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**[1]

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 [13] 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*.

$$Teach(X) \leftarrow Prof(X), not\ Admin(X). \quad (1)$$
$$Prof(John) \leftarrow . \quad (2)$$

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

[1] IIIS, Griffith University, Australia
[2] School of Computing Science, Simon Fraser University, Canada
[3] Department of Computing, Macquarie University, Australia

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

$$\leftarrow Teach(John). \quad (3)$$

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 can not 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

$$Admin(John) \leftarrow . \quad (4)$$

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*[4] 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 *llp-revision function*[5] 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 llp-revision function

[4] "s" stands for syntactic and "lp" for logic program.
[5] "l" 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:

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

where $m, n, o \geq 0$, $m + n + o > 0$, and $a_i, b_j, c_k \in \mathcal{A}$ for $\mathcal{A}$ a finite set of propositional atoms. We denote the set of all logic programs by $\mathcal{P}$. For each rule r, let $H(r) = \{a_1, \ldots, a_n\}$, $B^+(r) = \{b_1, \ldots, b_m\}$, 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 $\mathcal{A}$ 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 of P is denoted as $Mod(P)$. Given an interpretation Y, we write $Y \models P$ to mean Y is a classical model of P. The *reduct* of a program P with respect to an interpretation Y, denoted P^Y, is the set of rules:

$$\{H(r) \leftarrow B^+(r) \mid r \in P, B^-(r) \cap Y = \emptyset\}.$$

An *answer set* Y of P is a subset-minimal classical model of P^Y. The set of all answer sets of P is denoted as $AS(P)$.

An *SE interpretation* [27] is a pair (X, Y) of interpretations such that $X \subseteq Y \subseteq \mathcal{A}$. The set of all SE interpretations (over $\mathcal{A}$) is denoted $\mathcal{SE}$. The letters M and N are used to denote a set of SE interpretations throughout the paper. An SE interpretation is an *SE model* of a program P if $Y \models P$ and $X \models P^Y$. The set of all SE models of P is denoted as $SE(P)$. SE models are first proposed to capture *strong equivalence* [18] between programs that is $SE(P) = SE(Q)$ iff P and Q are strongly equivalent, thus they contain more informations than answer sets. For this reason, SE models have been used by many to characterise the logical content of a program [5, 7]. The following definitions and results are given in [10, 8]. A set M of SE interpretations is *well-defined* if $(X, Y) \in M$ implies $(Y, Y) \in M$. M is *complete* if it is well-defined and if $(X, Y) \in M$, $(Z, Z) \in M$ and $Y \subseteq Z$, then $(X, Z) \in M$. For each disjunctive logic program P, $SE(P)$ is complete and for each complete set M, there is a unique (up to strong equivalence) disjunctive logic program P such that $SE(P) = M$.

In this paper, we work with two notions of closure. The *closure* of a set M under completeness is denoted as $Cl(M)$. Formally, $Cl(M)$ is the minimal superset of M such that

1. if $(X, Y) \in M$, then $(Y, Y) \in Cl(M)$ and
2. if $(X, Y), (Z, Z) \in M$ and $Y \subseteq Z$, then $(X, Z) \in Cl(M)$.

The *closure* of a program P which intends to capture all logical consequence of P is denoted as $cl(P)$. Formally

$$cl(P) = \{r \mid SE(P) \subseteq SE(\{r\}) \text{ and } r \text{ is a program rule}\}.$$

We say P is *closed* if $P = cl(P)$.

The following two properties of SE models [27] are crucial to this paper:

1. $Y \in AS(P)$ iff $(Y, Y) \in SE(P)$ and there is no $(X, Y) \in SE(P)$ such that $X \subset Y$.
2. $(Y, Y) \in SE(P)$ iff $Y \in Mod(P)$.

So $SE(P) \neq \emptyset$ iff $Mod(P) \neq \emptyset$ but $SE(P) \neq \emptyset$ does not imply $AS(P) \neq \emptyset$. This gives rise to two notions of consistency.

Definition 1. *P is* consistent *iff $AS(P) \neq \emptyset$ and P is* m-consistent[6] *iff $SE(P) \neq \emptyset$.*

It follows from the SE model properties that consistency implies m-consistency and m-inconsistency implies inconsistency. In other words, a consistent program is m-consistent but not vice versa. For convenience, we say a set M of SE interpretations is consistent iff the program P such that $SE(P) = Cl(M)$ is consistent; we say (Y, Y) is an "answer set" in M iff $(Y, Y) \in M$ and there is no $(X, Y) \in M$ such that $X \subset Y$. Clearly, the consistency of M indicates the consistency of its corresponding logic program and (Y, Y) being an answer set in M indicates Y is an answer set of the corresponding program.

In subsequent sections, we will need to describe the difference between two logic programs and between their sets of SE models. For this purpose, we use the *symmetric difference* operator $\ominus$ which is defined as

$$X \ominus Y = (X \setminus Y) \cup (Y \setminus X)$$

for any sets X and Y.

3 SLP-Revision Functions

In this section, we give a syntax-based revision function $* : \mathcal{P} \times \mathcal{P} \mapsto \mathcal{P}$ 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 $R \subset R' \subseteq P$, 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-expansion 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 $P * Q$; 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 $P \updownarrow Q$, is such that $R \in P \updownarrow Q$ iff*
1. *$R \cup Q$ is consistent and*
2. *if $P \ominus R' \subset P \ominus R$, then $R' \cup Q$ is inconsistent.*

For example, let $P = \{a \leftarrow b, not\, c., \, b., \, e \leftarrow f, not\, g., \, f.\}$ and $Q = \{\leftarrow a., \leftarrow e.\}$. 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 $b \leftarrow$ from P (which eliminates the contradiction about a) and add $g \leftarrow$ to P (which eliminates the contradiction about e). The program thus obtained (i.e., $(P \setminus \{b.\}) \cup \{g.\}$) is a s-compatible program in $P \updownarrow Q$.

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, $P \updownarrow Q$ 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. $(P \uparrow Q) \cup (P \downarrow Q) \subseteq P \updownarrow Q$.

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 $Q = \{\leftarrow a.\}$, then $P \cup Q$ is inconsistent and we cannot restore consistency without removing $a \leftarrow$ 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 $P * Q$. 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 γ is a* selection function *for P iff for any program Q, $\gamma(P \updownarrow Q)$ returns a single element of $P \updownarrow Q$ whenever $P \updownarrow Q$ is non-empty; otherwise it returns P.*

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

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

$$P * Q = \gamma(P \updownarrow Q) \cup Q$$

for any program Q, where γ 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 $P = \{a \leftarrow not\, b, not\, c.\}$ and $Q = \{\leftarrow a.\}$. We can restore consistency of P with Q by, for instance, adding the rule $b \leftarrow$ to P which corresponds to the s-compatible program $P \cup \{b.\}$ or by adding the rule $c \leftarrow$ 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 $* : \mathcal{P} \times \mathcal{P} \mapsto \mathcal{P}$ is a function.

(s$*$s) $Q \subseteq P * Q$
(s$*$c) If Q is m-consistent, then $P * Q$ is consistent
(s$*$f) If Q is m-inconsistent, then $P * Q = P \cup Q$
(s$*$rr) If $\emptyset \neq R \subseteq P \setminus (P * Q)$, then $(P * Q) \cup R$ is inconsistent
(s$*$er) If $\emptyset \neq E \subseteq (P*Q) \setminus (P \cup Q)$, then $(P*Q) \setminus E$ is inconsistent
(s$*$mr) If $\emptyset \neq R \subseteq P \setminus (P * Q)$ and $\emptyset \neq E \subseteq (P * Q) \setminus (P \cup Q)$, then $((P * Q) \cup R) \setminus E$ is inconsistent
(s$*$u) If $P \updownarrow Q = P \updownarrow R$, then $P \setminus (P * Q) = P \setminus (P * R)$ and $(P * Q) \setminus (P \cup Q) = (P * R) \setminus (P \cup R)$

(s$*$s) (*Success*) states that a revision is always successful in incorporating the new beliefs. (s$*$c) (*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 $P = \{a.\}$ by $Q = \{b \leftarrow \; not \;\; b\}$. Although Q is inconsistent, we have $P \cup \{b.\}$ as a s-compatible program of P with respect to Q. Thus we can have $P \cup \{b.\} \cup Q$ as the revised logic program, which contains Q and is consistent. Here, a revision results in inconsistency only when the revising logic program is m-inconsistent. In such a case, (s$*$f) (*Failure*) states that the revision corresponds to the union of the original and revising logic program.

(s$*$rr) (*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. (s$*$er) (*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. It captures the intuition that nothing is added unless adding it contributes to making the revised logic program consistent. (s$*$mr) (*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∗rr) and (s∗er) together does not guarantee (s∗mr), 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 *Relevance* postulate [11].[7] The three postulates are the counterparts of *Relevance* 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∗u) (*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 (i.e., $P \setminus (P * Q)$) and the rules added to P (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 (i.e., $SE(Q) = SE(R)$) is a sufficient condition. However, it is too strong a requirement. Suppose $P = \{\leftarrow a.\}$, $Q = \{a.\}$, and $R = \{a \leftarrow b., b.\}$. Then the minimal change to P we have to made to accommodate Q and R are the same, that is we remove $\leftarrow a$. 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∗u).

We can show that these postulates are sufficient to characterise all slp-revision functions.

Theorem 1. *A function $*$ is a slp-revision function iff it satisfies (s∗s), (s∗c), (s∗f), (s∗rr), (s∗er), (s∗mr), and (s∗u).*

4 LLP-Revision Functions

Slp-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 *llp-revision function* that prioritises the preservation of logical content over that of syntactic information.

The main strategy of llp-revision functions is the same as that of slp-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 llp-revision is in the interpretation of "differs the least". Slp-revision interprets this notion as symmetric difference between the constituent rules whereas llp-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 *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* l-compatible programs *of P with respect to Q, denoted $P \Updownarrow Q$, is such that $R \in P \Updownarrow Q$ iff*

1. *$R = cl(R)$,*
2. *$R \cup Q$ is consistent, and*
3. *if $SE(P) \ominus SE(R') \subset SE(P) \ominus SE(R)$, 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 (condition 1) and whose set of SE models differ minimally from that of P (condition 2). Moreover, since we ignore syntactic difference, the logic program is closed (condition 1). We denote by $P \Downarrow Q$ and $P \Uparrow Q$ the set of *l-removal compatible programs* and *l-expansion compatible programs* such that $R \in P \Downarrow Q$ iff $SE(P) \subseteq SE(R)$ and $R \in P \Updownarrow Q$; $R \in P \Uparrow Q$ iff $SE(R) \subseteq SE(P)$ and $R \in P \Updownarrow Q$.

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 llp-revision functions. The theorem identifies the set of SE models for a l-compatible program under all possible situations.

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

1. *$P \cup Q$ is inconsistent. $SE(R) = Cl((SE(P) \cup \{(Y, Y)\}) \setminus M)$ where $(Y, Y) \in SE(Q) \setminus SE(P)$ and $M = \{(X, Z) \in SE(P) \mid (X, Y) \in SE(Q) \setminus SE(P)$ and $Z \subset Y\}$.*[8]
2. *$P \cup Q$ is inconsistent but m-consistent. $SE(R) = SE(P) \setminus M$ where there is $(Y, Y) \in SE(P) \cap SE(Q)$ such that $(W, W) \in SE(P) \cap SE(Q)$ implies $W \not\subset Y$ and $M = \{(X, Z) \in SE(P) \mid (X, Y) \in SE(Q) \cap SE(P), X \neq Y$ and $Z \subseteq Y\}$* [8]
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, $SE(P) \ominus SE(R)$ 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 shaped area. Such M is empty for (1.1) – (1.2) indicating that (Y, Y) is an answer set in $Cl(SE(P) \cup \{Y, Y\})$ and non-empty for (1.3) – (1.4) indicating that (Y, Y) is not an answer set in $Cl(SE(P) \cup \{Y, Y\})$ 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 $\{(Y, Y), (X, Y)\} \subseteq SE(R) \cap SE(Q) = Cl(((SE(P) \cup (Y, Y)) \setminus$

[7] If $\psi \in K$ and $\psi \notin K * \phi$, then there is some K' such that $K * \phi \subseteq K' \subseteq K \cup \{\phi\}$, K' is consistent but $K' \cup \{\psi\}$ is inconsistent.

[8] For $(X, Z) \in M$, X may be identical to Z.

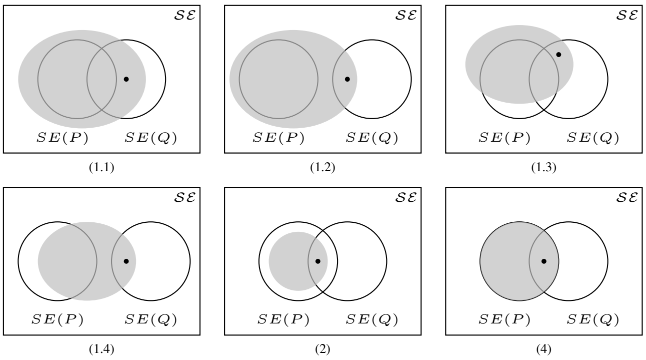

Figure 1. SE models for l-compatible programs of P w.r.t. Q

$M) \cup \{X, Z\}) \cap SE(Q)$, thus (Y, Y) 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 $P \cup Q$ 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 a 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*

$$P * Q = cl(\gamma(P \Updownarrow Q) \cup Q)$$

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 $* : \mathcal{P} \times \mathcal{P} \mapsto \mathcal{P}$ is a function.

(l$*$cl) $P * Q = cl(P * Q)$
(l$*$s) $Q \subseteq P * Q$
(l$*$c) If Q is m-consistent, then $P * Q$ is consistent
(l$*$f) If Q is m-inconsistent, then $P * Q = cl(P \cup Q)$
(l$*$rr) If $\emptyset \neq M \subseteq (SE(P) \cap SE(Q)) \setminus SE(P * Q)$, then $SE(P * Q) \cup M$ is inconsistent.
(l$*$er) If $\emptyset \neq N \subseteq SE(P * Q) \setminus SE(P)$, then either $SE(P * Q) \setminus N$ is inconsistent or $Cl(SE(P * Q) \setminus N) = SE(P * Q)$.
(l$*$mr) If $\emptyset \neq M \subseteq (SE(P) \cap SE(Q)) \setminus SE(P * Q)$ and $\emptyset \neq N \subseteq SE(P * Q) \setminus SE(P)$, then $(SE(P * Q) \cup M) \setminus N$ is inconsistent.
(l$*$u) If $P \Updownarrow Q = P \Updownarrow R$, then $Cl((SE(P) \cup SE(P * Q)) \setminus M) = Cl((SE(P) \cup SE(P * R)) \setminus N)$ for $M = \{(X, Z) \in SE(P) | Cl(SE(P * Q) \cup \{(X, Z)\}) \cap SE(Q)$ is inconsistent$\}$ and $N = \{(X, Z) \in SE(P) | Cl(SE(P * R) \cup \{(X, Z)\}) \cap SE(R)$ is inconsistent$\}$

(l$*$cl) (*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 $SE(P * Q)$ is 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 $P * Q$. (l$*$er) 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 $P * Q$ or to the closure of the set of SE models of $P * Q$. (l$*$mr) is a mixture of (l$*$rr) and (l$*$er), 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 $P \Updownarrow Q = P \Updownarrow R$ then $\gamma(P \Updownarrow Q) = \gamma(P \Updownarrow R)$ for γ a selection function. It follows from the other postulates that $Cl((SE(P) \cup SE(P * Q)) \setminus M) = SE(\gamma(P \Updownarrow Q))$ and $Cl((SE(P) \cup SE(P * R)) \setminus N) = SE(\gamma(P \Updownarrow R))$. The postulate is not as neat as its counterpart (s$*$u) for slp-revision functions. However the role of (s$*$u) in characterising slp-revision functions is the exactly the same as (l$*$u) 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$*$er), (l$*$mr), and (l$*$u).*

5 Related Work

There has been much work on belief revision for logic programs. Delgrande et al [7] generalise Satoh's [22] and Dalal's [4] revision operators to logic programs. Significantly, they bring SE model into the picture which has inspired several approaches that are based on SE models. Slota and Leite [24, 25, 26] investigate Katsuno and Mendelzon style [14] update for logic programs. Schwind and Inoue [23] provide a constructive characterisation for the revision operators in [7]. Delgrande et al [5] adapt the model-based revision of Katsuno and Mendelzon [15] to logic programs. Finally, Binnewies et al [3] 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 $a \leftarrow not\, a$ 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 is not sufficiently expressive to capture all aspects of a program and this gives rise to undesirable properties for the 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-Isberner [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-removal 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 $1 \leq j < j \leq n$ implies P_i 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 γ. We need to show $*$ satisfies (s$*$s), (s$*$c), (s$*$f), (s$*$rr), (s$*$er), (s$*$mr), and (s$*$u). (s$*$s), (s$*$c), and (s$*$f) follow immediately from the definition of slp-revision functions.

(s$*$rr): Suppose there is a set R such that $R \neq \emptyset$ and $R \subseteq P \setminus (P * Q)$. By the definition of slp-revision, we have $P * Q = \gamma(P \updownarrow Q) \cup Q$, hence $P \setminus (\gamma(P \updownarrow Q) \cup Q) \neq \emptyset$ which implies $\gamma(P \updownarrow Q) \neq P$. Then it follows from the definition of selection function that $P \updownarrow Q \neq \emptyset$ and $\gamma(P \updownarrow Q) \in P \updownarrow Q$. Let $\gamma(P \updownarrow Q) = X$. Then $(P * Q) \cup R = X \cup Q \cup R$. Since $\emptyset \neq R \subseteq P$, we have $((X \cup R) \ominus P) \subset (X \ominus P)$. By the definition of compatible program, $X \cup R \cup Q$ is inconsistent, that is $(P * Q) \cup R$ is inconsistent.

(s$*$er): Suppose there is a set E such that $E \neq \emptyset$ and $E \subseteq (P * Q) \setminus (P \cup Q)$. By the definition of slp-revision, we have $P * Q = \gamma(P \updownarrow Q) \cup Q$, hence $(\gamma(P \updownarrow Q) \cup Q) \setminus (P \cup Q) \neq \emptyset$ which implies $\gamma(P \updownarrow Q) \neq P$. Then it follows from the definition of selection function that $P \updownarrow Q \neq \emptyset$ and $\gamma(P \updownarrow Q) \in P \updownarrow Q$. Let $\gamma(P \updownarrow Q) = X$. Then $(P * Q) \setminus E = (X \cup Q) \setminus E$. Since $E \cap P = \emptyset$ and $\emptyset \neq E \subseteq X$, $((X \setminus E) \ominus P) \subset (X \ominus P)$. By the definition of compatible program, $(X \setminus E) \cup Q$ is inconsistent. Then since $E \cap Q = \emptyset$, we have $(X \setminus E) \cup Q = (X \cup Q) \setminus E = (P * Q) \setminus E$. Thus $(P * Q) \setminus E$ is inconsistent.

(s$*$mr): Can be proved by combining the proving method for (s$*$rr) and (s$*$er).

(s$*$u): Suppose $P \updownarrow Q = P \updownarrow R$. Then $\gamma(P \updownarrow Q) = \gamma(P \updownarrow R)$. If $P \updownarrow Q = P \updownarrow R = \emptyset$, then by the definition of slp-revision $P * Q = P \cup Q$ and $P * R = P \cup R$. Thus $P \setminus (P * Q) = P \setminus (P * R) = \emptyset$ and $(P * Q) \setminus (P \cup Q) = (P * R) \setminus (P \cup R) = \emptyset$. So suppose $P \updownarrow Q = P \updownarrow R \neq \emptyset$ and let $X = \gamma(P \updownarrow Q) = \gamma(P \updownarrow R)$. By the definition of slp-revision, we have $P \setminus (P * Q) = P \setminus (X \cup Q)$. Assume $\emptyset \neq P \cap Q \not\subseteq X$. Then since $X \cup (P \cap Q)$ is consistent with Q and $(X \cup (P \cap Q)) \ominus P \subset X \ominus P$, X is not a compatible program, a contradiction! Thus $P \cap Q = \emptyset$ or $P \cap Q \subseteq X$. In either case we have by set theory that $P \setminus (P * Q) = P \setminus (X \cup Q) = P \setminus X$. It can be shown in the same manner that $P \setminus (P * R) = P \setminus (X \cup R) = P \setminus X$. Thus $P \setminus (P * Q) = P \setminus (P * R)$. Again by the definition of slp-revision, we have $(P*Q) \setminus (P \cup Q) = (X \cup Q) \setminus (P \cup Q) = X \setminus P$. Similarly $(P * R) \setminus (P \cup R) = (X \cup R) \setminus (P \cup R) = X \setminus P$. Thus $(P * Q) \setminus (P \cup Q) = (P * R) \setminus (P \cup R)$.

For the other direction, suppose $*$ is a function that satisfies (s$*$s), (s$*$c), (s$*$f), (s$*$rr), (s$*$er), (s$*$mr), and (s$*$u). We need to show $*$ is a slp-revision function.

Let γ be defined as:

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

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

Part 1: For γ to be a selection function, it must be a function. Suppose $P \updownarrow Q = P \updownarrow R$. Then (s$*$u) implies $P \setminus (P * Q) = P \setminus (P * R)$ and $(P * Q) \setminus (P \cup Q) = (P * R) \setminus (P \cup R)$. Since $P = (P \setminus (P*Q)) \cup ((P*Q) \cap P) = (P \setminus (P*R)) \cup ((P*R) \cap P)$, $P \setminus (P * Q) = P \setminus (P * R)$ implies $(P * Q) \cap P = (P * R) \cap P$. Thus $(P * Q) \setminus (P \cup Q) = (P * R) \setminus (P \cup R)$ implies $((P * Q) \cap P) \cup ((P * Q) \setminus (P \cup Q)) = ((P * R) \cap P) \cup ((P * R) \setminus (P \cup R))$. Then by set theory, we have $((P * Q) \cap P) \cup ((P * Q) \setminus Q) = ((P * R) \cap P) \cup ((P * R) \setminus R)$. Finally, it follows from the definition of γ that $\gamma(P \updownarrow Q) = \gamma(P \updownarrow R)$.

If $P \updownarrow Q = \emptyset$, then we have to show $\gamma(P \updownarrow Q) = P$. $P \updownarrow Q = \emptyset$ implies Q is m-inconsistent, hence it follows from (s$*$f) that $P * Q = P \cup Q$. Then by the definition of γ, $\gamma(P \updownarrow Q) = ((P * Q) \cap P) \cup ((P * Q) \setminus Q) = ((P \cup Q) \cap P) \cup ((P \cup Q) \setminus Q) = P$.

If $P \updownarrow Q \neq \emptyset$, then we have to show $\gamma(P \updownarrow Q) \in P \updownarrow Q$. Since $P \updownarrow Q \neq \emptyset$, Q is m-consistent. Then (s$*$c) implies $P * Q$ is consistent. Since $\gamma(P \updownarrow Q) \cup Q = ((P*Q) \cap P) \cup ((P*Q) \setminus Q) \cup Q = P * Q$, $\gamma(P \updownarrow Q) \cup Q$ is consistent. Assume there is X s.t. $X \cup Q$ is consistent and $X \ominus P \subset \gamma(P \updownarrow Q) \ominus P$. Then we have three cases:

Case 1, there is R s.t. $\emptyset \neq R \subseteq P \setminus \gamma(P \updownarrow Q)$, and $X = \gamma(P \updownarrow Q) \cup R$: If $R \cap Q = \emptyset$, then since $\gamma(P \updownarrow Q) \cup Q = P * Q$, $R \cap (P * Q) = \emptyset$. Then it follows from (s$*$rr) that $(P * Q) \cup R$ is inconsistent. Since $X \cup Q = (P * Q) \cup R$, $X \cup Q$ is inconsistent, a contradiction! If $R \cap Q \neq \emptyset$, then since $R \subseteq P$, $R \cap P \cap Q \neq \emptyset$. Since (s$*$s) implies $Q \subseteq P * Q$, we have $Q \cap P \subseteq (P * Q) \cap P$, which implies $R \cap ((P * Q) \cap P) \neq \emptyset$. Then since $((P * Q) \cap P) \subseteq \gamma(P \updownarrow Q)$, $\gamma(P \updownarrow Q) \cap R \neq \emptyset$, a contradiction! Thus $R \cap Q \neq \emptyset$ is an impossible case.

Case 2, there is E s.t. $E \cap P = \emptyset$, $\emptyset \neq E \subseteq \gamma(P \updownarrow Q)$, and $X = \gamma(P \updownarrow Q) \setminus E$: Then $E \subseteq \gamma(P \updownarrow Q) \cup Q = P * Q$. If $E \cap Q = \emptyset$, then (s$*$er) implies $(P * Q) \setminus E$ is inconsistent. Since $X \cup Q = \gamma(P \updownarrow Q) \setminus E \cup Q = (P*Q) \setminus E$, $X \cup Q$ is inconsistent, a contradiction! If $E \cap Q \neq \emptyset$, then $E \not\subseteq (P*Q) \setminus Q$. Since $E \cap P = \emptyset$, we have $E \cap (P*Q) \cap P = \emptyset$. Thus $E \not\subseteq ((P*Q) \cap P) \cup ((P*Q) \setminus Q) = \gamma(P \updownarrow Q)$, a contradiction! Thus $E \cap Q \neq \emptyset$ is an impossible case.

Case 3, there are R and E s.t. $\emptyset \neq R \subseteq P$, $R \cap \gamma(P \updownarrow Q) = \emptyset$, $E \cap P = \emptyset$, $\emptyset \neq E \subseteq \gamma(P \updownarrow Q)$, and $X = (\gamma(P \updownarrow Q) \cup R) \setminus E$: Then we can show as in Case 1 and 2 that $R \cap P * Q = \emptyset$ and $E \subseteq P * Q$. If $R \cap Q = \emptyset$ and $E \cap Q = \emptyset$, then (s$*$mr) implies $((P*Q) \cup R) \setminus E$ is inconsistent. Thus $X \cup Q = ((\gamma(P \updownarrow Q) \cup R) \setminus E) \cup Q = ((P * Q) \cup R) \setminus E$ is inconsistent, a contradiction! Also we can show as in Case 1 and 2 that that $R \cap Q = \emptyset$ and $E \cap Q = \emptyset$ are impossible cases.

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

Proof for Theorem 2

For one direction, let $R \in P \Updownarrow Q$. Then $R = cl(R)$, $R \cup Q$ is consistent and $SE(P) \ominus SE(S) \subset SE(P) \ominus SE(R)$ implies $S \cup Q$ is inconsistent. Since $R \cup Q$ 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 $X \subset Y$. We have three cases:

Case 1, $P \cup Q$ is inconsistent and $(Y, Y) \in SE(Q) \setminus SE(P)$: Let $M = \{(X, Z) \in SE(P) | (X, Y) \in SE(Q) \setminus SE(P)$ and $Z \subset Y\}$. Let $SE(S) = Cl((SE(P) \cup \{(Y, Y)\}) \setminus M)$. 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 $X \subset Y$, hence $S \cup Q$ is consistent.

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

Case 2, $P \cup Q$ is inconsistent and $(Y, Y) \in SE(Q) \cap SE(P)$: Let $M = \{(X, Z) \in SE(P) \,|\, (X, Y) \in SE(Q) \setminus SE(P), X \neq Y$ and $Z \subseteq Y\}$. Note that $SE(P) \setminus M = Cl(SE(P) \setminus M)$. Let $SE(S) = SE(P) \setminus M$. 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 $X \subset Y$, hence $S \cup Q$ is consistent.

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

Assume there is $(W, W) \in SE(P) \cap SE(Q)$ such that $W \subset Y$. Let $SE(S) = SE(P) \setminus M'$ where $M' = \{(X, Z) \in SE(P) \,|\, (X, W) \in SE(Q) \setminus SE(P), X \neq W$ and $Z \subseteq W\}$. 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 $SE(P) \ominus SE(S) \subset SE(P) \ominus SE(R)$ which implies $S \cup Q$ is inconsistent, a contradiction! So there is no such $(W, W) \in SE(P) \cap SE(Q)$.

This case corresponds to condition condition 2 of Theorem 2.

Case 3, $P \cup Q$ is consistent: Assume $SE(R) \neq SE(P)$. Then
$SE(R) = SE(P) \cup M$,
$SE(R) = SE(P) \setminus N$, or
$SE(R) = (SE(P) \cup M) \setminus N$
for $M \cap SE(P) = \emptyset$ and $N \subseteq SE(P)$. For any of the possibilities, we have $SE(P) \ominus SE(P) \subset SE(P) \ominus SE(R)$, which means $P \cup Q$ 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 $R \in P \Updownarrow Q$. 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 $SE(P) \ominus SE(S) \subset SE(P) \ominus SE(R)$. Then
$SE(S) = SE(P) \setminus M$,
$SE(S) = (SE(P) \setminus M) \cup N$, or
$SE(S) = Cl(((SE(P) \cup (Y, Y)) \setminus M) \cup N$

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 l-compatible programs are trivially satisfied. Let $SE(P) \ominus SE(S) \subset SE(P) \ominus SE(R)$. 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 l-compatible programs are trivially satisfied. Since there is no S such that $SE(P) \ominus SE(S) \subset SE(P) \ominus SE(R)$, condition 3 is also trivially satisfied. □

Proof for Theorem 3

For one direction, suppose $*$ is a llp-revision function. We need to show $*$ satisfies (l$*$cl), (l$*$s), (l$*$c), (l$*$f), (l$*$rr), (l$*$er), (l$*$mr), and (l$*$u). (l$*$cl), (l$*$s), (l$*$f), and (l$*$c) follow immediately from the definition of llp-revision functions.

(l$*$rr): Let $\emptyset \neq J \subseteq (SE(P) \cap SE(Q)) \setminus SE(P*Q)$. We need to show $SE(P*Q) \cup J$ is inconsistent. By the definition of llp-revision function, we have $P * Q = cl(\gamma(P \Updownarrow Q) \cup Q)$. Let $\gamma(P \Updownarrow Q) = X$. Then we have $SE(P * Q) = SE(X) \cap SE(Q)$. According to Theorem 2 there are three cases:

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

Case 2, $P \cup Q$ is inconsistent but m-consistent and $SE(X) = SE(P) \setminus M$ for there is $(Y,Y) \in SE(P) \cap SE(Q)$ such that $(W,W) \in SE(P) \cap SE(Q)$ implies $W \not\subset Y$ and $M = \{(X,Z) \in SE(P) \mid (X,Y) \in SE(Q) \setminus SE(P), X \neq Y$ and $Z \subseteq Y\}$: Then $SE(P*Q) = (SE(P) \cap SE(Q)) \setminus M$ which implies $O \subseteq M$. So there is $(X,Y) \in Cl(SE(P*Q) \cup O)$ with $X \subset Y$ which means (Y,Y) is no longer an answer set in $SE(P*Q) \cup O$.

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

(l$*$er): Let $\emptyset \neq O \subseteq SE(P*Q) \setminus SE(P)$. We need to show $SE(P*Q) \setminus O$ is inconsistent or $Cl(SE(P*Q) \setminus O) = SE(P*Q)$. By the definition of llp-revision function, we have $P * Q = cl(\gamma(P \Updownarrow Q) \cup Q)$. Let $\gamma(P \Updownarrow Q) = X$. Then we have $SE(P*Q) = SE(X) \cap SE(Q)$. According to Theorem 2 there are three cases:

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

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

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

(l$*$mr): Can be proved by combining the proving methods for (l$*$rr) and (l$*$er).

(l$*$u): Suppose $P \Updownarrow Q = P \Updownarrow R$, we need to show $Cl((SE(P) \cup SE(P*Q)) \setminus M) = Cl((SE(P) \cup SE(P*R)) \setminus N)$ for $M = \{\mu \in SE(P) \mid Cl(SE(P*Q) \cup \{\mu\}) \cap SE(Q)$ is inconsistent$\}$ and $N = \{\mu \in SE(P) \mid Cl(SE(P*Q) \cup \{\mu\}) \cap SE(R)$ is inconsistent$\}$

Let γ be the selection function for P. Since $P \Updownarrow Q = P \Updownarrow R$, we have $\gamma P \Updownarrow Q = \gamma P \Updownarrow R$. Let $\gamma P \Updownarrow Q = X$. It is easy to see from the proof for other direction that $Cl((SE(P) \cup SE(P*Q)) \setminus M) = Cl((SE(P) \cup SE(P*R)) \setminus N) = SE(X)$.

For the other direction, suppose $*$ is a function that satisfies (l$*$cl), (l$*$s), (l$*$c), (l$*$f), (l$*$rr), (l$*$er), (l$*$mr), and (l$*$u). We need to show $*$ is a llp-revision function.

Let γ be defined as:

$$SE(\gamma(P \Updownarrow Q)) = Cl((SE(P) \cup SE(P*Q)) \setminus M)$$

for all Q, where $M = \{\mu \in SE(P) \mid Cl(SE(P*Q) \cup \{\mu\}) \cap SE(Q)$ is inconsistent$\}$. It remains to show $P * Q = cl(\gamma(P \Updownarrow Q) \cup Q)$ and γ is a selection function for P.

Part 1: Due to (l$*$cl), it suffices to show $SE(P*Q) = SE(\gamma(P \Updownarrow Q)) \cap SE(Q)$.

$\subseteq$: It follows from (l$*$s) that $SE(P*Q) \subseteq SE(Q)$, hence it suffices to show $SE(P*Q) \subseteq SE(\gamma(P \Updownarrow Q))$, that is $SE(P*Q) \subseteq Cl((SE(P) \cup SE(P*Q)) \setminus M)$. Let $\mu \in SE(P*Q)$. Then $Cl(SE(P*Q) \cup \{\mu\}) \cap SE(Q) = SE(Q) \cap SE(P*Q) = SE(P*Q)$. It follows from (l$*$c) that $SE(P*Q)$ is consistent, hence $Cl(SE(P*Q) \cup \{\mu\}) \cap SE(Q)$ is consistent. This means $\mu \notin M$, hence $\mu \in Cl((SE(P) \cup SE(P*Q)) \setminus M)$.

$\supseteq$: Assume there is $\mu \in SE(\gamma(P \Updownarrow Q)) \cap SE(Q)$ such that $\mu \notin SE(P*Q)$. Then $\mu \in Cl((SE(P) \cup SE(P*Q)) \setminus M) \cap SE(Q)$. It follows from $\mu \notin SE(P*Q)$ that $\mu \in SE(P) \setminus M$. Then it follows from $\mu \in SE(P) \cap SE(Q)$, $\mu \notin SE(P*Q)$, and (l$*$rr) that $SE(P*Q) \cup \{\mu\}$ is inconsistent. Since $SE(P*Q) \subseteq SE(Q)$ and $\mu \in SE(Q)$, we have $SE(Q) \cap Cl(SE(P*Q) \cup \{\mu\}) = Cl(SE(P*Q) \cup \{\mu\})$. Thus $SE(Q) \cap Cl(SE(P*Q) \cup \{\mu\})$ is inconsistent, which means $\mu \in M$, a contradiction.

Part 2: For γ to be a selection function, it must be a function. Suppose $P \Updownarrow Q = P \Updownarrow R$, we need to show $\gamma(P \Updownarrow Q) = \gamma(P \Updownarrow R)$, that is $Cl((SE(P) \cup SE(P*Q)) \setminus M) = Cl((SE(P) \cup SE(P*R)) \setminus N)$ for $M = \{\mu \in SE(P) \mid SE(R) \cap Cl(SE(P*Q) \cup \{\mu\})$ is inconsistent$\}$ and $N = \{\mu \in SE(P) \mid SE(R) \cap Cl(SE(P*R) \cup \{\mu\})$ is inconsistent$\}$. This follows immaculately from (l$*$u). It remains to show $\gamma(P \Updownarrow Q) \in P \Updownarrow Q$.

It has been shown in part 1 that $SE(P*Q) = SE(\gamma(P \Updownarrow Q)) \cap SE(Q)$. It follows from (l$*$c) that $SE(P*Q)$ is consistent, which means $SE(\gamma(P \Updownarrow Q)) \cap SE(Q)$ is consistent. Thus condition 1 for the definition of l-compatible program is satisfied. Now we focus on condition 2. Let $\gamma(P \Updownarrow Q) = R$ and $SE(P) \ominus SE(X) \subset SE(P) \ominus SE(R)$. We need to show $SE(X) \cap SE(Q)$ is inconsistent. There are three cases:

Case 1, $SE(X) = SE(R) \cup S$ for $S \subseteq SE(P)$ and $S \cap SE(R) = \emptyset$: Since $R = Cl((SE(P) \cup SE(P*Q)) \setminus M)$, we have $S \subseteq M$ which implies $SE(Q) \cap Cl(SE(P*Q) \cup S)$ is inconsistent. Since $SE(P*Q) = SE(R) \cap SE(Q)$, we have $SE(Q) \cap Cl(SE(P*Q) \cup S) = SE(Q) \cap Cl((SE(R) \cap SE(Q)) \cup S) = (SE(Q) \cap SE(R)) \cup (SE(Q) \cap S) = (SE(R) \cup S) \cap SE(Q) = SE(X) \cap SE(Q)$ is inconsistent. Case 2, $SE(X) = SE(R) \setminus T$ for $T \subseteq SE(R)$ and $T \cap SE(P) = \emptyset$: Since $SE(R) = Cl((SE(P) \cup SE(P*Q)) \setminus M)$, we have $T \subseteq Cl((SE(P) \cup SE(P*Q)) \setminus M)$. Let $T \cap SE(P*Q) = N$. Note that $N \neq \emptyset$ for otherwise $SE(X) \neq Cl(SE(X))$. Then $SE(X) \cap SE(Q) = (SE(R) \setminus T) \cap SE(Q) = (SE(R) \cap SE(Q)) \setminus T = SE(P*Q) \setminus T = SE(P*Q) \setminus N$. It follows from $N \subseteq SE(P*Q)$, $N \cap SE(P) = \emptyset$, and (l$*$er) that $SE(P*Q) \setminus N$ is inconsistent. Case 3, $SE(X) = (SE(R) \cup S) \setminus T$ for $S \subseteq SE(P)$, $S \cap SE(R) = \emptyset$, $T \subseteq SE(R)$ and $T \cap SE(P) = \emptyset$: Then from Case 1 we have $S \subseteq M$. Let $N = T \cap SE(P*Q)$. If $N = \emptyset$, then the $SE(X) = SE(R) \cup S$ and this situation has been taken care in Case 1. So suppose $N \neq \emptyset$. Now $SE(X) \cap SE(Q) = ((SE(R) \cup S) \setminus N) \cap SE(Q) = ((SE(R) \cap SE(Q)) \cup (S \cap SE(Q))) \setminus N$. If $S \cap SE(Q) = \emptyset$, then $SE(X) \cap SE(Q) = ((SE(R) \cap SE(Q)) \cup (S \cap SE(Q))) \setminus N = SE(P*Q) \setminus N$ and it follows from (l$*$er) that $SE(X) \cap SE(Q)$ is inconsistent. If $S \cap SE(Q) \neq \emptyset$, then it follows from (l$*$mr) that $SE(X) \cap SE(Q)$ is inconsistent. □

REFERENCES

[1] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson, 'On the logic of theory change: Partial meet contraction and revision functions', *The Journal of Symbolic Logic*, **50**(2), 510–530, (1985).

[2] José Júlio Alferes, João Alexandre Leite, Luís Moniz Pereira, Halina Przymusinska, and Teodor C. Przymusinski, 'Dynamic updates of non-monotonic knowledge bases', *J. Log. Program.*, **45**(1-3), 43–70, (2000).

[3] Sebastian Binnewies, Zhiqiang Zhuang, and Kewen Wang, 'Partial meet revision and contraction in logic programs', in *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-2015)*, (2015).

[4] Mukesh Dalal, 'Investigations into a theory of knowledge base revision', in *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI-1988)*, pp. 475–479, (1988).

[5] James P. Delgrande, Pavlos Peppas, and Stefan Woltran, 'Agm-style belief revision of logic programs under answer set semantics', in *Proceedings of the 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-2013)*, pp. 264–276, (2013).

[6] James P. Delgrande, Torsten Schaub, and Hans Tompits, 'A preference-based framework for updating logic programs', in *Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-2007)*, pp. 71–83, (2007).

[7] James P. Delgrande, Torsten Schaub, Hans Tompits, and Stefan Woltran, 'A model-theoretic approach to belief change in answer set programming', *ACM Trans. Comput. Log.*, **14**(2), (2013).

[8] Thomas Eiter, Michael Fink, Jörg Pührer, Hans Tompits, and Stefan Woltran, 'Model-based recasting in answer-set programming', *Journal of Applied Non-Classical Logics*, **23**(1-2), 75–104, (2013).

[9] Thomas Eiter, Michael Fink, Giuliana Sabbatini, and Hans Tompits, 'On properties of update sequences based on causal rejection', *TPLP*, **2**(6), 711–767, (2002).

[10] Thomas Eiter, Hans Tompits, and Stefan Woltran, 'On solution correspondences in answer-set programming', in *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-2015)*, pp. 463–498, (2005).

[11] Sven Ove Hansson, 'Reversing the Levi Identity', *Journal of Philosophical Logic*, **22**(6), 637–669, (1993).

[12] Sven Ove Hansson, 'Semi-revision', *Journal of Applied Non-Classical Logics*, **7**(1-2), 151–175, (1997).

[13] Sven Ove Hansson, *A Textbook of Belief Dynamics Theory Change and Database Updating*, Kluwer, 1999.

[14] Hirofumi Katsuno and Alberto O. Mendelzon, 'On the difference between updating a knowledge base and revising it', in *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR-1991)*, pp. 387–394, (1991).

[15] Hirofumi Katsuno and Alberto O. Mendelzon, 'Propositional knowledge base revision and minimal change', *Artificial Intelligence*, **52**(3), 263–294, (1992).

[16] Patrick Krümpelmann and Gabriele Kern-Isberner, 'Belief base change operations for answer set programming', in *Logics in Artificial Intelligence - 13th European Conference, JELIA 2012, Toulouse, France, September 26-28, 2012. Proceedings*, pp. 294–306, (2012).

[17] João Alexandre Leite, *Evolving Knowledge Bases: Specifications and Semantics*, IOS Press, 2003.

[18] Vladimir Lifschitz, David Pearce, and Agustín Valverde, 'Strongly equivalent logic programs', *ACM Trans. Comput. Logic*, **2**(4), 526–541, (2001).

[19] Mauricio Osorio and Víctor Cuevas, 'Updates in answer set programming: An approach based on basic structural properties', *TPLP*, **7**(4), 451–479, (2007).

[20] Teodor C. Przymusinski and Hudson Turner, 'Update by means of inference rules', *J. Log. Program.*, **30**(2), 125–143, (1997).

[21] Chiaki Sakama and Katsumi Inoue, 'An abductive framework for computing knowledge base updates', *TPLP*, **3**(6), 671–713, (2003).

[22] Ken Satoh, 'Nonmonotonic reasoning by minimal belief revision', in *Proceedings of the International Conference on Fifth Generation Computer Systems*, pp. 455–462, (1988).

[23] Nicolas Schwind and Katsumi Inoue, 'Characterization theorems for revision of logic programs', in *Proceedings of the 12th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-2013)*, pp. 485–498, (2013).

[24] Martin Slota and João Leite, 'On semantic update operators for answer-set programs', in *Proceedings of 19th European Conference on Artificial Intelligence (ECAI-2010)*, pp. 957–962, (2010).

[25] Martin Slota and João Leite, 'Robust equivalence models for semantic updates of answer-set programs', in *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR-2012)*, pp. 158–168, (2012).

[26] Martin Slota and João Leite, 'The rise and fall of semantic rule updates based on se-models', *Theory and Practice of Logic Programming*, **14**(6), 869–907, (2014).

[27] Hudson Turner, 'Strong equivalence made easy: Nested expressions and weight constraints', *Theory Pract. Log. Program.*, **3**(4), 609–622, (2003).

[28] Fernando Zacarías, Mauricio Osorio, Acosta Guadarrama, and Jürgen Dix, 'Updates in answer set programming based on structural properties', in *Proceedings of the 7th Internaltional Symposium on Logical Formalizations of Commonsense Reasoning*, pp. 213–219, (2005).

[29] Yan Zhang and Norman Y. Foo, 'Towards generalized rule-based updates', in *Proceedings of the 15th International Joint Conference on Artificial Intelligence, (IJCAI-1997) 97*, pp. 82–88, (1997).

[30] Yan Zhang and Norman Y. Foo, 'Updating logic programs', in *Proceedings of ECAI-1998*, pp. 403–407, (1998).