Defining Well-Founded Semantics for Revision Programming

Inna V. Pivkina

Department of Computer Science, New Mexico State University
P.O. Box 30001, MSC CS, Las Cruces, NM 88003, USA

Abstract

Revision programming is a formalism to describe and enforce constraints on belief sets, databases, and more generally, on arbitrary sets. In this paper we describe four approaches to defining well-founded semantics of revision programming and approximating justified revisions. The first two approaches are obtained via embeddings of revision programs into logic programs. The other two approaches are specific to revision programming and provide approximations for justified revisions.

1 Introduction

Revision programming is a formalism to describe and enforce constraints on belief sets, databases, and more generally, on arbitrary sets. Revision programming formalism was introduced by Marek and Truszczyński in [MT95, MT98]. The main concepts in the formalism are initial database, revision program, and justified revisions. Initial database is a set of atoms. It represents initial state of a belief set or a database. Constraints are represented by sets of revision rules (revision programs). Revision rules could be quite complex and are usually in a form of conditions (for instance, if these elements are present and those elements are absent, then this element must be absent). In addition to being a logical constraint, a revision rule specify a preferred way to satisfy the constraint. Justified revisions semantics assigns to any database a set (possibly empty) of revisions. Each revision satisfies the constraints, and all deletions and additions of elements in a transition from initial database to the revision are derived from revision rules.

Revision programming formalism is closely related to logic programming with stable model semantics. First, it was shown that logic programs with stable model semantics can be represented as revision programs with justified revision semantics ([MT98]). Then, Przymusinski and Turner proposed an embedding of revision programs into logic programs which involved an explicit representation of an initial database as part of the logic program ([PT97]). Later, an embedding of revision programs into logic programs which did not increase the size of the program was presented in ([MPT99]). The embedding is based on a so called Shifting Theorem ([MPT99]), which reflects the symmetry between **in**s and **out**s in revision programming.

Justified revisions semantics for revision programming has some drawbacks. First, an initial database may have none, one, or several justified revisions. This does not fit into a standard paradigm where a single "intended" model is desired. Second, deciding whether justified revisions exist is NP-complete problem.

Similar issues apply to logic programming with stable model semantics. One of the proposals to address the issue for logic programming was made by van Gelder, Ross, and Schlipf ([VRS88], [VRS91]). They assigned to an arbitrary logic program a single intended 3-valued model, a *well-founded model*. In this model, any logic program partitions the set of atoms into three groups: the well-founded atoms, the unfounded atoms, and the rest. An important feature of the well-founded semantics is that it is computable in polynomial time and, in some cases even in linear time ([BSJ95]).

In this paper we propose four approaches to defining well-founded semantics for revision programs and approximating justified revisions. First we consider two definitions of well-founded semantics which can be obtained via the two embeddings of revision programs into logic programs (Przymusinski–Turner translation and embedding via Shifting Theorem). We show that they are, in general, not comparable. Then, we propose a definition which is specific to revision programming. This definition allows us to specify well-founded revision literals if justified revisions exist. However, if there are no justified revisions, this approach may just tell us that there are no justified revisions and no well-founded revision literals are defined in such a case. This approach provides a good approximations for justified revisions since all well-founded revision literals are satisfied by justified revisions. The last approach also provides approximations for justified revisions or allows to conclude that they do not exist.

This work was started in [Piv01].

2 Preliminaries

In this section we give formal definition of revision programs with justified revision semantic and some of their properties as presented in [MT98, MT95].

Elements of some finite universe U are called *atoms*. Subsets of U are called *databases*. Expressions of the form $\mathbf{in}(a)$ or $\mathbf{out}(a)$, where a is an atom, are called *revision literals*. Revision literals will be denoted by greek letters α, etc. For a revision literal $\mathbf{in}(a)$, its *dual* is the revision literal $\mathbf{out}(a)$. Similarly, the *dual* of $\mathbf{out}(a)$ is $\mathbf{in}(a)$. The dual of a revision literal α is denoted by α^D.

For any set of atoms $B \subseteq U$, we define

$$B^c = \{\mathbf{in}(a) : a \in B\} \cup \{\mathbf{out}(a) : a \notin B\}.$$

We can think of B^c as a complete representation of B since for every atom $a \in U$, B^c shows whether a is in B or not.

For any set of literals L, we define

$$L^+ = \{a \in U \colon \mathbf{in}(a) \in L\}, \quad L^- = \{a \in U \colon \mathbf{out}(a) \in L\}.$$

A set of revision literals L is *coherent* if it does not contain a pair of dual literals, that is, $L^+ \cap L^- = \emptyset$. Given a database I and a coherent set of revision literals L, we define

$$I \oplus L = (I \setminus L^-) \cup L^+.$$

Notice that if L is coherent, then $(I \setminus L^-) \cup L^+ = (I \cup L^+) \setminus L^-$.

A *revision rule* is an expression of one of the following two types:

$$\mathbf{in}(a) \leftarrow \mathbf{in}(a_1), \ldots, \mathbf{in}(a_m), \mathbf{out}(b_1), \ldots, \mathbf{out}(b_n) \tag{1}$$

or

$$\mathbf{out}(a) \leftarrow \mathbf{in}(a_1), \ldots, \mathbf{in}(a_m), \mathbf{out}(b_1), \ldots, \mathbf{out}(b_n), \tag{2}$$

where a, a_i and b_i are atoms. Rules of the first type are called *in-rules* and rules of the second type are called *out-rules*. Revision rules have a declarative interpretation as constraints on databases. For instance, an in-rule (1) imposes on a database the following condition: a is *in* the database, or at least one a_i, $1 \leq i \leq m$, is *not* in the database, or at least one b_j, $1 \leq j \leq n$, is *in* the database.

Revision rules also have a computational (imperative) interpretation that expresses a preferred way to enforce a constraint. Namely, assume that all data items a_i, $1 \leq i \leq m$, belong to the current database, say I, and none of the data items b_j, $1 \leq j \leq n$, belongs to I. Then, to enforce the constraint (1), the item a must be added to the database (removed from it, in the case of the constraint (2)), rather than some item a_i removed or some item b_j added.

If a revision rule r is of the type (1), then the *head* of r, denoted *head*(r), is the revision literal $\mathbf{in}(a)$. If r is of the type (2), then the *head* of r, denoted *head*(r), is the revision literal *out*(a). If r is a revision rule of the type (1) or (2), then *body* of r is the set *body*$(r) = \{\mathbf{in}(a_1), \ldots, \mathbf{in}(a_m), \mathbf{out}(b_1), \ldots, \mathbf{out}(b_n)\}$.

A *revision program* is a collection of revision rules.

A set of atoms $B \subseteq U$ is a *model* of (or *satisfies*) a revision literal $\mathbf{in}(a)$ (resp., $\mathbf{out}(a)$), if $a \in B$ (resp., $a \notin B$). A set of atoms B is a *model* of (or *satisfies*) a revision rule r if either B is not a model of at least one revision literal from the body of r, or B is a model of *head*(r). A set of atoms B is a *model* of (or *satisfies*) a revision program P if B is a model of every rule in P.

Let P be a revision program. The *necessary change* of P, $NC(P)$, is the least model of P, when treated as a Horn program built of independent propositional atoms of the form $\mathbf{in}(a)$ and $\mathbf{out}(b)$. The necessary change describes all insertions and deletions that are enforced by the program, independently of the initial database.

The collection of all revision literals describing the elements that do not change their status in the transition from a database I to a database R is called the *inertia set* for I and R, and is defined as follows:

$$I(I, R) = \{\mathbf{in}(a) \colon a \in I \cap R\} \cup \{\mathbf{out}(a) \colon a \notin I \cup R\}.$$

By the *reduct* of P with respect to a pair of databases (I, R), denoted by $P_{I,R}$, we mean the revision program obtained from P by eliminating from the body of each rule in P

all revision literals in $I(I,R)$. The necessary change of the program $P_{I,R}$ provides a justification for some insertions and deletions. These are exactly the changes that are *a posteriori* justified by P in the context of the initial database I and a putative revised database R.

Definition 1 *A database R is a P-*justified revision *of I if the necessary change of $P_{I,R}$ is coherent and if $R = I \oplus NC(P_{I,R})$.* $\triangle$

There is an alternative definition of P-justified revisions also presented in [MT98]. It is based on a different notion of a reduct – a counterpart of the Gelfond-Lifschitz reduct in logic programming ([GL88]).

Definition 2 ([GL88]) *Let P be a logic program. The reduct of P relative to M, P^M, is obtained from P by*

1. *removing all clauses which contain "not q" such that q is true in M,*
2. *deleting all negative premises "not q" (for all $q \in U$) from the remaining clauses.* $\triangle$

Definition 3 ([MT98]) *Let P be a revision program, and let I and R be two databases. The* GL-reduct *of P with respect to (I,R) (denoted $P_R|I$) is obtained from P by*

1. *eliminating from P every rule whose body is not satisfied by R (the resulting program is denoted by P_R),*
2. *eliminating each literal that is satisfied by I from the body of each rule in P_R.* $\triangle$

Each of the reducts, $P_{I,R}$ and $P_R|I$, can be used to define the notion of P-justified revision, as the following theorem shows.

Theorem 1 ([MT98]) *Let P be a revision program and let I and R be two databases. The following two conditions are equivalent:*

- *$NC(P_{I,R})$ is coherent and $R = I \oplus NC(P_{I,R})$,*
- *$NC(P_R|I)$ is coherent and $R = I \oplus NC(P_R|I)$.*

For justified revisions, the necessary changes of both reducts are the same:

Theorem 2 ([MT98]) *Let P be a revision program and R be a P-justified revision of I. Then, $NC(P_{I,R}) = NC(P_R|I) = head(P_R)$.*

In the paper we will frequently use the following characterizations also given in [MT98].

Theorem 3 ([MT98]) *The following conditions are equivalent:*
1. A database R is a P-justified revision of a database I,
2. $NC(P \cup \{\alpha \leftarrow : \alpha \in I(I,R)\}) = R^c$;
3. $NC(P_{I,R}) \cup I(I,R) = R^c$.

The following lemma shows that a least model of a Horn program does not change if we add to the program rules which are satisfied by the least model.

Lemma 1 *Let M be a least model of a Horn program P and a model of a Horn program P'. Then, M is the least model of $P \cup P'$.*

2.1 Well-founded semantics.

The stable model semantics does not fit into a standard paradigm of logic programming languages. While standard approaches assign to a logic program a single "intended" model, stable model semantics assigns to a program a family (possibly empty) of "intended" models.

Proposals to address the issue for stable model semantics were of two types. Proposals of the first type attempt to salvage the notion of a single intended model at a cost of narrowing down the class of programs or weakening the semantics. Apt, Blair and Walker [ABW88] introduced the notion of *stratification*, a syntactic restriction on logic programs with negation. They assigned to each stratified program a single intended model, a perfect model.

Proposal of the second type for stable model semantics was made by van Gelder, Ross and Schlipf ([VRS88], [VRS91]). They assigned to an arbitrary logic program a single intended 3-valued model, a *well-founded model*. In this model, any logic program partitions the set of atoms into three groups: the well-founded atoms, the unfounded atoms, and the rest.

Definition 4 *For any program P, the operator γ_P is defined by the equation*

$$\gamma_P(X) = \mathit{Least}(P^X). \qquad \triangle$$

The operator γ_P is anti-monotone.

Proposition 1 *If T is an anti-monotone operator then*

1. *T^2 is monotone;*
2. *$T(\mathit{lfp}(T^2)) = \mathit{gfp}(T^2))$ and $T(\mathit{gfp}(T^2)) = \mathit{lfp}(T^2))$;*
3. *for any fixpoint X of T, $\mathit{lfp}(T^2) \subseteq X \subseteq \mathit{gfp}(T^2)$.*

By Proposition 1, the operator γ_P^2 is monotone, and has a least fixpoint and a greatest fixpoint.

The well-founded semantics of a logic program is defined as follows.

Definition 5 *The atoms that belong to the least fixpoint of γ_P^2 are* well-founded *relative to P. The atoms that do not belong to the greatest fixpoint of γ_P^2 are* unfounded *relative to P. The remaining atoms are called* unknown. $\triangle$

Proposition 2 ([Lif96]) *Any stable model for a logic program P*

- *includes all atoms that are well-founded relative to P, and*
- *includes no atoms unfounded relative to P.*

Theorem 4 ([VRS88]) *If every atom is either well-founded or unfounded relative to a logic program P then the set of well-founded atoms is the only stable model for P.*

An important feature of the well-founded semantics is that it is computable in polynomial time, in some cases even in linear time ([BSJ95]).

In revision programming a database may admit none, one or many revisions. What if we want to have a single "intended" model?

In the beginning of Section 2.1 we mentioned two types of approaches to address the same issue in logic programming. Because of the similarities between revision programming and logic programming the same approaches can be used for revision programming. Marek and Truszczyński in [MT98] presented a solution of the first type: they restrict class of revision programs to those that have the desired property: to every initial database they assign a unique justified revision.

Definition 6 ([MT98]) *A revision program is* safe *if for every literal* $\alpha \in head(P)$, $\alpha^D \notin var(P)$. △

Definition 7 ([MT98]) *A revision program* P *is* stratified *if there exists* $\{P_t\}_{0<t<n}$, *a partition of* P, *such that for every* $0 < t < n$:

1. P_t *is safe, and*
2. *if* $\alpha \in head(P_t)$ *then* $\alpha, \alpha^D \notin \cup_{q<t} var(P_q)$. △

Safe and stratified revision programs have exactly one justified revision for every initial database.

A solution of the second type is a well-founded semantics for revision programs, where each revision problem (revision program and initial database) is assigned a single 3-valued model on the set of revision literals. In this paper we discuss different ways of defining well-founded semantics for revision programs.

3 Definitions induced by embeddings into logic programming.

Revision programs and logic programs are closely related, as we discussed in [MPT99]. Therefore, given a revision problem, we can translate it into a logic program, find the well-founded semantics for the logic program, and declare the result to be a well-founded model of the revision program. The two ways of embedding of revision programs into logic programs presented in [PT97] and [MPT99] give rise to two definitions of well-founded semantics for revision programs. In this section we present these two definitions and show that they are indeed different.

3.1 A definition obtained via the Przymusinski–Turner translation

In this section we use the Przymusinski–Turner embedding of revision programs into logic programs described in [PT97] to compute well-founded semantics for revision programs.

The Przymusinski–Turner translation of a revision program RP and an initial database B_I into a logic program is as follows.

Definition 8 ([PT97]) The translation of the revision program RP and the initial database B_I into a logic program *is defined as the logic program* $\mathcal{P}(RP, B_I) = P_I \cup P_N \cup RP$ *over* $\mathcal{K}$ *consisting of the following three subprograms:*

Initial Knowledge Rules P_I*: All atoms* $q \in B_I$ *are initially in and all atoms* $s \notin B_I$ *are initially out:*

$$P_I = \{\mathbf{in}_I(q) \leftarrow : \ q \in B_I\} \cup \{\mathbf{out}_I(s) \leftarrow : \ s \notin B_I\};$$

Inertia Rules P_N*: If* q *was initially in (respectively, out) then after revision it remains in (respectively, out) unless it was forced out (respectively, in):*

$$P_N = \{\mathbf{in}(q) \leftarrow \mathbf{in}_I(q), \mathit{not}\ \mathbf{out}(q) \ \ ; \ \ \mathbf{out}(q) \leftarrow \mathbf{out}_I(q), \mathit{not}\ \mathbf{in}(q) \ : \ q \in U\};$$

Revision Rules RP*: All the revision rules that belong to the original revision program* RP*.* △

The translation $\mathcal{P}(RP, B_I)$ contains the original revision program RP, a complete explicit representation of the initial database as a set of facts, and inertia rules that specify that atoms do not change their status unless they are forced to.

The translation defines an embedding of revision programming into logic programming with stable model semantics.

Theorem 5 (Przymusinski and Turner [PT97]) *Let* RP *be a revision program and* B_I *be an initial database. There is a one-to-one correspondence between* RP*-justified revisions of* B_I *and coherent stable models of its translation* $\mathcal{P}(RP, B_I)$ *into a logic program.*

More precisely, to every RP*-justified revision* B_R *of* B_I *there corresponds a unique coherent stable model* M *of* $\mathcal{P}(RP, B_I)$ *such that:*

$$B_R = \{q : \mathbf{in}(q) \in M\}, \qquad U \setminus B_R = \{q : \mathbf{out}(q) \in M\},$$

and, conversely, for each coherent stable model M *of* $\mathcal{P}(RP, B_I)$ *the set* $B_R = \{q : \mathbf{in}(q) \in M\}$ *is an* RP*-justified revision of* B_I*.*

Given a revision program P and a database I, we compute the translation of P and I into a logic program, $\mathcal{P}(P, I)$ (as specified in Definition 8). Recall that $\mathcal{P}(P, I)$ is a logic program over a propositional language $\mathcal{K}$ with the set of propositional letters $\{\mathbf{in}(q) : q \in U\} \cup \{\mathbf{out}(q) : q \in U\} \cup \{\mathbf{in}_I(q) : q \in U\} \cup \{\mathbf{out}_I(q) : q \in U\}$. To find a well-founded semantics for $\mathcal{P}(P, I)$ we compute a least fixpoint and a greatest fixpoint of the operator $\gamma^2_{\mathcal{P}(P,I)}$. Then, for every revision literal $l \in \{\mathbf{in}(q) : q \in U\} \cup \{\mathbf{out}(q) : q \in U\}$ we can tell whether it is well-founded, unfounded or unknown relative to the logic program $\mathcal{P}(P, I)$. Therefore, we can define a well-founded semantics of revision programs (which we denote WFS^{PT}) as follows.

Definition 9 (WFS^{PT}) *Let P be a revision program. Let I be an initial database. The revision literals that belong to the least fixpoint of $\gamma^2_{\mathcal{P}(P,I)}$ are* well-founded^{PT} *relative to P and I. The revision literals that do not belong to the greatest fixpoint of $\gamma^2_{\mathcal{P}(P,I)}$ are* unfounded^{PT} *relative to P and I. The remaining revision literals are called* unknown^{PT}. $\triangle$

Let us illustrate the definition by an example.

Example 1 *Let $I = \emptyset$. Consider*

$$\begin{array}{rrcl} P: & \mathbf{out}(a) & \leftarrow & \mathbf{out}(b) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \\ & \mathbf{in}(a) & \leftarrow & \end{array}$$

The logic program $\mathcal{P} = \mathcal{P}(P, I)$ is the following.

$$\begin{array}{rrcl} \mathcal{P}: & \mathbf{out}_I(a) & \leftarrow & \\ & \mathbf{out}_I(b) & \leftarrow & \\ & \mathbf{in}(a) & \leftarrow & \mathbf{in}_I(a), \mathit{not}\ \mathbf{out}(a) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{in}_I(b), \mathit{not}\ \mathbf{out}(b) \\ & \mathbf{out}(a) & \leftarrow & \mathbf{out}_I(a), \mathit{not}\ \mathbf{in}(a) \\ & \mathbf{out}(b) & \leftarrow & \mathbf{out}_I(b), \mathit{not}\ \mathbf{in}(b) \\ & \mathbf{out}(a) & \leftarrow & \mathbf{out}(b) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \\ & \mathbf{in}(a) & \leftarrow & \end{array}$$

Thus, $\mathit{lfp}(\gamma^2_{\mathcal{P}}) = \{\mathbf{out}_I(a), \mathbf{out}_I(b), \mathbf{in}(a)\}$, $\mathit{gfp}(\gamma^2_{\mathcal{P}}) = \{\mathbf{out}_I(a), \mathbf{out}_I(b), \mathbf{in}(a), \mathbf{out}(a), \mathbf{in}(b), \mathbf{out}(b)\}$. Therefore, the only well-founded^{PT} revision literal is $\mathbf{in}(a)$. There are no unfounded^{PT} revision literals. Revision literals $\mathbf{out}(a)$, $\mathbf{in}(b)$, and $\mathbf{out}(b)$ are unknown^{PT}.

The following result shows that the definition agrees with intuitions. Namely, that well-founded revision literals must be satisfied by all justified revisions, and no justified revision may satisfy an unfounded revision literal.

Theorem 6 *Let P be a revision program. Let I be an initial database. Then, any P-justified revision of I*

- *satisfies all revision literals that are well-founded^{PT} relative to P and I, and*
- *satisfies no revision literals unfounded^{PT} relative to P and I.*

Proof.
Let R be a P-justified revision of I. Let l be a revision literal. By Theorem 5, there exists a unique coherent stable model M of $\mathcal{P}(P, I)$ such that $R = \{q : \mathbf{in}(q) \in M\}$.

If l is well-founded^{PT} relative to P and I, then, by definition, it belongs to the least fixpoint of $\gamma^2_{\mathcal{P}(P,I)}$. Hence, l is well-founded relative to $\mathcal{P}(P,I)$. Thus, by Proposition 2, $l \in M$. Since M is coherent, this implies that l is satisfied by R.

Similarly, if l is unfounded^{PT} relative to P and I, then it does not belong to the greatest fixpoint of $\gamma^2_{\mathcal{P}(P,I)}$. Hence, l is unfounded relative to $\mathcal{P}(P,I)$. Thus, by Proposition 2, $l \notin M$. Consequently, l is not satisfied by R. $\square$

Corollary 1 *If for some $a \in U$ both* $\mathbf{in}(a)$ *and* $\mathbf{out}(a)$ *are well-founded^{PT} relative to P and I, then there are no P-justified revisions of I.*

Proof.
By the theorem, if a justified revision exists it must satisfy both $\mathbf{in}(a)$ and $\mathbf{out}(a)$. No database has such a property. $\square$

WFS^{PT} assigns to every revision literal a value (well-founded, unfounded or unknown). We can think of it as assigning to each atom $a \in U$ a pair $\langle \alpha, \beta \rangle$, where $\alpha, \beta \in \{$well-founded, unfounded, unknown$\}$, and α (resp. β) is the value of $\mathbf{in}(a)$ (resp. $\mathbf{out}(a)$). The following theorem shows that not all pairs $\langle \alpha, \beta \rangle$ are valid assignments under WFS^{PT}.

Theorem 7 *Let P be a revision program. Let I be a database. Then, for any atom $a \in U$, if a revision literal* $\mathbf{in}(a)$ *(respectively, a revision literal* $\mathbf{out}(a)$*) is unfounded^{PT} then the revision literal* $\mathbf{out}(a)$ *(respectively,* $\mathbf{in}(a)$*) is well-founded^{PT}.*

Proof.
Assume that $\mathbf{in}(a)$ is unfounded^{PT} relative to P and I. That means that $\mathbf{in}(a)$ does not belong to the greatest fixpoint of $\gamma^2_{\mathcal{P}(P,I)}$. Let us show that $\mathbf{out}(a)$ belongs to the least fixpoint of $\gamma^2_{\mathcal{P}(P,I)}$.
Case 1: $a \in I$. Since $\mathbf{in}_I(a) \leftarrow$ is in $\mathcal{P}(P,I)$, $\mathbf{in}_I(a)$ is in all fixpoints of $\gamma^2_{\mathcal{P}(P,I)}$.

The logic program $\mathcal{P}(P,I)$ contains the rule

$$\mathbf{in}(a) \leftarrow \mathbf{in}_I(a), \mathit{not}\ \mathbf{out}(a).$$

Assume that $\mathbf{out}(a)$ is not in the least fixpoint. Then, $\mathbf{in}(a)$ must be in the greatest fixpoint. But this contradicts the assumption that $\mathbf{in}(a)$ is unfounded^{PT} relative to P and I. Consequently, $\mathbf{out}(a)$ must be in the least fixpoint.
Case 2: $a \notin I$. Then, $\mathbf{out}_I(a)$ is in all fixpoints of $\gamma^2_{\mathcal{P}(P,I)}$.

The logic program $\mathcal{P}(P,I)$ contains the rule

$$\mathbf{out}(a) \leftarrow \mathbf{out}_I(a), \mathit{not}\ \mathbf{in}(a).$$

By our assumption, $\mathbf{in}(a)$ is not in the greatest fixpoint. Therefore, $\mathbf{out}(a)$ must be in the least fixpoint.

We showed that if $\mathbf{in}(a)$ is unfounded^{PT} relative to P and I, then $\mathbf{out}(a)$ is well-founded^{PT}.

Similarly, we can show that if $\mathbf{out}(a)$ is unfounded^{PT} relative to P and I, then $\mathbf{in}(a)$ is well-founded^{PT}. $\square$

Table 1: Possible combinations of values for literals $\mathbf{in}(a)$ and $\mathbf{out}(a)$ under WFS^{PT}.

in(a)	**out**(a)	revision program P
T	T	$\{\mathbf{in}(a) \leftarrow ;\quad \mathbf{out}(a) \leftarrow \}$
T	U	$\{\mathbf{in}(a) \leftarrow ;\quad \mathbf{out}(a) \leftarrow \mathbf{out}(b)\ ;\quad \mathbf{in}(b) \leftarrow \mathbf{out}(b)\ \}$
T	F	$\{\mathbf{in}(a) \leftarrow \}$
U	T	$\{\mathbf{out}(a) \leftarrow ;\quad \mathbf{in}(a) \leftarrow \mathbf{out}(b)\ ;\quad \mathbf{in}(b) \leftarrow \mathbf{out}(b)\ \}$
U	U	$\{\mathbf{in}(a) \leftarrow \mathbf{out}(a)\ \}$
U	F	Impossible by Theorem 7
F	T	$\{\}$
F	U	Impossible by Theorem 7
F	F	Impossible by Theorem 7

Theorem 7 implies that there are no more than six pairs of values from {well-founded, unfounded, unknown} which an atom can have under WFS^{PT}. If we denote "well-founded" by T, "unfounded" by F, and "unknown" by U, then these pairs are $\langle$T, T$\rangle$, $\langle$T, U$\rangle$, $\langle$T, F$\rangle$, $\langle$U, T$\rangle$, $\langle$U, U$\rangle$, $\langle$F, T$\rangle$. Every one of these six pairs can be achieved. Table 1 provides examples of six revision programs which with an empty initial database give these six different assignments for atom a under WFS^{PT}. Therefore, we can think of WFS^{PT} as a three-valued model on a set of revision literals, or as a six-valued model on a set of atoms.

3.2 A definition obtained via Shifting Theorem

Another translation of revision programs into logic programs was described in [MPT99]. It is based on the Shifting Theorem. Similarly to the previous section, we take a revision program, translate it into a logic program, find the well-founded semantics of the logic program and declare the corresponding revision literals to be well-founded, unfounded or unknown. The main difference with the Przymusinski–Turner translation is that the Shifting Theorem allows us to translate revision problems into logic programs with constraints. Therefore, in addition to the well-founded semantics of the logic program part of the translation, we also use the constraints and the principle of inertia to define a well-founded semantics for revision programs.

We start by defining well-founded semantics in terms of revision programs only without explicitly mentioning translation into logic programs. Later we show that this definition is equivalent to the one obtained by translation into logic program and applying constraints to the result.

Let P be a revision program. Let I be an initial database.

Definition 10 *For a set of revision literals X define the* reduct *of (P, I) relative to X (denoted by $(P, I)^X$) to be the revision program obtained from P by*

1. *removing every rule $r \in P$ such that $body(r) \cap \{l^D : l \in X \setminus I^c\} \neq \emptyset$,*

2. deleting from the body of each remaining rule any revision literal that is in I^c. △

Definition 11 *For a revision program P and a database I, the operator $\gamma_{P,I}$ from sets of revision literals to sets of revision literals is defined by the equation*

$$\gamma_{P,I}(X) = NC((P,I)^X),$$

where $NC((P,I)^X)$ is the necessary change of $(P,I)^X$. △

Proposition 3 *Let P be a revision program, let I be a database. Let X be a set of revision literals. Let $I' \subseteq I^c$. Then, $(P,I)^X = (P,I)^{X\setminus I'}$ and $\gamma_{P,I}(X) = \gamma_{P,I}(X \setminus I')$.*

Proof.
Since $I' \subseteq I^c$, we have that $X \setminus I^c = (X \setminus I') \setminus I^c$. Thus, according to the definition of the reduct $(P,I)^X$, we have that $(P,I)^X = (P,I)^{X\setminus I'}$. Therefore, $\gamma_{P,I}(X) = NC((P,I)^X) = NC((P,I)^{X\setminus I'}) = \gamma_{P,I}(X \setminus I')$. □

Proposition 4 *For any revision program P and initial database I, the operator $\gamma_{P,I}$ is anti-monotone.*

Proof.
Let X_1 and X_2 be sets of revision literals, $X_1 \subseteq X_2$. Then, $X_1 \setminus I^c \subseteq X_2 \setminus I^c$. Hence, $\{l^D : l \in X_1 \setminus I^c\} \subseteq \{l^D : l \in X_2 \setminus I^c\}$. Therefore, $(P,I)^{X_2} \subseteq (P,I)^{X_1}$. Consequently, $\gamma_{P,I}(X_2) \subseteq \gamma_{P,I}(X_1)$. □

Since $\gamma_{P,I}$ is anti-monotone, the operator $\gamma^2_{P,I}$ is monotone. By the Knaster-Tarski Theorem, $\gamma^2_{P,I}$ has a least fixpoint and a greatest fixpoint.

Definition 12 (WFS^{Sh}) *Let P be a revision program. Let I be a database. The revision literals that belong to the $lfp(\gamma^2_{P,I}) \cup \{l \in I^c : l^D \notin gfp(\gamma^2_{P,I})\}$ are* well-founded^{Sh} *relative to (P,I). The revision literals that do not belong to the $gfp(\gamma^2_{P,I}) \cup I^c$ are* unfounded^{Sh} *relative to (P,I). The remaining revision literals are* unknown^{Sh} *relative to (P,I).* △

The following example illustrates the definition.

Example 2 *Let $I = \emptyset$. Consider*

$$\begin{array}{rrcl} P: & \mathbf{out}(a) & \leftarrow & \\ & \mathbf{out}(b) & \leftarrow & \\ & \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \\ & \mathbf{in}(a) & \leftarrow & \mathbf{out}(b) \end{array}$$

Then, $lfp(\gamma^2_{P,I}) = \{\mathbf{out}(a), \mathbf{out}(b)\}$, and $gft(\gamma^2_{P,I}) = \{\mathbf{out}(a),\ \mathbf{out}(b),\ \mathbf{in}(a),\ \mathbf{in}(b)\}$. Therefore, revision literals $\mathbf{out}(a)$, $\mathbf{out}(b)$ are well-founded^{Sh}, no revision literals are unfounded^{Sh}, revision literals $\mathbf{in}(a)$, $\mathbf{in}(b)$ are unknown^{Sh}.

Theorem 8 *Let Π be a logic program. Let W be a set of well-founded atoms relative to Π. Let N be a set of unfounded atoms relative to Π. Then, $\{\mathbf{in}(a) : a \in W\} \cup \{\mathbf{out}(a) : a \in N\}$ is the set of well-founded^{Sh} revision literals relative to $(rp(\Pi), \emptyset)$, and $\{\mathbf{in}(a) : a \in N\}$ is the set of unfounded^{Sh} revision literals.*

Proof.
For a rule $r \in rp(\Pi)$, $body(r) \cap \{l^D : l \in X \setminus \emptyset^c\} \neq \emptyset$ if and only if the body of r contains a revision literal $\mathbf{out}(b)$ for some $\mathbf{in}(b) \in X$. This is the case only when the logic program clause $lp(r)$ has the literal *not* b in the body for $b \in X^+$, where $X^+ = \{a \in U : \mathbf{in}(a) \in X\}$. Therefore, the first step in the definition of the reduct $(rp(\Pi), \emptyset)^X$ corresponds to the first step of the definition of the reduct Π^{X^+}. That is, $r \in rp(\Pi)$ is deleted on the first step of computing $(rp(\Pi), \emptyset)^X$ if and only if the corresponding logic program clause $lp(r) \in \Pi$ is deleted on the first step of computing Π^{X^+}.

In the second step of the definition of $(rp(\Pi), \emptyset)^X$ all revision literals of the form $\mathbf{out}(a)$ $(a \in U)$ are deleted from the remaining rules. It corresponds to deleting all atoms with negation as failure from the bodies of the remaining logic program clauses. Consequently, $rp(\Pi^{X^+}) = (rp(\Pi), \emptyset)^X$.

Since the revision program $(rp(\Pi), \emptyset)^X$ has only revision literals of the form $\mathbf{in}(a)$ $(a \in U)$, $\gamma_\Pi(X^+) = \gamma_{rp(\Pi),\emptyset}(X)$ for any set of revision literals X. Therefore, $Y \subseteq U$ is a fixpoint of γ^2_Π if and only if $\{\mathbf{in}(a) : a \in Y\}$ is a fixpoint of $\gamma^2_{rp(\Pi),\emptyset}$.

By definition, well-founded^{Sh} revision literals relative to $(rp(\Pi), \emptyset)$ are

$$lfp(\gamma^2_{rp(\Pi),\emptyset}) \cup \{l \in \emptyset^c : l^D \notin gfp(\gamma^2_{rp(\Pi),\emptyset})\} =$$

$$\{\mathbf{in}(a) : a \in lfp(\gamma^2_\Pi)\} \cup \{\mathbf{out}(a) : \mathbf{in}(a) \notin gfp(\gamma^2_{rp(\Pi),\emptyset})\} =$$

$$\{\mathbf{in}(a) : a \in W\} \cup \{\mathbf{out}(a) : a \notin gfp(\gamma^2_\Pi)\} =$$

$$\{\mathbf{in}(a) : a \in W\} \cup \{\mathbf{out}(a) : a \in N\}.$$

By definition, unfounded^{Sh} revision literals relative to $(rp(\Pi), \emptyset)$ are revision literals that do not belong to the $gfp(\gamma^2_{rp(\Pi),\emptyset}) \cup I^c$. Hence, $\{\mathbf{in}(a) : \mathbf{in}(a) \notin gfp(\gamma^2_{rp(\Pi),\emptyset})\}$ is the set of unfounded^{Sh} revision literals. However,

$$\{\mathbf{in}(a) : \mathbf{in}(a) \notin gfp(\gamma^2_{rp(\Pi),\emptyset})\} =$$

$$\{\mathbf{in}(a) : a \notin gfp(\gamma^2_\Pi)\} = \{\mathbf{in}(a) : a \in N\}.$$

This finishes the proof. □

Lemma 2 *Let P be a revision program. Let I be an initial database. Let R be a P-justified revision of I. Let X be a set of heads of P_R. Then, X is a fixpoint of $\gamma_{P,I}$.*

Proof.
By definition of P-justified revision, $R = I \oplus NC(P_R|I)$. By Theorem 2, $X = NC(P_R|I)$. Thus, $R = I \oplus X$.

Let us compare the reducts $(P, I)^X$ and $P_R|I$. Recall, that $P_R|I$ (Definition 3) is obtained from P by

1. removing every rule $r \in P$ whose body is not satisfied by R;
2. deleting from the body of each remaining rule any revision literal that is in I^c.

Assume that rule $r \in P$ is removed during the first step of computation of the reduct $(P, I)^X$. By definition of the reduct, there exists revision literal l such that $l^D \in \mathit{body}(r)$, $l \in X$, and $l \notin I^c$. From $R = I \oplus X$ and $l \in X$ it follows that $R \models l$. Hence, $R \not\models l^D$. Consequently, the body of rule r is not satisfied by R. Hence, r is removed during the first step of computation of the reduct $P_R|I$. The second steps of computation of the reducts $(P, I)^X$ and $P_R|I$ are the same. Therefore, $P_R|I \subseteq (P, I)^X$.

Assume that rule $r \in P$ is removed during the first step of computation of the reduct $P_R|I$. It means that there exists literal l in the body of r such that $R \not\models l$. Thus, $R \models l^D$. Since $R = I \oplus X$, only the following three cases are possible.

1. $l^D \in X$ and $l^D \in I^c$. Since X is coherent, $l \notin X$. If there are no other literals that cause r to be deleted during the first step of computation of $(P, I)^X$, then the reduct $(P, I)^X$ contains rule r' which is obtained from r by deleting from the body all revision literals that are in I^c. Note, that r' contains l in the body.

2. $l^D \in X$ and $l^D \notin I^c$. Hence, $l^D \in X \setminus I^c$, and $l = (l^D)^D$ is in the body of r. Thus, rule r is deleted during the first step of computation of the reduct $(P, I)^X$.

3. $l^D \notin X$, $l^D \in I^c$, and $l \notin X$. Similarly to case 1, if r is *not* deleted during the first step of computation of $(P, I)^X$, then the reduct $(P, I)^X$ contains rule r' which is obtained from r by deleting from the body all revision literals that are in I^c. Note, that r' contains l in the body.

4. $l^D \in X$ and $l^D \notin I^c$. Hence, $l^D \in X \setminus I^c$, and $l = (l^D)^D$ is in the body of r. Thus, rule r is deleted during the first step of computation of the reduct $(P, I)^X$.

5. $l^D \notin X$, $l^D \in I^c$, and $l \notin X$. Similarly to case 1, if r is *not* deleted during the first step of computation of $(P, I)^X$, then the reduct $(P, I)^X$ contains rule r' which is obtained from r by deleting from the body all revision literals that are in I^c. Note, that r' contains l in the body.

From the above observations it follows that $(P, I)^X = (P_R|I) \cup S$, where S is a set of rules (possibly empty) with the property that every rule $r \in S$ contains in its body a literal l such that $l \notin X$. We have that $X = NC(P_R|I)$. Since all rules in S have literals that are *not* in X in their bodies, $NC((P, I)^X) = NC((P_R|I) \cup S) = NC(P_R|I) = X$. Thus, $\gamma_{P,I}(X) = NC((P, I)^X) = X$. □

Theorem 9 *Let P be a revision program. Let I be an initial database. Then, any P-justified revision of I*

- *satisfies all revision literals that are well-founded^{Sh} relative to P and I, and*
- *satisfies no revision literals unfounded^{Sh} relative to P and I.*

Proof.
By definition of P-justified revision, $R = I \oplus NC(P_R|I)$. Let X be a set of heads of P_R. By Theorem 2, $X = NC(P_R|I)$. Thus, $R = I \oplus X$.

By Lemma 2 X is a fixpoint of $\gamma_{P,I}$. Therefore, X is a fixpoint of $\gamma^2_{P,I}$.

By Proposition 1, $lfp(\gamma^2_{P,I}) \subseteq X \subseteq gfp(\gamma^2_{P,I})$.

Let l be a well-founded^{Sh} revision literal relative to P and I. By definition, $l \in lfp(\gamma^2_{P,I}) \cup \{l \in I^c : l^D \notin gfp(\gamma^2_{P,I})\}$. There are two possible cases.

Case 1: $l \in lfp(\gamma^2_{P,I})$. Then, $l \in X$. Hence, $R = I \oplus X$ implies $R \models l$.

Case 2: $l \in I^c$ and $l^D \notin gfp(\gamma^2_{P,I})$. Then, $l^D \notin X$. Hence, $R = I \oplus X$ implies $R \models l$.

Therefore, R satisfies all well-founded^{Sh} revision literals.

Let l be an unfounded^{Sh} revision literal relative to P and I. By definition, $l \notin gfp(\gamma^2_{P,I}) \cup I^c$. Thus, $l \notin I^c$ and $l \notin X$. In other words, $l^D \in I^c$ and $(l^D)^D \notin X$. This implies that $R \models l^D$. Hence, $R \not\models l$. Consequently, R satisfies no unfounded^{Sh} revision literals. $\square$

In the rest of the section we show that well-founded^{Sh} semantics is equivalent to well-founded semantics obtained by translation into logic program (via Shifting theorem) and applying constraints to the result.

Theorem 10 (Shifting preserves WFS^{Sh}) *Let P be a revision program. Let I, J be databases. Let $W = I \div J$ be symmetric difference of I and J. Then, F, U, N are sets of well-founded^{Sh}, unknown^{Sh}, and unfounded^{Sh} revision literals relative to (P, I), respectively, if and only if $T_W(F)$, $T_W(U)$, $T_W(N)$ are sets of well-founded^{Sh}, unknown^{Sh}, and unfounded^{Sh} revision literals relative to $(T_W(P), J)$, respectively.*

Proof.
The W-transformation can be viewed as a renaming of propositional atoms of the form **in**(a) and **out**(b) which preserves duality. That is, $T_W(\alpha^D) = (T_W(\alpha))^D$. If we apply T_W simultaneously to P, I and any set of revision literals X, and compute $(T_W(P), T_W(I))^{T_W(X)}$, then the result will coincide with $T_W((P, I)^X)$.

When calculating the necessary change, we treat literals as propositional atoms of the form **in**(a) and **out**(b). The W-transformation can be viewed as a renaming of these atoms. If we rename all atoms in the Horn program, find the least model of the obtained program, and then rename the atoms back, we will get the least model of the original program.

In other words,
$$NC((P, I)^X) = T_W(NC(T_W((P, I)^X))).$$

Therefore, for any X, $\gamma_{P,I}(X) = T_W(\gamma_{T_W(P),T_W(I)}(T_W(X)))$. Hence, $lfp(\gamma^2_{P,I}) = T_W(lfp(\gamma^2_{T_W(P),T_W(I)}))$ and $gfp(\gamma^2_{P,I}) = T_W(gfp(\gamma^2_{T_W(P),T_W(I)}))$. This implies the statement of the theorem. $\square$

The following result shows that for a revision program P and a database I, WFS^{Sh} relative to (P, I) can be obtained from WFS^{Sh} relative to (P', I) where $P' = \{r \in P : head(r) \notin \text{I}\}$.

Theorem 11 *Let P be a revision program. Let F, N be sets of well-founded^{Sh} and unfounded^{Sh} revision literals relative to (P, I), respectively. Let $P' = \{r \in P : head(r) \notin I\}$. Let $P'' = P \setminus P'$. Let F', N' be sets of well-founded^{Sh} and unfounded^{Sh} revision literals relative to (P', I), respectively. Then, $F = F' \cup \{\mathbf{out}(a) : \mathbf{out}(a) \leftarrow body \in P''$, and body is satisfied by $F'^{+}\}$ and $N = N'$.*

Proof.
Let X be a set of revision literals. Clearly, $(P, I)^X = (P', I)^X \cup (P'', I)^X$. By the definition of the reduct $(P, I)^X$, bodies of the rules in $(P, I)^X$ contain no revision literals from I^c. The only way to get literals form I^c in $NC((P, I)^X)$ is through rules in $(P'', I)^X$. Therefore, we get that $NC((P, I)^X) = NC((P', I)^X) \cup \{l \in I^c : l \leftarrow body \in (P'', I)^X$, and *body* is satisfied by $NC((P', I)^X)\}$.

By definition,

$$\gamma_{P,I}(X) = NC((P, I)^X).$$

Therefore, we get that for any set X of revision literals,
$\gamma_{P,I}(X) = \gamma_{P',I}(X) \cup \{l \in I^c : l \leftarrow body \in (P'', I)^X$, and *body* is satisfied by $\gamma_{P',I}(X)\}$.

Let us substitute $Y = \gamma_{P,I}(X)$ instead of X in the above equation. We get
$\gamma_{P,I}(Y) = \gamma_{P',I}(Y) \cup \{l \in I^c : l \leftarrow body \in (P'', I)^Y$, and *body* is satisfied by $\gamma_{P',I}(Y)\}$.
Let us consider left-hand side (LHS) and right-hand side (RHS) of the equation.
LHS $= \gamma_{P,I}(Y) = \gamma_{P,I}(\gamma_{P,I}(X)) = \gamma^2_{P,I}(X)$.
Since $Y = \gamma_{P',I}(X) \cup \{l \in I^c : l \leftarrow body \in (P'', I)^X$, and *body* is satisfied by $\gamma_{P',I}(X)\}$, by Proposition 3, we get $\gamma_{P',I}(Y) = \gamma_{P',I}(\gamma_{P',I}(X)) = \gamma^2_{P',I}(X)$. Therefore,
RHS $= \gamma^2_{P',I}(X) \cup \{l \in I^c : l \leftarrow body \in (P'', I)^{\gamma_{P,I}(X)}$, and *body* is satisfied by $\gamma^2_{P',I}(X)\}$.
LHS=RHS implies that
$\gamma^2_{P,I}(X) = \gamma^2_{P',I}(X) \cup \{l \in I^c : l \leftarrow body \in (P'', I)^{\gamma_{P,I}(X)}$, and *body* is satisfied by $\gamma^2_{P',I}(X)\}$.

Let Y be a set of revision literals of the form $\mathbf{in}(a)$. One can show that Y is a fixpoint of $\gamma^2_{P',\emptyset}$ if and only if $Y \cup \{\mathbf{out}(a) : \mathbf{out}(a) \leftarrow body \in (P'', \emptyset)^Y$, and *body* is satisfied by $Y\}$ is a fixpoint of $\gamma^2_{P,\emptyset}$. □

Theorem 12 *Let P be a revision program. Let I be an initial database. Let F, U, N be sets of well-founded^{Sh}, unknown^{Sh}, and unfounded^{Sh} revision literals relative to (P, I), respectively. Let P' be a set of in-rules from $T_I(P)$. Let Π be a logic program $\Pi = lp(P')$. Let F', U', N' be sets of well-founded, unknown, and unfounded atoms relative to Π, respectively. Then, the following is satisfied.*
$F = NC(\{T_I(\mathbf{in}(a)) \leftarrow : a \in F'\}) \cup \{\alpha : \alpha \leftarrow body \in P,\ \alpha \in I^c\}$.

Proof.
Let us consider the process of computing well-founded semantics as a process of computing the least fixed points of corresponding operators. For well-founded semantics relative to (P, I), it is computing $lfp(\gamma^2_{P,I})$, and for well-founded semantics relative to Π, it is computing $lfp(\gamma^2_{\Pi})$.

One can prove that by induction on k. □

3.3 Comparison of the two definitions

In this section we compare the two definitions of well-founded semantics for revision programs from Section 3.1 and Section 3.2.

We say that one well-founded semantics is *better* (or *more informative*) than the other one on a particular instance of a revision problem if the union of well-founded and unfounded revision literals in the first semantics contains the union of well-founded and unfounded revision literals in the second semantics.

We present examples that show that these definitions are indeed different and neither of them is always better than the other. In particular, WFS^{Sh} is more informative than WFS^{PT} in Example 4, whereas WFS^{PT} is better than WFS^{Sh} in Example 3.

The following example shows that WFS^{PT} may be more informative than WFS^{Sh}.

Example 3 *Let $I = \emptyset$. Consider*

$$\begin{array}{rrcl} P: & \mathbf{in}(a) & \leftarrow & \mathbf{out}(b) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \\ & \mathbf{out}(b) & \leftarrow & \end{array}$$

1. *Compute WFS^{Sh}. We have: $lfp(\gamma^2_{P,I}) = \{\mathbf{out}(b)\}$, and $gft(\gamma^2_{P,I}) = \{\mathbf{in}(a), \mathbf{in}(b), \mathbf{out}(b)\}$. Hence, well-founded^{Sh} revision literals are $\{\mathbf{out}(b)\}$. There are no unfounded^{Sh} revision literals.*

2. *Compute WFS^{PT}. The logic program $\mathcal{P} = \mathcal{P}(P, I)$ is the following.*

$$\begin{array}{rrcl} \mathcal{P}: & \mathbf{out}_I(a) & \leftarrow & \\ & \mathbf{out}_I(b) & \leftarrow & \\ & \mathbf{in}(a) & \leftarrow & \mathbf{in}_I(a), \mathit{not}\ \mathbf{out}(a) \\ & \mathbf{out}(a) & \leftarrow & \mathbf{out}_I(a), \mathit{not}\ \mathbf{in}(a) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{in}_I(b), \mathit{not}\ \mathbf{out}(b) \\ & \mathbf{out}(b) & \leftarrow & \mathbf{out}_I(b), \mathit{not}\ \mathbf{in}(b) \\ & \mathbf{in}(a) & \leftarrow & \mathbf{out}(b) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \\ & \mathbf{out}(b) & \leftarrow & \end{array}$$

   Thus, $lfp(\gamma^2_{\mathcal{P}}) = gfp(\gamma^2_{\mathcal{P}}) = \{\mathbf{out}_I(a),\ \mathbf{out}_I(b),\ \mathbf{in}(a),\ \mathbf{out}(b)\}$. Therefore, revision literals $\mathbf{in}(a)$, $\mathbf{out}(b)$ are well-founded^{PT}. Revision literals $\mathbf{out}(a)$, $\mathbf{in}(b)$ are unfounded^{PT}.

Consequently, WFS^{PT} is more informative than WFS^{Sh} relative to P and I.

It is easy to see that $R = \{a\}$ is a P-justified revision of I. WFS^{PT} allows us to conclude that a must be in possible revisions, and b must be out. Whereas WFS^{Sh} only implies that P-justified revisions of $\emptyset$ must satisfy $\mathbf{out}(b)$.

The following example shows that WFS^{Sh} may be more informative than WFS^{PT}.

Example 4 *Let $I = \emptyset$. Consider*

$$\begin{array}{rrcl} P: & \mathbf{in}(a) & \leftarrow & \mathbf{out}(b) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \\ & \mathbf{in}(c) & \leftarrow & \mathbf{out}(d) \\ & \mathbf{in}(d) & \leftarrow & \mathbf{out}(c), \mathbf{out}(f) \\ & \mathbf{out}(f) & \leftarrow & \mathbf{in}(a), \mathbf{in}(b) \\ & \mathbf{in}(f) & \leftarrow & \\ & \mathbf{in}(a) & \leftarrow & \mathbf{in}(c), \mathbf{in}(f) \end{array}$$

1. *Compute WFS^{Sh}. We start with $\emptyset$ and obtain iterations of $\gamma_{P,\emptyset}$:*

$$\emptyset \mapsto \left\{ \begin{array}{c} \mathbf{in}(a) \\ \mathbf{in}(b) \\ \mathbf{in}(c) \\ \mathbf{in}(d) \\ \mathbf{out}(f) \\ \mathbf{in}(f) \end{array} \right\} \mapsto \{\mathbf{in}(f)\} \mapsto \left\{ \begin{array}{c} \mathbf{in}(a) \\ \mathbf{in}(b) \\ \mathbf{in}(c) \\ \mathbf{out}(f) \\ \mathbf{in}(f) \end{array} \right\} \mapsto \left\{ \begin{array}{c} \mathbf{in}(c) \\ \mathbf{in}(f) \\ \mathbf{in}(a) \end{array} \right\} \mapsto \left\{ \begin{array}{c} \mathbf{in}(a) \\ \mathbf{in}(c) \\ \mathbf{in}(f) \end{array} \right\}.$$

Thus, $lfp(\gamma^2_{P,I}) = gft(\gamma^2_{P,I}) = \{\mathbf{in}(a), \mathbf{in}(c), \mathbf{in}(f)\}$. Hence, $\mathbf{in}(a)$, $\mathbf{in}(c)$, $\mathbf{in}(f)$, $\mathbf{out}(b)$, $\mathbf{out}(d)$ are well-founded^{Sh}. Revision literals $\mathbf{in}(b)$, $\mathbf{in}(d)$ are unfounded^{Sh}.

2. *Compute WFS^{PT}. The logic program $\mathcal{P} = \mathcal{P}(P, \emptyset)$ is the following.*

$$\begin{array}{rrcl} \mathcal{P}: & \mathbf{out}_I(a) & \leftarrow & \\ & \mathbf{out}_I(b) & \leftarrow & \\ & \mathbf{out}_I(c) & \leftarrow & \\ & \mathbf{out}_I(d) & \leftarrow & \\ & \mathbf{out}_I(f) & \leftarrow & \\ & \mathbf{in}(a) & \leftarrow & \mathbf{in}_I(a), \mathit{not}\ \mathbf{out}(a) \\ & \mathbf{out}(a) & \leftarrow & \mathbf{out}_I(a), \mathit{not}\ \mathbf{in}(a) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{in}_I(b), \mathit{not}\ \mathbf{out}(b) \\ & \mathbf{out}(b) & \leftarrow & \mathbf{out}_I(b), \mathit{not}\ \mathbf{in}(b) \\ & \mathbf{in}(c) & \leftarrow & \mathbf{in}_I(c), \mathit{not}\ \mathbf{out}(c) \\ & \mathbf{out}(c) & \leftarrow & \mathbf{out}_I(c), \mathit{not}\ \mathbf{in}(c) \\ & \mathbf{in}(d) & \leftarrow & \mathbf{in}_I(d), \mathit{not}\ \mathbf{out}(d) \\ & \mathbf{out}(d) & \leftarrow & \mathbf{out}_I(d), \mathit{not}\ \mathbf{in}(d) \\ & \mathbf{in}(f) & \leftarrow & \mathbf{in}_I(f), \mathit{not}\ \mathbf{out}(f) \\ & \mathbf{out}(f) & \leftarrow & \mathbf{out}_I(f), \mathit{not}\ \mathbf{in}(f) \\ & \mathbf{in}(a) & \leftarrow & \mathbf{out}(b) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \end{array}$$

$$\begin{array}{rcl}
\mathbf{in}(c) & \leftarrow & \mathbf{out}(d) \\
\mathbf{in}(d) & \leftarrow & \mathbf{out}(c), \mathbf{out}(f) \\
\mathbf{out}(f) & \leftarrow & \mathbf{in}(a), \mathbf{in}(b) \\
\mathbf{in}(f) & \leftarrow & \\
\mathbf{in}(a) & \leftarrow & \mathbf{in}(c), \mathbf{in}(f)
\end{array}$$

We have the following computation (iterations of $\gamma_{\mathcal{P}}$ starting from $\emptyset$):

$$\emptyset \mapsto \left\{ \begin{array}{c} \mathbf{out}_I(a), \mathbf{out}_I(b), \mathbf{out}_I(c), \mathbf{out}_I(d), \mathbf{out}_I(f), \mathbf{out}(a), \mathbf{out}(b), \\ \mathbf{out}(c), \mathbf{out}(d), \mathbf{out}(f), \mathbf{in}(a), \mathbf{in}(b), \mathbf{in}(c), \mathbf{in}(d), \mathbf{in}(f) \end{array} \right\} \mapsto$$

$$\mapsto \left\{ \begin{array}{c} \mathbf{out}_I(a), \mathbf{out}_I(b), \\ \mathbf{out}_I(c), \mathbf{out}_I(d), \\ \mathbf{out}_I(f), \mathbf{in}(f) \end{array} \right\} \mapsto \left\{ \begin{array}{c} \mathbf{out}_I(a), \mathbf{out}_I(b), \mathbf{out}_I(c), \mathbf{out}_I(d), \mathbf{out}_I(f), \\ \mathbf{out}(a), \mathbf{out}(b), \mathbf{out}(c), \mathbf{out}(d), \mathbf{in}(a), \\ \mathbf{in}(b), \mathbf{in}(c), \mathbf{out}(f), \mathbf{in}(f), \mathbf{in}(d) \end{array} \right\}.$$

Thus, $lfp(\gamma^2_{\mathcal{P}}) = \{\mathbf{in}(f)\} \cup \{\mathbf{out}_I(x) : x \in \{a,b,c,d,f\}\}$ and $gft(\gamma^2_{\mathcal{P}}) = \{\mathbf{out}(a)$, $\mathbf{out}(b)$, $\mathbf{out}(c)$, $\mathbf{out}(d)$, $\mathbf{in}(a)$, $\mathbf{in}(b)$, $\mathbf{in}(c)$, $\mathbf{out}(f)$, $\mathbf{in}(f)$, $\mathbf{in}(d)\} \cup \{\mathbf{out}_I(x) : x \in \{a,b,c,d,f\}\}$. Therefore, the only well-founded^{PT} revision literal is $\mathbf{in}(f)$. There are no unfounded^{PT} revision literals.

Consequently, WFS^{Sh} is more informative than WFS^{PT} relative to P and I.

Moreover, WFS^{Sh} allows us to conclude that the only P-justified revision of $\emptyset$ is $R = \{a,c,f\}$. Whereas WFS^{PT} only imply that P-justified revisions of $\emptyset$ must satisfy $\mathbf{in}(f)$.

4 Definition specific to revision programming

In this section we give a definition of well-founded semantics which is specific for revision programming.

4.1 Well-founded semantics for revision programming

Definition 13 *Given a revision program P and a set of literals A, the revision program $Simpl(P,A)$ is obtained from P by*

1. *removing all rules $r \in P$ such that $body(r) \cap \{l^D : l \in A\} \neq \emptyset$;*
2. *remove all rules with heads from A;*
3. *remove from the bodies of the remaining rules all revision literals that are in A.* $\triangle$

Lemma 3 *Let P_1, P_2 be revision programs. Let A_1, A_2 be sets of revision literals. Then,*

1. *$Simpl(P_1 \cup P_2, A_1) = Simpl(P_1, A_1) \cup Simpl(P_2, A_2)$, and*
2. *$Simpl(P_1, A_1 \cup A_2) = Simpl(Simpl(P_1, A_1), A_2)$.*

Definition 14 *Let us define a sequence of triples (P_k, A_k, X_k), $k = 0, 1, \ldots$, as follows.*

- $k = 0$. *Let* $P_0 = P$, $A_0 = \emptyset$, $X_0 = \emptyset$.
- $k \geq 1$. *If* $k = 2i + 1$ ($i \in \mathbb{N}$), *then* $A'_{2i+1} = NC(P_{2i})$.
  Else if $k = 2i$ ($i \in \mathbb{N}$), *then* $A'_{2i} = \{l \in I^c : (l^D \notin X_{2i-1})\ \&\ (l \notin A_{2i-1})\ \&\ (l^D \notin A_{2i-1})\}$.
  Let $A_k = A_{k-1} \cup A'_k$.
  If A_k *is incoherent then for all* $k' \geq k$ *let* $A_{k'} = A_k$, $X_{k'} = \emptyset$, *and* $P_{k'} = P_{k-1}$.
  Otherwise, let $P_k = Simpl(P_{k-1}, A'_k)$, $X_k = \gamma_{P_k,I}(X_{k-1})$. $\triangle$

To illustrate the notion, let us compute a sequence of triples for the following example.

Example 5 *Let* $I = \emptyset$. *Consider*

$$\begin{array}{rrcl} P: & \mathbf{out}(a) & \leftarrow & \\ & \mathbf{in}(a) & \leftarrow & \mathbf{out}(b) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \\ & \mathbf{in}(c) & \leftarrow & \mathbf{in}(a), \mathbf{in}(b) \\ & \mathbf{out}(d) & \leftarrow & \mathbf{out}(c) \\ & \mathbf{in}(d) & \leftarrow & \mathbf{out}(e) \\ & \mathbf{in}(e) & \leftarrow & \mathbf{out}(d) \\ & \mathbf{in}(f) & \leftarrow & \mathbf{in}(d), \mathbf{in}(e) \\ & \mathbf{out}(g) & \leftarrow & \mathbf{out}(f) \\ & \mathbf{in}(g) & \leftarrow & \mathbf{out}(h) \\ & \mathbf{in}(h) & \leftarrow & \mathbf{out}(g) \\ & \mathbf{in}(i) & \leftarrow & \mathbf{in}(g), \mathbf{in}(h) \end{array}$$

We have the following computation:

$$A_0 = \emptyset, X_0 = \emptyset, P_0 = P \mapsto$$

$$A_1 = \left\{ \begin{array}{c} \mathbf{out}(a) \\ \mathbf{in}(b) \end{array} \right\}, P_1 = \left\{ \begin{array}{rcl} \mathbf{out}(d) & \leftarrow & \mathbf{out}(c) \\ \mathbf{in}(d) & \leftarrow & \mathbf{out}(e) \\ \mathbf{in}(e) & \leftarrow & \mathbf{out}(d) \\ \mathbf{in}(f) & \leftarrow & \mathbf{in}(d), \mathbf{in}(e) \\ \mathbf{out}(g) & \leftarrow & \mathbf{out}(f) \\ \mathbf{in}(g) & \leftarrow & \mathbf{out}(h) \\ \mathbf{in}(h) & \leftarrow & \mathbf{out}(g) \\ \mathbf{in}(i) & \leftarrow & \mathbf{in}(g), \mathbf{in}(h) \end{array} \right\}, X_1 = \left\{ \begin{array}{c} \mathbf{out}(d), \\ \mathbf{in}(d), \\ \mathbf{in}(e), \\ \mathbf{in}(f), \\ \mathbf{out}(g), \\ \mathbf{in}(g), \\ \mathbf{in}(h), \\ \mathbf{in}(i) \end{array} \right\} \mapsto$$

$$\mapsto A_2 = \left\{ \begin{array}{c} \mathbf{out}(a) \\ \mathbf{in}(b) \\ \mathbf{out}(c) \end{array} \right\}, P_2 = \left\{ \begin{array}{rcl} \mathbf{out}(d) & \leftarrow & \\ \mathbf{in}(d) & \leftarrow & \mathbf{out}(e) \\ \mathbf{in}(e) & \leftarrow & \mathbf{out}(d) \\ \mathbf{in}(f) & \leftarrow & \mathbf{in}(d), \mathbf{in}(e) \\ \mathbf{out}(g) & \leftarrow & \mathbf{out}(f) \\ \mathbf{in}(g) & \leftarrow & \mathbf{out}(h) \\ \mathbf{in}(h) & \leftarrow & \mathbf{out}(g) \\ \mathbf{in}(i) & \leftarrow & \mathbf{in}(g), \mathbf{in}(h) \end{array} \right\}, X_2 = \left\{ \ \mathbf{out}(d) \ \right\} \mapsto$$

$$\mapsto A_3 = \left\{ \begin{array}{c} \mathbf{out}(a), \\ \mathbf{in}(b), \\ \mathbf{out}(c), \\ \mathbf{out}(d), \\ \mathbf{in}(e) \end{array} \right\}, P_3 = \left\{ \begin{array}{rcl} \mathbf{out}(g) & \leftarrow & \mathbf{out}(f) \\ \mathbf{in}(g) & \leftarrow & \mathbf{out}(h) \\ \mathbf{in}(h) & \leftarrow & \mathbf{out}(g) \\ \mathbf{in}(i) & \leftarrow & \mathbf{in}(g), \mathbf{in}(h) \end{array} \right\}, X_3 = \left\{ \begin{array}{c} \mathbf{out}(g) \\ \mathbf{in}(g) \\ \mathbf{in}(h) \\ \mathbf{in}(i) \end{array} \right\} \mapsto$$

$$\mapsto A_4 = \left\{ \begin{array}{c} \mathbf{out}(a), \\ \mathbf{in}(b), \\ \mathbf{out}(c), \\ \mathbf{out}(d), \\ \mathbf{in}(e), \\ \mathbf{out}(f) \end{array} \right\}, P_4 = \left\{ \begin{array}{rcl} \mathbf{out}(g) & \leftarrow & \\ \mathbf{in}(g) & \leftarrow & \mathbf{out}(h) \\ \mathbf{in}(h) & \leftarrow & \mathbf{out}(g) \\ \mathbf{in}(i) & \leftarrow & \mathbf{in}(g), \mathbf{in}(h) \end{array} \right\}, X_4 = \left\{ \ \mathbf{out}(g) \ \right\} \mapsto$$

$$\mapsto A_5 = \left\{ \ \mathbf{out}(a), \mathbf{in}(b), \mathbf{out}(c), \mathbf{out}(d), \mathbf{in}(e), \mathbf{out}(f), \mathbf{out}(g), \mathbf{in}(h) \ \right\}, P_5 = \emptyset, X_5 = \emptyset.$$

For $i \geq 6$, $A_i = \left\{ \ \mathbf{out}(a), \mathbf{in}(b), \mathbf{out}(c), \mathbf{out}(d), \mathbf{in}(e), \mathbf{out}(f), \mathbf{out}(g), \mathbf{in}(h), \mathbf{out}(i) \ \right\}$, $P_i = \emptyset$, and $X_i = \emptyset$.

Lemma 4 *Let P be a revision program. Let I be an initial database. Let R be a P-justified revision of I. Then, $R^c = \gamma_{P,I}(R^c) \cup I(I,R)$.*

Proof.
By Theorem 3, $R^c = NC(P_{I,R}) \cup I(I,R)$. By definition, $\gamma_{P,I}(R^c) = NC((P,I)^{R^c})$. Let us show that $NC((P,I)^{R^c}) = NC(P_{I,R})$.

Indeed, the reduct $P_{I,R}$ is obtained from P by removing from the bodies of rules literals from $I(I,R)$. Consider the second step in the definition of the reduct $(P,I)^{R^c}$. Assume that a revision literal l is removed from the body of a rule r. Then, $l \in I^c$. If $l \notin R^c$, then $l^D \in R^c \setminus I^c$, and rule r would have been removed at the first step of the definition of the reduct $(P,I)^{R^c}$. Hence, $l \in R^c$, and thus, $l \in I(I,R)$. Therefore, at the second step of the definition of the reduct $(P,I)^{R^c}$, literals from $I(I,R)$ are removed from the bodies of the remaining rules. Thus, $(P,I)^{R^c} \subseteq P_{I,R}$.

Bodies of the rules which are removed during the first step of the definition of the reduct $(P,I)^{R^c}$ are *not* satisfied by R^c. Since $NC(P_{I,R}) \subseteq R^c$, they are not satisfied by $NC(P_{I,R})$, too. The necessary change of a program remains the same if we remove from the program rules that are not satisfied by the necessary change. Consequently, $NC((P,I)^{R^c}) = NC(P_{I,R})$. Therefore, $R^c = \gamma_{P,I}(R^c) \cup I(I,R)$. $\square$

Lemma 5 *Let P be a revision program. Let I be an initial database. Let R be a P-justified revision of I. Let A be a set of literals such that $A \subseteq R^c$. Let $P' = Simpl(P, A) \cup \{l \leftarrow: l \in A\}$. Then, R is a P'-justified revision of I.*

Proof.
By Theorem 3,

$$NC(P \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}) = R^c.$$

By definition of the necessary change, R^c is the least model of $P \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}$ when treated as a Horn program built of independent propositional atoms of the form $\mathbf{in}(a)$ and $\mathbf{out}(b)$.

Since $A \subseteq R^c$, rules $\{l \leftarrow: l \in A\}$ are satisfied in R^c. Therefore by Lemma 1,

$$NC(P \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}) = NC(P \cup \{l \leftarrow: l \in A\} \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}).$$

Let $Q = P\cup \{l \leftarrow: l \in A\} \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}$. We have $NC(Q) = R^c$. It is easy to see that applying steps 1, 2, and 3 from the definition of $Simpl(P, A)$ to the part P of the program Q does not change the least model of the program. In other words,

$$NC(Q) = NC(Simpl(P, A) \cup \{l \leftarrow: l \in A\} \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}).$$

Thus, $R^c = NC(P' \cup \{\alpha \leftarrow: \alpha \in I(I, R)\})$. Therefore, by Theorem 3, R is a P'-justified revision of I. □

Lemma 6 *Let P be a revision program. Let I be an initial database. Let $A = NC(P)$. Let $P' = Simpl(P, A) \cup \{l \leftarrow: l \in A\}$. Then, R is a P-justified revision of I if and only if R is a P'-justified revision of I.*

Proof.
($\Longrightarrow$) Let R be a P-justified revision of I. Then, by Theorem 3,

$$NC(P \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}) = R^c.$$

Thus, $A \subseteq R^c$. By Lemma 5, R is a P'-justified revision of I.
($\Longleftarrow$) Let R be a P'-justified revision of I. Then,

$$NC(Simpl(P, A) \cup \{l \leftarrow: l \in A\} \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}) = R^c.$$

Notice that we can "undo" steps 3, 2, 1 in the definition of $Simpl(P, A)$ without changing the necessary change of the program

$$(Simpl(P, A) \cup \{l \leftarrow: l \in A\} \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}).$$

Thus,

$$R^c = NC(P \cup \{l \leftarrow: l \in A\} \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}).$$

Since $A = NC(P)$, we get that

$$NC(P \cup \{l \leftarrow: l \in A\} \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}) = NC(P \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}).$$

Consequently, $R^c = NC(P \cup \{\alpha \leftarrow: \alpha \in I(I, R)\})$. Therefore, by Theorem 3, R is a P-justified revision of I. □

Lemma 7 *Let Q be a revision program. Let I be an initial database. Let X be a set of revision literals such that for every R which is a Q-justified revision of I, $R^c \subseteq X \cup I^c$. Let $A = \{l \in I^c : l^D \notin X\}$. Let $Q' = Simpl(Q, A) \cup \{l \leftarrow: l \in A\}$. Then, R is a Q-justified revision of I if and only if R is a Q'-justified revision of I.*

Proof.
($\Longrightarrow$) Let R be a Q-justified revision of I. Then, $R^c \subseteq X \cup I^c$. Let us show that $A \subseteq R^c$. Let $l \in A$. Then, $l \in I^c$ and $l^D \notin X$. Since I^c does not contain dual revision literals, $l \in I^c$ implies $l^D \notin I^c$. Hence, $l^D \notin X \cup I^c$. Thus, $l^D \notin R^c$. That is, $l \in R^c$. Therefore, $A \subseteq R^c$. By Lemma 5, R is a Q'-justified revision of I.
($\Longleftarrow$) Let R be a Q'-justified revision of I. Then,

$$NC(Simpl(Q, A) \cup \{l \leftarrow: l \in A\} \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}) = R^c.$$

Notice that we can "undo" steps 3, 2, 1 in the definition of $Simpl(Q, A)$ without changing the necessary change of the program

$$(Simpl(Q, A) \cup \{l \leftarrow: l \in A\} \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}).$$

Thus,

$$R^c = NC(Q \cup \{l \leftarrow: l \in A\} \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}).$$

Let us show that

$$\{l \leftarrow: l \in A\} \subseteq \{\alpha \leftarrow: \alpha \in I(I, R)\}). \qquad (3)$$

Indeed, let $l \in A$. Then, $l \in R^c$ and $l \in I^c$. Therefore, $l \in I(I, R)$. Consequently, equation (3) holds. We get that

$$NC(Q \cup \{l \leftarrow: l \in A\} \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}) = NC(Q \cup \{\alpha \leftarrow: \alpha \in I(I, R)\}).$$

Hence, $R^c = NC(Q \cup \{\alpha \leftarrow: \alpha \in I(I, R)\})$. Therefore, by Theorem 3, R is a Q-justified revision of I. $\square$

Theorem 13 *Let P be a revision program. Let I be an initial database. Let (P_k, A_k, X_k), $k \geq 0$, be the sequence of triples constructed according to the Definition 14. Then, for every $k \geq 0$ the following holds.*

1. *R is a P-justified revision of I if and only if R is a $(P_k \cup \{l \leftarrow : l \in A_k\})$-justified revision of I.*

2. *For every R which is a P-justified revision of I, the following holds. If $k = 2i$ then $(X_k \cup A_k) \subseteq R^c$, and if $k = 2i + 1$ then $R^c \subseteq (X_k \cup A_k \cup I^c)$.*

Proof.
By induction on k.
Basis step ($k = 0$). Both statements 1 and 2 trivially hold because $P_0 = P$, $A_0 = \emptyset$, $X_0 = \emptyset$.

Inductive step. Assume that the statement of the theorem is true for $k-1$. We need to prove that it holds for k ($k \geq 1$).

Case 1. Let $k = 2i+1$. By inductive hypothesis,
(1) R is a P-justified revision of I if and only if R is a $(P_{2i} \cup \{l \leftarrow\ : l \in A_{2i}\})$-justified revision of I, and
(2) for every R which is a P-justified revision of I, $(X_{2i} \cup A_{2i}) \subseteq R^c$.

We need to prove that
(3) R is a P-justified revision of I if and only if R is a $(P_{2i+1} \cup \{l \leftarrow\ : l \in A_{2i+1}\})$-justified revision of I, and
(4) for every R which is a P-justified revision of I, $R^c \subseteq (X_{2i+1} \cup A_{2i+1} \cup I^c)$.

By definition, $A'_{2i+1} = NC(P_{2i})$, $A_{2i+1} = A_{2i} \cup A'_{2i+1}$, $P_{2i+1} = Simpl(P_{2i}, A'_{2i+1})$, $X_{2i+1} = \gamma_{P_{2i+1},I}(X_{2i})$.

Proof of (3):
Consider revision program $Q = P_{2i} \cup \{l \leftarrow\ : l \in A_{2i}\}$. Since P_{2i} does not contain in the bodies of its rules literals from $A_{2i} \cup A_{2i}^D$, we get that $NC(Q) = NC(P_{2i} \cup \{l \leftarrow\ : l \in A_{2i}\}) = NC(P_{2i}) \cup A_{2i} = A'_{2i+1} \cup A_{2i} = A_{2i+1}$. Also, $P_{2i+1} = Simpl(P_{2i}, A'_{2i+1}) = Simpl(P_{2i}, A_{2i+1})$. From this and the fact that $A_{2i} \subseteq A_{2i+1}$, it follows that

$$P_{2i+1} \cup \{l \leftarrow\ : l \in A_{2i+1}\} =$$

$$Simpl(P_{2i}, A_{2i+1}) \cup \{l \leftarrow\ : l \in A_{2i+1}\} =$$

$$Simpl(P_{2i} \cup \{l \leftarrow\ : l \in A_{2i}\}, A_{2i+1}) \cup \{l \leftarrow\ : l \in A_{2i+1}\} =$$

$$Simpl(Q, NC(Q)) \cup \{l \leftarrow\ : l \in NC(Q)\}.$$

Therefore, by Lemma 6, R is a $(P_{2i} \cup \{l \leftarrow\ : l \in A_{2i}\})$-justified revision of I if and only if R is a $P_{2i+1} \cup \{l \leftarrow\ : l \in A_{2i+1}\}$-justified revision of I. Now, using the inductive assumption (1), we get statement (3).

Proof of (4):
Assume that R is a P-justified revision of I. By definition, $X_{2i+1} = \gamma_{P_{2i+1},I}(X_{2i})$. It is easy to see that

$$X_{2i+1} \cup A_{2i+1} = \gamma_{P_{2i+1},I}(X_{2i}) \cup A_{2i+1} =$$

$$\gamma_{(P_{2i+1} \cup \{l \leftarrow\ : l \in A_{2i+1}\}),I}(X_{2i} \cup A_{2i}).$$

Let $P' = P_{2i+1} \cup \{l \leftarrow\ : l \in A_{2i+1}\}$. By (3), R is a P'-justified revision of I. Since γ is anti-monotone and $(X_{2i} \cup A_{2i}) \subseteq R^c$, we get that

$$\gamma_{P',I}(R^c) \subseteq \gamma_{P',I}(X_{2i} \cup A_{2i}).$$

Hence,

$$\gamma_{P',I}(R^c) \cup I^c \subseteq \gamma_{P',I}(X_{2i} \cup A_{2i}) \cup I^c.$$

Therefore, by Lemma 4,

$$R^c = \gamma_{P',I}(R^c) \cup I(I,R) \subseteq \gamma_{P',I}(X_{2i} \cup A_{2i}) \cup I^c = X_{2i+1} \cup A_{2i+1} \cup I^c.$$

Case 2. $k = 2i$, $k \geq 2$. By inductive hypothesis,
(5) R is a P-justified revision of I if and only if R is a $(P_{2i-1} \cup \{l \leftarrow\ : l \in A_{2i-1}\})$-justified

revision of I, and
(6) for every R which is a P-justified revision of I, $R^c \subseteq (X_{2i-1} \cup A_{2i-1} \cup I^c)$.

We need to prove that
(7) R is a P-justified revision of I if and only if R is a $(P_{2i} \cup \{l \leftarrow\ : l \in A_{2i}\})$-justified revision of I, and
(8) for every R which is a P-justified revision of I, $(X_{2i} \cup A_{2i}) \subseteq R^c$.

By definition, $A'_{2i} = \{l \in I^c : l^D \notin X_{2i-1} \ \&\ l \notin A_{2i-1} \ \&\ l^D \notin A_{2i-1}\}$, $A_{2i} = A_{2i-1} \cup A'_{2i}$, $P_{2i} = \mathit{Simpl}(P_{2i-1}, A'_{2i})$, $X_{2i} = \gamma_{P_{2i},I}(X_{2i-1})$.

Proof of (7):
Let $Q = P_{2i-1} \cup \{l \leftarrow\ : l \in A_{2i-1}\}$. Let $X = X_{2i-1} \cup A_{2i-1}$. Then, from (5) and (6) it follows that for every R which is a Q-justified revision of I, $R^c \subseteq X \cup I^c$.

Let $A = \{l \in I^c : l^D \notin X\}$. Let $Q' = \mathit{Simpl}(Q, A) \cup \{l \leftarrow\ : l \in A\}$. By Lemma 7, R is a Q-justified revision of I if and only if R is a Q'-justified revision of I. Using (5) we get that R is a P-justified revision of I if and only if R is a Q'-justified revision of I.

Therefore, to prove (7) we need to show that $Q' = P_{2i} \cup \{l \leftarrow\ : l \in A_{2i}\}$.

Indeed,

$$Q' = \mathit{Simpl}(Q, A) \cup \{l \leftarrow\ : l \in A\} =$$
$$= \mathit{Simpl}(P_{2i-1} \cup \{l \leftarrow\ : l \in A_{2i-1}\}, A) \cup \{l \leftarrow\ : l \in A\} =$$
$$= \mathit{Simpl}(P_{2i-1}, A) \cup \mathit{Simpl}(\{l \leftarrow\ : l \in A_{2i-1}\}, A) \cup \{l \leftarrow\ : l \in A\} =$$
$$= \mathit{Simpl}(P_{2i-1}, A) \cup \{l \leftarrow\ : l \in A_{2i-1} \setminus A\} \cup \{l \leftarrow\ : l \in A\} =$$
$$= \mathit{Simpl}(P_{2i-1}, A) \cup \{l \leftarrow\ : l \in A_{2i-1} \cup A\}.$$

By definition, $A = \{l \in I^c : l^D \notin X\} = \{l \in I^c : l^D \notin X_{2i-1} \ \&\ l^D \notin A_{2i-1}\}$. Consequently, $A = A'_{2i} \cup \{l \in I^c : l^D \notin X_{2i-1} \ \&\ l^D \notin A_{2i-1} \ \&\ l \in A_{2i-1}\}$. Thus, $A_{2i-1} \cup A = A_{2i-1} \cup A'_{2i} = A_{2i}$. Hence,

$$Q' = \mathit{Simpl}(P_{2i-1}, A) \cup \{l \leftarrow\ : l \in A_{2i}\}.$$

Also, if a revision literal $l \in A_{2i-1}$, then l and l^D do not appear in the bodies of the rules from P_{2i-1}, and l does not appear in the heads of the rules from P_{2i-1}. Therefore, $\mathit{Simpl}(P_{2i-1}, A) = \mathit{Simpl}(P_{2i-1}, A \setminus A_{2i-1}) = \mathit{Simpl}(P_{2i-1}, A'_{2i}) = P_{2i}$. Consequently,

$$Q' = P_{2i} \cup \{l \leftarrow\ : l \in A_{2i}\}.$$

This finishes the proof of (7).

Proof of (8):
Assume that R is a P-justified revision of I. By definition, $X_{2i} = \gamma_{P_{2i},I}(X_{2i-1})$. It is easy to see that

$$X_{2i} \cup A_{2i} = \gamma_{P_{2i},I}(X_{2i-1}) \cup A_{2i} = \gamma_{(P_{2i} \cup \{l \leftarrow\ : l \in A_{2i}\}),I}(X_{2i-1} \cup A_{2i-1}).$$

Let $P' = P_{2i} \cup \{l \leftarrow\ : l \in A_{2i}\}$. By (7), R is a P'-justified revision of I. Since γ is anti-monotone and $R^c \subseteq (X_{2i-1} \cup A_{2i-1} \cup I^c)$, we get that

$$\gamma_{P',I}(R^c) \supseteq \gamma_{P',I}(X_{2i-1} \cup A_{2i-1} \cup I^c).$$

From the definition of γ it follows that

$$\gamma_{P',I}(X_{2i-1} \cup A_{2i-1} \cup I^c) = \gamma_{P',I}(X_{2i-1} \cup A_{2i-1}).$$

Hence,

$$\gamma_{P',I}(R^c) \supseteq \gamma_{P',I}(X_{2i-1} \cup A_{2i-1}) = X_{2i} \cup A_{2i}.$$

By Lemma 4, $R^c = \gamma_{P',I}(R^c) \cup I(I,R)$. Therefore,

$$R^c \supseteq \gamma_{P',I}(R^c) \supseteq X_{2i} \cup A_{2i}.$$

That is, (8) holds. □

Corollary 2 *Let R be a P-justified revision of I. Then, for every $k \geq 0$, $R \models A_k$.*

Proof.
By Theorem 13, R is a $(P_k \cup \{l \leftarrow\, : l \in A_k\})$-justified revision of I. Hence, $R \models A_k$. □

Corollary 3 *If in the sequence of triples (P_k, A_k, X_k) there exists k such that A_k is incoherent, then there are no P-justified revisions of I.*

Proof.
Follows from Corollary 2. □

Lemma 8 *In the Definition 14, $X_{2i} \subseteq A'_{2i+1}$ for $i \geq 0$.*

Proof.
If $i = 0$, then $X_0 = \emptyset$. Thus, $X_0 \subseteq A'_1$.

Let $i \geq 1$. Then, by definition, $A'_{2i+1} = NC(P_{2i})$, and $X_{2i} = \gamma_{P_{2i},I}(X_{2i-1})$. By definition of operator γ, $X_{2i} = NC((P_{2i}, I)^{X_{2i-1}})$. By definition of the reduct, $(P_{2i}, I)^{X_{2i-1}}$ is obtained from P_{2i} by

1. removing every rule $r \in P_{2i}$ such that $\mathit{body}(r) \cap \{l^D : l \in X_{2i-1} \setminus I^c\} \neq \emptyset$,
2. deleting from the body of each remaining rule any revision literal that is in I^c.

Therefore, for every rule $r \in (P_{2i}, I)^{X_{2i-1}}$, there is a rule $r' \in P_{2i}$, such that r' is *not* removed during step 1 of computing the reduct $(P_{2i}, I)^{X_{2i-1}}$, its head $\mathit{head}(r) = \mathit{head}(r')$ and $\mathit{body}(r) \subseteq \mathit{body}(r')$. We will prove that in fact, $\mathit{body}(r) = \mathit{body}(r')$.

Assume the contrary. Assume that $l \in \mathit{body}(r')$ and $l \notin \mathit{body}(r)$. Then, by the second step of the definition of the reduct, $l \in I^c$. Hence, $l^D \notin I^c$.

Let us show that $l^D \in X_{2i-1}$. Assume the contrary, that $l^D \notin X_{2i-1}$. Because of the properties of *Simpl* transformation, if $l \in A_{2i-1}$ or $l^D \in A_{2i-1}$, then l can not be in the body of a rule in P_{2i}. However, $l \in \mathit{body}(r')$, where $r' \in P_{2i}$. Therefore, $l \notin A_{2i-1}$ and $l^D \notin A_{2i-1}$. Then, by construction of triples (P_k, A_k, X_k), $l \in \{l \in I^c : l^D \notin X_{2i-1} \& l \notin A_{2i-1} \& l^D \notin A_{2i-1}\}$. That is, $l \in A'_{2i}$. Hence, by properties of *Simpl* transformation, l

can not be in the body of a rule from P_{2i}. This contradicts the fact that $l \in body(r')$ and $r' \in P_{2i}$. Consequently, $l^D \in X_{2i-1}$.

We have: $l^D \notin I^c$ and $l^D \in X_{2i-1}$. Thus, $l^D \in X_{2i-1} \backslash I^c$. At the same time, $l \in body(r')$. According to step 1 in the definition of the reduct $(P_{2i}, I)^{X_{2i-1}}$, rule r' will be removed. This contradicts the fact that r' is not removed during step 1 of computing the reduct $(P_{2i}, I)^{X_{2i-1}}$. Therefore, for every rule $r \in (P_{2i}, I)^{X_{2i-1}}$, there is a rule $r' \in P_{2i}$, such that $head(r) = head(r')$ and $body(r) = body(r')$. Hence, $NC((P_{2i}, I)^{X_{2i-1}}) \subseteq NC(P_{2i})$. In other words, $X_{2i} \subseteq A'_{2i+1}$. $\Box$

Corollary 4 *In Definition 14, for every $i \geq 0$, $X_{2i} \cup A_{2i} \subseteq X_{2i+2} \cup A_{2i+2}$.*

Proof.
By Definition 14, $A_{2i} \subseteq A_{2i+2}$. By Lemma 8, $X_{2i} \subseteq A_{2i+1} \subseteq A_{2i+2}$. Hence, $X_{2i} \cup A_{2i} \subseteq X_{2i+2} \cup A_{2i+2}$. $\Box$

Definition 15 *Let P be a revision program. Let I be an initial database. Let (P_k, A_k, X_k), $k = 0, 1, \ldots$ be a sequence of triples as defined in Definition 14. Let $F = \cup_{i \geq 0}(X_{2i} \cup A_{2i})$. If F is coherent, then revision literals from F are called* well-founded *with respect to P and I.* $\triangle$

Theorem 14 *Let P be a revision program. Let I be initial database. Then, any P-justified revision of I satisfies all revision literals that are well-founded with respect to P and I.*

Proof.
Let R be a P-justified revision of I. Let l be a revision literal that is well-founded with respect to P and I. By definition, $l \in \cup_{i \geq 0}(X_{2i} \cup A_{2i})$. Hence, there exists $i \geq 0$ such that $l \in (X_{2i} \cup A_{2i})$. By Theorem 13, $(X_{2i} \cup A_{2i}) \subseteq R^c$. Therefore, $l \in R^c$. In other words, R satisfies l. $\Box$

By Theorem 13, if F is not coherent in the definition 15, then there are no P-justified revisions of I.

Key features of the well-founded semantics of revision programs specified by Definition 15:

1. As soon as it is establish that a revision literal and its dual are both well-founded, no further computation of well-founded semantics is done.

2. There is no need for unfounded revision literals. If a literal $l \notin I^c$ is shown to be unfounded then is dual l^D automatically becomes well-founded by inertia. (In logic programming they only talk about unfounded atoms which correspond to revision literals not in I^c.)

4.1.1 Towards simplifying the definition

Lemma 9 *Let P be a revision program. Let I be an initial database. Let (P_k, A_k, X_k), $k \geq 0$, be the sequence of triples constructed according to the Definition 14. Then, for every $k \geq 0$, if A_k is coherent then $P_k = Simpl(P, A_k)$.*

Proof.
By induction on k.

Basis step ($k = 0$). The statement holds because both P_0 and $Simpl(P, A_0)$ are equal to P.

Inductive step. Assume that the statement of the theorem is true for $k-1$. That is, $P_{k-1} = Simpl(P, A_{k-1})$. We need to prove that it holds for k ($k \geq 1$).

By definition, if A_k is coherent, then $P_k = Simpl(P_{k-1}, A'_k)$. By inductive assumption, we get $P_k = Simpl(Simpl(P, A_{k-1}), A'_k)$. By Lemma 3, $Simpl(Simpl(P, A_{k-1}), A'_k) = Simpl(P, A_{k-1} \cup A'_k)$. By definition, $A_k = A_{k-1} \cup A'_k$. Therefore, $P_k = Simpl(P, A_k)$. □

4.2 Comparison with definitions induced by embeddings

The following is an example when the new definition of well-founded semantics gives a better result than WFS^{Sh}.

Example 6 *Let $I = \emptyset$. Consider revision program from Example 3*

$$\begin{array}{rrcl} P: & \mathbf{in}(a) & \leftarrow & \mathbf{out}(b) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \\ & \mathbf{out}(b) & \leftarrow & \end{array}$$

We have the following computation:

$$A_0 = \emptyset, P_0 = P, X_0 = \emptyset \mapsto$$

$$\mapsto A_1 = \left\{ \begin{array}{c} \mathbf{out}(b), \\ \mathbf{in}(a) \end{array} \right\}, P_1 = \emptyset, X_1 = \emptyset.$$

The WFS^{Sh} for P was found in Example 3. It is less informative than the new definition of WFS.

The following is an example when the new definition of well-founded semantics gives better result than WFS^{PT}.

Example 7 *Let $I = \emptyset$. Consider revision program from Example 4*

$$\begin{array}{rrcl} P: & \mathbf{in}(a) & \leftarrow & \mathbf{out}(b) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \\ & \mathbf{in}(c) & \leftarrow & \mathbf{out}(d) \\ & \mathbf{in}(d) & \leftarrow & \mathbf{out}(c), \mathbf{out}(f) \\ & \mathbf{out}(f) & \leftarrow & \mathbf{in}(a), \mathbf{in}(b) \\ & \mathbf{in}(f) & \leftarrow & \\ & \mathbf{in}(a) & \leftarrow & \mathbf{in}(c), \mathbf{in}(f) \end{array}$$

We have the following computation:

$$A_0 = \emptyset, P_0 = P, X_0 = \emptyset \mapsto$$

$$\mapsto A_1 = \left\{\ \mathbf{in}(f)\ \right\}, P_1 = \left\{ \begin{array}{rcl} \mathbf{in}(a) & \leftarrow & \mathbf{out}(b) \\ \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \\ \mathbf{in}(c) & \leftarrow & \mathbf{out}(d) \\ \mathbf{out}(f) & \leftarrow & \mathbf{in}(a), \mathbf{in}(b) \\ \mathbf{in}(a) & \leftarrow & \mathbf{in}(c) \end{array} \right\}, X_1 = \left\{ \begin{array}{c} \mathbf{in}(a), \\ \mathbf{in}(b), \\ \mathbf{in}(c), \\ \mathbf{out}(f) \end{array} \right\} \mapsto$$

$$\mapsto A_2 = \left\{ \begin{array}{c} \mathbf{in}(f), \\ \mathbf{out}(d) \end{array} \right\}, P_2 = \left\{ \begin{array}{rcl} \mathbf{in}(a) & \leftarrow & \mathbf{out}(b) \\ \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \\ \mathbf{in}(c) & \leftarrow & \\ \mathbf{out}(f) & \leftarrow & \mathbf{in}(a), \mathbf{in}(b) \\ \mathbf{in}(a) & \leftarrow & \mathbf{in}(c) \end{array} \right\}, X_2 = \left\{ \begin{array}{c} \mathbf{in}(c), \\ \mathbf{in}(a) \end{array} \right\} \mapsto$$

$$\mapsto A_3 = \left\{ \begin{array}{c} \mathbf{in}(f), \\ \mathbf{out}(d), \\ \mathbf{in}(c), \\ \mathbf{in}(a) \end{array} \right\}, P_3 = \left\{\ \mathbf{out}(f) \ \leftarrow\ \mathbf{in}(b)\ \right\}, X_3 = \emptyset \mapsto$$

$$\mapsto A_4 = \left\{ \begin{array}{c} \mathbf{in}(f), \\ \mathbf{out}(d), \\ \mathbf{in}(c), \\ \mathbf{in}(a), \\ \mathbf{out}(b) \end{array} \right\}, P_4 = \emptyset, X_4 = \emptyset.$$

The WFS^{PT} for P was found in Example 4. It is less informative than the new definition of WFS.

5 Yet another definition

Let P be a revision program. Let I be an initial database.

Definition 16 *For a set of revision literals X define the* reduct *of $[P,\ I]$ relative to X (denoted by $[P,I]^X$) to be the revision program obtained from P by*

1. *removing every rule $r \in P$ such that body$(r) \cap \{l^D : (l \in X \setminus I^c) \wedge (l^D \notin X)\} \neq \emptyset$,*
2. *deleting from the body of each remaining rule any revision literal that is in $I^c \setminus \{l \in I^c : (l \in X) \wedge (l^D \in X)\}$.* $\triangle$

Notice that the above definition can be rewritten in the following way.

Definition 17 *For a set of revision literals X define the* reduct *of $[P,\ I]$ relative to X (denoted by $[P,I]^X$) to be the revision program obtained from P by*

1. *removing every rule $r \in P$ such that body$(r) \cap \{l : (l^D \in X \setminus I^c) \wedge (l \notin X)\} \neq \emptyset$,*

2. *deleting from the body of each remaining rule any revision literal that is in* $\{l \in I^c : l^D \notin X\}$. △

Proposition 5 *Definition 16 and Definition 17 are equivalent.*

Proof.
Steps 1 in both definitions are the same, only in Definition 16 we use l^D to represent a literal while in Definition 17 we use l. Let us show that conditions in steps 2 of the definitions are equivalent when step 1 is executed. In Definition 17, $\{l \in I^c : l^D \notin X\} = I^c \setminus \{l \in I^c : (l^D \in X)\}$. Thus, we need to show that after execution of step 1, $(l^D \in X)$ is equivalent to $(l \in X) \wedge (l^D \in X)$. Let $r \in P$. Assume that r was not removed in step 1. Then, $\mathit{body}(r) \cap \{l : (l^D \in X \setminus I^c) \wedge (l \notin X)\} = \emptyset$. Assume that revision literal $l \in \mathit{body}(r) \cap I^c$. Since r was not removed at step 1 we have that condition $(l^D \in X \setminus I^c) \wedge (l \notin X)$ is not true. Assume that $l^D \in X$. Since $l \in I^c$, literal $l^D \notin I^c$. Therefore, $l^D \in X \setminus I^c$. Because condition $(l^D \in X \setminus I^c) \wedge (l \notin X)$ is false, we have $l \in X$. Therefore, after execution of step 1, condition $t^D \in X$ in step 2 implies $l \in X$. Thus, conditions in steps 2 of the definitions are equivalent when step 1 is executed. □

The intuitive meaning of these definitions of the $[P, I]^X$ reduct is as follows. A revision literal α ($\alpha = \mathbf{in}(a)$ or $\alpha = \mathbf{out}(a)$ for some $a \in U$) is satisfied by a justified revision if it is a head of a rule which fires while computing the revision or it is satisfied by initial database I and status of a does not change while computing the revision. We say that in the first case α is derived and in the second case α remains satisfied by inertia. This results in two different treatments of revision literals which are satisfied by I and occur in a body of a rule. If there is a possibility for such a literal to be derived then it remains in the body of the rule in the weak reduct. If there is no possibility for the literal to be derived then it is either removed from the body or the rule itself is removed from the reduct.

Definition 18 *For a revision program P and an initial database I, the operator $\gamma_{[P,I]}$ from sets of revision literals to sets of revision literals is defined by the equation*

$$\gamma_{[P,I]}(X) = NC([P, I]^X),$$

where $NC([P, I]^X)$ is the necessary change of $[P, I]^X$. △

In general, operator $\gamma_{[P,I]}$ is not anti-monotone as we can see from the following example.

Example 8 *Let $I = \emptyset$. Consider*

$$\begin{aligned} P: \quad \mathbf{out}(a) &\leftarrow \\ \mathbf{in}(b) &\leftarrow \mathbf{out}(a) \end{aligned}$$

Let $X_1 = \{\mathbf{in}(a)\}$ and $X_2 = \{\mathbf{in}(a), \mathbf{out}(a)\}$. Then, $\gamma_{[P,I]}(X_1) = \{\mathbf{out}(a)\}$ and $\gamma_{[P,I]}(X_2) = \{\mathbf{out}(a), \mathbf{in}(b)\}$. Clearly, $X_1 \subseteq X_2$ and $\gamma_{[P,I]}(X_1) \subseteq \gamma_{[P,I]}(X_2)$. Therefore, operator $\gamma_{[P,I]}$ is not anti-monotone.

Lemma 10 *If X is coherent, then $\gamma_{[P,I]}(X) = \gamma_{P,I}(X)$.*

Proof.
If $l \in X \setminus I^c$ and X is coherent, then $l^D \notin X$. Therefore, $\{l^D : (l \in X \setminus I^c) \wedge (l^D \notin X)\} = \{l^D : l \in X \setminus I^c)$. Also, $\{l \in I^c : (l \in X) \wedge (l^D \in X)\} = \emptyset$. Hence, $(P,I)^X = [P,I]^X$. This implies that $\gamma_{[P,I]}(X) = NC([P,I]^X) = NC((P,I)^X) = \gamma_{P,I}(X)$. □

Definition 19 *Let P be a revision program. Let I be an initial database. Let $X_0 = \emptyset$ and $X_k = \gamma_{[P,I]}(X_{k-1})$ for $k > 0$.* △

To illustrate the notion, let us compute a sequence of X_k for the following example.

Example 9 *Let $I = \emptyset$. Consider*

$$\begin{array}{rrcl} P: & \mathbf{out}(a) & \leftarrow & \\ & \mathbf{in}(a) & \leftarrow & \mathbf{out}(b) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \\ & \mathbf{in}(c) & \leftarrow & \mathbf{in}(a), \mathbf{in}(b) \\ & \mathbf{out}(d) & \leftarrow & \mathbf{out}(c) \\ & \mathbf{in}(d) & \leftarrow & \mathbf{out}(e) \\ & \mathbf{in}(e) & \leftarrow & \mathbf{out}(d) \\ & \mathbf{in}(f) & \leftarrow & \mathbf{in}(d), \mathbf{in}(e) \\ & \mathbf{out}(g) & \leftarrow & \mathbf{out}(f) \\ & \mathbf{in}(g) & \leftarrow & \mathbf{out}(h) \\ & \mathbf{in}(h) & \leftarrow & \mathbf{out}(g) \\ & \mathbf{in}(i) & \leftarrow & \mathbf{in}(g), \mathbf{in}(h) \end{array}$$

We have the following sequence:

$$X_0 = \emptyset,$$

$$X_1 = \{\mathbf{out}(a), \mathbf{in}(a), \mathbf{in}(b), \mathbf{in}(c), \mathbf{out}(d), \mathbf{in}(d), \mathbf{in}(e), \mathbf{in}(f), \mathbf{out}(g), \mathbf{in}(g), \mathbf{in}(h), \mathbf{in}(i)\},$$

$$[P,I]^{X_1} = \left\{ \begin{array}{rcl} \mathbf{out}(a) & \leftarrow & \\ \mathbf{in}(b) & \leftarrow & \mathbf{out}(a) \\ \mathbf{in}(c) & \leftarrow & \mathbf{in}(a), \mathbf{in}(b) \\ \mathbf{in}(e) & \leftarrow & \mathbf{out}(d) \\ \mathbf{in}(f) & \leftarrow & \mathbf{in}(d), \mathbf{in}(e) \\ \mathbf{in}(h) & \leftarrow & \mathbf{out}(g) \\ \mathbf{in}(i) & \leftarrow & \mathbf{in}(g), \mathbf{in}(h) \end{array} \right\}, X_2 = \{\mathbf{out}(a), \mathbf{in}(b)\},$$

$$[P,I]^{X_2} = \left\{ \begin{array}{rcl} \mathbf{out}(a) & \leftarrow & \\ \mathbf{in}(b) & \leftarrow & \\ \mathbf{in}(c) & \leftarrow & \mathbf{in}(a), \mathbf{in}(b) \\ \mathbf{out}(d) & \leftarrow & \\ \mathbf{in}(d) & \leftarrow & \\ \mathbf{in}(e) & \leftarrow & \\ \mathbf{in}(f) & \leftarrow & \mathbf{in}(d), \mathbf{in}(e) \\ \mathbf{out}(g) & \leftarrow & \\ \mathbf{in}(g) & \leftarrow & \\ \mathbf{in}(h) & \leftarrow & \\ \mathbf{in}(i) & \leftarrow & \mathbf{in}(g), \mathbf{in}(h) \end{array} \right\}, X_3 = \left\{ \begin{array}{c} \mathbf{out}(a), \\ \mathbf{in}(b), \\ \mathbf{out}(d), \\ \mathbf{in}(d), \\ \mathbf{in}(e), \\ \mathbf{in}(f), \\ \mathbf{out}(g), \\ \mathbf{in}(g), \\ \mathbf{in}(h), \\ \mathbf{in}(i) \end{array} \right\},$$

$$[P,I]^{X_3} = \left\{ \begin{array}{rcl} \mathbf{out}(a) & \leftarrow & \\ \mathbf{in}(b) & \leftarrow & \\ \mathbf{in}(c) & \leftarrow & \mathbf{in}(a), \mathbf{in}(b) \\ \mathbf{out}(d) & \leftarrow & \\ \mathbf{in}(e) & \leftarrow & \mathbf{out}(d) \\ \mathbf{in}(f) & \leftarrow & \mathbf{in}(d), \mathbf{in}(e) \\ \mathbf{in}(h) & \leftarrow & \mathbf{out}(g) \\ \mathbf{in}(i) & \leftarrow & \mathbf{in}(g), \mathbf{in}(h) \end{array} \right\}, X_4 = \left\{ \begin{array}{c} \mathbf{out}(a), \\ \mathbf{in}(b), \\ \mathbf{out}(d), \\ \mathbf{in}(e) \end{array} \right\},$$

$$[P,I]^{X_4} = \left\{ \begin{array}{rcl} \mathbf{out}(a) & \leftarrow & \\ \mathbf{in}(b) & \leftarrow & \\ \mathbf{in}(c) & \leftarrow & \mathbf{in}(a), \mathbf{in}(b) \\ \mathbf{out}(d) & \leftarrow & \\ \mathbf{in}(e) & \leftarrow & \\ \mathbf{in}(f) & \leftarrow & \mathbf{in}(d), \mathbf{in}(e) \\ \mathbf{out}(g) & \leftarrow & \\ \mathbf{in}(g) & \leftarrow & \\ \mathbf{in}(h) & \leftarrow & \\ \mathbf{in}(i) & \leftarrow & \mathbf{in}(g), \mathbf{in}(h) \end{array} \right\}, X_5 = \left\{ \begin{array}{c} \mathbf{out}(a), \\ \mathbf{in}(b), \\ \mathbf{out}(d), \\ \mathbf{in}(e), \\ \mathbf{out}(g), \\ \mathbf{in}(g), \\ \mathbf{in}(h), \\ \mathbf{in}(i) \end{array} \right\},$$

$$[P,I]^{X_5} = \left\{ \begin{array}{rcl} \mathbf{out}(a) & \leftarrow & \\ \mathbf{in}(b) & \leftarrow & \\ \mathbf{in}(c) & \leftarrow & \mathbf{in}(a), \mathbf{in}(b) \\ \mathbf{out}(d) & \leftarrow & \\ \mathbf{in}(e) & \leftarrow & \\ \mathbf{in}(f) & \leftarrow & \mathbf{in}(d), \mathbf{in}(e) \\ \mathbf{out}(g) & \leftarrow & \\ \mathbf{in}(h) & \leftarrow & \mathbf{out}(g) \\ \mathbf{in}(i) & \leftarrow & \mathbf{in}(g), \mathbf{in}(h) \end{array} \right\}, X_6 = \left\{ \begin{array}{c} \mathbf{out}(a), \\ \mathbf{in}(b), \\ \mathbf{out}(d), \\ \mathbf{in}(e), \\ \mathbf{out}(g), \\ \mathbf{in}(h) \end{array} \right\},$$

$$[P,I]^{X_6} = \left\{ \begin{array}{rcl} \mathbf{out}(a) & \leftarrow & \\ \mathbf{in}(b) & \leftarrow & \\ \mathbf{in}(c) & \leftarrow & \mathbf{in}(a), \mathbf{in}(b) \\ \mathbf{out}(d) & \leftarrow & \\ \mathbf{in}(e) & \leftarrow & \\ \mathbf{in}(f) & \leftarrow & \mathbf{in}(d), \mathbf{in}(e) \\ \mathbf{out}(g) & \leftarrow & \\ \mathbf{in}(h) & \leftarrow & \\ \mathbf{in}(i) & \leftarrow & \mathbf{in}(g), \mathbf{in}(h) \end{array} \right\}, X_7 = \left\{ \begin{array}{c} \mathbf{out}(a), \\ \mathbf{in}(b), \\ \mathbf{out}(d), \\ \mathbf{in}(e), \\ \mathbf{out}(g), \\ \mathbf{in}(h) \end{array} \right\}.$$

Since $X_6 = X_7$ we have $X_k = \{\mathbf{out}(a), \mathbf{in}(b), \mathbf{out}(d), \mathbf{in}(e), \mathbf{out}(g), \mathbf{in}(h)\}$ for $k \geq 6$.

Theorem 15 *Let P be a revision program. Let I be an initial database. Let R be a P-justified revision of I. Then, for all $k \geq 0$, the following holds. If $k = 2i$ then $X_{2i} \subseteq R^c$. If $k = 2i+1$ then $R^c \subseteq X_{2i+1} \cup I^c$.*

Proof.
By induction on k.

Basis step ($k = 0$). $X_0 = \emptyset \subseteq R^c$.

Inductive step. Assume that the statement of the theorem is true for $k-1$. We need to prove that it holds for k ($k \geq 1$).

Case 1. Let $k = 2i+1$. By inductive hypothesis, $X_{2i} \subseteq R^c$. Since R^c is coherent, X_{2i} is also coherent. By lemma 10, $X_{2i+1} = \gamma_{P,I}(X_{2i})$. Operator $\gamma_{P,I}$ is anti-monotone. Hence, $\gamma_{P,I}(R^c) \subseteq \gamma_{P,I}(X_{2i})$. Therefore, $\gamma_{P,I}(R^c) \cup I^c \subseteq \gamma_{P,I}(X_{2i}) \cup I^c = X_{2i+1} \cup I^c$. Let X be a set of heads of P_R. By Theorem 2, $R = I \oplus X$. Therefore, $R^c = X \cup (I^c \setminus X)$. By definition, $\gamma_{P,I}(R^c) = NC((P,I)^{R^c})$. In step 1 of computing the reduct $(P,I)^{R^c}$ every rule $r \in P$ such that $body(r) \cap \{l^D : l \in (R^c) \setminus I^c\} \neq \emptyset$. However, $\{l^D : l \in (R^c) \setminus I^c\} = \{l^D : l \in X \setminus I^c\}$. Hence, $(P,I)^{R^c} = (P,I)^X$. Thus, $\gamma_{P,I}(R^c) = NC((P,I)^{R^c}) = NC((P,I)^X) = \gamma_{P,I}(X)$. By Lemma 2, $\gamma_{P,I}(X) = X$. Therefore, $\gamma_{P,I}(R^c) = X$. We have:

$$R^c = X \cup (I^c \setminus X) \subseteq X \cup I^c = \gamma_{P,I}(R^c) \cup I^c \subseteq X_{2i+1} \cup I^c.$$

Hence, $R^c \subseteq X_{2i+1} \cup I^c$.

Case 2. Let $k = 2i+2$. By inductive hypothesis, $R^c \subseteq X_{2i+1} \cup I^c$. We need to prove that $X_{2i+2} \subseteq R^c$. By the definition of justified revisions and by Theorem 1, $NC(P_R|I)$ is coherent and $R = I \oplus NC(P_R|I)$. By definition, $X_{2i+2} = NC([P,I]^{X_{2i+1}})$. Let us compare the reducts $[P,I]^{X_{2i+1}}$ and $P_R|I$. In both reducts some rules are removed in step 1 and some literals from the bodies of the remaining rules are deleted in step 2.

In step 2 of computing the reduct $P_R|I$ all literals from I^c are deleted from the bodies of remaining rules. In step 2 of computing the reduct $[P,I]^{X_{2i+1}}$ some subset of literals in I^c is deleted. Therefore, if a rule r is *not* removed during step 1 of computing $P_R|I$ and $[P,I]^{X_{2i+1}}$, then its body in $P_R|I$ will be a subset of its body in $[P,I]^{X_{2i+1}}$.

Assume that a rule $r \in P$ was removed during step 1 of computing $P_R|I$, but was *not* removed during step 1 of computing $[P,I]^{X_{2i+1}}$. Then, by definition of $P_R|I$, there is a revision literal $t^D \in body(r)$ which is not satisfied by R. Thus, $t^D \notin R^c$ and $t \in R^c$. Since $R^c \subseteq X_{2i+1} \cup I^c$, we have that $t \in X_{2i+1} \cup I^c$. On the other hand, by definition of

$[P,I]^{X_{2i+1}}$, $t^D \notin \{l^D : (l \in X_{2i+1} \setminus I^c) \wedge (l^D \notin X_{2i+1})\}$. Therefore, either $t \notin (X_{2i+1} \setminus I^c)$ or $(t \in X_{2i+1} \setminus I^c) \wedge (t^D \in X_{2i+1})$.

Case 2.1: $t \notin (X_{2i+1} \setminus I^c)$. Hence, $t \notin X_{2i+1}$. However, $t \in X_{2i+1} \cup I^c$. Therefore, $t \in I^c$. Thus, $t^D \notin I^c$. This implies that t^D will remain in the body of rule r after step 2 of computing the reduct $[P,I]^{X_{2i+1}}$.

Case 2.2: $(t \in X_{2i+1} \setminus I^c) \wedge (t^D \in X_{2i+1})$. Hence, $t^D \in I^c$ and $(t \in X_{2i+1}) \wedge (t^D \in X_{2i+1})$. Therefore, $t^D \in \{l \in I^c : (l \in X_{2i+1}) \wedge (l^D \in X_{2i+1})\}$. This implies that t^D will remain in the body of rule r after step 2 of computing the reduct $[P,I]^{X_{2i+1}}$.

Therefore, every rule that was removed during step 1 of computing $P_R|I$, but was *not* removed during step 1 of computing $[P,I]^{X_{2i+1}}$ contains revision literals in its body that are *not* satisfied by R. For any such rule to fire during computation of $NC([P,I]^{X_{2i+1}})$ there must be a rule in $[P,I]^{X_{2i+1}}$ which allows to derive a literal that is not satisfied by R. Hence, there must exist a rule, say r, in $[P,I]^{X_{2i+1}}$, body of which contains only literals satisfied by R, and $head(r)$ is a literal that is not satisfied by R, which fires during computation of $NC([P,I]^{X_{2i+1}})$. From our comparisons of $P_R|I$ and $[P,I]^{X_{2i+1}}$ it follows that this rule r must be in $P_R|I$ as well and it must fire during computation of $NC(P^R|I)$. Therefore $head(r)$ must be satisfied by R. This contradicts the fact that $head(r)$ is not satisfied by R. Therefore, none of the rules that were removed during step 1 of computing $P_R|I$, but were *not* removed during step 1 of computing $[P,I]^{X_{2i+1}}$ fire during computation of $NC([P,I]^{X_{2i+1}})$. This together with our earlier observation that if a rule r is *not* removed during step 1 of computing $P_R|I$ and $[P,I]^{X_{2i+1}}$, then its body in $P_R|I$ will be a subset of its body in $[P,I]^{X_{2i+1}}$, implies that $NC([P,I]^{X_{2i+1}}) \subseteq NC(P^R|I)$. Consequently, $X_{2i+2} = NC([P,I]^{X_{2i+1}}) \subseteq NC(P^R|I) \subseteq R^c$. □

Corollary 5 *Let P be a revision program. Let I be an initial database. If there exists i such that X_{2i} contains a pair of dual revision literals, then no P-justified revisions of I exists.*

Theorem 15 and its corollary show that $\{X_k\}_{k=0,1,\ldots}$ can be used to approximate justified revisions or conclude that no justified revisions exist. Let $L = \bigcup_{i=0}^{\infty} X_{2i}$ and $U = \bigcap_{i=0}^{\infty} X_{2i+1}$. If R is a P-justified revision of I, then $U \subseteq R^c$ and $R^c \subseteq U \cup I^c$.

The following example illustrates a case when for some i, X_{2i} contains a pair of dual revision literals. Then the sequence $\{X_{2i}\}_{i=0,1,\ldots}$, in general, is not monotonic.

Example 10 *Let $I = \emptyset$. Consider*

$$\begin{array}{rrcl} P: & \mathbf{in}(a) & \leftarrow & \\ & \mathbf{in}(z) & \leftarrow & \mathbf{out}(a) \\ & \mathbf{in}(b) & \leftarrow & \mathbf{out}(z) \\ & \mathbf{in}(c) & \leftarrow & \mathbf{out}(b) \\ & \mathbf{out}(a) & \leftarrow & \mathbf{in}(b), \mathbf{out}(c) \end{array}$$

We have the following sequence:

$$X_0 = \emptyset,$$

$$X_1 = \{\mathbf{in}(a), \mathbf{in}(z), \mathbf{in}(b), \mathbf{in}(c), \mathbf{out}(a)\}, \},$$

$$[P, I]^{X_1} = \left\{ \begin{array}{rcl} \mathbf{in}(a) & \leftarrow & \end{array} \right\}, X_2 = \{\mathbf{in}(a)\},$$

$$[P, I]^{X_2} = \left\{ \begin{array}{rcl} \mathbf{in}(a) & \leftarrow & \\ \mathbf{in}(b) & \leftarrow & \\ \mathbf{in}(c) & \leftarrow & \\ \mathbf{out}(a) & \leftarrow & \mathbf{in}(b) \end{array} \right\}, X_3 = \{\mathbf{in}(a), \mathbf{in}(b), \mathbf{in}(c), \mathbf{out}(a)\},$$

$$[P, I]^{X_3} = \left\{ \begin{array}{rcl} \mathbf{in}(a) & \leftarrow & \\ \mathbf{in}(z) & \leftarrow & \mathbf{out}(a) \\ \mathbf{in}(b) & \leftarrow & \end{array} \right\}, X_4 = \{\mathbf{in}(a), \mathbf{in}(b)\},$$

$$[P, I]^{X_4} = \left\{ \begin{array}{rcl} \mathbf{in}(a) & \leftarrow & \\ \mathbf{in}(b) & \leftarrow & \\ \mathbf{out}(a) & \leftarrow & \mathbf{in}(b) \end{array} \right\}, X_5 = \{\mathbf{in}(a), \mathbf{in}(b), \mathbf{out}(a)\},$$

$$[P, I]^{X_5} = \left\{ \begin{array}{rcl} \mathbf{in}(a) & \leftarrow & \\ \mathbf{in}(z) & \leftarrow & \mathbf{out}(a) \\ \mathbf{in}(b) & \leftarrow & \\ \mathbf{out}(a) & \leftarrow & \mathbf{in}(b) \end{array} \right\}, X_6 = \{\mathbf{in}(a), \mathbf{in}(b), \mathbf{out}(a), \mathbf{in}(z)\},$$

$$[P, I]^{X_6} = \left\{ \begin{array}{rcl} \mathbf{in}(a) & \leftarrow & \\ \mathbf{out}(a) & \leftarrow & \mathbf{in}(b) \end{array} \right\}, X_7 = \{\mathbf{in}(a)\} = X_2, \quad X_8 = X_3.$$

Notice, that X_6 contains a pair of dual revision literals. This indicates that there are no P-justified revisions of I, which is indeed the case. The sequence $\{X_{2i}\}_{i=0,1,\ldots}$ is not monotonic since $X_6 \not\subseteq X_8$.

References

[ABW88] K.R. Apt, H.A. Blair, and A. Walker. Towards a theory of declarative knowledge. In J. Minker, editor, *Foundations of deductive databases and logic programming. Papers from the workshop held in Washington, D.C., August 18–22, 1986*, pages 89–148, Palo Alto, CA, 1988. Morgan Kaufmann.

[BSJ95] K. Berman, J. Schlipf, and J.Franco. Computing the well-founded semantics faster. In *Logic Programming and Nonmonotonic Reasoning (Lexington, KY, 1995)*, volume 928 of *Lecture Notes in Computer Science*, pages 113–125, Berlin, 1995. Springer.

[GL88] M. Gelfond and V. Lifschitz. The stable semantics for logic programs. In R. Kowalski and K. Bowen, editors, *Proceedings of the 5th International Symposium on Logic Programming*, pages 1070–1080, Cambridge, MA, 1988. MIT Press.

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

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

[MT95] W. Marek and M. Truszczyński. Revision programming, database updates and integrity constraints. In *Proceedings of the 5th International Conference on Database Theory — ICDT 95*, volume 893 of *Lecture Notes in Computer Science*, pages 368–382. Berlin: Springer-Verlag, 1995.

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

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

[PT97] T. C. Przymusinski and H. Turner. Update by means of inference rules. *Journal of Logic Programming*, 30(2):125–143, 1997.

[VRS88] A. Van Gelder, K.A. Ross, and J.S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *Proceedings of the Seventh ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 1988) (March 21-23, 1988, Austin, Texas)*, pages 221–230, New York, 1988. ACM Press.

[VRS91] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.