Reconsidering AGM-Style Belief Revision in the Context of Logic Programs

Zhiqiang Zhuang\(^1\) and James Delgrande\(^2\) and Abhaya Nayak\(^3\) and Abdul Sattar\(^4\)

Abstract.

Belief revision has been studied mainly with respect to background logics that are monotonic in character. In this paper we study belief revision when the underlying logic is non-monotonic instead—an inherently interesting problem that is under explored. In particular, we will focus on the revision of a body of beliefs that is represented as a logic program under the answer set semantics, while the new information is also similarly represented as a logic program. Our approach is driven by the observation that unlike in a monotonic setting where, when necessary, consistency in a revised body of beliefs is maintained by jettisoning some old beliefs, in a non-monotonic setting consistency can be restored by adding new beliefs as well. We will define two revision functions through syntactic and model-theoretic methods respectively and subsequently provide representation theorems for characterising them.

1 Introduction

The ability to change one’s beliefs when presented with new information is crucial for any intelligent agent. In the area of belief change, substantial effort has been made towards the understanding and realisation of this process. Traditionally, it is assumed that the agent’s reasoning is governed by a monotonic logic. For this reason, traditional belief change is inapplicable when the agent’s reasoning is non-monotonic. Our goal in this research program is to extend the established (AGM) belief set \(^1\) and belief base \(^2\) approaches in belief revision to nonmonotonic setting. In this paper, we focus on disjunctive logic programs, as a well-studied and well-known approach to nonmonotonic reasoning that also has efficient implementations.

Much, if not most, of our day-to-day reasoning involves non-monotonic reasoning. To illustrate issues that may arise, consider the following example. In a university, professors generally teach, unless they have an administrative appointment. Assume we know that John is a professor. Since most faculty do not have an administrative appointment, and there is no evidence that John does, we conclude that he teaches. This reasoning is a classical form of non-monotonic reasoning, namely using the closed world assumption. It can be represented by the following logic program under the answer set semantics.

\[
\begin{align*}
Teach(X) & \leftarrow Prof(X), not\ Admin(X). \\
Prof(John) & \leftarrow.
\end{align*}
\]

The answer set \(\{Prof(John), Teach(John)\}\) for this logic program corresponds exactly to the facts we can conclude.

Suppose we receive information that John does not teach, which we can represent by the rule

\[
Teach(John).
\]

Now our beliefs about John are contradictory; and it is not surprising that the logic program consisting of rules (1) – (3) has no answer set. For us or any intelligent agent in this situation to function properly, we need a mechanism to resolve this inconsistency. This is a typical belief revision problem; however, the classical (AGM) approach cannot be applied, as we are reasoning non-monotonically.

It is not hard to suggest possible causes of the inconsistency and to resolve it. It could be that some of our beliefs are wrong; perhaps professors with administrative duties may still need to do teaching or perhaps John is not a professor. Thus we can restore consistency by removing rule (1) or (2). Alternatively and perhaps more interestingly, it could be that assuming that John is not an administrative staff via the absence of evidence is too adventurous; that is he may indeed be an administrative staff member but we don’t know it. Thus we can also restore consistency by adding the missing evidence of John being an administrative staff member by

\[
Admin(John) \leftarrow.
\]

The second alternative highlights the distinction for belief revision in monotonic and non-monotonic settings. In the monotonic setting, an inconsistent body of knowledge will remain inconsistent no matter how much extra information is supplied. On the other hand, in the non-monotonic setting, inconsistency can be resolved by either removing old information, or adding new information, or both. Therefore, belief revision functions in a non-monotonic setting should allow a mixture of removal and addition of information for inconsistency-resolution. In this paper, we will define two such revision functions for disjunctive logic programs under the answer set semantics.

Our first revision function called slp-revision\(^3\) is like belief base revision which takes syntactic information into account. In revising \(P\) by \(Q\), a slp-revision function first obtains a logic program \(R\) that is consistent with \(Q\) and differs minimally from \(P\), then combines \(R\) with \(Q\). For example, if \(P = \{(1), (2)\}\) and \(Q = \{(3)\}\), then \(R\) could be \(\{(1)\}\) (i.e., resolving inconsistency by removing \(2\)); \(\{(2)\}\) (i.e., resolving inconsistency by removing \(1\)); or \(\{(1), (2), (4)\}\) (i.e., resolving inconsistency by adding \(4\)). Our second revision function called lrp-revision function\(^4\) is like AGM belief set revision which ignores syntactic difference and focuses on the logical content of a knowledge base. So in revising \(P\) by \(Q\), a lrp-revision function

---

1 IIIS, Griffith University, Australia
2 School of Computing Science, Simon Fraser University, Canada
3 Department of Computing, Macquarie University, Australia
4 “s” stands for syntactic and “lp” for logic program.
5 “f” stands for logical content and “lp” for logic program.
will instead obtain a logic program $R$ whose logical content differ the least from that of $P$ where the logical content is characterised by strong equivalent (SE) models [27].

The next section gives logical preliminaries. The following two sections develop our approach to slp-revision and llp-revision, in each case providing postulates, a semantic construction, and a representation result. This is followed by a comparison to other work, and a brief conclusion.

2 Preliminary Considerations

In this paper, we consider only fully grounded disjunctive logic programs. Thus a logic program (or program for short) here is a finite set of rules of the form:

$$a_1;\ldots;a_m \leftarrow b_1,\ldots,b_n, \text{not } c_1,\ldots,\text{not } c_o$$

where $m, n, a \geq 0$, $m + n + a > 0$, and $a_i, b_j, c_k \in A$ for $A$ a finite set of propositional atoms. We denote the set of all logic programs by $P$. For each rule $r$, let $H(r) = \{a_1,\ldots,a_m\}$, $B^+(r) = \{b_1,\ldots,b_n\}$, and $B^-(r) = \{c_1,\ldots,c_o\}$. The letters $P, Q$ and $R$ are used to denote a logic program throughout the paper.

An interpretation is represented by the subset of atoms in $A$ that are true in the interpretation. A classical model of a program $P$ is an interpretation in which all rules of $P$ are true according to the standard definition of truth in propositional logic, and where default negation is treated as classical negation. The set of classical models is denoted as $\text{Mod}(P)$. An interpretation in which all rules of a program hold is called a classical model of $P$. The set of all classical models of $P$ is denoted as $\text{Cl}(P)$. A program $P$ is consistent if and only if $\text{Mod}(P) \neq \emptyset$.

3 SLP-Revision Functions

In this section, we give a syntax-based revision function $* : P \times P \rightarrow P$ for revising one logic program by another. The function takes a logic program $P$ called the original logic program and a logic program $Q$ called the revising logic program, and returns another logic program $P*Q$ called the revised logic program. Following AGM belief revision, we want to have $Q$ contained in $P*Q$ (i.e., $Q \subseteq P*Q$) and $P*Q$ is consistent whenever possible.

A main task in defining $*$ is to deal with the possible inconsistency between $Q$ and $P$. As illustrated in the teaching example, one means of ensuring that $P*Q$ is consistent is to remove a minimal set of beliefs from $P$ so that adding $Q$ to the result is consistent. Of course there may be more than one way to remove beliefs from $P$. Following this intuition, we obtain all maximal subsets of $P$ that are consistent with $Q$, which we call the s-removal compatible programs of $P$ with respect to $Q$.

Definition 2. The set of s-removal compatible programs of $P$ with respect to $Q$, denoted $P \downarrow Q$, is such that $R \in P \downarrow Q$ iff

1. $R \subseteq P$.
2. $R \cup Q$ is consistent, and
3. if $R \subseteq R' \subseteq P$, then $R' \cup Q$ is inconsistent.

The notion of s-removal compatible programs is not new, classical revision functions [1, 11] are based on more or less the same notion. The difference is that this notion alone is sufficient to capture the inconsistency-resolution strategy of classical belief revision, but there is more that one can do in non-monotonic belief revision.

In our non-monotonic setting, we are able to express assumptions (i.e., negation as failure) and to reason with them. Earlier, we assumed John is not an administrator, in the absence of evidence to the contrary. With this, we came to the conclusion that he has to teach.

6 “m” stands for “monotonic” which indicates that the notion of m-consistency is based on a monotonic characterisation (i.e., SE models) for logic programs.
Consequently, if we learn that John does not teach, as in our example, one way of resolving this inconsistency is by adding a minimal set of information so that our assumption does not hold. Following this intuition, we obtain all the minimal supersets of $P$ that are consistent with $Q$, which we call the s-compatible programs of $P$ with respect to $Q$.

**Definition 3.** The set of s-expansion compatible programs of $P$ with respect to $Q$, denoted $P \uparrow Q$, is such that $R \in P \uparrow Q$ iff

1. $P \subseteq R$,
2. $R \cup Q$ is consistent, and
3. if $P \subseteq R' \subset R$, then $R' \cup Q$ is inconsistent.

Since the s-expansion and s-removal compatible programs are consistent with $Q$ and are obtained by removing or adding minimal sets of rules from or to $P$, the union of $Q$ with any of these sets is consistent and comprises a least change made to $P$ in order to achieve consistency. These programs clearly should be candidates for forming the revised logic program $P \ast Q$; however, they do not form the set of all candidates. In particular, we can obtain a program that differs the least from $P$ and is consistent with $Q$ by removing some beliefs of $P$ and at the same time adding some new beliefs to $P$. Thus we consider all those logic programs that differ the least from $P$ and are consistent with $Q$; these are called the s-compatible programs of $P$ with respect to $Q$.

**Definition 4.** The set of s-compatible programs of $P$ with respect to $Q$, denoted $P \uparrow Q$, is such that $R \in P \uparrow Q$ iff

1. $R \cup Q$ is consistent and
2. if $P \subseteq R' \subset P$, then $R' \cup Q$ is inconsistent.

For example, let $P = \{a \leftarrow b, \neg c, b, e \leftarrow f, \neg g, a, f\}$ and $Q = \{\neg a, \neg e\}$. Then $P \cup Q$ is inconsistent since $a$ and $e$ can be concluded from $P$ but they contradict the rules of $Q$. To resolve the inconsistency via making the least change to $P$, we could remove $b \leftarrow f$ (which eliminates the contradiction about $b$) and add $g \leftarrow P$ (which eliminates the contradiction about $e$). The program thus obtained (i.e., $(P \setminus \{b, f\}) \cup \{g\}$) is a s-compatible program in $P \uparrow Q$.

It is obvious, but worth noting that the notion of s-compatible program subsumes those of s-removal and s-expansion compatible programs. In the above example, $P \uparrow Q$ also contains $P \setminus \{b, f\}$ and $P \cup \{c, g\}$, which are respectively an s-removal and an s-expansion compatible program of $P$ with respect to $Q$.

**Proposition 1.** $(P \uparrow Q) \cup (P \downarrow Q) \subseteq P \uparrow Q$.

There are cases in which we cannot resolve inconsistency by only adding new beliefs which means the set of s-expansion compatible programs is empty. For example, if $P = \{a\}$ and $Q = \{\neg a\}$, then $P \cup Q$ is inconsistent and we cannot resolve consistency without removing $a$ from $P$. In these cases, the inconsistency is due to contradictory facts that can be concluded without using any reasoning power beyond that of classical logic. Clearly, the inconsistency is of a monotonic nature, that is, in our terminology, m-inconsistency.

**Proposition 2.** If $P \cup Q$ is m-inconsistent, then $P \uparrow Q = \emptyset$.

So far, we have identified the candidates for forming $P \ast Q$. It remains to pick the “best” one. Such extralogical information is typically modelled by a selection function, which we do next.

**Definition 5.** A function $\gamma$ is a selection function for $P$ iff for any program $Q$, $\gamma(P \uparrow Q)$ returns a single element of $P \uparrow Q$ whenever $P \uparrow Q$ is non-empty; otherwise it returns $P$.

The revised logic program $P \ast Q$ is then formed by combining $Q$ with the s-compatible program picked by the selection function for $P$. We call the function $\ast$ defined in this way a slp-revision function for $P$.

**Definition 6.** A function $\ast$ is a slp-revision function for $P$ iff

\[ P \ast Q = \gamma(P \uparrow Q) \cup Q \]

for any program $Q$, where $\gamma$ is a selection function for $P$.

In classical belief revision, multiple candidates maybe chosen by a selection function, and their intersection is combined with the new belief to form the revision result. There, a selection function that picks out a single element is called a maxichoice function [1]. In classical logic, maxichoice selection functions lead to undesirable properties for belief set revision but not for belief base revision. In our non-monotonic setting, picking multiple candidates does not make sense, as intersection of s-compatible programs may not be consistent with the revising program. For example, let $P = \{a \leftarrow \neg b, \neg c\}$ and $Q = \{\neg a\}$. We can restore consistency of $P$ with $Q$ by, for instance, adding the rule $b \leftarrow P$ which corresponds to the s-compatible program $P \cup \{b\}$ or by adding the rule $c \leftarrow P$ which corresponds to the s-compatible program $P \cup \{c\}$. However, the intersection of the two s-compatible programs is inconsistent with $Q$.

We turn next to properties of slp-revision functions. Consider the following set of postulates where $\ast : P \times P \rightarrow P$ is a function.

\[ \text{(s}s\text{s})\text{ }Q \subseteq P \ast Q \]

\[ \text{(s}c\text{)}\text{ }\text{If } Q \text{ is m-consistent, then } P \ast Q \text{ is consistent} \]

\[ \text{(s}f\text{)}\text{ }\text{If } Q \text{ is m-inconsistent, then } P \ast Q = P \cup Q \]

\[ \text{(s}r\text{R)}\text{ If } \emptyset \neq R \subseteq P \setminus (P \ast Q) \text{, then } (P \ast Q) \cup R \text{ is inconsistent} \]

\[ \text{(s}e\text{R)}\text{ If } \emptyset \neq E \subseteq (P \ast Q) \setminus (P \cup Q) \text{, then } (P \ast Q) \setminus E \text{ is inconsistent} \]

\[ \text{(s}m\text{R)}\text{ If } \emptyset \neq R \subseteq P \setminus (P \ast Q) \text{ and } \emptyset \neq E \subseteq (P \ast Q) \setminus (P \cup Q) \text{, then } ((P \ast Q) \cup R) \setminus E \text{ is inconsistent} \]

\[ \text{(s}u\text{)}\text{ If } P \uparrow Q = P \downarrow R, \text{ then } P \setminus (P \ast Q) = P \setminus (P \ast R) \text{ and } (P \ast Q) \setminus (P \cup Q) = (P \ast R) \setminus (P \cup R) \]

\[ \text{(s}s\text{s})\text{ (Success) states that a revision is always successful in incorporating the new beliefs. (s}c\text{) (Consistency) states that a revision ensures consistency of the revised logic program whenever possible. In the monotonic setting, a revision results in inconsistency only when the new beliefs are themselves inconsistent. This is not the case in the non-monotonic setting. For example, consider the revision of } P = \{a\} \text{ by } Q = \{b \leftarrow \neg b\}. \text{ Although } Q \text{ is inconsistent, we have } P \cup \{b\} \text{ as a s-compatible program of } P \text{ with respect to } Q. \text{ Thus we can have } P \cup \{b\} \cup Q \text{ as the revised logic program, which contains } Q \text{ and is consistent. Here, a revision results in inconsistency only when the revising logic program is m-inconsistent. In such a case, (s}f\text{) (Failure) states that the revision corresponds to the union of the original and revising logic program.} \]

\[ \text{(s}r\text{R)}\text{ (Removal Relevance) states that if some rules are removed from the original logic program for the revision, then adding them to the revised logic program results in inconsistency. It captures the intuition that nothing is removed unless its removal contributes to making the revised logic program consistent.} \]

\[ \text{(s}e\text{R)}\text{ (Expansion Relevance) states that if some new rules other than those in the revising logic program are added to the original logic program for the revision, then removing them from the revised logic program results in inconsistency.} \]

\[ \text{(s}m\text{R)}\text{ (Mixed Relevance) states that if some rules are removed from} \]


the original logic program and some new rules other than those in the revising logic program are added to the original logic program for the revision, then adding back the removed ones and removing the added ones result in inconsistency. Its intuition is a mixture of the two above. Note that putting \((s \cap r)\) and \((s \cup r)\) together does not guarantee \((s \cap m)\), nor the reverse. In summary, these three postulates express the necessity of adding and/or removing certain belief for resolving inconsistency and hence to accomplish a revision. In classical belief revision, inconsistency can only be resolved by removing old beliefs; the necessity of removing particular beliefs is captured by the \(s\) postulate \([11]\). ¹ The three postulates are the counterparts of \(R\) in our non-monotonic setting, and we need all three of them to deal respectively with addition, removal, and a mixture of addition and removal.

Finally, \((s \cup a)\) \(\text{(Uniformity)}\) states the condition under which two revising logic programs \(Q\) and \(R\) trigger the same changes to the original logic program \(P\). That is the rules removed from \(P\) \(\text{(i.e., } P \setminus (P + Q))\) and the rules added to \(P\) \(\text{(i.e., } (P + Q) \setminus (P \cup Q))\) for accommodating \(Q\) are identical to those for accommodating \(R\). Certainly having \(Q\) and \(R\) be strongly equivalent \(\text{(i.e., } SE(Q) = SE(R))\) is a sufficient condition. However, it is too strong a requirement. Suppose \(P = \{a \leftarrow b\}, Q = \{a\}\), and \(R = \{a \leftarrow b, b\}\). Then the minimal change to \(P\) we have to make to accommodate \(Q\) and \(R\) are the same, that is we remove \(-a\). However \(Q\) and \(R\) are not strongly equivalent, even though they incur the same change to \(P\). The essential point of this example is that instead of a global condition like strong equivalence, we need a condition that is local to the original logic program \(P\). Unfortunately, it seems there is no existing notion in the logic programming literature that captures this local condition. Thus we use our newly defined notion of \(s\)-compatible programs and come up with the local but more appropriate condition in \((s \cup a)\).

We can show that these postulates are sufficient to characterise all \(s\)-revision functions.

**Theorem 1.** A function \(*\) is a \(s\)-revision function iff it satisfies \((s \cap a), (s \cap c), (s \cap f), (s \cup r), (s \cup e),\) \((s \cap m)\), and \((s \cup a)\).

### 4. LLP-Revision Functions

\(s\)-revision functions preserve the syntactic structure of the original logic program as much as possible. Thus is most useful for scenarios in which syntactic information is prioritised over that of logical content. In this section, we provide a revision function called \(\text{lp} \cap \text{revision function}\) that prioritises the preservation of logical content over that of syntactic information.

The main strategy of \(\text{lp} \cap \text{revision functions}\) is the same as that of \(\text{slp} \cap \text{revision functions}\), which is to first obtain a logic program that differs the least from the original one and that is consistent with the revising one, and then combine it with the revising program. The distinguishing feature of \(\text{lp} \cap \text{revision}\) is in the interpretation of “differs the least”. \(\text{slp} \cap \text{revision}\) interprets this notion as symmetric difference between the constituent rules whereas \(\text{lp} \cap \text{revision}\) interprets it as between the logical content. Since the standard approach in characterising the logical content of logic programs is through their SE models, the difference between logical content is represented as the difference between sets of SE models. This brings about the following dual notion of \(s\)-compatible program which we call \(|\text{l-compatible program}|\).

Note that although we are using SE models, we concern

with the stronger notion of consistency (i.e. \(P\) is consistent if it has an answer set).

**Definition 7.** The set of \(|\text{l-compatible programs of } P\) with respect to \(Q\), denoted \(P \uplus Q\), is such that \(R \in P \uplus Q\) iff

1. \(R = cl(R)\),
2. \(R \cup Q\) is consistent, and
3. if \(SE(P) \cap SE(R^*) \subseteq SE(P) \cap SE(R)\), then \(R^* \cup Q\) is inconsistent.

A \(l\)-compatible program of \(P\) with respect to \(Q\) is a logic program that is consistent with \(Q\) \(\text{(condition 1)}\) and whose set of SE models differ minimally from that of \(P\) \(\text{(condition 2)}\). Moreover, since we ignore syntactic difference, the logic program is closed \(\text{(condition 1)}\). We denote by \(P \uplus Q\) and \(P \uplus Q\) the set of \(l\)-removal compatible programs and \(l\)-expansion compatible programs such that \(R \in P \uplus Q\) iff \(SE(P) \subseteq SE(R)\) and \(R \in P \uplus Q\) iff \(SE(P) \subseteq SE(R)\) and \(P \in R \uplus Q\).

We have given a declarative definition for \(l\)-compatible programs. The following theorem serves as a constructive one which is crucial for investigating the behaviour of \(\text{lp} \cap \text{revision functions}\). The theorem identifies the set of SE models for a \(l\)-compatible program under all possible situations.

**Theorem 2.** \(R \in P \uplus Q\) iff \(R = cl(R)\) and one of the following conditions holds:

1. \(P \cup Q\) is inconsistent. \(SE(R) = Cl((SE(P) \cup \{(W, Y)\}) \setminus M)\) where \((W, Y) \in SE(Q) \cap SE(P)\) and \(M = \{(X, Z) \in SE(P) | (X, Y) \in SE(Q) \cap SE(P)\} \setminus Z \subseteq Y\).
2. \(P \cup Q\) is inconsistent but \(m\)-consistent. \(SE(R) = SE(P) \setminus M\) where \((W, Y) \in SE(P) \cap SE(Q)\) such that \((W, Y) \in SE(P) \cap SE(Q)\) implies \(W \not\subseteq Y\) and \(M = \{(X, Z) \in \{X \in SE(P) | (X, Y) \in SE(Q) \cap SE(P)\} | X \neq Y\) and \(Z \subseteq Y\) \(\)(\(3\)).
3. \(P \cup Q\) is consistent. \(SE(R) = SE(P)\).

Due to the mixture of the minimality (i.e., for any \(l\)-compatible program \(R, SE(P) \subseteq SE(R)\) is minimal among programs consistent with \(P\)) and completeness requirement (i.e., \(SE(R)\) is complete), it is a challenging task identifying such models. We will explain the process with the aid of Figure 1 which gives a visualisation of Theorem 2.

In Figure 1, a rectangle represents the space of all SE interpretations. In each rectangle, the left circle represents the SE models of \(P\) and the right one represents those of \(Q\). When the two circles intersect as in diagram \((1.1), (1.3), (2)\) and \((4)\), it means the SE models of \(P\) intersect with those of \(Q\). Finally, the shaded area in each rectangle represents the SE models of a \(l\)-compatible program and the black dot represents the SE model \((Y, Y)\) such that \(Y\) is the answer set of the \(l\)-compatible program (i.e., the SE model \((Y, Y)\) in Theorem 2).

Condition 1 of Theorem 2 corresponds to diagrams \((1.1)\) – \((1.4)\) in which the SE models of the \(l\)-compatible program consists of the closure of all or part of \(SE(P)\) and an SE model \((Y, Y)\) from \(SE(Q) \setminus SE(P)\). The set \(M\) corresponds to the area in \(SE(P)\) that is not intersecting with the shaded area. Such \(M\) is empty for \((1.1)\) – \((1.2)\) indicating that \((Y, Y)\) is an answer set in \(Cl(SE(P) \cup \{(W, Y)\})\) and non-empty for \((1.3)\) – \((1.4)\) indicating that \((Y, Y)\) is not an answer set in \(Cl(SE(P) \cup \{(W, Y)\})\) but will be one after the removal of \(M\) from \(SE(P)\). To see why all elements of \(M\) have to be removed, suppose \((X, Z) \in M\) is not removed. Then we have \(\{(Y, Y), (X, Z)\} \subseteq SE(R) \cap SE(Q) = Cl(((SE(P) \cup \{(Y, Y)\})\)
SE models of the l-compatible program are contained in SE compatible programs; that in diagram (2) is a l-expansion compatible that the l-compatible programs in diagrams (1.1) – (1.2) are l-removal condition 3 corresponds to diagram (3) in which the SE models of the (tent, we don't have to make any change to K l-compatible programs in these diagrams do not contain all those of is evident in diagram (1.3), (1.4) and (2). Since the SE models of the other options that are characteristic of our non-monotonic setting. In consistent with P of postulates where A function is a llp-revision function for (l∗) is the “best” l-compatible program. Then the closure of the chosen l-

\[ P \cap \{X, Z\} \in SE(Q), \text{ thus } (Y, Y') \text{ no longer leads to an answer set for } R. \]

Condition 2 corresponds to diagram (2) in which the SE models of the l-compatible program are contained in SE(P). As for condition 1, the set M corresponds to the area in SE(P) that is not intersecting with the shaped area and the removal of M is necessary for guaranteeing (Y, Y') is an answer set in SE(R). Condition 3 corresponds to diagram (3) in which the SE models of the l-compatible program consists all of SE(P). Since P ∪ Q is consistent, we don’t have to make any change to SE(P). It is easy to see that the l-compatible programs in diagrams (1.1) – (1.2) are l-removal compatible programs; that in diagram (2) is a l-expansion compatible program; and that in diagram (4) is a l-removal compatible program as well as a l-expansion compatible program.

In the AGM setting, a belief set K is inconsistent with another one K' iff the (classical) models of K disjoint with those of K'. So if a belief set K'' differs minimally from K, but is consistent with K', then the models of K'' consist of the models of K together with a single model of K. In our non-monotonic setting, P can be inconsistent with Q even though the (SE) models of P intersect with those of Q. In this setting, there are more options for the program to be consistent with Q and such that its models differ minimally from those of P. Diagram (1.2) represents the counterpart of the only option in the AGM setting. Diagrams (1.1), (1.3), (1.4) and (2) represent the other options that are characteristic of our non-monotonic setting. In particular, the inconsistency resolving strategy by adding new rules is evident in diagram (1.3), (1.4) and (2). Since the SE models of the l-compatible programs in these diagrams do not contain all those of P, the l-compatible programs must contains rules outside of P.

Now we give the definition of llp-revision functions. As for slp-revision functions, a selection function γ is assumed for choosing the “best” l-compatible program. Then the closure of the chosen l-compatible program and the revising logic program is returned as the revised logic program.

**Definition 8.** A function * is a llp-revision function for P iff

\[ P * Q = cl(\gamma(P \downarrow \downarrow Q) \cup Q) \]

for any program Q, where γ is a selection function for P.

For properties of llp-revision functions, consider the following set of postulates where * : \( P \times P \rightarrow P \) is a function.

1. \( P * Q = cl(P * Q) \)
2. \( Q \subseteq P * Q \)
3. If Q is m-consistent, then \( P * Q \) is consistent

\[ \text{if } Q \text{ is m-inconsistent, then } P * Q = cl(P \cup Q) \]

\[ \text{if } \emptyset \neq M \subseteq (SE(P) \cap SE(Q)) \setminus SE(P * Q), \text{ then } SE(P * Q) \cup M \text{ is inconsistent.} \]

\[ \text{if } \emptyset \neq N \subseteq SE(P \cup Q) \setminus SE(P * Q), \text{ then either } SE(P * Q) \cup N \text{ is inconsistent or } CL(SE(P * Q) \setminus N) = SE(P * Q). \]

\[ \text{if } \emptyset \neq M \subseteq (SE(P) \cap SE(Q)) \setminus SE(P * Q) \text{ and } \emptyset \neq N \subseteq SE(P * Q) \setminus SE(P), \text{ then } (SE(P * Q) \cup M) \cup N \text{ is inconsistent.} \]

\[ \text{if } P \nsubseteq Q = P \nsubseteq R; \text{ then } CL((SE(P) \cup SE(P * Q)) \setminus M) = CL((SE(P) \cup SE(P * R)) \setminus N) \text{ for } M = \{(X, Z) \in SE(P)/|CL(SE(P * Q) \cup \{(X, Z)\}) \setminus SE(Q) \text{ is inconsistent} \}

\[ \text{and } N = \{(X, Z) \in SE(P)/|CL(SE(P * R) \cup \{(X, Z)\}) \setminus SE(R) \text{ is inconsistent} \}. \]

\( P \downarrow \downarrow Q \) (Closure) states that the revised logic program is closed. (l*s), (l*c) and (l*f) are the same as their counterparts for slp-revision function, except for (l*f) in which the revised logic program has to be closed. Since (l*s) requires that \( SE(P * Q) = \) a subset of \( SE(Q) \), SE models outside of \( SE(Q) \) is of no concern. Then the principle of minimal change dictates that as much as possible the intersecting models of P and Q are preserved and as least as possible the models outside of \( SE(P) \) are added. (l*rr) states that if a subset M of \( SE(P) \cap SE(Q) \) is not in \( SE(P * Q) \), then adding M to \( SE(P * Q) \) results in inconsistency. The postulate captures the intuition that no intersecting models of P and Q is excluded in the revision unless this contributes to the consistency of \( P \cup Q \). (l*e) states that if a subset N of \( SE(P * Q) \) is not in \( SE(P) \), then removing N from \( SE(P * Q) \) results in inconsistency or its inclusion in \( SE(P * Q) \) is required for \( SE(P * Q) \) to be closed under completeness. Recall that the set of SE models of any logic program has to be closed under completeness. The postulate captures the intuition that no SE models outside \( SE(P) \) is included in those of \( P * Q \) unless this contributes to the consistency of \( P \cup Q \) or to the closure of the set of SE models of \( P * Q \). (l*mr) is a mixture of (l*rr) and (l*e), capturing the necessity of exclusion of SE models in \( SE(P) \cap SE(Q) \) and inclusion of SE models not in \( SE(P) \). Finally, (l*u) states that the selection function that determines a llp-revision function is indeed a function, that is if \( P \nsubseteq Q = P \nsubseteq R \) then \( \gamma(P \nsubseteq Q) = \gamma(P \nsubseteq R) \) for \( \gamma \) a selection function. It follows from the other postulates that \( CL((SE(P) \cup SE(P * Q)) \setminus M) = SE(\gamma(P \nsubseteq Q)) \text{ and } \)

\[ CL((SE(P) \cup SE(P * R)) \setminus N) = SE(\gamma(P \nsubseteq R)) \text{. The postulate is not as neat as its counterpart } (s*su) \text{ for slp-revision functions. However the role of } (s*su) \text{ in characterising slp-revision functions is the exactly the same as } (l*u) \text{ in characterising llp-revision functions. We can show that these postulates are sufficient to characterise all llp-revision functions.} \]

**Theorem 3.** A function * is a llp-revision function iff * satisfies (l*cl), (l*s), (l*c), (l*f), (l*rr), (l*e), (l*mr), and (l*u).

### 5 Related Work

provide a variant of partial meet revision and contraction for logic programs.

Comparing with our llp-revision function which also makes use of SE models, these approaches assume a weaker notion of consistency, that is m-consistency. For this reason, some contradictions will not be dealt with in these approaches. For instance, the contradictory rule $a \leftarrow \neg a$ is m-consistent thus is considered to be an acceptable state of belief. Also in our teaching example, as the program consisting of rules (1) – (3) is m-consistent, no attempt will be made to resolve the contradiction about John’s teaching duty by the SE model approaches. Therefore for application scenarios in which such contradictions can not be tolerant, our llp-revision function is clearly a better choice. It worth noting that Slota and Leite [24, 26] argue that SE model approaches. llp-revision functions also suffer from this defect of SE models, however our slp-revision function, since it is purely syntactic, avoids all such undesirable properties.

Apart from the SE model approaches, Krümpelmann and Kern-Ibserner [16] provide a revision function for logic programs that originates from Hansson’s semi-revision [12]. For reference we call the revision function base revision function as the revision they considered is belief base revision. Since they assume the same notion of consistency as ours, all the above mentioned contradictions will be resolved in their approach. As we have noted, classical belief revision is defined for monotonic setting, not for non-monotonic ones. Inconsistency can be caused by wrong assumptions in the non-monotonic setting but not in the monotonic setting. Such causes are not considered in [16]. Consequently, their approach only supports one of the many possible inconsistency-resolution strategies we have developed. Specifically, in [16], inconsistency can be resolved only by removing old beliefs; this strategy is captured by a notion analogous to s-revision compatible programs. The inconsistency-resolution strategies captured by the notion of s-expansion compatible program and s-compatible program in general are not considered.

A group of work under the title of update [2, 6, 9, 17, 20, 21, 28, 19, 29, 30] also deals with changes of logic programs. The update however is different from the Katsuno and Mendelzon style update [14]. Following [2, 17], a typical problem setting is to consider a sequence $P_1, P_2, \ldots, P_n$ of programs such that $1 \leq j < j < n$ implies $P_j$ has higher priority over $P_j$. The goal of the update then is to obtain a set of answer sets from such a program sequence that in some sense respects the priority ordering. Clearly, these approaches have very different focus from ours and from those of [16, 7, 24, 25, 26, 5] in which a single new logic program is returned.

6 Conclusion and Future Work

Depending on the application, the logic governing an agent’s beliefs could be either monotonic or non-monotonic. Traditional belief revision assumes that an agent reasons monotonically; therefore, by definition, it is applicable to such situations only. Here we have aimed to study belief revision for situations in which the agent reasons non-monotonically. To this end, we defined slp-revision function and llp-revision function for disjunctive logic programs under the answer set semantics, catering respectively for application scenarios that prioritise the preservation of the syntactic structure and that prioritise the preservation of logical content.

Inconsistency-resolution is an essential task for belief revision. However, the strategies used in traditional belief revision functions are limited to situations when the agent reasons monotonically. With a logic program we have the luxury of making assumptions via lack of contrary evidence, and we can deduce certain facts from such assumptions. Thus if a set of beliefs is inconsistent, then one possible cause is that we made the wrong assumption. In such cases, we can resolve the inconsistency by adding some new rules so that the assumption can no longer be made. Such a cause of inconsistency and the associated inconsistency-resolution strategy is beyond the scope of traditional belief revision, but is crucial for non-monotonic belief revision. We argue that this rationale, which is encoded in our belief revision functions, captures the fundamental difference between monotonic and non-monotonic belief revision.

This paper then has explored AGM-style revision and belief base revision in the non-monotonic setting of disjunctive logic programs; in future work we propose to extend this to a general approach to belief revision in arbitrary non-monotonic settings.

Appendix: Proof of Results

Proof for Proposition 2

Suppose $P \cup Q$ is m-inconsistent. We need to show $P \uparrow Q = \emptyset$. Since $P \cup Q$ is m-inconsistent, we have $SE(P) \cap SE(Q) = \emptyset$. By the definition of expansion compatible program, any element in $P \uparrow Q$ has to be a superset of $P$ and consistent with $Q$. However, for any superset $R$ of $P$, $SE(R) \subseteq SE(P)$. Thus $SE(R) \cap SE(Q) = \emptyset$ which implies $R \cup Q$ is m-inconsistent.

Proof for Theorem 1

For one direction, suppose $*$ is a slp-revision function for $P$ and the associated selection function is $\gamma$. We need to show $*$ satisfies $(s\gamma s), (s\gamma e), (s\gamma r), (s\gamma er)$, $(s\gamma mr)$, and $(s\gamma u), (s\gamma s), (s\gamma c)$ and $(s\gamma)$ follow immediately from the definition of slp-revision functions.

$(s\gamma r)$: Suppose there is a set $R$ such that $R \not\in P \cup Q$. By the definition of slp-revision, we have $P \star Q = \gamma(P \uparrow Q) \cup Q$, hence $P \setminus \gamma(P \uparrow Q) \cup Q \not\in \emptyset$ which implies $\gamma(P \uparrow Q) \not\in P$. Then it follows from the definition of selection function that $P \uparrow Q \not\in \emptyset$ and $\gamma(P \uparrow Q) \in P \uparrow Q$. Let $\gamma(P \uparrow Q) = X$. Then $(P \cup Q) R = X \cup Q R$. Since $\emptyset \not\in R \cup P$, we have $(X \cup Q R) \subseteq (X \cup P)$. By the definition of compatible program, $X \cup Q R$ is inconsistent, that is $(P \cup Q) R$ is inconsistent.

$(s\gamma er)$: Suppose there is a set $E$ such that $E \not\subseteq X \cup Q \cup P$. By the definition of slp-revision, we have $P \star Q = \gamma(P \uparrow Q) \cup Q$, hence $\gamma(P \uparrow Q) \cup Q \not\in X \cup Q \cup P$. Then it follows from the definition of selection function that $P \uparrow Q \cup Q \not\in X \cup Q \cup P$. Let $\gamma(P \uparrow Q) = X$. Then $(P \cup Q) \subseteq Y \subseteq (X \cup Q) \cup E$. Since $E \cap P = \emptyset$ and $E \not\subseteq X \cup Q \cup P \subseteq (X \cup Q \cup E)$. By the definition of compatible program, $(X \cup Q \cup E) \cup Q$ is inconsistent. Since then $X \cap Q = Q$, we have $(X \setminus Q) \cup X = (X \cup Q) \cup E = (P \cup Q) \cup E$. Thus $(P \cup Q) \subseteq E$ is inconsistent.

$(s\gamma mr)$: Can be proved by combining the proving method for $(s\gamma r)$ and $(s\gamma er)$.

$(s\gamma u)$: Suppose $P \uparrow Q \subseteq P \cup R$. Then $\gamma(P \uparrow Q) = \gamma(P \uparrow R)$. If $P \uparrow Q \subseteq P \uparrow R \not\subseteq X$, then by the definition of slp-revision $P \star Q = P \cup Q$ and $P \star R = P \cup R$. Thus $P \setminus \gamma(P \uparrow Q) = P \setminus \gamma(P \uparrow R) = \emptyset$ and $(P \cup Q) \setminus (P \cup Q) = (P \cup R) \setminus (P \cup R) = \emptyset$. So suppose $P \uparrow Q \subseteq P \uparrow R \not\subseteq X$. Then $X = \gamma(P \uparrow Q) = \gamma(P \uparrow R)$. By the definition of slp-revision, we have $P \setminus (P \cup Q) = P \setminus (X \cup Q)$. Assume $\emptyset \not\subseteq X$. Then since $X \cup (P \cup Q) = \emptyset$, $P \cup Q \subseteq X$. In either case we have by set theory that $P \setminus (P \cup Q) = P \setminus (X \cup Q) = P \setminus X$. It can be
shown in the same manner that $P \setminus (P \cap R) = P \setminus (X \cup R) = P \setminus X$. Thus $P \setminus (P \setminus R) = P \setminus (X \cup R) = P \setminus X$. Again by the definition of slp-revision, we have $P \setminus (P \setminus Q) = (X \setminus Q) \cup (P \cup Q) = X \setminus P$. Similarly $(P \setminus R) \setminus (P \setminus R) = (X \setminus R) \cup (P \setminus R) = X \setminus P$. Thus $(P \setminus Q) \setminus (P \setminus Q) = (P \setminus R) \setminus (P \setminus R)$.

For the other direction, suppose $*$ is a function that satisfies ($s$+$s$), ($s$+$s$+$s$), ($s$+$s$+$t$), ($s$+$t$+$t$), ($s$+$t$+$m$), and ($s$+$u$). We need to show $*$ is a slp-revision function.

Let $\gamma$ be defined as:

$$\gamma(P \setminus Q) = ((P \setminus Q) \cap P) \cup ((P \setminus Q) \setminus Q)$$

for all $Q$. It suffices to show $\gamma$ is a selection function for $P$ and $P \setminus Q = \gamma(P \setminus Q) \cup Q$.

Part 1: For $\gamma$ to be a selection function, it must be a function.

Suppose $P \setminus Q = P \setminus R$. Then ($s$+$u$) implies $P \setminus (P \setminus Q) = P \setminus (P \setminus R) \setminus (P \setminus R) \setminus P$. Since $P \setminus (P \setminus Q) \cap (P \setminus Q) \setminus P = P \setminus (P \setminus R) \cap (P \setminus R) \setminus P$, $P \setminus (P \setminus Q) = P \setminus (P \setminus R) \setminus P$. Then $P \setminus (P \setminus Q) \setminus P = (P \setminus R) \setminus P$. Thus $P \setminus (P \setminus Q) \setminus P \cup (P \setminus R) \setminus P = (P \setminus R) \setminus P$. Then by set theory, we have $(P \setminus Q) \cap P \cup (P \setminus Q) \setminus P \cap P = (P \setminus R) \setminus P$. Finally, it follows from the definition of $\gamma$ that $\gamma(P \setminus Q) = P \setminus R$.

If $P \setminus Q = \emptyset$, then we have to show $\gamma(P \setminus Q) = P \setminus Q$. $P \setminus Q = \emptyset$ implies $Q$ is m-inconsistent, hence it follows from ($s$+$t$) that $P \setminus Q = Q \setminus P \setminus Q$. Then by the definition of $\gamma$, $\gamma(P \setminus Q) = ((P \setminus Q) \cap Q) \cup ((P \setminus Q) \setminus Q) \cap Q = Q \setminus P \setminus Q$. $P \setminus Q \setminus Q$ is inconsistent. Assume there is $X \setminus T \subseteq Q \setminus T$ and $X \setminus T \subseteq X \setminus T$.

Case 1, there is $R \subseteq T \subseteq X \setminus T$, $\gamma(P \setminus Q) = X \setminus T$, and $X = \gamma(P \setminus Q) \setminus R$. If $R \setminus T = \emptyset$, then since $\gamma(P \setminus Q) = \emptyset$, $R \setminus T = \emptyset$. Then it follows from ($s$+$t$+$m$) that $(P \setminus Q) \setminus R$ is inconsistent. Since $X \setminus T = (P \setminus Q) \setminus R$, $X \setminus T$ is inconsistent, a contradiction! Hit $X \setminus T \subseteq X \setminus T$.

Case 2, there is $E \subseteq T \subseteq X \setminus T$, $\gamma(P \setminus Q) = E$, and $X = \gamma(P \setminus Q) \setminus E$. If $E \subseteq T \subseteq X \setminus T$, then $\gamma(P \setminus Q) \setminus E$ is inconsistent. Since $X \setminus T = (P \setminus Q) \setminus E$, $X \setminus T$ is inconsistent, a contradiction! Hit $E \subseteq T \subseteq X \setminus T$.

Case 3, there are $R \subseteq T \subseteq X \setminus T$, $\gamma(P \setminus Q) = \emptyset$, $E \subseteq T \subseteq X \setminus T$, and $X = \gamma(P \setminus Q) \setminus E$. Then we can show as in Case 1 and 2 that $R \setminus T \subseteq X \setminus T$.

Part 2: By set theory, $\gamma(P \setminus Q) \setminus Q = ((P \setminus Q) \cap P) \cup ((P \setminus Q) \setminus Q) \setminus Q = ((P \setminus Q) \cap P) \cup (P \setminus Q) = P \setminus Q$.

Proof for Theorem 2

For one direction, let $R \subseteq P \setminus Q$. Then $R = cl(R)$, $R \setminus Q$ is consistent and $SE(P) \setminus SE(S) \subseteq SE(P) \setminus SE(R)$ implies $S \cup Q$ is inconsistent. Since $R \setminus Q$ is consistent, there is $(Y, Y) \in SE(R) \cap SE(Q)$ and there is no $(X, Y) \in SE(R) \cap SE(Q)$ with $X \setminus Y$. We have three cases:

Case 1, $P \setminus Q$ is inconsistent and $(Y, Y) \in SE(Q) \setminus SE(P)$: Let $M = \{(X, Z) \in SE(P) \setminus (Y, Y) \setminus SE(Q) \cap SE(Z) \subset Y\}$. Then $SE(S) = Cl((SE(P) \setminus (Y, Y)) \cup M)$. Then $(Y, Y) \in SE(S) \cap SE(Q)$ and the removal of all elements in $M$ guarantees that there is no $(X, Y) \in SE(S) \cap SE(Q)$ with $X \setminus Y$, hence $S \cup Q$ is consistent.

Assume $SE(R) \neq Cl((SE(P) \setminus (Y, Y)) \cup M))$. Then $SE(R) = Cl((SE(P) \setminus (Y, Y)) \setminus M) \cup \{N\}$, or $SE(R) = Cl((SE(P) \setminus (Y, Y)) \setminus M) \cup \{N\}$ where $G \neq \emptyset, G \subseteq SE(P) \setminus M, N \neq \emptyset$, and $N \setminus SE(P) = \emptyset$. For any of the possibilities, we have by basic set theory that $SE(P) \setminus SE(R) \subseteq SE(R)$, which means $P \setminus Q$ is inconsistent, a contradiction! So $SE(R) = Cl((SE(P) \setminus (Y, Y)) \setminus M)$. This case corresponds to condition 1 of Theorem 2.

Case 2, $P \setminus Q$ is inconsistent and $(Y, Y) \in SE(Q) \setminus SE(P)$: Let $M = \{(X, Z) \in SE(P) \setminus (Y, Y) \setminus SE(Q) \setminus SE(P), X \neq Y, Z \subseteq Y\}$. Note that $SE(S) \setminus M = Cl((SE(P) \setminus M)$, $SE(S) = SE(P) \setminus M$. Then $(Y, Y) \in SE(S) \cap SE(Q)$ and the removal of all elements in $M$ guarantees that there is no $(X, Y) \in SE(S) \cap SE(Q)$ with $X \setminus Y$, hence $S \cup Q$ is consistent.

Assume $SE(R) \subseteq \{M\} \cup \{N\}$. Then $SE(R) = (SE(P) \setminus \{M\} \cup \{N\}) \cup H$, or $SE(R) = (SE(P) \setminus \{M\} \cup \{N\}) \cup H$ for $G \neq \emptyset, G \subseteq SE(P) \setminus M, H \neq \emptyset$, and $H \setminus SE(P) = \emptyset$. For any of the possibilities, we have $SE(P) \setminus SE(S) \subseteq SE(S) \cap SE(R)$, which means $P \setminus Q$ is inconsistent, a contradiction! $SE(R) = SE(P) \setminus M$.

Assume there is $(W, W) \in SE(P) \cap SE(Q)$ such that $W \subset Y$. Let $SE(S) = SE(P) \setminus M'$ where $M' = \{(X, Z) \in SE(P) \setminus (W, W) \setminus SE(Q) \setminus SE(P), X \neq W, Z \subseteq W\}$. Then $(W, W) \in SE(S) \cap SE(Q)$ and the removal of all elements in $M'$ guarantees that there is no $(X, W) \in SE(S) \cap SE(Q)$ with $X \subset W$, hence $S \cup Q$ is consistent. By the completeness of SE models, it follows from $(X, W) \in SE(Q) \setminus SE(P)$ and $W \subset Y$ that $(X, Y) \in SE(Q) \setminus SE(P)$. Then it is easy to see that $M' \subset M$. Thus $SE(P) \setminus SE(S) \subseteq SE(R) \setminus SE(P)$ which implies $S \cup Q$ is inconsistent, a contradiction! So there is no such $(W, W) \in SE(P) \setminus SE(Q)$.

This case corresponds to condition 2 of Theorem 2.

Case 3, $P \setminus Q$ is consistent: Assume $SE(R) \neq SE(P)$. Then $SE(R) = (SE(P) \setminus \{M\} \cup \{N\}) \setminus N$, or $SE(R) = (SE(P) \setminus M) \setminus N$ for $M \setminus SE(P) = \emptyset$ and $N \subseteq SE(P)$. For any of the possibilities, we have $SE(P) \setminus SE(S) \subseteq SE(P) \setminus SE(R)$, which means $P \setminus Q$ is inconsistent, a contradiction! Thus $SE(R) = SE(P)$. This case corresponds to condition 3 of Theorem 2.

For the other direction, we need to show if $R = cl(R)$ and any of the three conditions of Theorem 2 is true, then $R \subseteq P \setminus Q$. So we have three cases which correspond to the three conditions.

Case 1: Conditions 1 and 2 for l-compatible programs are trivially satisfied. Let $SE(P) \setminus SE(S) \subseteq SE(P) \setminus SE(R)$. Then $SE(S) = SE(P) \setminus \{M\}$, $SE(S) = (SE(P) \setminus \{M\}) \setminus N$, or $SE(S) = Cl((SE(P) \setminus (Y, Y)) \setminus M) \setminus N$.
for $\emptyset \neq N \subseteq M$. It is easy to see that for all cases, $S \cup Q$ is inconsistent.

Case 2: Conditions 1 and 2 for 1-compatible programs are trivially satisfied. Let $SE(P) \cup SE(S) \subseteq SE(P) \cup SE(R)$. Then $SE(S) = (SE(P) \setminus M) \cup G$ for $\emptyset \neq G \subseteq M$. It is easy to see that $S \cup Q$ is inconsistent.

Case 3: Conditions 1 and 2 for 1-compatible programs are trivially satisfied. Since there is no $S$ such that $SE(P) \cup SE(S) \subseteq SE(P) \cup SE(R)$, condition 3 is also trivially satisfied.

\textbf{Proof for Theorem 3}

For one direction, suppose * is a lfp-revision function. We need to show * satisfies (i), (ii), (i+), (i+), (i+), (i+), (i+), and (i+). (i+), (i+), (i+), (i+), (i+), and (i+). We need to show $SE(SE(P) \cup SE(S)) \subseteq SE(P \cup Q)$.

Theorem 2 there are three cases:

Case 1: As for (l), we have $SE(P) = \emptyset$. Then we have $SE(P \cup Q) = SE(X) \cap SE(Q)$.

According to Theorem 2 there are three cases:

Case 1: $P \cup Q$ is inconsistent and $SE(X) = Cl((SE(P) \cup \{Y \cup Q\}) \setminus M) = \{X, Y \in SE(P) \setminus SE(Q) \mid X \neq Y \setminus \{X, Y\}\}$. Then $SE(P \cup Q) = Cl(SE(P) \cup \{Y \cup Q\}) \cup \{X \neq Y \setminus \{X, Y\}\}$ which implies $O \subseteq M$. So there is $(X, Y) \in Cl((SE(P) \cup Q) \cup O)$ with $X \in Y$ which means $(Y, Y)$ no longer an answer set in $SE(P \cup Q) \cap O$.

Case 2: $P \cup Q$ is inconsistent but m-consistent and $SE(X) = SE(P) \setminus M$ for there is $(Y, Y) \in SE(P) \setminus SE(Q)$ such that $W, Y \in SE(P) \setminus SE(Q)$ implies $W \neq Y$ and $M = \{(X, Z) \in SE(P) \setminus SE(Q) \mid X \neq Y \setminus \{X, Y\}\}$. Then $SE(P \cup Q) = Cl(SE(P) \cup \{Q\}) \cup \{X \neq Y \setminus \{X, Y\}\}$. Thus there is $(X, Y) \in Cl(SE(P) \cup Q) \cup \{X \neq Y \setminus \{X, Y\}\}$ which implies $O \subseteq M$. So there is $(X, Y) \in Cl(SE(P) \cup Q) \cup O$ with $X \neq Y \setminus \{X, Y\}$ which means $(Y, Y)$ no longer an answer set in $SE(P \cup Q) \cap O$.

Case 3: Since $SE(X) = SE(P)$, we have $SE(P \cup Q) = SE(P \cup Q) \subseteq SE(Q)$ which means $O$ does not exist. So this is an impossible case.

Case 1: For (i+), we have $SE(P) = Cl(SE(P) \cup \{Q\})$. According to Theorem 2 there are three cases:

Case 1: As for (i+) we have $SE(P) = Cl(SE(P) \cup \{Q\}) \setminus M$ for $(Y, Y) \in O$, then $SE(P \cup Q) \setminus O$ no longer contains any answer set. If $(Y, Y) \notin O$, then since $O \cap SE(P) \setminus SE(Q) = \emptyset$, we have $Cl(SE(P) \cup \{Q\}) \subseteq SE(P \cup Q)$.

Case 2: As for (i+) we have $SE(P \cup Q) = (SE(P) \cup SE(Q)) \setminus M$. This means $O$ does not exist. So this is an impossible case.

Case 3: As for (i+) we have $SE(P \cup Q) = SE(P) \cup SE(Q)$. This means $O$ does not exist. So this is an impossible case.

(i+): Can be proved by combining the proving proofs for (i+)+ and (i+).

(i+): Suppose $P \subseteq Q = P \setminus R$, we need to show $Cl(SE(P) \cup SE(S) \setminus M) = Cl(SE(P) \cup SE(R) \setminus \{Q\})$. For $\emptyset \neq R \subseteq M$ which implies $SE(Q) \subseteq SE(P) \cup SE(Q) \subseteq \emptyset$ is inconsistent. Since $SE(P \cup Q) \subseteq S$, we have $Cl(SE(P) \cup SE(Q) \setminus M) \cup Cl(SE(P) \cup SE(Q) \setminus \{Q\}) \subseteq SE(R) \cup Cl(SE(P) \cup SE(Q) \setminus \{Q\}) \subseteq SE(S) \setminus SE(Q)$ is inconsistent. Case 2, $SE(X) = SE(R) \cup T$ for $T \subseteq SE(R)$ and $T \cap SE(P) = \emptyset$. Then $SE(P \cup Q) = N$. Note that $SE(X) \neq Cl(SE(X))$. Then $SE(X) \setminus SE(Q) = (SE(R) \cup T) \setminus SE(Q) = (SE(R) \cup SE(Q)) \setminus T = (SE(P) \cup T) \setminus T = SE(P) \cup T$. It follows from $N \subseteq SE(P) \cup T$, $N \setminus SE(P) = \emptyset$, and (i+), then $SE(P \cup Q) \setminus N$ is inconsistent. Case 3, $SE(X) = SE(R) \cup S \cup T$ for $T \subseteq SE(R)$, $S \subseteq SE(R)$ and $T \setminus SE(P) = \emptyset$. Then from Case 1 we have $S \subseteq M$. Let $N \subseteq T \setminus SE(P)$. If $N = \emptyset$, then $SE(X) = SE(R) \cup S$ and this situation has been taken care in Case 1. So suppose $N \neq \emptyset$. Now $SE(X) \setminus SE(Q) = (SE(R) \setminus T) \setminus SE(Q) = (SE(R) \setminus SE(Q)) \setminus T = SE(P) \setminus T = SE(P) \setminus T$. If $N \neq \emptyset$, then $SE(X) \setminus SE(Q) = (SE(R) \setminus SE(Q)) \setminus (S \in SE(Q)) \setminus N$ and it follows from (i+) that $SE(X) \setminus SE(Q)$ is inconsistent. If $S \neq SE(Q)$, then it follows from (i+), then that $SE(X) \setminus SE(Q)$ is inconsistent.
REFERENCES


