P, L) \cup \{p \leftarrow \mid p \in L\})$. ■

Example 1 *Consider the logic program P_1:*

$a \leftarrow b, \quad c \leftarrow a, \quad b \leftarrow c, \quad c \leftarrow d, \quad a \leftarrow f,$

$d \leftarrow not\ e, \quad e \leftarrow not\ d, \quad \leftarrow not\ a, \quad f \leftarrow a.$

$L = \{a, f\}$ is a consequence of P_1, and $tr_p(P_1, L)$ is:

$c \leftarrow, \quad b \leftarrow c, \quad c \leftarrow d, \quad d \leftarrow not\ e, \quad e \leftarrow not\ d.$

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

Given a consequence L of a DLP P, we can simplify the program by $tr_n(P, L)$ which would no longer contain atoms in L^-. However, we cannot use $tr_p(P, L)$ to simplify P in general. In this paper, 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 the computational complexities 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 $head(r) \cap X \subseteq E$, $body^+(r) \subseteq X \cup U$, and $body^-(r) \cap (Y \cup U) = \emptyset$.

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, $U \setminus E \subseteq X$ 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 $tr_p(P, U) \cup \{p \leftarrow \mid p \in U\}$ if and only if U is a strong reliable set of P.*

Proof Sketch: $\Leftarrow$: 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 $(X \setminus E, Y)$ is not a SE-model of $tr_p(P, U) \cup \{p \leftarrow \mid p \in U\}$. ■

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 $X' \setminus U \subset X \setminus U$ and $(X' \cup U, Y)$ is a SE-model of P. We show that, a U-equilibrium model of P is related to an answer set of $tr_p(P, U)$.

Lemma 1 *Let P be a DLP and U a set of atoms. A SE-interpretation (X, Y) is a SE-model of $tr_p(P, U)$ if and only if there exists a SE-model (X^*, Y^*) of P such that $X^* = X \cup U$ and $Y^* = Y \cup U$.*

Proposition 6 *Let P be a DLP and U a set of atoms. A set S is an answer set of $tr_p(P, U)$ if and only if there exists a U-equilibrium model (X, Y) of P such that $X \setminus U = Y \setminus U = S$.*

Proof Sketch: $\Leftarrow$: (X, Y) is a U-equilibrium model of P, then $X \cup U = Y \cup U$ and $(X \cup U, Y \cup U)$ is a SE-model of P. So $(X \setminus U, Y \setminus U)$ is a SE-model of $tr_p(P, U)$. Meanwhile, there does not exist a set X' such that $X' \setminus U \subset X \setminus U$ and $(X' \cup U, Y \cup U)$ is a SE-model of P. From Lemma 1, $(X' \setminus U, Y \setminus U)$ is a SE-model of $tr_p(P, U)$. So $X \setminus U$ is an answer set of $tr_p(P, U)$.

$\Rightarrow$: S is answer set of $tr_p(P, U)$, then $(S \cup U, S \cup U)$ is a SE-model of P and there does not exists a set S' such that $S' \subset S$ and (S', S) is a SE-model of $tr_p(P, U)$. 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 $head(r) \cap X \subseteq E$, $body^+(r) \subseteq X \cup U$, and $body^-(r) \cap (Y \cup U) = \emptyset$.

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

$\Rightarrow$: If U is not a weak reliable set of P, then there exists a U-equilibrium model (X, Y) of P and a nonempty subset E of U that prevent U to be a weak reliable set of P. Similar to the proof for Theorem 1, $(X \setminus E, Y)$ is also a U-equilibrium model of P. Then Y is not an answer set of P but Y is an answer set of $tr_p(P, U) \cup \{p \leftarrow | \ p \in U\}$. ■

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 of P.

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 $\mathcal{C}$ of clauses, and $Atoms(\mathcal{C})$ the set of atoms in $\mathcal{C}$. We can construct a DLP P from $\mathcal{C}$ by:

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

It can be verified that $U = Atoms(\mathcal{C}) \cup \{t, e\}$ is a strong reliable set of P if and only if $\mathcal{C}$ 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 $\mathcal{C}$ of clauses, and $Atoms(\mathcal{C})$ the set of atoms in $\mathcal{C}$. We can construct an NLP P from $\mathcal{C}$ by:

- adding rules $e \leftarrow t$, $t \leftarrow not\, t'$, $t' \leftarrow not\, t$, and $\leftarrow t', e$,
- for each atom $p \in Atoms(\mathcal{C})$, adding the rule $p \leftarrow e$, and
- for each clause $C \in \mathcal{C}$, adding the rule $e \leftarrow \bigwedge_{\neg p \in C} p \wedge \bigwedge_{q \in C} \neg q$.

It can be verified that $U = Atoms(\mathcal{C}) \cup \{e, t\}$ is a strong reliable set of P if and only if $\mathcal{C}$ is not satisfiable. Moreover, U is a strong reliable set of P iff U is a weak 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 another 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 $X \cup \hat{Y}$ is a model of $\pi(P)$, where $\hat{Y}$ 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 $\pi(P) \models \bigwedge U$ 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 $head(r) \setminus L^- \subseteq E$ and $U \cup (L \setminus L^+) \models body(r)$.

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:*

$$P \cup \{\leftarrow not\, p \mid p \in L\} \cup \{\leftarrow p \mid \neg p \in L\}.$$

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 $head(r) \cap X \subseteq E$, $body^+(r) \subseteq X \cup U$ and $body^-(r) \cap (Y \cup U) = \emptyset$. 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 P under L, then*

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

(ii) *$P \cup \{\leftarrow not\, p \mid p \in L\} \cup \{\leftarrow p \mid \neg p \in L\}$ is strongly equivalent to*

$$tr_p(P, U) \cup \{p \leftarrow| \ p \in U\} \cup \{\leftarrow not\, p \mid p \in L\} \cup \{\leftarrow p \mid \neg p \in L\}.$$

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.*

$$\{r \mid r \in P,\ head(r) \setminus L^- \subseteq U \cup L^+\} \cup \{\leftarrow p \mid \neg p \in L\} \cup \{\leftarrow not\, p \mid p \in L\}$$

is strongly equivalent to

$$\{p \leftarrow \mid p \in U\} \cup \{\leftarrow p \mid \neg p \in L\} \cup \{\leftarrow not\, p \mid p \in L\}$$

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.

$\Leftarrow$: 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 $GRS(P, L)$, *i.e.*, the union of all possible reliable sets of P under L.

Example 1 (Continued) *Recall that $L = \{a, f\}$ is a consequence of P_1. From the definition, $GRS(P_1, L) = \emptyset$. $L' = \{\neg e\}$ is another consequence of P_1, and $GRS(P_1, L') = \{a, b, c, d, f\}$.*

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 $GRS(P, L)$ is coNP-hard.*
- *Deciding whether an atom p is in $GRS(P, L)$ 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 $U = Atoms(\mathcal{C}) \cup \{t, e\}$ is a strong reliable set of P iff U is a reliable set of P under $\emptyset$ iff $U = GRS(P, \emptyset)$ iff $e \in GRS(P, \emptyset)$. ■

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

Let P be a DLP, X a set of atoms and L a set of literals. $T_{P,L}(X) = \{p \mid$ there is a rule $r \in P$ such that $p \in head(r)$,

$$(L \setminus L^+) \cup X \models body(r) \wedge \bigwedge_{q \in head(r) \setminus \{p\}} \neg q\}.$$

$T_{P,L}$ 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

$GRS(P, L) \subseteq T(P, L)$. ■

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:*

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

Example 2 *Consider the DLP P_2:*

$$a \vee b \leftarrow, \qquad a \leftarrow b, \qquad b \leftarrow a.$$

$GRS(P_2, \emptyset) = \{a, b\}$, $T(P_2, \emptyset) = \emptyset$, and $WFM(P_2) = \emptyset$.

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

```
A := X;
H := {head(r) \ L⁻ | r ∈ P, head(r) ∩ body⁺(r) = ∅,
        X ∩ head(r) = ∅, and
     X ∪ (L \ L⁺) ⊨ body(r)};
for each C ∈ H do
    if |C| = 1 or
       for each p ∈ C, C ⊆ RS_{P,L}(X ∪ {p}) then
        A := A ∪ C;
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 the proof for 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 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^* and $R = \{p \leftarrow \mid p \in WFM(P)\}$. 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 $P^* = \{p \vee q \leftarrow,\ q \leftarrow not\ r,\ \leftarrow r\}$, the corresponding set of clauses is $\{p \vee q,\ q \vee r,\ \neg r\}$, and the consequence computed by applying unit propagation is $\{\neg r, q\}$. Based on such consequence L, we can compute $T(P^*, L)$. 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 $T(P^*, L)$.

We have implemented a program to compute GRS^* for programs grounded by gringo (version 4.4.0). Table 1 contains average sizes of consequences and GRS^* of P^* for different instances from 3 classes of NLPs and 2 classes of DLPs, and average times for computing these notions[1]. 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, and a number under "#WFM" denotes the average number of facts in grounded programs, which is also the size of positive consequence in $WFM(P')$.

Table 1: Comparing WFM and GRS^*

Benchmark	#ins	#WFM	Consequence		GRS^*	
			size	time(s)	size	time(s)
15-Puzzle(N)	9	85	1452	1379.27	0	5.96
Factoring(N)	5	0	2805	435.82	148	23.49
SchurNumbers(N)	5	48	284	46.60	91	0.74
Mutex(D)	7	3224	2935	128.10	0	0.01
RQBF(D)	15	80	11	0.07	0	0.00

The result shows that for some programs in benchmarks, GRS^* is larger than the results computed from the well-founded operator. Note that, 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 Conclusion

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 variables in the consequence and the answer sets can be constructed from the answer sets of the resulting program. We introduce notions of the strong and the weak reliable set of a program. We show that, a consequence can simplify a logic program if and only if it is a weak reliable set. Moreover, the simplified program with rules constructed from the consequence is strongly equivalent to the original program if and only if the consequence is a strong reliable set. We explore the computational complexities on identifying a strong or a weak reliable set and provide an approach to verify a strong reliable set. These notions provide a guideline to explore classes of consequences that could be used to simplify a logic program. As an application, 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.

[1] For more information, please visit the website.

References

[Chen *et al.*, 2013] Xiaoping Chen, Jianmin Ji, and Fangzhen Lin. Computing loops with at most one external support rule. *ACM Transactions on Computational Logic (TOCL)*, 14(1):3–40, 2013.

[Clark, 1978] Keith L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases*, pages 293–322. Plenum Press, New York, 1978.

[Denecker *et al.*, 2009] Marc Denecker, Joost Vennekens, Stephen Bond, Martin Gebser, and Mirosław Truszczyński. The second answer set programming competition. In *Proceedgins of the 10th International Converence on Logic Programming and Nonmonotonic Reasoning (LPNMR-09)*, pages 637–654. Springer, 2009.

[Drescher *et al.*, 2008] Christian Drescher, Martin Gebser, Torsten Grote, Benjamin Kaufmann, Arne König, Max Ostrowski, and Torsten Schaub. Conflict-driven disjunctive answer set solving. In *Proceedings of the 11th International Conference on Principles of Knowledge Respresentation and Reasoning (KR-08)*, pages 422–432, 2008.

[Ferraris, 2005] Paolo Ferraris. Answer sets for propositional theories. In *Proceedings of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-05), Diamante, Italy, September 5-8, 2005*, pages 119–131, 2005.

[Gebser *et al.*, 2007a] Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub. clasp: A conflict-driven answer set solver. In *Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-07)*, pages 260–265. Springer, 2007.

[Gebser *et al.*, 2007b] Martin Gebser, Torsten Schaub, and Sven Thiele. Gringo: A new grounder for answer set programming. In *Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-07)*, pages 266–271. Springer, 2007.

[Gebser *et al.*, 2013] Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. Advanced conflict-driven disjunctive answer set solving. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-13)*, pages 912–918. AAAI Press, 2013.

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

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

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

[Leone *et al.*, 2002] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Francesco Calimeri, Tina Dell'Armi, Thomas Eiter, Georg Gottlob, Giovambattista Ianni, Giuseppe Ielpa, Christoph Koch, et al. The dlv system. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA-02)*, pages 537–540. Springer-Verlag, 2002.

[Lierler and Maratea, 2004] Yuliya Lierler and Marco Maratea. Cmodels-2: Sat-based answer set solver enhanced to non-tight programs. In *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-04)*, pages 346–350. Springer, 2004.

[Lifschitz *et al.*, 2001] Vladimir Lifschitz, David Pearce, and Agustín Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic (TOCL)*, 2(4):526–541, 2001.

[Lin and Zhao, 2004] Fangzhen Lin and Yuting Zhao. Assat: Computing answer sets of a logic program by sat solvers. *Artificial Intelligence*, 157(1):115–137, 2004.

[Pearce *et al.*, 2001] David Pearce, Hans Tompits, and Stefan Woltran. Encodings for equilibrium logic and logic programs with nested expressions. In *Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA-01)*, pages 306–320. Springer, 2001.

[Syrjänen and Niemelä, 2001] Tommi Syrjänen and Ilkka Niemelä. The smodels system. In *Proceedings of the 6th International Conference on Logic Programming and Nonmotonic Reasoning (LPNMR-01)*, pages 434–438. Springer, 2001.

[Syrjänen, 2000] Tommi Syrjänen. Lparse 1.0 user's manual. 2000.

[Turner, 2003] Hudson Turner. Strong equivalence made easy: nested expressions and weight constraints. *Theory and Practice of Logic Programming*, 3(4+ 5):609–622, 2003.

[Van Gelder *et al.*, 1991] Allen Van Gelder, Kenneth A Ross, and John S Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM (JACM)*, 38(3):619–649, 1991.