# Approximating Answer Sets of Unitary Lifschitz-Woo Programs

Victor W. Marek[1], Inna Pivkina[2], and Mirosław Truszczyński[1]

[1] Department of Computer Science, University of Kentucky
Lexington, KY 40506-0046, USA
[2] Department of Computer Science, New Mexico State University
P.O. Box 30001, MSC CS, Las Cruces, NM 88003, USA

**Abstract.** We investigate techniques for approximating answer sets of general logic programs of Lifschitz and Woo, whose rules have single literals as heads. We propose three different methods of approximation and obtain results on the relationship between them. Since general logic programs with single literals as heads are equivalent to revision programs, we obtain results on approximations of justified revisions of databases by revision programs.

## 1 Introduction

General logic programs were introduced by Lifschitz and Woo [LW92]. Their syntax follows closely that of disjunctive logic programs but there is one essential difference. The operator $\mathbf{not}$, representing the *default negation* is no longer confined to the bodies of program rules but may appear in their heads, as well. Lifschitz and Woo [LW92] showed that the semantics of answer sets introduced for disjunctive logic programs in [GL91] can be lifted to the class of general logic programs.

In this paper, we study the class of those general programs that do not contain disjunctions in the heads of their rules. We call such programs *unitary*. Unitary general programs are of interest for two reasons. First, they go beyond the class of normal logic programs by allowing the default-negation operator in the rule heads. Second, in a certain precise sense, unitary general programs are equivalent to the class of revision programs [MT98,MPT02], which provide a formalism for describing and enforcing database revisions. Consequently, results for unitary general programs extend to the case of revision programs.

The problem we focus on in this paper is that of approximating answer sets of unitary general programs. The problem to decide whether a unitary logic program has an answer set is NP-complete[3]. Consequently, computing answer sets of unitary general programs is hard and it is important to establish efficient ways to approximate them. On one hand, such approximations can be sufficient for some reasoning tasks. On the other hand, they can be used by programs computing answer sets to prune the search space and can improve their performance significantly.

[3] Without the restriction to unitary programs (and assuming that the polynomial hierarchy does not collapse) the problem is even harder — $\Sigma_2^P$-complete.

In the case of normal logic programs the well-founded model [VRS88] provides an effective approximation to all answer sets[4]. It can be computed in polynomial time and is known to provide an effective pruning mechanism for programs computing stable models [SNV95,SNS02]. An obvious approach to the problem at hand seems to be then to extend the well-founded model and its properties to the class of unitary programs. However, despite similarities between normal and unitary programs, no counterpart of the well-founded model has been proposed for the latter class so far, and whether it can be done remains unresolved.

Thus, we approach the problem not by attempting to generalize the well-founded semantics but by exploiting this semantics in some other, less direct ways. Namely, we introduce three operators for unitary general programs and use them to define the approximations. The first two operators are antimonotone and are closely related to operators behind the well-founded semantics of normal logic programs. Iterating them yields *alternating* sequences. We use the limits of these sequences to construct our first two approximations to answer sets of unitary general programs. The two approximations we obtain in this way are not comparable (neither is stronger than the other one). The third operator is not antimonotone in general. However, in the case of unitary general programs that have answer sets, iterating this operator results in an alternating sequence and the limit of this sequence yields yet another approximation to answer sets of unitary general programs. We show that this third approximation is stronger than the other two. We also show that all three approaches imply sufficient conditions for the *non-existence* of answer sets of unitary programs.

As we noted, unitary programs are related to revision programs [MT98,MPT99]. Having introduced approximations to answer sets of unitary general programs, we show that our results apply in a direct way to the case of revision programming.

All programs we consider in the paper are *finite*. That assumption simplifies arguments. However, all our results can be extended to the case of infinite programs.

## 2 Preliminaries

**Atoms and literals.** In the paper we consider a fixed set $U$ of (propositional) atoms. Expressions of the form $a$ and $\mathbf{not}(a)$, where $a \in U$, are *literals* (over $U$). We denote the set of all literals over $U$ by $Lit(U)$. A set of literals $L \subseteq Lit(U)$ is *coherent* if there is no $a \in U$ such that both $a \in L$ and $\mathbf{not}(a) \in L$. A set of literals $L \subseteq Lit(U)$ is *complete* if for every $a \in U$, $a \in L$ or $\mathbf{not}(a) \in L$ (it is possible that for some $a$, both $a \in L$ and $\mathbf{not}(a) \in L$).

For a set $M$ of atoms, $M \subseteq U$, we define

$$\mathbf{not}(M) = \{\mathbf{not}(a) \colon a \in M\} \text{ and } M^c = M \cup \mathbf{not}(U \setminus M).$$

The mapping $M \mapsto M^c$ is a bijection between subsets of $U$ and coherent and complete sets of literals contained in $Lit(U)$.

[4] In the context of normal logic programming, answer sets are more commonly known as *stable models*.

**Unitary general programs.** A *unitary general logic program*, or *UG-program* is a collection of rules of the form:

$$\alpha \leftarrow \alpha_1, \ldots, \alpha_m \tag{1}$$

where $\alpha$, $\alpha_1, \ldots, \alpha_m$ are literals from $Lit(U)$. The literal $\alpha$ is the *head* of the rule. The set of literals $\{\alpha_1, \ldots, \alpha_m\}$ is the *body* of the rule.

Let $P$ be a UG-program. We write $P^+$ (respectively, $P^-$) to denote programs consisting of all rules in $P$ that have an atom (respectively, a negated atom) as the head.

**Satisfaction and models.** A set of atoms $M \subseteq U$ *satisfies* (is a *model* of) an atom $a \in U$ (respectively, a literal $\mathbf{not}(a) \in Lit(U)$), if $a \in M$ (respectively, $a \notin M$). The concept of satisfaction (being a model of) extends in a standard way to rules and programs. As usual, we write $\models$ to denote the satisfaction relation.

**Sets of literals closed under UG-programs.** In addition to models, we also associate with a UG-program $P$ sets of literals that are closed under rules in $P$. A set $L$ of literals is *closed* under a UG-program $P$ if for every rule $r = \alpha \leftarrow Body \in P$ such that $Body \subseteq L$, $\alpha \in L$. One can show that every UG-program $P$ has a least set of literals closed under its rules[5]. We denote it by $P^*$. We observe that if $P$ is a definite Horn program, $P^*$ consists of atoms only and coincides with the least model of $P$.

**Stable models of normal logic programs.** Models are too weak for knowledge representation applications. In the case of normal logic programs, the appropriate semantic concept is that of a stable model. We recall that according to the original definition [GL88], a set of atoms $M$ is a stable model of a normal logic program $P$ if

$$[P^M]^* = M, \tag{2}$$

where $P^M$ is the *Gelfond-Lifschitz* reduct of $P$ with respect to $M$. The following characterization of stable models is well known [BTK93]: $M$ is a stable model of a normal logic program $P$ if and only if

$$[P \cup \mathbf{not}(U \setminus M)]^* \cap U = M. \tag{3}$$

**Answer sets of UG-programs.** Lifschitz and Woo [LW92] extended the concept of a stable model to the case of arbitrary general programs and called the resulting semantic object an *answer set*. Rather than to give the original definition from [LW92], we recall a basic characterization of answer sets of UG-programs that will be of use in the paper. Its proof can be found in [Lif96,MPT99].

**Proposition 1.** *Let $P$ be a UG-program. A set of atoms $M$ is an answer set to $P$ if and only if $M$ is a stable model of $P^+$ and a model of $P^-$. In particular, if $M$ is an answer set to $P$ then $M$ is a model of $P$.*

**Alternating sequences.** All approximations to answer sets of UG-programs we study in this paper are defined in terms of alternating sequences and their limits. A sequence $(X_i)$ of sets of literals is *alternating* if

[5] If we treat literals $\mathbf{not}(a)$ as new atoms, $P$ becomes a Horn program and its least model is the least set of literals closed under $P$.

1. $X_0 \subseteq X_2 \subseteq X_4 \subseteq \ldots$
2. $X_1 \supseteq X_3 \supseteq X_5 \supseteq \ldots$
3. $X_{2i} \subseteq X_{2i+1}$, for every non-negative integer $i$.

If $(X_i)$ is an alternating sequence, we define $X^l = \bigcup_{i=0}^{\infty} X_{2i}$ and $X^u = \bigcap_{i=0}^{\infty} X_{2i+1}$. We call the pair $(X^l, X^u)$ the *limit* of the alternating sequence $(X_i)$. It follows directly from the definition that for every non-negative integers $i$ and $j$,

$$X_{2i} \subseteq X^l \subseteq X^u \subseteq X_{2j+1}$$

Alternating sequences are often defined by means of operators that are antimonotone. An operator $\gamma$ defined on $Lit(U)$ is *antimonotone* if for every two sets $X \subseteq Y \subseteq Lit(U)$, $\gamma(Y) \subseteq \gamma(X)$. Let $\gamma$ be antimonotone. We define $X_0 = \emptyset$ and $X_{i+1} = \gamma(X_i)$. It is well known (and easy to show) that the sequence $(X_i)$ is alternating. We call $(X_i)$ the *alternating* sequence of $\gamma$.

We will consider in the paper the following two operators:

$$\gamma_{P,U}(X) = [P \cup \mathbf{not}(U \setminus X)]^* \cap U \text{ and } \gamma_P(X) = [P \cup \mathbf{not}(U \setminus X)]^*.$$

Both operators are antimonotone and give rise to alternating sequences, say $(W_i)$ and $(Y_i)$. Let $(W^l, W^u)$ and $(Y^l, Y^u)$ be the limits of these sequences, respectively. One can verify that these limits form *alternating pairs*. That is, we have

$$\gamma_{P,U}(W^l) = W^u \text{ and } \gamma_{P,U}(W^u) = W^l \tag{4}$$

and

$$\gamma_P(Y^l) = Y^u \text{ and } \gamma_P(Y^u) = Y^l. \tag{5}$$

One can show that if $P$ is a normal logic program then the alternating sequence of $\gamma_{P,U}$ is precisely the alternating sequence defining the well-founded semantics of $P$ [VRS88,Van93].

One can also show that the limit of the alternating sequence of $\gamma_P$ is the well-founded model of the normal logic program $P'$ obtained from $P$ by replacing every literal $\mathbf{not}(a)$ with a *new* atom, say $a'$, and adding rules of the form $a' \leftarrow \mathbf{not}(a)$ (the claim holds modulo the correspondence $a' \leftrightarrow \mathbf{not}(a)$). The mapping $P \mapsto P'$ was introduced and studied in [PT95] in the context of revision programs.

**Approximating sets of atoms.** Let $M$ be a set of atoms. Every pair of sets $(T, S)$ that *approximates* $M$, that is, such that $T \subseteq M \subseteq S$, implies a lower bound on the complete representation $M^c$ of $M$:

$$T \cup \{\mathbf{not}(U \setminus S)\} \subseteq M^c.$$

Conversely, every set $L$ of literals such that $L \subseteq M^c$ determines an *approximation* $(T, S)$ of $M$, where $T = U \cap L$ and $S = \{a \in U \colon \mathbf{not}(a) \notin L\}$. Indeed,

$$U \cap L \subseteq M \subseteq \{a \in U \colon \mathbf{not}(a) \notin L\}.$$

In this way, we establish a bijection between approximations to a set of atoms $M$ and subsets of $M^c$. It follows that approximations of answer sets can be represented as subsets of their complete representations. We have the following fact.

**Proposition 2.** *Let $P$ be a UG-program and let $T$ and $S$ be two sets of atoms. For every answer set $M$ of $P$, if $T \subseteq M \subseteq S$ then $[P \cup T \cup \mathbf{not}(U \setminus S)]^* \subseteq M^c$.*

Proof: We have $T \subseteq M \subseteq S$. Thus, $T \cup \mathbf{not}(U \setminus S) \subseteq M^c$. Let $r = \alpha \leftarrow Body$ be a rule in $P$ such that $Body \subseteq M^c$. It follows that $M$ satisfies the body of $r$. Since $M$ is an answer set of $P$, $M$ satisfies $\alpha$ and so, $\alpha \in M^c$. Thus, $T \cup \mathbf{not}(U \setminus S) \subseteq M^c$ and $M^c$ is closed under $P$. Consequently, $[P \cup T \cup \mathbf{not}(U \setminus S)]^* \subseteq M^c$. $\Box$

In the case of normal logic programs, the well-founded model, that is, the limit $(W^l, W^u)$ of the alternating sequence $(W_i)$ of the operator $\gamma_{P,U}$, approximates every stable model (if they exist) and, in some cases determines the existence of a unique stable model.

**Theorem 1 ([VRS88,Lif96]).** *Let $(W^l, W^u)$ be the well-founded model of a normal logic program $P$.*

1. *For every stable model $M$ of $P$, $W^l \cup \mathbf{not}(U \setminus W^u) \subseteq M^c$.*
2. *If $W^l = W^u$, then $W^l$ is a unique stable model for $P$.*

In the remainder of the paper, we will propose approximations to answer sets of UG-programs generalizing Theorem 1.

## 3 Approximating answer sets using operators $\gamma_{P,U}$ and $\gamma_P$

Our first approach exploits the fact that every answer set of a UG-program $P$ is a stable model of $P^+$ (Proposition 1). Let $P$ be a UG-program and let $(W^l, W^u)$ be the limit of the alternating sequence of the operator $\gamma_{P^+,U}$. As we observed, $(W^l, W^u)$ is the well-founded model of $P^+$. We define

$$Appx_1(P) = [P \cup \mathbf{not}(U \setminus W^u)]^*.$$

By (4), $W^l = [P \cup \mathbf{not}(U \setminus W^u)]^* \cap U$. Hence, $W^l \subseteq Appx_1(P)$ and so, $Appx_1(P)$ contains all literals that are true in the well-founded model $(W^l, W^u)$.

**Theorem 2.** *Let $P$ be a UG-program. For every answer set $M$ of $P$, $Appx_1(P) \subseteq M^c$. In addition, if $Appx_1(P)$ is incoherent then $P$ has no answer sets.*

Proof: Let $M$ be an answer set of $P$. By Proposition 1, $M$ is a stable model of $P^+$. Let $(W^l, W^u)$ be the well-founded model of $P^+$. By Theorem 1, $\mathbf{not}(U \setminus W^u) \subseteq M^c$. Moreover, since $M$ is an answer set of $P$, $M$ is a model of $P$ (Proposition 1, again) and so, $M^c$ is closed under $P$. Since $Appx_1(P)$ is the least set of literals containing $\mathbf{not}(U \setminus W^u)$ and closed under $P$, $Appx_1(P) \subseteq M^c$, as claimed. The second part of the assertion follows from the first one. $\Box$

We will illustrate this approach with an example.

*Example 1.* Let us consider the following UG-program $P$:

$$\begin{array}{ll} a \leftarrow \mathbf{not}(b), \mathbf{not}(c) & d \leftarrow \mathbf{not}(b) \\ c \leftarrow c, \mathbf{not}(b) & \mathbf{not}(b) \leftarrow \\ b \leftarrow \mathbf{not}(d) & \end{array}$$

All but the last rule belong to $P^+$. The operator $\gamma_{P^+,U}$ determines the following alternating sequence $(W_i)$ of sets:

$$\emptyset \mapsto \{a, b, d\} \mapsto \emptyset \ldots .$$

It follows that the well-founded model of $P^+$ is $(W^l, W^u) = (\emptyset, \{a, b, d\})$. Consequently,

$$Appx_1(P) = [P \cup \{\mathbf{not}(c)\}]^* = \{a, d, \mathbf{not}(b), \mathbf{not}(c)\}.$$

In this case, the well-founded model of $P^+$ alone provides a weak bound on answer sets of $P$. The improved bound $Appx_1(P)$, which closes the model under $P$, provides a much stronger approximation. In fact, only one set $M$ is approximated by $\{a, d, \mathbf{not}(b), \mathbf{not}(c)\}$. This set is $\{a, d\}$ and it happens to be a unique answer set of $P$.

Let $Q = P \cup \{\mathbf{not}(a) \leftarrow d\}$. Since $Q^+ = P^+$, it follows that $Appx_1(Q) = [Q \cup \{\mathbf{not}(c)\}]^* = \{a, d, \mathbf{not}(a), \mathbf{not}(b), \mathbf{not}(c)\}$. Since $Appx_1(Q)$ is incoherent, $Q$ has no answer sets, a fact that can be verified directly. $\square$

The approximation $Appx_1(P)$, where $P$ is the first program from Example 1, is complete and coherent, and we noted that the unique set of atoms that $Appx_1(P)$ approximates is a unique answer set of $P$. It is a general property extending Theorem 1(2).

**Corollary 1.** *Let $P$ be a UG-program. If $Appx_1(P)$ is coherent and complete then $Appx_1(P) \cap U$ is a unique answer set of $P$.*

Proof: Since $Appx_1(P)$ is coherent and complete, Theorem 2 implies that $P$ has at most one answer set. To prove the assertion it is then enough to show that $M = Appx_1(P) \cap U$ is an answer set of $P$.

Let $(W^l, W^u)$ be the well-founded model of $P^+$. Since $Appx_1(P) = [P \cup \mathbf{not}(U \setminus W^u)]^*$, $[P \cup \mathbf{not}(U \setminus W^u)]^*$ is coherent and complete. Consequently,

$$M^c = [P \cup \mathbf{not}(U \setminus W^u)]^*.$$

It follows that $\mathbf{not}(U \setminus W^u) \subseteq \mathbf{not}(U \setminus M)$. Thus, $M^c \subseteq [P \cup \mathbf{not}(U \setminus M)]^*$. It also follows that $M^c$ is closed under the rules in $P$. Since $\mathbf{not}(U \setminus M) \subseteq M^c$, $[P \cup \mathbf{not}(U \setminus M)]^* \subseteq M^c$. Thus,

$$M^c = [P \cup \mathbf{not}(U \setminus M)]^*.$$

It follows now that $M$ is a model of $P^-$. Moreover, it also follows that $M = [P^+ \cup \mathbf{not}(U \setminus M)]^*$ and so, $M$ is a stable model of $P^+$. Thus, $M$ is an answer set of $P$. $\square$

We will now introduce another approximation to answer sets of a UG-program $P$. This time, we will use the operator $\gamma_P$. Let $Y_i$ be the alternating sequence of the operator $\gamma_P$ and let $(Y^l, Y^u)$ be the limit of $(Y_i)$. We define

$$Appx_2(P) = Y^l.$$

**Theorem 3.** *Let $P$ be a UP-program. If $M$ is an answer-set for $P$ then $Appx_2(P) \subseteq M^c$. In addition, if $Appx_2$ is incoherent, then $P$ has no answer sets.*

Proof: Let $M$ be an answer set of $P$ and let $(Y_i)$ be the alternating sequence for the operator $\gamma_P$. We will show by induction that for every $i \geq 0$, $Y_{2i} \cap U \subseteq M \subseteq Y_{2i+1}$.

Since $Y_0 = \emptyset$, $Y_0 \cap U \subseteq M$. We will now assume that $Y_{2i} \cap U \subseteq M$ and show that $M \subseteq Y_{2i+1}$. Our assumption implies that $\mathbf{not}(U \setminus M) \subseteq \mathbf{not}(U \setminus Y_{2i})$. Thus, since $M$ is a stable model of $P^+$, it follows from (3) that

$$M = [P^+ \cup \mathbf{not}(U \setminus M)]^* \cap U \subseteq [P \cup \mathbf{not}(U \setminus M)]^* \subseteq [P \cup \mathbf{not}(U \setminus Y_{2i})]^* = Y_{2i+1}.$$

Next, we assume that $M \subseteq Y_{2i+1}$ and show that $Y_{2i+2} \cap U \subseteq M$. The assumption implies that $\mathbf{not}(U \setminus Y_{2i+1}) \subseteq \mathbf{not}(U \setminus M)$. Thus,

$$\begin{aligned} Y_{2i+2} \cap U &= [P \cup \mathbf{not}(U \setminus Y_{2i+1})]^* \cap U \subseteq [P \cup \mathbf{not}(U \setminus M)]^* \cap U \\ &= [P^+ \cup \mathbf{not}(U \setminus M)]^* \cap U = M. \end{aligned}$$

The last but one equality follows from the fact that $M$ is a model of $P^-$ and the last inequality follows from the fact that $M$ is a stable model of $P^+$.

From the claim it follows that $M \subseteq Y^u$. Thus, $\mathbf{not}(U \setminus Y^u) \subseteq M^c$. Since $M$ is a model of $P$, $M^c$ is closed under $P$. Thus, $Y^l = [P \cup \mathbf{not}(U \setminus Y^u)]^* \subseteq M^c$. □

As before, if the approximation provided by $Appx_2(P)$ is complete and coherent, $P$ has a unique answer set.

**Corollary 2.** *Let $P$ be a UG-program such that $Appx_2(P)$ is complete and coherent. Then, $Appx_2(P) \cap U$ is a unique answer set of $P$.*

The following example illustrates our second approach.

*Example 2.* Let $U = \{a, b\}$. Let $P$ be a UG-program consisting of rules:

$$\begin{aligned} \mathbf{not}(a) &\leftarrow \mathbf{not}(b) \\ b &\leftarrow \mathbf{not}(a) \\ a &\leftarrow \end{aligned}$$

Iterating the operator $\gamma_P$ results in the following alternating sequence:

$$\emptyset \mapsto \{a, b, \mathbf{not}(a), \mathbf{not}(b)\} \mapsto \{a\} \mapsto \{a, b, \mathbf{not}(a), \mathbf{not}(b)\} \mapsto \ldots .$$

Its limit is $(\{a\}, \{a, b, \mathbf{not}(a), \mathbf{not}(b)\})$ and so, $Appx_2(P) = \{a\}$. □

We conclude this section by showing that the approximations $Appx_1$ and $Appx_2$ are, in general, not comparable.

The following example shows that there is a UG-program $P$ such that $Appx_1(P)$ and $Appx_2(P)$ are coherent and $Appx_2(P)$ is a *proper* subset of $Appx_1(P)$.

*Example 3.* Let $U = \{a, b, c, d, e\}$ and let $P$ be a UG-program consisting of the rules:

$$\begin{aligned} a &\leftarrow \mathbf{not}(a) & d &\leftarrow \mathbf{not}(c), \mathbf{not}(e) \\ b &\leftarrow \mathbf{not}(a) & e &\leftarrow \\ c &\leftarrow \mathbf{not}(d) & a &\leftarrow c, e \\ & & \mathbf{not}(e) &\leftarrow a, b \end{aligned}$$

**Computing** $Appx_1(P)$**.** The program $P^+$ consists of all rules of $P$ except the last one. The alternating sequence of $\gamma_{P+,U}$ starts as follows:

$$\emptyset \mapsto \{\, a, b, c, d, e \,\} \mapsto \{e\} \mapsto \{\, a, b, c, e \,\} \mapsto \{\, a, c, e \,\} \mapsto \{\, a, c, e \,\} \mapsto \dots \,.$$

Thus, its limit is $(\{a, c, e\}, \{a, c, e\})$ and

$$Appx_1(P) = [P \cup \{a, c, e\} \cup \{\mathbf{not}(b), \mathbf{not}(d)\}]^* = \{a, c, e, \mathbf{not}(b), \mathbf{not}(d)\}.$$

**Computing** $Appx_2(P)$**.** Iterating the operator $\gamma_P$ yields the following sequence:

$$\emptyset \mapsto Lit(U) \mapsto \{e\} \mapsto Lit(U) \mapsto \dots \,.$$

Thus, the limit is $(\{e\}, Lit(U))$ and so, $Appx_2(P) = \{e\}$. □

The next example shows that for some programs the opposite is true and the second approximation is strictly more precise.

*Example 4.* Let $U = \{a, b, c\}$ and let $P$ be a UG-program consisting of the rules:

$$\begin{array}{ll} a \leftarrow \mathbf{not}(b) & \qquad c \leftarrow a, b \\ b \leftarrow \mathbf{not}(a) & \qquad \mathbf{not}(a) \leftarrow \end{array}$$

**Computing** $Appx_1(P)$**.** The alternating sequence of the operator $\gamma_{P+,U}$ is

$$\emptyset \mapsto \{a, b, c\} \mapsto \emptyset \mapsto \dots \,.$$

Thus,

$$Appx_1(P) = P^* = \{\mathbf{not}(a), b\}.$$

**Computing** $Appx_2(P)$**.** Iterating $\gamma_P$ yields:

$$\emptyset \mapsto Lit(U) \mapsto \{\mathbf{not}(a), b\} \mapsto \{\mathbf{not}(a), b, \mathbf{not}(c)\} \mapsto \{\mathbf{not}(a), b, \mathbf{not}(c)\} \mapsto \dots \,.$$

Thus, $Appx_2(P) = \{\mathbf{not}(a), b, \mathbf{not}(c)\}$. □

## 4 Strong approximation

Let $P$ be a UG-program and $Z \subseteq Lit(U)$ a set of literals (not necessarily *coherent*). By the *weak reduct* of $P$ with respect to $Z$ we mean the program $P^Z_w$ obtained from $P$ by:

1. removing all rules that contain in the body a literal $\mathbf{not}(a)$ such that $a \in Z$ and $\mathbf{not}(a) \notin Z$;
2. removing from the bodies of the remaining rules all literals $\mathbf{not}(a)$ such that $a \notin Z$.

Let us note that if $a \in Z$ and $\mathbf{not}(a) \in Z$, $\mathbf{not}(a)$ will not be removed from the rules that remain after Step 1.

Let $Z$ be a set of literals, $Z \subseteq Lit(U)$. We define

$$\gamma^w_P(Z) = [P^Z_w]^*.$$

In general, the operator $\gamma^w_P$ is not antimonotone. Thus, the sequence $(Z_i)$ obtained by iterating $\gamma^w_P$ (starting with the empty set) in general is not alternating.

*Example 5.* Let $P$ be a UG-program consisting of the rules:

$$\begin{array}{rcr} a \leftarrow \mathbf{not}(b) & \qquad & c \leftarrow \mathbf{not}(d) \\ b \leftarrow & & d \leftarrow \\ \mathbf{not}(b) \leftarrow \mathbf{not}(c) & & \end{array}$$

By the definition, $Z_0 = \emptyset$. When computing $P^{Z_0}$, no rule is removed in Step 1 of the definition of the weak reduct, and every literal of the form $\mathbf{not}(a)$ is removed from the bodies of rules in $P$. Thus, $Z_1 = \{a, b, c, d, \mathbf{not}(b)\}$. When computing $P_w^{Z_1}$, we observe that $\mathbf{not}(b) \in Z_1$. Thus, the first rule is not removed despite the fact that $b \in Z_1$. Hence, we have:

$$P_w^{Z_1} = \left\{ \begin{array}{l} a \leftarrow \mathbf{not}(b) \\ b \leftarrow \\ d \leftarrow \end{array} \right\}, \quad \text{and so, } Z_2 = \{b, d\}.$$

In the next step, we compute:

$$P_w^{Z_2} = \left\{ \begin{array}{r} b \leftarrow \\ \mathbf{not}(b) \leftarrow \\ d \leftarrow \end{array} \right\}, \quad \text{and so, } Z_3 = \{b, d, \mathbf{not}(b)\}.$$

When computing $P_w^{Z_3}$, the rule $a \leftarrow \mathbf{not}(b)$ is again *not* removed in Step 1. Thus,

$$P_w^{Z_3} = \left\{ \begin{array}{r} a \leftarrow \mathbf{not}(b) \\ b \leftarrow \\ \mathbf{not}(b) \leftarrow \\ d \leftarrow \end{array} \right\}, \quad \text{and so, } Z_4 = \{a, b, d, \mathbf{not}(b)\}.$$

We note that $Z_4$ is *not* a subset of $Z_3$. Thus, for this program $P$, the sequence $(Z_i)$ is not alternating. □

In the remainder of this section we show that under some conditions the sequence $(Z_i)$ is alternating and may be used to approximate answer sets of UG-programs. We first establish a lemma providing conditions, under which $[P_w^X]^*$ is antimonotone in $X$.

**Lemma 1.** *Let $P$ be a UG-program, $X$ and $X'$ be sets of literals such that $X \subseteq X'$. Moreover, let at least one of the following conditions hold:*

1. *$X'$ is coherent*
2. *$X \subseteq [P_w^{X'}]^*$ and $[P_w^{X'}]^*$ is coherent.*
3. *$[P_w^{X'}]^* \subseteq X$*
4. *$X \subseteq [P_w^X]^*$ and $[P_w^X]^*$ is coherent.*

*Then $[P_w^{X'}]^* \subseteq [P_w^X]^*$.*

The next lemma describes two properties of $[P_w^X]^*$ under the assumption that $X$ is coherent.

**Lemma 2.** *Let $P$ be a UG-program and $X$ a coherent set of literals, $X \subseteq Lit(U)$.*

1. $[P_w^X]^* = [P_w^{X \cap U}]^*$.
2. $[P_w^X]^* = [(P^+)_w^{X \cap U}]^* \cup \mathbf{not}(X') = [(P^+)^{X \cap U}]^* \cup \mathbf{not}(X')$,
   *where $X'$ is the set of atoms such that $a \in X'$ if and only if there is a rule $\mathbf{not}(a) \leftarrow Body$ in $(P^-)_w^X$ such that $[(P^+)_w^{X \cap U}]^* \models Body$.*

We can now prove the following characterization of answer sets of UG-programs.

**Lemma 3.** *Let $P$ be a UG-program, $M \subseteq U$ a set of atoms, and $N$ a set of atoms consisting of all atoms $a \in U$ such that $a \notin M$ and there is a rule $\mathbf{not}(a) \leftarrow Body$ in $P$ such that $M \models Body$. Then $M$ is an answer set of $P$ if and only if $[P_w^M]^* = M \cup \mathbf{not}(N)$.*

Proof: ($\Rightarrow$) By Proposition 1, $M$ is a stable model of $P^+$ and a model of $P^-$. In particular, $[(P^+)^M]^* = M$. Let $X'$ be the set specified in Lemma 2(2), defined for $X = M$. Since $[(P^+)^M]^* = M$ and $M$ is a model of $P^-$, for every $a \in X'$, $a \notin M$. Thus, $X' = N$ and the assertion follows from Lemma 2(2).
($\Leftarrow$) It follows from Lemma 2(2) that $M = [(P^+)^M]^*$. Thus, $M$ is stable model of $P^+$. Let us consider a rule $\mathbf{not}(a) \leftarrow Body$ from $P^-$ such that $M$ satisfies $Body$. Let $Body'$ consist of all atoms in $Body$. It follows that $\mathbf{not}(a) \leftarrow Body'$ is a rule in $(P^-)_w^M$. Since $M \models Body$, $M \models Body'$. Thus, by Lemma 2(2), $\mathbf{not}(a) \in [P_w^M]^*$. Since $[P_w^M]^* = M \cup \mathbf{not}(N)$, $a \in \mathbf{not}(N)$ which, in turn, implies $a \notin M$. It follows that $M$ is a model of $P^-$ and so, an answer set of $P$. □

The results we presented above allow us to prove that as long as the lower (even) terms of the sequence $(Z_i)$ are coherent, the sequence behaves as an alternating one.

**Proposition 3.** *Let $i$ be an integer, $i \geq 0$, such that $Z_{2i}$ is coherent. Then*

1. $Z_0 \subseteq Z_2 \subseteq \ldots \subseteq Z_{2i}$
2. $Z_1 \supseteq Z_3 \supseteq \ldots \supseteq Z_{2i+1}$
3. $Z_{2i} \subseteq Z_{2i+1}$.

This last proposition is crucial for the definition of our third approximation. Let us consider the sequence $(Z_i)$. If for every $i$, $Z_{2i}$ is coherent, Proposition 3 implies that the sequence $(Z_i)$ is alternating. Let $(Z^l, Z^u)$ be the limit of $(Z_i)$. We define

$$Appx_3(P) = Z^l \cup \{\mathbf{not}(a)\colon a \in U \setminus Z^u\}.$$

Otherwise, there is $i$ such that $Z_{2i}$ is incoherent. In this case, we say that $Appx_3(P)$ is undefined.

**Theorem 4.** *Let $P$ be a UG-program. If $M$ is an answer set of $P$ then $Appx_3(P)$ is defined and $Appx_3(P) \subseteq M^c$. If $Appx_3(P)$ is not defined, then $P$ has no answer sets.*

Proof: The second part of the assertion follows from the first one. To prove the first part of the assertion, we will show that for every $i \geq 0$, $Z_{2i} \subseteq M^c$, and $M \subseteq Z_{2i+1}$.

We proceed by induction on $i$. If $i = 0$, then $Z_0 = \emptyset \subseteq M^c$. We now assume that $Z_{2i} \subseteq M^c$ and prove that $M \subseteq Z_{2i+1}$.

Since $Z_{2i} \subseteq M^c$ and $M^c$ is coherent, $Z_{2i}$ is coherent, too. By Lemma 1 (applied to $X = Z_{2i}$ and $X' = M^c$, under the assumption (4)), $[P_w^{M^c}]^* \subseteq [P_w^{Z_{2i}}]^*$. Thus, $[P_w^{M^c}]^* \subseteq Z_{2i+1}$. By Lemma 2(1), $[P_w^M]^* \subseteq Z_{2i+1}$. By Lemma 3, $M \subseteq [P_w^M]^*$. Therefore, $M \subseteq Z_{2i+1}$.

Next, we assume that $M \subseteq Z_{2i+1}$ and prove that $Z_{2i+2} \subseteq M^c$. Let us note that $Z_{2i+2} = [P_w^{Z_{2i+1}}]^*$ and that by Lemma 3, $[P_w^M]^* \subseteq M^c$. Thus, it will suffice to show that $[P_w^{Z_{2i+1}}]^* \subseteq [P_w^M]^*$. To this end, we note that by Lemma 3, $M \subseteq [P_w^M]^*$ and so Lemma 1 applies (under the condition (4)) to $X = M$ and $X' = Z_{2i+1}$, and implies the required inclusion.

It follows that $Z^l \subseteq M^c$ and that $M \subseteq Z^u$. If $a \notin Z^u$, then $a \notin M$ and so, $\mathbf{not}(a) \in M^c$. Thus, $Appx_3(P) = Z^l \cup \mathbf{not}(U \setminus Z^u) \subseteq M^c$. □

*Example 6.* Let $P$ be a UG-program consisting of the rules:

$$\begin{array}{rlcrl}
\mathbf{not}(a) & \leftarrow & \qquad & \mathbf{not}(d) & \leftarrow \mathbf{not}(c) \\
a & \leftarrow \mathbf{not}(b) & & d & \leftarrow \mathbf{not}(e) \\
b & \leftarrow \mathbf{not}(a) & & e & \leftarrow \mathbf{not}(d) \\
c & \leftarrow a, b & & f & \leftarrow d, e
\end{array}$$

Iterating the operator $\gamma_P^w$ results in the following sequence:

$$\emptyset \mapsto \{a, b, c, d, e, f, \mathbf{not}(a), \mathbf{not}(d)\} \mapsto \{\mathbf{not}(a), b\} \mapsto \{b, d, e, f, \mathbf{not}(a), \mathbf{not}(d)\}$$
$$\mapsto \{b, e, \mathbf{not}(a), \mathbf{not}(d)\} \mapsto \{b, e, \mathbf{not}(a), \mathbf{not}(d)\} \mapsto \dots .$$

Thus, the sequence $(Z_i)$ is alternating. Its limit is $(Z^l, Z^u)$, where $Z^l = Z^u = \{b, e, \mathbf{not}(a), \mathbf{not}(d)\}$. Thus,

$$Appx_3(P) = Z^l \cup \mathbf{not}(U \setminus Z^u) = \{b, e, \mathbf{not}(a), \mathbf{not}(c), \mathbf{not}(d), \mathbf{not}(f)\}.$$

Since $Appx_3(P)$ is coherent and complete, $P$ has a unique answer set, $\{b, e\}$. This example also demonstrates that $Z^u$ can improve on the bound provided by $Z^l$ itself. □

## 5 Properties of $Appx_3$

In this section we will show that if $Appx_3$ is defined then it is stronger than the other two approximations. We recall that if $Appx_3(P)$ is undefined, then $P$ has no answer sets, that is, $P$ is *inconsistent*. It follows that for all *consistent* UG-programs, $Appx_3$ is stronger than the the other two approximations.

**Theorem 5.** *LetP be a UG-program. If $Appx_3(P)$ is defined then*

$$Appx_1(P) \cup Appx_2(P) \subseteq Appx_3(P)$$

There are programs which show that$Appx_3$ is strictly stronger.

*Example 7.* Let $P$ be the UG-program from Example 4. We recall that $Appx_1(P) = \{\mathbf{not}(a), b\}$. Let us compute $Appx_3(P)$. By iterating the operator $\gamma_w^P$, we obtain the following sequence:

$$Z_0 = \emptyset \mapsto Z_1 = \{a, b, c, \mathbf{not}(a)\} \mapsto Z_2 = \{\mathbf{not}(a), b\} \mapsto Z_3 = \{\mathbf{not}(a), b\} \dots .$$

Hence, $Appx_3(P) = \{\mathbf{not}(a), b, \mathbf{not}(c)\}$ and $Appx_1(P)$ is a *proper* subset of $Appx_3(P)$. $\square$

*Example 8.* Let $P$ be the UG-program from Example 3. We recall that $Appx_2(P) = \{e\}$. To compute $Appx_3(P)$, we note that by iterating the operator $\gamma_w^P$ we get the following sequence:

$$Z_0 = \emptyset \mapsto Z_1 = \{a, b, c, d, e, \mathbf{not}(e)\} \mapsto Z_2 = \{e\} \mapsto$$

$$Z_3 = \{a, b, c, e, \mathbf{not}(e)\} \mapsto Z_4 = \{a, c, e\} \mapsto Z_5 = \{a, c, e\} \dots .$$

Hence, $Appx_3(P) = \{a, \mathbf{not}(b), c, \mathbf{not}(d), e\}$ and $Appx_2(P)$ is a *proper* subset of $Appx_3(P)$. $\square$

Finally, we show that if $Appx_3(P)$ is defined and complete then $P$ has a unique answer set.

**Corollary 3.** *Let $P$ be a UG-program such that $Appx_3(P)$ is defined and complete. Then $Appx_3(P) \cap U$ is an answer set of $P$ and $P$ has no other answer sets.*

## 6 Corollaries for the case of revision programs

Revision programming [MT98] is a formalism for describing and enforcing constraints on databases. The main concepts in the formalism are an initial database, a revision program, and justified revisions.

Expressions of the form **in**$(a)$ and **out**$(a)$ ($a \in U$) are *revision literals*. Intuitively, **in**$(a)$ (respectively, **out**$(a)$) means that atom $a$ is in (respectively, is not in) a database.

A *revision program* consists of rules $\alpha \leftarrow \alpha_1, \dots, \alpha_n$, where $\alpha$, $\alpha_i, \dots, \alpha_n$ are revision literals. Given a revision program $P$ and an initial database $I$, [MT98] defined *$P$-justified revisions* of $I$ to represent revisions that satisfy the constraints of $P$, are "grounded" in $P$ and $I$, and differ minimally from the initial database.

As we mentioned earlier, unitary general programs are equivalent to revision programs. The equivalence is established by the so called *shifting theorem* [MPT99], which allows us to reduce any pair $(P, I)$, where $P$ is a revision program and $I$ is an initial database, to a unitary general program so that $P$-justified revisions of $I$ correspond to answer sets of the unitary general program. Consequently, all results of our paper imply results about approximations of justified revisions. Formal descriptions of $Appx_1$, $Appx_2$, and $Appx_3$ for revision programs can be found in [Piv05]. Approximations $Appx_1$ and $Appx_2$ for revision programs were originally described in [Piv01].

**Acknowledgments.** Inna Pivkina was supported by the NSF-funded ADVANCE Institutional Transformation Program at New Mexico State University, Grant No. 0123690, and NMSU College of Arts and Sciences Research Center Grant No. 01-3-43891. The other two authors were supported by the NSF Grants No. 0097278 and 0325063.

## References


[BTK93] A. Bondarenko, F. Toni, and R.A. Kowalski. An assumption-based framework for non-monotonic reasoning. In A. Nerode and L. Pereira, editors, *Logic programming and non-monotonic reasoning (Lisbon, 1993)*, pages 171–189, Cambridge, MA, 1993. MIT Press.

[GL88] M. Gelfond and V. Lifschitz. The stable semantics for logic programs. In *Proceedings of the 5th International Conference on Logic Programming*, pages 1070–1080. MIT Press, 1988.

[GL91] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.

[Lif96] V. Lifschitz. Foundations of logic programming. In *Principles of Knowledge Representation*, pages 69–127. CSLI Publications, 1996.

[LW92] V. Lifschitz and T.Y.C. Woo. Answer sets in general nonmonotonic reasoning. In *Proceedings of the 3rd international conference on principles of knowledge representation and reasoning, KR '92*, pages 603–614, San Mateo, CA, 1992. Morgan Kaufmann.

[MPT99] V. W. Marek, I. Pivkina, and M. Truszczyński. Revision programming = logic programming + integrity constraints. In G. Gottlob, E. Grandjean, and K. Seyr, editors, *Computer Science Logic, 12th International Workshop, CSL'98*, volume 1584 of *Lecture Notes in Computer Science*, pages 73–89. Springer, 1999.

[MPT02] V. W. Marek, I. Pivkina, and M. Truszczyński. Annotated revision programs. *Artificial Intelligence Journal*, 138:149–180, 2002.

[MT98] W. Marek and M. Truszczyński. Revision programming. *Theoretical Computer Science*, 190(2):241–277, 1998.

[Piv01] I. Pivkina. Revision programming: a knowledge representation formalism. PhD dissertation, University of Kentucky, 2001.

[Piv05] I. Pivkina. Defining well-founded semantics for revision programming Technical Report NMSU-CS-2005-001, New Mexico State University, Computer Science Department, 2005.

[PT95] T.C. Przymusiński and H. Turner. Update by means of inference rules. In *Logic programming and nonmonotonic reasoning (Lexington, KY, 1995)*, volume 928 of *Lecture Notes in Computer Science*, pages 156–174, Berlin, 1995. Springer.

[SNS02] P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138:181–234, 2002.

[SNV95] V.S. Subrahmanian, D. Nau, and C. Vago. WFS + branch bound = stable models. *IEEE Transactions on Knowledge and Data Engineering*, 7:362–377, 1995.

[Van93] A. Van Gelder. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences*, 47(1):185–221, 1993.

[VRS88] A. Van Gelder, K.A. Ross, and J.S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *ACM Symposium on Principles of Database Systems*, pages 221–230, 1988.