Journal of Artificial Intelligence Research 66 (2019) 989-1029 Submitted 04/2019; published 12/2019

A Semantic Characterization for ASP Base Revision

Laurent Garcia LAURENT.GARCIA@UNIV-ANGERS.FR
Claire Lefèvre CLAIRE.LEFEVRE@UNIV-ANGERS.FR
Igor Stéphan IGOR.STEPHAN@UNIV-ANGERS.FR
LERIA, University of Angers, France

Odile Papini ODILE.PAPINI@UNIV-AMU.FR
Éric Würbel ERIC.WURBEL@UNIV-AMU.FR
Aix Marseille University, University of Toulon
CNRS, LIS, Marseille, France

Abstract

The paper deals with base revision for Answer Set Programming (ASP). Base revision in classical logic is done by the removal of formulas. Exploiting the non-monotonicity of ASP allows one to propose other revision strategies, namely addition strategy or removal and/or addition strategy. These strategies allow one to define families of rule-based revision operators. The paper presents a semantic characterization of these families of revision operators in terms of answer sets. This semantic characterization allows for equivalently considering the evolution of syntactic logic programs and the evolution of their semantic content. It then studies the logical properties of the proposed operators and gives complexity results.

1. Introduction

Answer Set Programming (ASP) is an efficient unified formalism for both knowledge representation and reasoning in Artificial Intelligence (AI). It has its roots in non-monotonic reasoning and logic programming and gave rise to intensive research since Gelfond and Lifschitz's seminal paper (Gelfond & Lifschitz, 1988). ASP has an elegant and conceptually simple theoretical foundation and has been proved useful for solving a wide range of problems in various domains (Schaub & Woltran, 2018). Beyond its ability to formalize various problems from AI and to encode combinatorial problems (Baral, 2003; Niemelä, 1999), ASP provides also an interesting way to practically solve such problems since some efficient solvers are available (Gebser, Kaufmann, & Schaub, 2012; Leone, Pfeifer, Faber, Eiter, Gottlob, Perri, & Scarcello, 2006). But in most domains, information is evolving and subject to change, it is thus necessary to modify ASP logic programs when new information is received.

Belief change in a classical logic setting, in particular belief revision, has been extensively studied for decades. It applies to situations where an agent faces incomplete or uncertain information and where new and more reliable information may be contradictory with its initial beliefs. Belief revision consists in modifying the initial agent's beliefs while taking into account new information and ensuring the consistency of the result. Belief revision relies on three main principles: (i) *Success*: Change must succeed, new information has to be accepted, (ii) *Consistency*: The result of the revision operation must be a consistent set of beliefs, and (iii) *Minimal change*: The initial beliefs have to be changed as little as possible.

Two main frameworks became standard according to the nature of the involved representation of beliefs. AGM paradigm (Alchourrón & Makinson, 1985) for deductively closed set of beliefs

or theory revision, rephrased by Katsuno and Mendelzon (1991) for model-based revision, consists in providing the models of new information which are the closest (according to some criteria) to the models of the initial beliefs. On contrast, formula-based revision (or base revision) introduced by Hansson (1999) consists in selecting subsets of the initial beliefs maximally (according to some criteria) consistent with new information. Several concrete base revision operators have been proposed. Most approaches focus on the construction of consistent subbases maximal with respect to several criteria (Benferhat, Cayrol, Dubois, Lang, & Prade, 1993; Lehmann, 1995). From a dual point of view, others stem from the minimal withdrawal of formulas in order to restore consistency with new information like Kernel revision (Hansson, 1997) or like Removed Sets Revision (RSR) (Papini, 1992; Würbel, Jeansoulin, & Papini, 2000; Benferhat, Bennaim, Papini, & Würbel, 2010) that focuses on subsets of formulas minimal with respect to cardinality to remove. All these approaches require selection functions that encode the revision strategies for selecting among subbases or among subsets of formulas to remove.

This paper aims at studying base revision when beliefs are represented by ASP logic programs. Due to the non-monotonic nature of logic programs under answer set semantics, the problem of change in ASP is different and can be solved with more strategies than in classical logic. To illustrate this issue, consider the following example. In a medical context, patients suffering from a certain disease ($disease_1$) generally take a drug ($drug_1$), unless they take another drug ($drug_2$). Suppose we know that *Lea* is suffering from $disease_1$ and that in this case the medical team does not recommend her hospitalization. We can conclude that *Lea* takes $drug_1$. It can be represented by the following ASP logic program:

$$take\_drug_1(X) \leftarrow suffer\_disease_1(X), not\, take\_drug_2(X). \quad (1)$$
$$suffer\_disease_1(lea). \quad (2)$$
$$\leftarrow hospitalization(lea). \quad (3)$$

The answer set $\{suffer\_disease_1(lea), take\_drug_1(lea)\}$ for the above logic program corresponds to the facts we can conclude. Suppose we receive the information that *Lea* is allergic to $drug_1$ and that if the fever does not drop she should be hospitalized for further tests. This can be represented by the following ASP logic program:

$$\leftarrow take\_drug_1(lea). \quad (4)$$
$$hospitalization(lea) \leftarrow not\, fever\_drop(lea). \quad (5)$$

Now our beliefs about *Lea* are contradictory. Indeed the ASP logic program consisting in rules $(1)-(5)$ has no answer set. Resolving this inconsistency requires a revision mechanism. Inconsistency source is twofold, the first one concerns the drugs that *Lea* has to take and the second one concerns her hospitalization. The belief that *Lea* suffers from $disease_1$ (2) may be wrong, or rule (1) may be questioned. Moreover the constraint (3) may be too strong. In the spirit of monotonic revision we can restore consistency by removing rules (1) and (3), or (2) and (3). Alternatively, within a non-monotonic setting, in absence of information on the facts that *Lea* takes $drug_2$ and that *Lea*'s fever is dropping we can restore consistency by adding $take\_drug_2(lea)$ and $fever\_drop(lea)$. Besides, combining the above approaches we can also restore consistency by removing rule (1) or (2) and adding $fever\_drop(lea)$ or by removing rule (3) and adding $take\_drug_2(lea)$. In this paper, we propose three families of ASP base revision stemming from the addition and/or removal of some rules.

The first approaches dealing with logic programs in a dynamic setting focused on the problem of logic program change (Zhang & Foo, 1998; Sakama & Inoue, 1999; Alferes, Leite, Pereira, Przymusinska, & Przymusinski, 2000; Eiter, Fink, Sabbatini, & Tompits, 2002). The first work bridging ASP in a dynamic setting and belief change has been proposed by Delgrande, Schaub, Tompits and Woltran (2008, 2009, 2013). Their approach uses a semantic of logic programs in terms of SE-models (Turner, 2003). Model-based revision and merging stemming from a distance between interpretations have been extended to logic programs. Revision in this context consists in providing the sets of SE-models of the new logic program closest to the SE-models of the initial one. However, they noted that this approach has the drawback that arbitrary sets of SE-models may not necessarily be expressed via a logic program. Recently this drawback has been avoided for classes of logic programs satisfying an AGM-compliance condition on SE-models and a new postulate (Delgrande, Peppas, & Woltran, 2013). Model-based update has been addressed in the same spirit by Slota and Leite (2010, 2014).

From a syntactic point of view, formula-based belief merging and revision have been extended to ASP. The "removed sets" approach for fusion and revision (RSF) (Hué, Papini, & Würbel, 2008) and (RSR) (Benferhat et al., 2010) respectively, proposed in propositional logic have been extended to ASP (Hué, Papini, & Würbel, 2009, 2013). Besides, Krümpelmann and Kern-Isberner extended to ASP the "remainder sets" approach for screened consolidation initially defined in a classical setting. The strategy of these two approaches stems from the removal of some rules in order to restore consistency. More recently, a new approach for extending belief base revision to ASP has been proposed with additional strategies stemming from the addition and the addition and/or removal of some rules (Zhuang, Delgrande, Nayak, & Sattar, 2016b).

This paper focuses on three different families of ASP base revision operators. It first reviews the RSR family, it then introduces the notions of "added Sets" and "modified Sets" and proposes the Added Set Revision (ASR) and the Modified Set Revision (MSR). Note that these families of operators differ from the ones provided by Zhuang et al. (2016b) since the minimality criterion for the removed, added or modified sets is cardinality and not set inclusion. Indeed, cardinality refines set inclusion and has interesting properties as pointed by Benferhat, Dubois and Prade (1995). For each family of ASP base revision operators the paper provides a semantic counter-part that characterizes the operators in terms of answer sets.

The main contribution of the paper is the characterization of ASP base revision operators which also covers the family of *SLP* operators proposed by Zhuang et al. (2016b). This is an important result since it provides a new semantic characterization of logic program revision in terms of answer sets and allows one to change the focus from the evolution of a syntactic logic program to the evolution of its semantic content.

The paper is organized as follows. Section 2 gives a refresher on ASP and on belief base revision. Section 3 recalls RSR revision and provides a semantic characterization of removed sets. Section 4 introduces the notions of added set and ASR revision, it then gives a semantic characterization of added sets. Section 5 introduces the notions of modified set and MSR revision, it then provides a semantic characterization of modified sets. Section 6 reformulates the well-known Hansson's postulates defined for proposition belief base revision within the ASP framework and gives logical properties of RSR, ASR and MSR revision operators. Section 7 presents a study on the computational complexity of RSR, ASR and MSR revision operators. Finally Section 8 presents some related works and Section 9 concludes the paper.

This article is an extension of the conference paper (Garcia, Lefèvre, Papini, Stéphan, & Würbel, 2017). Besides providing full proofs for all results in the Appendix, we add here also a study of logical properties of the proposed revision operators (Section 6) as well as complexity results (Section 7).

2. Preliminaries

In this section, we present all the notions that will be usefull in this paper.

2.1 Answer Set Programming

Let $\mathscr{A}$ be a set of propositional atoms, a *logic program* is a finite set of rules of the form:

$$(c \leftarrow a_1,\ \ldots,\ a_n,\ not\ b_1,\ \ldots,\ not\ b_m.)\ \ n \geq 0, m \geq 0$$

where $c, a_1, \ldots, a_n,\ b_1, \ldots b_m \in \mathscr{A}$. The set of all logic programs is denoted by $\mathscr{P}$. The symbol "not" represents default negation and such a program may be seen as a sub-case of the default theory of Reiter (Reiter, 1980). A negation-free program is a *definite program*. For each rule r, let $head(r) = c$, $body^+(r) = \{a_1, \ldots, a_n\}$, $body^-(r) = \{b_1, \ldots, b_m\}$ and $r^+ = (head(r) \leftarrow body^+(r).)$. If $body^+(r) = \emptyset$ and $body^-(r) = \emptyset$ then the rule is simply written $(c.)$ and is called a *fact*. For a set of rules R, $Head(R) = \{head(r) \mid r \in R\}$ and $R^+ = \{r^+ \mid r \in R\}$.

Let X be a set of atoms. A rule r is *applicable in* X if $body^+(r) \subseteq X$. $App(P,X)$ denotes the set of applicable rules of P in X. The *least Herbrand model* of a definite program P, denoted $Cn(P)$, is the smallest set of atoms closed under P and can be computed as the least fix-point of the following consequence operator: $T_P : 2^{\mathscr{A}} \rightarrow 2^{\mathscr{A}}$ such that $T_P(X) = Head(App(P,X))$.

The Gelfond-Lifschitz *reduct* of a program P by a set of atoms X (Gelfond & Lifschitz, 1988) is the program $P^X = \{r^+ \mid r \in P \text{ and } body^-(r) \cap X = \emptyset\}$. Since it has no default negation, such a program is definite and then it has a unique minimal Herbrand model. By definition, an *answer set* (or stable model) of P is a set of atoms $X \subseteq \mathscr{A}$ such that $X = Cn(P^X)$. The set of answer sets of a logic program P is denoted by $AS(P)$. If $AS(P) \neq \emptyset$ the program is said *consistent* otherwise it is said *inconsistent*.

$GR(P,X) = \{r \in P \mid body^+(r) \subseteq X \text{ and } body^-(r) \cap X = \emptyset\}$ denotes the set of the *generating rules* of a logic program P with respect to a set of atoms X. A set of rules $R \subseteq P$ is *grounded* if there exists some enumeration $\langle r_i \rangle_{i=1}^n$ of the rules of R such that $\forall i > 0$, $body^+(r_i) \subseteq \{head(r_j) \mid j < i\}$. With those definitions the following result holds: $X \in AS(P)$ if and only if $X = Cn(GR(P,X)^+)$ if and only if $X = Head(GR(P,X))$ and $GR(P,X)$ is grounded.

A *constraint* is a rule without head $(\leftarrow a_1,\ \ldots,\ a_n,\ not\ b_1,\ \ldots,\ not\ b_m.)$ that should be read as $(h \leftarrow a_1,\ \ldots,\ a_n,\ not\ b_1,\ \ldots,\ not\ b_m,\ not\ h.)$ where h is a new atom symbol appearing nowhere else in the program.

For an atom a, $fact(a)$ denotes the corresponding fact $(a.)$ and $atom(a.)$ denotes the corresponding atom (a). These notations are extended to sets as usual: $fact(A) = \{a. \mid a \in A\}$ and $atom(F) = \{a \mid a. \in F\}$.

Example 1. In the following example:

$$take\_drug_1(lea) \leftarrow suffer\_disease_1(lea), not\ take\_drug_2(lea). \quad (1)$$
$$suffer\_disease_1(lea). \quad (2)$$
$$\leftarrow hospitalization(lea). \quad (3)$$

Rule (2) is a fact meaning that Lea suffers from disease 1 and rule (3) is a constraint that forbids each answer set containing that Lea is hospitalized. There is one answer set: $\{suffer\_disease_1(lea), take\_drug_1(lea)\}$. Moreover, if we add the fact $take\_drug_2(lea)$, the only answer set becomes: $\{suffer\_disease_1(lea), take\_drug_2(lea)\}$.

2.2 Classical Interpretations and Models

We now consider a rule $(c \leftarrow a_1, \ldots, a_n, not\ b_1, \ldots, not\ b_m.)$ as a classical implication $(a_1 \wedge \cdots \wedge a_n \wedge \neg b_1 \wedge \cdots \wedge \neg b_m \rightarrow c)$ which is equivalent to $(\neg a_1 \vee \cdots \vee \neg a_n \vee b_1 \vee \cdots \vee b_m \vee c)$. Hence, an *interpretation* of P is a set of atoms $m \subseteq \mathscr{A}$. An interpretation m *satisfies* a rule r if $body^+(r) \not\subseteq m$ or $body^-(r) \cap m \neq \emptyset$ or $head(r) \in m$. Conversely, an interpretation m falsifies a rule r if m does not satisfy r: $body^+(r) \subseteq m$ and $body^-(r) \cap m = \emptyset$ and $head(r) \notin m$. An interpretation m is a *model* of a program P if m satisfies all rules from P. $Mod(P)$ denotes the set of all the models of a logic program P. A logic program P is *m-consistent*[1] if $Mod(P) \neq \emptyset$ otherwise P is said *m-inconsistent*.

The following property links answer sets and classical models.

Theorem 1. *For any logic programs P and Q,* $AS(P \cup Q) \subseteq Mod(P)$.

Note that, if Q is empty, $AS(P) \subseteq Mod(P)$ holds. The converse of this property is obviously false, even for an empty set Q: a classical model is not necessary an answer set. For example $P = \{a \leftarrow not\ b., b \leftarrow not\ c., c \leftarrow not\ a\}$, we have $AS(P) = \emptyset$ but $Mod(P) \neq \emptyset$.

2.3 Belief Base Revision

Belief revision consists in incorporating a new belief, changing as few as possible the original beliefs while preserving consistency. Belief revision has been extensively studied within the framework of classical logic. In this paper we focus on belief base (finite (not closed) set of formulas) revision. More formally, let $\mathscr{L}$ be the classical language, a base revision operator denoted by $*$ is a function from $2^{\mathscr{L}} \times \mathscr{L}$ to $2^{\mathscr{L}}$ that maps a belief base $\mathscr{B}$ (the initial agent's beliefs) and a formula μ (new information) to a new belief base $\mathscr{B} * \mu$ (the revised agent's beliefs). Within this context, Hansson (1999) proposed postulates that any base revision operation should satisfy. Such postulates are as follows: let $\mathscr{B}, \mathscr{B}'$ be consistent belief bases and μ, ϕ be formulas,

1. The term is from Delgrande and al. (Zhuang et al., 2016b)

Success	$\mu \in \mathscr{B} * \mu$
Inclusion	$\mathscr{B} * \mu \subseteq \mathscr{B} \cup \{\mu\}$
Consistency	If μ is consistent then $\mathscr{B} * \mu$ is consistent
Vacuity	If $\mathscr{B} \cup \{\mu\}$ is consistent then $\mathscr{B} * \mu = \mathscr{B} \cup \{\mu\}$
$Core - retainment$	If $\phi \in \mathscr{B}$ and $\phi \notin \mathscr{B} * \mu$ then there exists $\mathscr{B}'$ such that $\mathscr{B}' \subseteq \mathscr{B} \cup \{\mu\}$, $\mathscr{B}'$ is consistent but $\mathscr{B}' \cup \{\phi\}$ is inconsistent.
Relevance	If $\phi \in \mathscr{B}$ and $\phi \notin \mathscr{B} * \mu$ then there exists $\mathscr{B}'$ such that $\mathscr{B} * \mu \subseteq \mathscr{B}' \subseteq \mathscr{B} \cup \{\mu\}$, $\mathscr{B}'$ is consistent but $\mathscr{B}' \cup \{\phi\}$ is inconsistent
Uniformity	If for all subsets $\mathscr{B}' \subseteq \mathscr{B}$, $\mathscr{B}' \cup \{\mu\}$ is inconsistent if and only if $\mathscr{B}' \cup \{\phi\}$ is inconsistent then $\mathscr{B} \setminus (\mathscr{B} * \mu) = \mathscr{B} \setminus (\mathscr{B} * \phi)$

The meaning of these postulates is the following. *Success* and *Consistency* express basic revision principles. *Inclusion* states that the union of the initial belief bases is the upper bound of any revision operation. *Vacuity* says that if new information is consistent with the belief base then the result of revision equals the non closing expansion. $Core - retainment$ and *Relevance* express the intuition that nothing is removed from the original belief bases unless its removal in some way contributes to make the result consistent. *Uniformity* determines that if two formulas are consistent with the same subsets of the original belief base then the respective erased formulas should be identical.

These postulates have been proposed for base revision within a monotonic setting. The starting hypothesis is that the initial belief base is consistent, moreover, thanks to the compactness property of a monotonic setting, the only way to restore consistency of belief bases is to remove some formulas. Within the ASP non-monotonic setting we relax the consistency hypothesis of the logic programs. Moreover restoring consistency can be performed by removing rules only but also by adding rules only or by removing and adding rules, as illustrated by the example in the introduction.

Considering consistency, within the ASP framework consistency means existence of an answer set while in a classical setting consistency is the existence of a model (monotonic consistency). Interpreting rules of a logic program as classical implications (see Section 2.2) leads to the notion of *m*-consistency of a logic program and Theorem 1 shows that for any normal logic program, consistency implies *m*-consistency (of course the converse does not hold). Investigating belief base revision within the ASP framework, we have to consider both consistency and *m*-consistency. When the revising logic program is not consistent but *m*-consistent there is a possibility to restore consistency while when the revising logic program is not *m*-consistent there is no possibility to restore consistency.

Example 2. Let P and Q be two normal logic programs, $P = \{c \leftarrow b, not\ a.\}$ and $Q = \{\leftarrow a., a \leftarrow b., b.\}$. P is consistent $AS(P) = \{\emptyset\}$, Q is m-inconsistent $AS(Q) = \emptyset$ and $P \cup Q$ is m-inconsistent.

Keeping Q unchanged, removing the rules from P leads to an m-inconsistent logic program, and adding some rule to $P \cup Q$ also leads to a m-inconsistent logic program.

2.4 Other Notions and Notations

We review some notions and notations useful in subsequent sections.

Definition 1 (preorders)**.** Let A be a set.

- A preorder on A is a reflexive and transitive binary relation.
- A total preorder on A, denoted by $\leq$, is a preorder such that $\forall x, y \in A$ either $x \leq y$ or $y \leq x$ holds.
- An equivalence relation, denoted by $\simeq$, is such that $x \simeq y$ if and only if $x \leq y$ and $y \leq x$.
- A strict total preorder, denoted by $<$, is such that $x < y$ if and only if $x \leq y$ holds but $x \simeq y$ does not hold.
- Let M be a subset of A, the set of minimal elements of M with respect to $\leq$, denoted by $Min(M, \leq)$, is defined as: $Min(M, \leq) = \{x \mid x \in M \text{ and } \nexists y (y \in M \text{ and } y < x)\}$.

Definition 2 (preference relation)**.** Let X and Y be two sets, $|X|$ (resp. $|Y|$) denotes the cardinality of X (resp. of Y) and $X \leq Y$ if $|X| \leq |Y|$. $X \leq Y$ means that X is preferred to Y.

Definition 3 (selection function)**.** Let A be a finite set, a *selection function* denoted by f is a function from $2^A \setminus \emptyset$ to A which for any set $X \in 2^A$ returns an element $f(X)$ such that $f(X) \in X$.

2.5 Revision, Non-Monotony and Inconsistency

Let P and Q be logic programs, possibly inconsistent. Throughout the paper Q is the program used to revise the initial program P. Revising P by Q amounts to providing a new consistent logic program containing Q and differing as little as possible from P. Revision is not always possible (for example, when Q does not admit any classical model since the revision strategy only processes information from P).

If $P \cup Q$ is consistent, no revision is necessary. Note that, in ASP, $P \cup Q$ can be consistent even if P and Q are both inconsistent. This is due to the fact that some information of P can block the rules responsible for the inconsistency of Q (and vice versa).

When $P \cup Q$ is inconsistent, there are several ways to revise P by Q. If we refer to the classical setting, let P and Q be two belief bases, revising P by Q often consists in removing as few pieces of information from P as possible. However, in a non-monotonic setting, restoring consistency can be done by removing some rules responsible for the inconsistency (like in the classical setting) but also by blocking these rules (by keeping these rules and adding pieces of information blocking them). Another way to restore consistency is to combine both removal and addition of rules. In view of these observations, we propose three ASP base revision methods.

3. ASP Base Revision by Removal

This section is dedicated to ASP base revision by removal. This revision strategy stems from the suppression of rules of P when $P \cup Q$ is inconsistent. This strategy is a direct application of the

one used for revising belief bases in a classical setting, however it differs from it due to the non-monotonicity of logic programs. Hué et al. (2013) proposed a syntactic characterization of ASP Revision by removal based on removed sets. We recall here this syntactic proposal and we add its semantical counterpart.

3.1 Rule-Based Revision by Removal

We review Removed Sets Revision (RSR) extended to ASP (Hué et al., 2013). This strategy focuses on the minimal number of rules needed to remove from P in order to restore the consistency. We first review the notion of *potential removed set* for which the minimality criterion is set inclusion.

Definition 4 (potential removed set, Hué et al., 2013)**.** Let P and Q be two logic programs. A potential removed set X is a set of rules such that:

- $X \subseteq P$.
- $(P \setminus X) \cup Q$ is consistent.
- For each $X' \subset X$, $(P \setminus X') \cup Q$ is inconsistent.

$\mathscr{PR}(P,Q)$ denotes the set of potential removed sets for P and Q.

Example 3. Let P and Q be two logic programs such that

$$P = \left\{ \begin{array}{ll} r_1 : a \leftarrow b., & r_4 : c \leftarrow not\ a., \\ r_2 : a., & r_5 : d., \\ r_3 : b., & r_6 : d \leftarrow not\ b. \end{array} \right\} \text{ and } Q = \{\leftarrow a, b., \quad \leftarrow not\ c, d.\}.$$

These two logic programs are consistent since $AS(P) = \{\{a,b,d\}\}$ and $AS(Q) = \{\emptyset\}$ but $P \cup Q$ is inconsistent. $\mathscr{PR}(P,Q) = \{\{r_3, r_5, r_6\}, \{r_2, r_3\}, \{r_1, r_2\}\}$.

According to the definition, if $P \cup Q$ is consistent then $\mathscr{PR}(P,Q) = \{\emptyset\}$. Since potential removed sets are built by removing only rules from P in order to restore consistency of $P \cup Q$, it may be possible that the set of potential removed sets for an inconsistent set Q is empty.

Example 4. Let $Q = \{\leftarrow not\ a.\}$. If $P = \{a.\}$ then $\mathscr{PR}(P,Q) = \{\emptyset\}$ since $P \cup Q$ is consistent. But if $P = \{b.\}$ then $\mathscr{PR}(P,Q) = \emptyset$, it is not possible to restore consistency by removal.

Since revision must change the program as little as possible, we have to choose among the potential removed sets those that are minimal. The minimality criterion chosen for RSR is cardinality. We thus review the notion of *removed set* by selecting the potential removed sets minimal with respect to cardinality.

Definition 5 (removed set, Hué et al., 2013)**.** Let P and Q be two logic programs. A removed set X is a set of rules such that:

- X is a potential removed set.
- There is no potential removed set Y such that $Y < X$.

$\mathscr{R}(P,Q)$ denotes the set of removed sets for P and Q. According to the definition $\mathscr{R}(P,Q) = Min(\mathscr{PR}(P,Q), \leq)$[2] and if $P \cup Q$ is consistent then $\mathscr{R}(P,Q) = \{\emptyset\}$.

2. We recall that when we write $Min(\mathscr{PR}(P,Q), \leq)$, the minimality criterion is cardinality, see Definition 2.

Example 5 (Example 3 continued)**.** $\mathscr{R}(P,Q) = \{\{r_2, r_3\}, \{r_1, r_2\}\}$.

A revision operator $\star$ defines, for all programs P and Q, the revised program $P \star Q$. In the present case, for two programs P and Q, there can be several removed sets, thus there can be several revision operators, as many as removed sets. The role of the selection function is to choose one removed set among the set $\mathscr{R}(P,Q)$. More formally, f is such that $f(\mathscr{R}(P,Q))) = X$, with $X \in \mathscr{R}(P,Q)$. We now review the *Removed Set Revision* (RSR) family of operators.

Definition 6 (RSR operators, Hué et al., 2013)**.** Let P and Q be two logic programs, $\mathscr{R}(P,Q)$ the set of removed sets and f a selection function. The revision operator denoted by $\star_{RSR(f)}$ is a function from $\mathscr{P} \times \mathscr{P}$ to $\mathscr{P}$ such that $P \star_{RSR(f)} Q = (P \setminus f(\mathscr{R}(P,Q))) \cup Q$.

Note that if $\mathscr{R}(P,Q) = \emptyset$, $f(\mathscr{R}(P,Q))$ is not defined. This means that the program P cannot be revised by Q.

Example 6 (Example 3 continued)**.** There are two selection functions, $f_1(\mathscr{R}(P,Q)) = \{r_2, r_3\}$ and $f_2(\mathscr{R}(P,Q)) = \{r_1, r_2\}$, therefore

$$P \star_{RSR(f_1)} Q = \left\{ \begin{array}{ll} r_1 : a \leftarrow b., & r_6 : d \leftarrow not\ b., \\ r_4 : c \leftarrow not\ a., & \leftarrow a, b., \\ r_5 : d., & \leftarrow not\ c, d. \end{array} \right\} \text{ with } AS(P \star_{RSR(f_1)} Q) = \{\{c,d\}\} \text{ and}$$

$$P \star_{RSR(f_2)} Q = \left\{ \begin{array}{ll} r_3 : b., & r_6 : d \leftarrow not\ b., \\ r_4 : c \leftarrow not\ a., & \leftarrow a, b., \\ r_5 : d., & \leftarrow not\ c, d. \end{array} \right\} \text{ with } AS(P \star_{RSR(f_2)} Q) = \{\{b,c,d\}\}.$$

3.2 Semantic Characterization of ASP Base Revision by Removal

We now present the semantical counterparts of the potential removed set and removed set notions.

Given P and Q two logic programs, the candidate models to be an answer set of the revised program must be models of Q, since Q is the smallest set of rules that we have to keep (and, by Theorem 1, for all R, $AS(Q \cup R) \subseteq Mod(Q)$). Thus, we are interested in the rules from P that must be removed so that a model of Q becomes an answer set of the revision. These rules are those that are falsified by the model of Q. $Fal(P,m)$ denotes the set of the rules of a logic program P that are falsified with respect to an interpretation m.

Definition 7 (falsified rules)**.** Let P be a logic program and m be an interpretation, the set of falsified rules of P with respect to m is the set $Fal(P,m) = \{r \in P \mid body^+(r) \subseteq m \text{ and } body^-(r) \cap m = \emptyset \text{ and } head(r) \notin m\}$.

We now introduce the notion of *canonical removed set*: it is a set of rules of P falsified by a model m of Q with the additional condition that, if these rules are removed, the resulting program admits m as an answer set. Note that m is not necessarily an answer set of the resulting program (see Example 7 below).

Definition 8 (canonical removed set)**.** Let P and Q be two logic programs and m a model of Q. A *canonical removed set* X is such that:

- $X = Fal(P,m)$
- $m \in AS((P \setminus X) \cup Q)$

$CR(P,Q,m) = \{X \mid X = Fal(P,m)$ and $m \in AS((P \setminus X) \cup Q)\}$ denotes the set of all canonical removed sets for m and $CR(P,Q) = \bigcup_{m \in Mod(Q)} CR(P,Q,m)$ denotes the union of sets of all canonical removed sets for the models of a program Q with respect to a program P. Note that for a given interpretation m, there is zero or one canonical removed set and if Q has no model then $CR(P,Q) = \emptyset$.

Example 7 (Example 3 continued)**.** The logic programs P and Q are such that

$$P = \left\{ \begin{array}{ll} r_1 : a \leftarrow b., & r_4 : c \leftarrow not\ a., \\ r_2 : a., & r_5 : d., \\ r_3 : b., & r_6 : d \leftarrow not\ b. \end{array} \right\} \text{ and } Q = \{\leftarrow a, b., \quad \leftarrow not\ c, d.\}.$$

Let us consider the model of Q, $m = \{c,d\}$. The rules r_2 and r_3 are falsified by m and, if these rules are removed, m is an answer set of the resulting revised program. Thus $\{r_2, r_3\}$ is a canonical removed set. On the other hand, if we consider the model of Q, $m = \{a,c,d\}$, the only falsified rule is r_3 but, if it is removed, $\{a,c,d\}$ is not an answer set because c has no support (there is no rule that can be applied to prove it). Thus $\{r_3\}$ is not a canonical removed set.

The following table gives, for each model of Q (first column), the corresponding set of canonical removed sets (last column). If this set is not empty, it only contains the set of rules X in bold in the second column.

$m \in Mod(Q)$	$X = Fal(P,m)$	$AS((P \setminus X) \cup Q)$	$CR(P,Q,m)$
$\emptyset$	$\{\mathbf{r_2}, \mathbf{r_3}, \mathbf{r_4}, \mathbf{r_5}, \mathbf{r_6}\}$	$\{\emptyset\}$	$\{Fal(P,\emptyset)\}$
$\{a\}$	$\{\mathbf{r_3}, \mathbf{r_5}, \mathbf{r_6}\}$	$\{\{a\}\}$	$\{Fal(P,\{a\})\}$
$\{b\}$	$\{\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_4}, \mathbf{r_5}\}$	$\{\{b\}\}$	$\{Fal(P,\{b\})\}$
$\{c\}$	$\{\mathbf{r_2}, \mathbf{r_3}, \mathbf{r_5}, \mathbf{r_6}\}$	$\{\{c\}\}$	$\{Fal(P,\{c\})\}$
$\{a,c\}$	$\{r_3, r_5, r_6\}$	$\{\{a\}\}$	$\emptyset$
$\{b,c\}$	$\{\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_5}\}$	$\{\{b,c\}\}$	$\{Fal(P,\{b,c\})\}$
$\{c,d\}$	$\{\mathbf{r_2}, \mathbf{r_3}\}$	$\{\{c,d\}\}$	$\{Fal(P,\{c,d\})\}$
$\{a,c,d\}$	$\{r_3\}$	$\emptyset$	$\emptyset$
$\{b,c,d\}$	$\{\mathbf{r_1}, \mathbf{r_2}\}$	$\{\{b,c,d\}\}$	$\{Fal(P,\{b,c,d\})\}$

Hence, if we restrict our attention to minimal canonical removed sets with respect to inclusion, we have $Min(CR(P,Q),\subseteq) = \{\{r_3,r_5,r_6\},\{r_2,r_3\},\{r_1,r_2\}\}$ which corresponds to potential removed sets and, if we consider minimality with respect to cardinality, we have $Min(CR(P,Q),\leq) = \{\{r_2,r_3\},\{r_1,r_2\}\}$ which corresponds to removed sets.

The following theorems give the equivalence between syntactic (potential) removed sets and semantic canonical removed sets.

Theorem 2. *Let P and Q be logic programs. We have $\mathscr{PR}(P,Q) = Min(CR(P,Q),\subseteq)$.*

The following theorem is a direct consequence of Definition 5 and Theorem 2.

Theorem 3. *Let P and Q be logic programs. We have $\mathscr{R}(P,Q) = Min(CR(P,Q),\leq)$.*

We now define the semantical counterpart of the RSR family of operators. We introduce a preference relation between interpretations, denoted by $<_{R(P)}$ as follows.

Definition 9 (preference relation on interpretations)**.** Let m and m' be two interpretations and P be a logic program, $m <_{R(P)} m'$ if $|Fal(P,m)| < |Fal(P,m')|$. [3]

The following result directly follows from Theorem 3 and Definition 6. It provides a semantic characterization of the RSR family of operators for logic programs[4].

Theorem 4. *Let P and Q be logic programs. Let $M = \{m \mid m \in Mod(Q)$ and $CR(P,Q,m) \neq \emptyset\}$. The following properties hold:*

- *For each selection function f, if $m \in AS(P \star_{RSR(f)} Q)$ then $m \in Min(M, \leq_{R(P)})$.*
- *If $m \in Min(M, \leq_{R(P)})$ then there exists a selection function f s.t. $m \in AS(P \star_{RSR(f)} Q)$.*

Example 8 (Example 3 continued)**.** The logic programs P and Q are such that

$$P = \left\{ \begin{array}{ll} r_1 : a \leftarrow b., & r_4 : c \leftarrow not\ a., \\ r_2 : a., & r_5 : d., \\ r_3 : b., & r_6 : d \leftarrow not\ b. \end{array} \right\} \text{ and } Q = \{\leftarrow a, b., \quad \leftarrow not\ c, d.\}.$$

From the table in Example 7 we have $\mathscr{PR}(P,Q) = Min(CR(P,Q), \subseteq)$, $\mathscr{R}(P,Q) = Min(CR(P,Q), \leq)$ and $Min(M, \leq_{R(P)}) = \{\{c,d\}, \{b,c,d\}\}$. Let f_1 and f_2 be the functions that select respectively $\{r_2, r_3\}$ and $\{r_1, r_2\}$. The respective revised logic programs are $P \star_{RSR(f_1)} Q = (P \setminus \{r_2, r_3\}) \cup Q$ and $P \star_{RSR(f_2)} Q = (P \setminus \{r_1, r_2\}) \cup Q$ with $AS(P \star_{RSR(f_1)} Q) = \{\{c,d\}\}$ and $AS(P \star_{RSR(f_2)} Q) = \{\{b,c,d\}\}$.

To conclude, we wonder which are the conditions under which revision by removal is defined. By Theorem 1, we know that revision by removal is defined only if there exists a model of Q (since Q is the minimum that we have to keep). Thus m-consistency of Q is a necessary condition but it is not sufficient. Recall Example 4 where $Q = \{\leftarrow not\ a.\}$ and $P = \{b.\}$, it is not possible to restore consistency by removal even if Q is m-consistent.

4. ASP Base Revision by Addition

This section is dedicated to ASP base revision by addition. Let P and Q be logic programs, this revision strategy stems from the addition of rules to P when $P \cup Q$ is inconsistent. This strategy relies on the non-monotonicity of the ASP framework. Indeed, adding a new rule may prevent the application of a rule (*block* the rule) which contributes to inconsistency.

Note that, with this strategy, revision is not always possible, even if P and Q are consistent (see Example 10 below). Moreover, since the addition of a new rule must block an existing rule, we restrict addition to the vocabulary of $P \cup Q$. Revising by addition allows for adding any kind of rules but adding a set of facts is sufficient and makes the revision process easier.

3. To be in accordance with the corresponding definitions for addition (Def. 15) and modification (Def. 21), the preference relation can also be defined by $m <_{R(P,Q)} m'$ if $Min(CR(P,Q,m), \leq) < Min(CR(P,Q,m'), \leq)$.
4. Note that, if we define a family of revision operators selecting one potential removed set (where the minimality criterion is set inclusion instead of cardinality), we can easily provide a semantic characterization of this family of operators.

4.1 Rule-Based Revision by Addition

The strategy of Added Set Revision (ASR) focuses on the minimal number of new rules to add in order to restore consistency. We first introduce the notion of *potential added set* for which the minimality criterion is set inclusion.

Definition 10 (potential added set)**.** Let P and Q be two logic programs. A potential added set Y is a set of rules made from the vocabulary of $P \cup Q$ such that:

- $(P \cup Y) \cup Q$ is consistent.
- For each $Y' \subset Y$, $(P \cup Y') \cup Q$ is inconsistent.

$\mathscr{PA}(P,Q)$ denotes the set of potential added sets for P and Q. Note that rules from a potential added set Y cannot already belong to $P \cup Q$. Indeed, if $(P \cup Y) \cup Q$ is consistent and some rule r from Y already belongs to $P \cup Q$, then there exists some $Y' = Y \setminus \{r\}$ such that $Y' \subset Y$ and $(P \cup Y') \cup Q = (P \cup Y) \cup Q$ is consistent.

According to the definition if $P \cup Q$ is consistent then $\mathscr{PA}(P,Q) = \{\emptyset\}$.

Example 9. Let P and Q be two logic programs such that $P = \{a \leftarrow not\ b.\}$ and $Q = \{\leftarrow a, not\ c., \leftarrow a, not\ d.\}$. P has an answer set $\{a\}$ that violates the two constraints from Q. For restoring consistency, we can add c and d so that the constraints are blocked or, alternatively, we can add b so that a cannot be deduced any more. Thus, if we restrict ourselves to the addition of facts, the potential removed sets are $\mathscr{PA}(P,Q) = \{\{c.,\ d.\}, \{b.\}\}$.

Note that revision by addition is not always feasible, even if P and Q are consistent.

Example 10. Let P and Q be two logic programs such that $P = \{a.\}$ and $Q = \{\leftarrow a.\}$. We have $\mathscr{PA}(P,Q) = \emptyset$. This is because, even if the constraint $(\leftarrow a.)$ can be read as $(h \leftarrow a, not\ h.)$, we do not allow using the implicit atom h for adding rules and thus the constraint $(\leftarrow a.)$ cannot be blocked by addition.

Among these potential added sets, we now select the ones that are minimal with respect to cardinality to minimize the changes in the initial knowledge. This leads to the notion of *added set*.

Definition 11 (added set)**.** Let P and Q be two logic programs. An added set Y is a set of rules such that:

- Y is a potential added set.
- There is no potential added set Z such that $Z < Y$.

$\mathscr{A}(P,Q)$ denotes the set of added sets for P and Q. According to the definition $\mathscr{A}(P,Q) = Min(\mathscr{PA}(P,Q), \leq)$ and if $P \cup Q$ is consistent then $\mathscr{A}(P,Q) = \{\emptyset\}$.

Example 11 (Example 9 continued)**.** For these programs, $\mathscr{A}(P,Q) = \{\{b.\}\}$ is reduced to only one added set.

Example 12. Let P and Q be two logic programs such that

$$P = \left\{ \begin{array}{l} a \leftarrow b., \\ b \leftarrow a., \\ c. \end{array} \right\} \text{ and } Q = \{\leftarrow c, not\ a.\}.$$

There are two (potential) added sets, $\mathscr{PA}(P,Q) = \mathscr{A}(P,Q) = \{\{a.\}, \{b.\}\}$.

We now define the *Added Set Revision* family of operators. Since there can be several added sets, there can be several revision operators for two programs P and Q, one for each selection function f that chooses an added set among $\mathscr{A}(P,Q)$. More formally, f is such that $f(\mathscr{A}(P,Q))) = Y$, with $Y \in \mathscr{A}(P,Q)$.

Definition 12 (ASR operators)**.** Let P and Q be two logic programs, $\mathscr{A}(P,Q)$ the set of added sets and f a selection function. The revision operator denoted by $\star_{ASR(f)}$ is a function from $\mathscr{P} \times \mathscr{P}$ to $\mathscr{P}$ such that $P \star_{ASR(f)} Q = (P \cup f(\mathscr{A}(P,Q))) \cup Q$.

Note that if $\mathscr{A}(P,Q) = \emptyset$, $f(\mathscr{A}(P,Q))$ is not defined. That means that the program cannot be revised by addition.

Example 13 (Example 12 continued)**.** There are two selection functions $f_1(\mathscr{A}(P,Q)) = \{a.\}$ and $f_2(\mathscr{A}(P,Q)) = \{b.\}$, therefore

$$P \star_{ASR(f_1)} Q = \left\{ \begin{array}{l} a \leftarrow b., \\ b \leftarrow a., \\ c., \\ a. \\ \leftarrow c, not\ a. \end{array} \right\} \text{ with } AS(P \star_{ASR(f_1)} Q) = \{\{a,b,c\}\} \text{ and}$$

$$P \star_{ASR(f_2)} Q = \left\{ \begin{array}{l} a \leftarrow b., \\ b \leftarrow a., \\ c., \\ b. \\ \leftarrow c, not\ a. \end{array} \right\} \text{ with } AS(P \star_{ASR(f_2)} Q) = \{\{a,b,c\}\}.$$

4.2 Semantic Characterization of ASP Base Revision by Addition

We now present the semantic counterparts of the potential added set and added set notions.

Given P and Q two logic programs, the candidate models to be an answer set of the revised program must be models of $P \cup Q$. Indeed $P \cup Q$ is the smallest set of rules that we have to keep and, by Theorem 1, we know that every answer set of a program containing $P \cup Q$ is a model of $P \cup Q$: for all Y, $AS(P \cup Q \cup Y) \subseteq Mod(P \cup Q)$. Thus, we are interested in what is missing in $P \cup Q$ (the Y part that we have to add) so that a model of $P \cup Q$ becomes an answer set of the result of revision.

If we consider a model m of $P \cup Q$, we know (see Section 2.1) that m is an answer set of $P \cup Q$ iff $m = Cn(GR(P \cup Q, m)^+)$. Intuitively, $Cn(GR(P \cup Q, m)^+)$ represents the part of m that can be deduced from the rules of $P \cup Q$. If it does not match with m, that means that m contains supplementary atoms that cannot be deduced from $P \cup Q$. These supplementary atoms are the "missing part" Y. If it is added to $P \cup Q$ [5], it allows m to become an answer set of $P \cup Q \cup Y$, that is, $m = Cn(GR(P \cup Q \cup Y, m)^+)$ holds. This set of atoms from a model m that cannot be deduced from the rules of $P \cup Q$ is exactly $m \backslash Cn(GR(P \cup Q, m)^+)$. When atoms are replaced by facts, the corresponding set is denoted by $Nded(m, P \cup Q)$.

Definition 13 (non-deduced atoms)**.** Let P be a logic program and m an interpretation. $Nded(m,P) = fact(m \backslash Cn(GR(P,m)^+))$.

Then the following theorem holds.

5. in the form of a set of facts Y

Theorem 5. *Let P be a logic program. We have, $\forall m \in Mod(P)$, $m \in AS(P \cup Nded(m,P))$.*

We now introduce the notion of *canonical added set*. Given P and Q two logic programs, a *canonical added set* is a set of facts corresponding to the smallest subset (with respect to inclusion) of atoms to add to $P \cup Q$ so that a model of $P \cup Q$ becomes an answer set. Note that the non-deduced atoms (facts) from $Nded(m, P \cup Q)$ is a superset of what is needed. Indeed a subset of $Nded(m, P \cup Q)$ can be sufficient for deducing the whole set (see Example 14 below). Then, there may be several canonical added sets for the same model. Note also that, by Theorem 5, there is at least one canonical added set for each model of $P \cup Q$.

Definition 14 (canonical added set)**.** Let P and Q be two logic programs and m a model of $P \cup Q$. A *canonical added set* Y is such that:

- $Y \subseteq Nded(m, P \cup Q)$
- $m \in AS(P \cup Q \cup Y)$
- $\forall Y' \subset Y, m \notin AS(P \cup Q \cup Y')$

$CA(P,Q,m)$ denotes the set of all canonical added sets for m and $CA(P,Q) = \bigcup_{m \in Mod(P \cup Q)} CA(P,Q,m)$. Note that $CA(P,Q,m) = Min(\{Y \mid Y \subseteq Nded(m, P \cup Q)$ and $m \in AS(P \cup Q \cup Y)\}, \subseteq)$.

Example 14. Let P and Q be two logic programs such that

$$P = \left\{ \begin{array}{ll} r_1 : a \leftarrow b, not\ c., & r_3 : b \leftarrow d., \\ r_2 : c \leftarrow d, e, not\ a., & r_4 : d \leftarrow b., \\ & r_5 : e. \end{array} \right\} \text{ and } Q = \left\{ \begin{array}{l} \leftarrow not\ a, not\ c., \\ \leftarrow a, not\ b, not\ c., \\ \leftarrow c, not\ d, not\ a. \end{array} \right\}$$

If we consider the model $m = \{a,b,d,e\}$ of $P \cup Q$, the generating rules of $P \cup Q$ with respect to m (see Section 2.1) are $\{r_1, r_3, r_4, r_5\}$ (r_2 is blocked by a). But, with these rules, only e can be deduced, $\{a., b., d.\}$ are missing but it is sufficient to add b (or d) so that a and d (resp. a and b) can be deduced too. So the minimal set to add is $\{b.\}$ (or, alternatively, $\{d.\}$). The set of canonical added sets for this model $m = \{a,b,d,e\}$ is then $CA(P,Q,m) = \{\{b.\}, \{d.\}\}$.

The last column of the following table gives the set of canonical added sets corresponding to each classical model of $P \cup Q$ given in the first column of the table.

$m \in Mod(P \cup Q)$	$GR(P \cup Q, m)$	$Nded(m, P \cup Q)$	Y	$AS(P \cup Q \cup Y)$	$CA(P,Q,m)$
$\{a,c,e\}$	$\{r_5\}$	$\{a., c.\}$	$\{a., c.\}$	$\{\{a,c,e\}\}$	$\{\{a., c.\}\}$
$\{a,b,d,e\}$	$\{r_1, r_3, r_4, r_5\}$	$\{a., b., d.\}$	$\{b.\}$	$\{\{a,b,d,e\}, \{b,c,d,e\}\}$	
			$\{d.\}$	$\{\{a,b,d,e\}, \{b,c,d,e\}\}$	$\{\{b.\}, \{d.\}\}$
$\{b,c,d,e\}$	$\{r_2, r_3, r_4, r_5\}$	$\{b., c., d.\}$	$\{b.\}$	$\{\{a,b,d,e\}, \{b,c,d,e\}\}$	
			$\{d.\}$	$\{\{a,b,d,e\}, \{b,c,d,e\}\}$	$\{\{b.\}, \{d.\}\}$
$\{a,b,c,d,e\}$	$\{r_3, r_4, r_5\}$	$\{a., b., c., d.\}$	$\{a., b., c.\}$	$\{\{a,b,c,d,e\}\}$	$\{\{a., b., c.\},$
			$\{a., c., d.\}$	$\{\{a,b,c,d,e\}\}$	$\{a., c., d.\}\}$

Note that $Min(CA(P,Q),\subseteq) = \{\{a.,c.\},\{b.\},\{d.\}\}$ which corresponds to the set of potential added sets and $Min(CA(P,Q),\leq) = \{\{b.\},\{d.\}\}$ which corresponds to the set of added sets.

Note that canonical added sets only consist of facts. Thus the semantic characterization of ASR operators is limited to ASR operators that require the addition of facts (not the addition of general rules).

Theorem 6. *Let P and Q be logic programs. We have $\mathscr{PA}(P,Q) = Min(CA(P,Q),\subseteq)$.*

The following theorem is a direct consequence of Theorem 6 and Definition 11.

Theorem 7. *Let P and Q be logic programs. We have $\mathscr{A}(P,Q) = Min(CA(P,Q),\leq)$.*

We introduce a preference relation between interpretations, denoted by $<_{A(P,Q)}$ as follows.

Definition 15 (preference relation on interpretations)**.** Let m and m' be two interpretations and P and Q be two logic programs, $m <_{A(P,Q)} m'$ if $Min(CA(P,Q,m),\leq) < Min(CA(P,Q,m'),\leq)$.

Note that, for each model m of $P \cup Q$, $CA(P,Q,m) \neq \emptyset$ thus the preference relation is always defined on $Mod(P \cup Q)$.

The following result directly follows from Theorem 7 and Definition 12. It provides a semantic characterization of the ASR family of revision operators for logic programs[6].

Theorem 8. *Let P and Q be two logic programs and $M = Mod(P \cup Q)$. The following properties hold:*

- *For each selection function f, if $m \in AS(P \star_{ASR(f)} Q)$ then $m \in Min(M, \leq_{A(P,Q)})$.*
- *If $m \in Min(M, \leq_{A(P,Q)})$ then there exists a selection function f s.t. $m \in AS(P \star_{ASR(f)} Q)$.*

Example 15 (Example 14 continued)**.** From the table in Example 14 we have $\mathscr{PA}(P,Q) = Min(CA(P,Q),\subseteq)$, $\mathscr{A}(P,Q) = Min(CA(P,Q),\leq) = \{\{b.\},\{d.\}\}$. $Min(M,\leq_{A(P,Q)}) = \{\{a,b,d,e\},\{b,c,d,e\}\}$. Let f_1 and f_2 be the functions that select respectively $\{b.\}$ and $\{d.\}$ the respective revised logic programs are $P \star_{ASR(f_1)} Q = P \cup Q \cup \{b.\}$ and $P \star_{ASR(f_2)} Q = P \cup Q \cup \{d.\}$ with $AS(P \star_{ASR(f_1)} Q) = AS(P \star_{ASR(f_2)} Q) = \{\{a,b,d,e\},\{b,c,d,e\}\}$.

To conclude, we wonder which are the conditions under which revision by addition is defined. By Theorem 1, we know that revision by addition is defined only if there exists a model of $P \cup Q$. Thus m-consistency of $P \cup Q$ is a necessary condition. It is also a sufficient condition. Indeed, revision by adding all atoms non-deduced from a model of $P \cup Q$ is always possible, it then suffices to minimize it.

Theorem 9. *Let P and Q be logic programs, there exists a selection function f such that $P \star_{ASR(f)} Q$ is defined if and only if $P \cup Q$ is m-consistent.*

6. Note that, if we define a family of revision operators selecting one potential added set (where the minimality criterion is set inclusion instead of cardinality), we can easily provide a semantic characterization of this family of operators.

5. ASP Base Revision by Modification

This section focuses on ASP base revision by modification. Modification strategy means combining the removal strategy and the addition one. Let P and Q be logic programs, removing some rules from P and in the same time adding some new rules to P allows one to construct a new logic program which is consistent with Q and differs as little as possible from P. Indeed, revision by removal and revision by addition can be viewed as particular cases of revision by modification.

5.1 Rule-Based Revision by Modification

The strategy of Modified Set Revision (MSR) focuses on the minimal number of rules to remove and/or to add in order to restore consistency. A (potential) modified set is a pair of sets of rules, where the first component is the set of rules to remove and the second one is the set of new rules to add. We first define preference relations between pairs of sets of rules with respect to set inclusion and with respect to cardinality as follows.

Definition 16. Let X, Y, X', Y' be sets of rules.

- $(X',Y') \subset (X,Y)$ if $X' \subset X$ and $Y' \subseteq Y$, or $X' \subseteq X$ and $Y' \subset Y$.
- $(X',Y') \leq (X,Y)$ if $|X' \cup Y'| \leq |X \cup Y|$.

We now introduce the notion of *potential modified set*.

Definition 17 (potential modified set)**.** Let P and Q be two logic programs. A potential modified set (X,Y) is a pair of sets of rules such that:

- $X \subseteq P$.
- $(P \backslash X) \cup Y \cup Q$ is consistent.
- For each (X',Y') such that $(X',Y') \subset (X,Y)$, $(P \backslash X') \cup Y' \cup Q$ is inconsistent.

$\mathscr{PM}(P,Q)$ denotes the set of potential modified sets for P and Q. According to the definition if $P \cup Q$ is consistent then $\mathscr{PM}(P,Q) = \{(\emptyset,\emptyset)\}$.

Example 16. Let P and Q be two logic programs such that

$$P = \left\{ \begin{array}{l} r_1 : a \leftarrow not\ b., \\ r_2 : c \leftarrow not\ b., \\ r_3 : \leftarrow f. \end{array} \right\} \text{ and } Q = \left\{ \begin{array}{l} \leftarrow a., \\ \leftarrow c., \\ f \leftarrow not\ g., \\ f \leftarrow not\ h. \end{array} \right\}.$$

P has an answer set $\{a,c\}$ and Q has also one, $\{f\}$, but a and c violate the constraints from Q and f violates the constraint from P. There are four ways to restore consistency: (1) to remove all rules from P, (2) to add $\{b.,g.,h.\}$ so that a, c and f cannot be deduced any more, or a mix of both: (3) to remove r_3 and to add b or (4) to remove r_1 and r_2 and to add g and h. We have thus
$\mathscr{PM}(P,Q) = \{(\{r_1,r_2,r_3\},\emptyset),\ (\emptyset,\{b.,g.,h.\}),\ (\{r_3\},\{b.\}),\ (\{r_1,r_2\},\ \{g.,h.\})\}$.

As for RSR and ASR, the minimality criterion chosen for MSR is cardinality, we introduce the notion of *modified set* by selecting the potential modified sets minimal with respect to cardinality.

Definition 18 (modified set)**.** Let P and Q be two logic programs. A modified set (X,Y) is a pair of sets of rules such that:

- (X,Y) is a potential modified set.
- There is no potential modified set (X',Y') such that $(X',Y') < (X,Y)$.

We denote by $\mathscr{M}(P,Q)$ the set of modified sets. According to the definition $\mathscr{M}(P,Q) = Min(\mathscr{P}\mathscr{M}(P,Q),\leq)$ and if $P \cup Q$ is consistent then $\mathscr{M}(P,Q) = \{(\emptyset,\emptyset)\}$.

Example 17 (Example 16 continued)**.** $\mathscr{M}(P,Q) = \{(\{r_3\},\{b.\})\}$.

We now define the *Modified Set Revision* family of operators. Since there can be several modified sets, there can be several revision operators for two programs P and Q, one for each selection function f that chooses a modify set among $\mathscr{M}(P,Q)$. More formally, f is such that $f(\mathscr{M}(P,Q))) = (X,Y)$, with $(X,Y) \in \mathscr{M}(P,Q)$.

Definition 19 (MSR operators)**.** Let P and Q be two logic programs, $\mathscr{M}(P,Q)$ the set of modified sets and f a selection function. The revision operator denoted by $\star_{MSR(f)}$ is a function from $\mathscr{P} \times \mathscr{P}$ to $\mathscr{P}$ such that $P \star_{MSR(f)} Q = (P \setminus X) \cup Y \cup Q$ where $(X,Y) = f(\mathscr{M}(P,Q))$.

Example 18 (Example 16 continued)**.** The logic programs P and Q are such that

$$P = \left\{ \begin{array}{l} r_1 : a \leftarrow not\ b., \\ r_2 : c \leftarrow not\ b., \\ r_3 :\leftarrow f. \end{array} \right\} \text{ and } Q = \left\{ \begin{array}{l} \leftarrow a., \\ \leftarrow c., \\ f \leftarrow not\ g., \\ f \leftarrow not\ h. \end{array} \right\}.$$

There is only one selection function f and $f(\mathscr{M}(P,Q)) = (\{\leftarrow f.\},\{b.\})$, therefore

$$P \star_{MSR(f)} Q = \left\{ \begin{array}{ll} r_1 : a \leftarrow not\ b., & \leftarrow a., \\ r_2 : c \leftarrow not\ b., & \leftarrow c., \\ b., & f \leftarrow not\ g., \\ & f \leftarrow not\ h. \end{array} \right\}$$

with $AS(P \star_{MSR(f)} Q) = \{\{b,g,h\},\{b,g,f\},\{b,h,f\},\{b,f\}\}$.

5.2 Semantic Characterization of ASP Base Revision by Modification

We now present the semantic counterparts of potential modified set and modified set notions.

Given P and Q two logic programs, the candidate models to be an answer set of the revised program must be models of Q, since Q is the smallest set of rules that we have to keep (and, by Theorem 1, for all R, $AS(Q \cup R) \subseteq Mod(Q)$). Thus, we are interested in what must be removed from P (falsified rules X) *and* what is missing in the resulting program $(P \setminus X) \cup Q$ (the non-deduced atoms) so that a model of Q becomes an answer set of the result of revision. We first introduce the notion of *canonical modified set* which captures this notion.

Definition 20 (canonical modified set)**.** Let P and Q be two logic programs and m be a model of Q. A *canonical modified set* (X,Y) is such that:

- $X = Fal(P,m)$.
- $Y \subseteq Nded(m,(P \setminus X) \cup Q)$.

- $m \in AS((P \setminus X) \cup Q \cup Y)$.
- $\forall Y' \subset Y, m \notin AS((P \setminus X) \cup Q \cup Y')$.

The set of all canonical modified sets for m is denoted by $CM(P,Q,m)$. According to the definition 20, $CM(P,Q,m) = Min(\{(X,Y) \mid X = Fal(P,m), Y \subseteq Nded(m,(P \setminus X) \cup Q)$ and $m \in AS((P \setminus X) \cup Q \cup Y)\}, \subseteq)$. The set of all possible canonical modified sets (for all models of Q) is denoted by $CM(P,Q) = \bigcup_{m \in Mod(Q)} CM(P,Q,m)$.

Note that there is at least one canonical modified set for each model of Q. Indeed, by definition of Fal, any interpretation m is a model of $P \setminus Fal(P,m)$. Then, if $m \in Mod(Q)$, $m \in Mod((P \setminus Fal(P,m)) \cup Q)$. Thus, by Theorem 5, $m \in AS((P \setminus Fal(P,m)) \cup Q \cup Nded(m,(P \setminus Fal(P,m)) \cup Q))$.

Example 19 (Example 16 continued)**.** Let P and Q be two logic programs such that

$$P = \left\{ \begin{array}{l} r_1 : a \leftarrow not\ b., \\ r_2 : c \leftarrow not\ b., \\ r_3 : \leftarrow f. \end{array} \right\} \text{ and } Q = \left\{ \begin{array}{l} \leftarrow a., \\ \leftarrow c., \\ f \leftarrow not\ g., \\ f \leftarrow not\ h. \end{array} \right\}.$$

If we consider the model of Q, $m = \{b,f,g\}$, the only falsified rule is r_3. If we remove it, atoms b and g cannot be deduced from the resulting program. Now, if we remove r_3 and add b and g, the resulting program admits m as an answer set. Thus $(\{r_3\}, \{b.,g.\})$ is a canonical modified set (since it is minimal).

The second and the third column of the following table give the first and the second component respectively of the canonical modified set corresponding to a classical model of Q given in the first column of the table.

$m \in Mod(Q)$	$X = Fal(P,m)$	$Y \subseteq Nded((P \setminus X) \cup Q, m)$	$AS((P \setminus X) \cup Q \cup Y)$
$\{f\}$	$\{r_1,r_2,r_3\}$	$\emptyset$	$\{\{f\}\}$
$\{b,f\}$	$\{r_3\}$	$\{b.\}$	$\{\{b,f\}\}$
$\{f,g\}$	$\{r_1,r_2,r_3\}$	$\{g.\}$	$\{\{f,g\}\}$
$\{f,h\}$	$\{r_1,r_2,r_3\}$	$\{h.\}$	$\{\{f,h\}\}$
$\{g,h\}$	$\{r_1,r_2\}$	$\{g.,h.\}$	$\{\{g,h\}\}$
$\{b,f,g\}$	$\{r_3\}$	$\{b.,g.\}$	$\{\{b,f,g\}\}$
$\{b,f,h\}$	$\{r_3\}$	$\{b.,h.\}$	$\{\{b,f,h\}\}$
$\{b,g,h\}$	$\emptyset$	$\{b.,g.,h.\}$	$\{\{b,g,h\}\}$
$\{f,g,h\}$	$\{r_1,r_2,r_3\}$	$\{f.,g.,h.\}$	$\{\{f,g,h\}\}$
$\{b,f,g,h\}$	$\emptyset$	$\{b.,f.,g.,h.\}$	$\{\{b,f,g,h\}\}$

Hence $Min(CM(P,Q),\subseteq) = \{(\{r_1,r_2,r_3\},\emptyset),(\{r_3\},\{b.\}),(\{r_1,r_2\},\{g.,h.\}),(\emptyset,\{b.,g.,h.\})\}$ which corresponds to the set of potential modified sets and $Min(CM(P,Q),\leq) = \{(\{r_3\},\{b.\})\}$ which corresponds to the set of modified sets.

The following theorems give the equivalence between syntactic (potential) modified sets and semantic canonical modified sets.

Theorem 10. *Let P and Q be programs. We have* $\mathscr{PM}(P,Q) = Min(CM(P,Q),\subseteq)$.

The following theorem is a direct consequence of Theorem 10 and Definition 18.

Theorem 11. *Let P and Q be logic programs. We have $\mathscr{M}(P,Q) = Min(CM(P,Q),\leq)$.*

We introduce a preference relation between interpretations denoted by $<_{M(P,Q)}$.

Definition 21 (preference relation on interpretations)**.** Let m and m' be two interpretations and P and Q be two logic programs, $m <_{M(P,Q)} m'$ if $Min(CM(P,Q,m),\leq) < Min(CM(P,Q,m'),\leq)$.

Note that, for each model m of Q, $CM(P,Q,m) \neq \emptyset$ thus the preference relation is always defined on $Mod(Q)$.

The following result directly follows from Theorem 11 and Definition 19. It provides a semantic characterization of the MSR family of revision operators[7].

Theorem 12. *Let P and Q be logic programs and $M = Mod(Q)$. The following properties hold:*

- *For each selection function f, if $m \in AS(P \star_{MSR(f)} Q)$ then $m \in Min(M, \leq_{M(P,Q)})$*
- *If $m \in Min(M, \leq_{M(P,Q)})$ then there exists a selection function f s.t. $m \in AS(P \star_{MSR(f)} Q)$.*

Example 20 (Example 16 continued)**.** The logic programs P and Q are such that

$$P = \left\{ \begin{array}{l} r_1 : a \leftarrow not\ b., \\ r_2 : c \leftarrow not\ b., \\ r_3 : \leftarrow f. \end{array} \right\} \text{ and } Q = \left\{ \begin{array}{l} \leftarrow a., \\ \leftarrow c., \\ f \leftarrow not\ g., \\ f \leftarrow not\ h. \end{array} \right\}.$$

From the table in Example 19 we have $\mathscr{PM}(P,Q) = Min(CM(P,Q),\subseteq)$, $\mathscr{M}(P,Q) = Min(CM(P,Q),\leq) = \{(\{r_3\},\{b.\})\}$ and $Min(M,\leq_{M(P,Q)}) = \{\{b,f\}\}$. There is only one modified set thus f selects $(\{r_3\},\{b.\})$ and $P \star_{MSR(f)} Q = \{r_1,\ r_2\} \cup \{b.\} \cup Q$ and $AS(P \star_{MSR(f)} Q) = \{\{b,f\}\}$.

To conclude, we wonder which are the conditions under which revision by modification is defined. By Theorem 1, we know that revision by modification is defined only if there exists a model of Q (since Q is the minimum that we have to keep). Thus m-consistency of Q is a necessary condition. It is also a sufficient condition. Indeed, revision by removing all rules from P and adding all atoms non-deduced from a model of Q is always possible, it then suffices to minimize changes.

Theorem 13. *Let P and Q be logic programs, there exists a selection function f such that $P \star_{MSR(f)} Q$ is defined if and only if Q is m-consistent.*

5.3 Applying Revision by Modification

We come back to our initial example to illustrate syntactic and semantic aspects of revision. We only consider revision by modification since it is more general than the two other revisions.

Example 21 (Example 1 continued)**.** Let P and Q be two logic programs such that

$$P = \left\{ \begin{array}{l} r_1 : td_1 \leftarrow sd_1, not\ td_2., \\ r_2 : sd_1., \\ r_3 : \leftarrow h. \end{array} \right\} \text{ and } Q = \left\{ \begin{array}{l} \leftarrow td_1., \\ h \leftarrow not\ fd. \end{array} \right\}$$

7. Note that, if we define a family of revision operators selecting one potential modified set (where the minimality criterion is set inclusion instead of cardinality), we can easily provide a semantic characterization of this family of operators.

where td_1 stands for $take\_drug_1(lea)$, td_2 for $take\_drug_2(lea)$, sd_1 for $suffer\_disease_1(lea)$, h for $hospitalization(lea)$ and fd for $fever\_drop(lea)$.

From a syntactic point of view, there are six ways to restore consistency. The set of modified sets is $\mathscr{M}(P,Q) = \{(\{r_1,r_3\},\emptyset),(\{r_2,r_3\},\emptyset),(\emptyset,\{fd.,td_2.\}),(\{r_1\},\{fd.\}),(\{r_2\},\{fd.\}),(\{r_3\},\{td_2.\})\}$. Let us note that, in this example $\mathscr{P}\mathscr{M}(P,Q) = \mathscr{M}(P,Q)$.

Now, from the semantic point of view, we start from the models of Q and compute the canonical modified sets. The second and the third column of the following table give the first and the second component respectively of the canonical modified set corresponding to a classical model of Q given in the first column of the table.

$m \in Mod(Q)$	$X = Fal(P,m)$	$Y \subseteq Nded((P \setminus X) \cup Q, m)$	$AS((P \setminus X) \cup Q \cup Y)$
$\{h\}$	$\{r_2,r_3\}$	$\emptyset$	$\{\{h\}\}$
$\{h,sd_1\}$	$\{r_1,r_3\}$	$\emptyset$	$\{\{h,sd_1\}\}$
$\{h,td_2\}$	$\{r_2,r_3\}$	$\{td_2.\}$	$\{\{h,td_2\}\}$
$\{h,sd_1,td_2\}$	$\{r_3\}$	$\{td_2.\}$	$\{\{h,sd_1,td_2\}\}$
$\{fd\}$	$\{r_2\}$	$\{fd.\}$	$\{\{fd\}\}$
$\{fd,sd_1\}$	$\{r_1\}$	$\{fd.\}$	$\{\{fd,sd_1\}\}$
$\{fd,td_2\}$	$\{r_2\}$	$\{fd.,td_2.\}$	$\{\{fd,td_2\}\}$
$\{fd,sd_1,td_2\}$	$\emptyset$	$\{fd.,td_2.\}$	$\{\{fd,sd_1,td_2\}\}$
$\{h,fd\}$	$\{r_2,r_3\}$	$\{h.,fd.\}$	$\{\{h,fd\}\}$
$\{h,fd,sd_1\}$	$\{r_1,r_3\}$	$\{h.,fd.\}$	$\{\{h,fd,sd_1\}\}$
$\{h,fd,td_2\}$	$\{r_2,r_3\}$	$\{h.,fd.,td_2.\}$	$\{\{h,fd,td_2\}\}$
$\{h,fd,sd_1,td_2\}$	$\{r_3\}$	$\{h.,fd.,td_2.\}$	$\{\{h,fd,sd_1,td_2\}\}$

The set of minimal canonical modified sets is $Min(CM(P,Q),\subseteq) = Min(CM(P,Q),\leq) = \{(\{r_1,r_3\},\emptyset),(\{r_2,r_3\},\emptyset),(\emptyset,\{fd.,td_2.\}),(\{r_1\},\{fd.\}),(\{r_2\},\{fd.\}),(\{r_3\},\{td_2.\})\}$ and corresponds to the set of modified sets.

6. Logical Properties

We now provide logical properties of the proposed operators through a set of postulates. As mentioned in Section 2.3, Hansson and Wassermann (2002) formulated postulates for characterizing belief base revision in a classical (monotonic) setting. According to Zhuang et al. (2016b) we now adapt them within the non-monotonic ASP framework.

Let P, Q, R, X, Y be normal logic programs and $\star$ be a revision operator.

Success	$Q \subseteq P \star Q$.
Inclusion	$P \star Q \subseteq P \cup Q$.
Inclusion^{-}	$P \cup Q \subseteq P \star Q$.
Consistency	If Q is m-consistent then $P \star Q$ is consistent.
Vacuity	If $P \cup Q$ is consistent then $P \star Q = P \cup Q$.
$R-$*Relevance*	If $D \neq \emptyset$ and $D \subseteq P \setminus (P \star Q)$ then $(P \star Q) \cup D$ is inconsistent.
$A-$*Relevance*	If $E \neq \emptyset$ and $E \subseteq (P \star Q) \setminus (P \cup Q)$ then $(P \star Q) \setminus E$ is inconsistent.
$M-$*Relevance*	If $(D,E) \neq (\emptyset,\emptyset)$, $D \subseteq P \setminus (P \star Q)$ and $E \subseteq (P \star Q) \setminus (P \cup Q)$ then $((P \star Q) \cup D) \setminus E$ is inconsistent.
Uniformity	If for all subsets D of P, $D \cup Q$ is consistent if and only if $D \cup R$ is consistent and if for all supersets E of P, $E \cup Q$ is consistent if and only if $E \cup R$ is consistent then $P \setminus (P \star Q) = P \setminus (P \star R)$ and $(P \star Q) \setminus (P \cup Q) = (P \star R) \setminus (P \cup R)$.

The meaning of the postulates is the following. *Success* gives priority to new information. *Consistency* expresses that the result of revision is consistent whenever the input is m-consistent. This differs from the monotonic setting where postulate *Consistency* requires the consistency of new information. *Inclusion* states that the union of the initial logic programs is the upper bound of any revision operation. *Inclusion*$^{-}$, inverse Inclusion, states that the union of the initial logic programs is the lower bound of any revision operation. Within a monotonic setting, the only way to restore consistency is to drop some beliefs and this is expressed by *Inclusion*, however within an ASP framework, we can restore consistency by adding rules we thus introduce *Inclusion*$^{-}$. *Vacuity* establishes that if the input is consistent with the initial logic program then the revised logic program equals the union of them. $R-$*Relevance* expresses the intuition that nothing is removed from the original logic program unless its removal contributes in some way to make the result consistent. $A-$*Relevance* expresses the intuition that nothing is added from the original logic program unless its addition contributes in some way to make the result consistent. $M-$*Relevance* expresses the intuition that nothing is removed and nothing is added from the original logic program unless its removal and addition contributes in some way to make the result consistent. Note that if $D = \emptyset$, respectively $E = \emptyset$, then $M-$*Relevance* turns out to be $A-$*Relevance*, respectively $R-$*Relevance*. Within a monotonic setting, since the only strategy is to remove beliefs, Postulate *Relevance* corresponds to $R-$*Relevance*, however within an ASP framework the addition strategy leads to $A-$*Relevance* and the removal and addition strategy leads to $M-$*Relevance*. *Uniformity* determines the condition under which two revising logic program Q and R produce the same changes to the initial logic program P. More precisely, the rules removed from P and the rules added to P in presence of Q are the same ones to those in presence of R.

Theorem 14. *For any selection function f,*

- *if $\star_{RSR(f)}$ is defined, $\star_{RSR(f)}$ satisfies Success, Consistency, Inclusion, Vacuity, $R-$Relevance and Uniformity,*
- *if $\star_{ASR(f)}$ is defined, $\star_{ASR(f)}$ satisfies Success, Consistency, Inclusion^{-}, Vacuity, $A-$Relevance and Uniformity,*
- *if $\star_{MSR(f)}$ is defined, $\star_{MSR(f)}$ satisfies Success, Consistency, Vacuity, $M-$Relevance and Uniformity.*

The families of $\star_{RSR(f)}$ $\star_{ASR(f)}$, and $\star_{MSR(f)}$ operators satisfy most of the properties that capture a good expected behaviour of revision operators, that is, success: the new logic program is accepted, consistency: the result of revision is consistent and minimal change: the initial logic program is changed as little as possible according to minimality with respect to set inclusion.

7. Complexity

This section deals with the complexity of the three revision operators.

7.1 Complexity of Rule-Based Revision by Removal

Traditionally, in the field of belief revision, the central problem addressed to assess the complexity of a revision operator is the model checking problem, who asks, given a belief base K, a formula ϕ, a revision operation $*$, and a model m, whether $m \models K * \phi$ holds. Translated in our framework, this problems could be stated as follows: given two normal logic programs P and Q, a revision operator $\star_{RSR(f)}$ and a set of atoms X, does X belongs to $AS(P\star_{RSR(f)}Q)$? Unfortunately, this question depends on the complexity of the selection function f, whose goal is to select a removed set among $\mathscr{R}(P,Q)$. So we will study another problem, which is more meaningful in our case, the ASPMODELCHECKING(RS) problem, which is defined as follows:

Name: ASPMODELCHECKING(RS)
Input: P and Q two logic programs, X a set of atoms.
Question: does there exist $R \in \mathscr{R}(P,Q)$ such that X belongs to $AS((P \setminus R) \cup Q)$?
And the following result holds.

Theorem 15. ASPMODELCHECKING(RS) *is* **DP**-*complete.*

7.2 Complexity of Rule-Based Revision by Addition

The decision problem we are studying is the ASPMODELCHECKING(AS) problem, which is defined as follows:

Name: ASPMODELCHECKING(AS)
Input: P and Q two logic programs, X a subset of $atom(P \cup Q)$.
Question: does there exist $Y \in \mathscr{A}(P,Q)$, such that X belongs to $AS((P \cup Y \cup Q)$?
And the following result holds.

Theorem 16. ASPMODELCHECKING(AS) *is* **DP**-*complete.*

7.3 Complexity of Rule-Based Revision by Modification

The decision problem under study is the ASPMODELCHECKING(MS) problem, which is defined as follows:

Name : ASPMODELCHECKING(MS)
Input : P and Q two logic programs, X a subset of $atom(P \cup Q)$.
Question : does there exist $(R,Y) \in \mathscr{M}(P,Q)$, such that X belongs to $AS((P \setminus R) \cup Y \cup Q)$?

And the following result holds.

Theorem 17. ASPMODELCHECKING(MS) *is* **DP**-*complete.*

8. Related Works

As mentioned in the introduction, the first approaches dealing with logic programs in a dynamic setting focused on logic programs update (Zhang & Foo, 1998; Sakama & Inoue, 1999; Alferes et al., 2000; Eiter et al., 2002).

Logic programs change in the same spirit as belief sets change in a propositional setting has first been addressed by Eiter et al. (2002). Belief sets for logic programs are defined as sets of rules satisfied by the interpretations corresponding to answer sets. However the proposed approach violates most of the AGM postulates for revision and update postulates.

Delgrande et al. (2008, 2009, 2013) generalized classical model-based revision operators to logic programs. Belief sets are then represented by SE-models and change operations (revision and merging) based on distance between interpretations have then been extended to logic programs with SE-model semantics. The proposed change operations stemming from a distance between SE-models have been implemented in ASP. Schwind and Inoue (2016) provided a constructive characterization of logic programs revision operators in terms of preorders over interpretations with further conditions specific to SE-models. Belief sets update has also been addressed in the same spirit by Slota and Leite (2010, 2014). Binnewies, Zhuang, and Wang (2015) generalized partial meet revision and contraction to logic programs under SE-models semantics. More recently, Zhuang et al. (2016b) proposed a new revision operator, called llp revision, using a strategy based on the removal or the addition and/or removal of rules and stemming from the minimization of the symmetric difference between SE-models. SE-model-based approaches are belief set revision while we address belief base revision. Moreover, the notion of consistency dealt with SE-model-based approaches is not the same since it stems from the existence of SE-models.

Krümpelmann and Kern-Isberner (2012) extended belief base revision to ASP with the "Remainder Sets" approach for screened consolidation, stemming from semi-revision proposed by Hansson (1997). The "Removed Sets" approach for merging and revision has been extended to ASP (Hué et al., 2009, 2013), called respectively, ΠRSF and ΠRSR. These two approaches only use a strategy stemming from the removal of rules, but they differ on the minimality criteria, set inclusion for the "Remainder Sets" approach and cardinality for ΠRSR. No semantic characterization was provided for these two approaches, nor computational complexity study.

More recently, another approach called SLP revision (Zhuang, Delgrande, Nayak, & Sattar, 2016a; Zhuang et al., 2016b) has been proposed. The strategy stems from the removal or the addition and/or removal of rules. Let P be the initial logic program and Q be the new one. The removal (respectively addition and addition and/or removal) strategies stem from the construction

of "s-removal" (respectively "s-expansion" and "s-compatible") logic programs which are subsets of P consistent with Q maximal with respect to set inclusion (respectively sets of rules containing P consistent with Q and minimal with respect to set inclusion, and a combination of both for the third strategy). They are the dual sets of potential removed sets, potential added sets and potential modified sets respectively.

Note that the families of revision operators proposed in this paper differ from SLP revision operators since the maximality criterion is set inclusion for SLP whereas the minimality criterion for the RSR, ASR and MSR revision operators is cardinality. Moreover, in this paper we go a step further since we provide a semantic characterization in terms of answer sets for the RSR, ASR and MSR revision operators which also covers SLP revision operators. Besides we provide complexity results for RSR, ASR and MSR revision operators.

9. Concluding Discussion

The paper addresses the problem of base revision in logic programming under Answer Set semantics. It proposes new families of revision operators within the framework of Answer Set Programming. ASP Base Revision by removal (RSR) is a direct extension of base revision within the monotonic classical setting stemming from a removal of rules strategy. ASP base Revision by Addition (ASR) and ASP base Revision by Modification (MSR) exploit the non-monotony of the ASP framework allowing addition or removal and/or addition of rules strategies for revision. The paper provides a semantic characterization for each family of operators in terms of answer sets. This is an important contribution since it allows one to go from the evolution of a syntactic rule-based revision operator to the evolution of its semantic content. The paper then presents a study of logical properties of the proposed revision operators in terms of satisfaction of a set of postulates adapted within the non-monotonic ASP framework from the Hansson's postulates for base revision in a classical setting. Finally, the paper gives computational complexity results for each proposed family of operators. We would have liked to compare these complexity results to the ones pertaining to SLP revision. We proved that SLP belongs to the **DP** class. However, deciding if SLP is *DP*-complete is still an open question.

There are several issues to address as future work. Providing a representation theorem for RSR, ASR and MSR is an interesting issue. To this end, we need to investigate for additional postulates capturing a refined notion of relevance with respect to cardinality for each family of operators. Future work will also be dedicated to the ASP implementation of the proposed families of revision operators and to an experimental study.

Another issue is introducing uncertainty and studying belief base revision when beliefs are represented by possibilistic logic programs under possibilistic answer set semantics (Nicolas, Garcia, Stéphan, & Lefèvre, 2006; Bauters, Schockaert, Cock, & Vermeir, 2015). Recently, Garcia et al. (2018) introduced a "RSR possibilistic ASP base revision" which is based on the strategy of removing a minimum number of the least certain possibilistic rules, keeping the possibilistic rules not involved in inconsistency. A future issue is to investigate how to define ASP revision operators stemming from the addition or removal and/or addition of rules strategies within the framework of possibilistic ASP.

Acknowledgments

This work was supported by the ANR project ASPIQ (ANR-12-BS02-0003).

Appendix A. Proofs of Theorems

Proof of Theorem 1. Let $m \in AS(P \cup Q)$. Then, by definition, $Head(GR(P \cup Q, m)) = m$. If $m \notin Mod(P)$ then there exists a rule $r \in P$ such that $body^+(r) \subseteq m$ and $body^-(r) \cap m = \emptyset$ and $head(r) \notin m$. By definition of GR, $r \in GR(P \cup Q, m)$. Since $m = Head(GR(P \cup Q, m))$, $head(r) \in m$ and there is a contradiction. Thus $m \in Mod(P)$ □

Lemma 1. *Let P be a logic program and r be a rule. Let $m \in AS(P)$. If m satisfies r then $m \in AS(P \cup \{r\})$.*

Proof of Lemma 1. Let $m \in AS(P)$, then $m = Cn(GR(P,m)^+)$. $GR(P,m) = \{r \in P$ such that $body^+(r) \subseteq m$, $body^-(r) \cap m = \emptyset\}$. If m satisfies r then (i) $body^+(r) \not\subseteq m$ or $body^-(r) \cap m \neq \emptyset$, or (ii) $head(r) \in m$.

(i) if $body^+(r) \not\subseteq m$ or $body^-(r) \cap m \neq \emptyset$, then $r \notin GR(P \cup \{r\}, m)$. Since GR is increasing monotonic when the program increases, $GR(P \cup \{r\}, m) = GR(P,m)$ thus $Cn(GR(P \cup \{r\}, m)^+) = Cn(GR(P,m)^+) = m$, therefore $m \in AS(P \cup \{r\})$.

(ii) if $body^+(r) \subseteq m$, $body^-(r) \cap m = \emptyset$ and $head(r) \in m$ then $r \in GR(P \cup \{r\}, m)$. Thus $GR(P \cup \{r\}, m) = GR(P,m) \cup \{r\}$ and $Cn(GR(P \cup \{r\}, m)^+) = m$, therefore $m \in AS(P \cup \{r\})$.

□

Lemma 2. *Let P, Q and X be logic programs with $X \subseteq P$. We have that $\forall m \in AS((P \setminus X) \cup Q)$, if $X \in \mathscr{PR}(P,Q)$ then $X = Fal(P,m)$.*

Proof of Lemma 2. Let $X \in \mathscr{PR}(P,Q)$ and $m \in AS((P \setminus X) \cup Q)$. m is an answer set of $(P \setminus X) \cup Q$ thus $m = Head(GR((P \setminus X) \cup Q, m))$. We show that $X = Fal(P,m)$.

- We first show that $Fal(P,m) \subseteq X$.
  Let $r \in Fal(P,m)$. By definition, $r \in P$, $body(r)^+ \subseteq m$, $body(r)^- \cap m = \emptyset$ and $head(r) \notin m$. Since $head(r) \notin m$, we have $r \notin GR((P \setminus X) \cup Q, m) = \{r \in (P \setminus X) \cup Q$ s.t. $body(r)^+ \subseteq m$ and $body(r)^- \cap m = \emptyset\}$, thus $r \notin (P \setminus X) \cup Q$ thus $r \notin (P \setminus X)$ but $r \in P$. Therefore $r \in X$.

- We now show that $X \subseteq Fal(P,m)$.
  Let us suppose that there exists $r \in X$ s.t. $r \notin Fal(P,m)$, thus m satisfies r. Since $m \in AS((P \setminus X) \cup Q)$, by Lemma 1, $m \in AS((P \setminus X) \cup Q \cup \{r\})$. Thus, there exists $X' = X \setminus \{r\}$ such that $m \in AS((P \setminus X') \cup Q)$. This contradicts the minimality with respect to set inclusion of Definition 4.

□

Proof of Theorem 2. Let P and Q be two logic programs.

- We fist show that $\mathscr{PR}(P,Q) \subseteq Min(CR(P,Q),\subseteq)$.
  We first prove that $\forall X \in \mathscr{PR}(P,Q)$, $\exists m \in Mod(Q)$ s.t. $X \in CR(P,Q,m)$.
  Let $X \in \mathscr{PR}(P,Q)$. By Definition 4, $AS((P \setminus X) \cup Q) \neq \emptyset$. Let $m \in AS((P \setminus X) \cup Q)$. By Theorem 1, $m \in Mod(Q)$ and by Lemma 2, $X = Fal(P,m)$. Therefore $X \in CR(P,Q,m)$.

  $\forall X \in \mathscr{PR}(P,Q)$, $\exists m \in Mod(Q)$ s.t. $X \in CR(P,Q,m)$, thus $\forall X \in \mathscr{PR}(P,Q)$, $X \in CR(P,Q)$. If $X \notin Min(CR(P,Q),\subseteq)$ then $\exists X' \subset X$ s. t. $X' \in CR(P,Q,m)$, therefore $\exists m' \in Mod(Q)$ s.t. $m' \in AS((P \setminus X') \cup Q)$ therefore $AS((P \setminus X') \cup Q) \neq \emptyset$ which contradicts the minimality with respect to set inclusion of Definition 4.

- We now show that $Min(CR(P,Q),\subseteq) \subseteq \mathscr{PR}(P,Q)$.
  $\forall X \in Min(CR(P,Q),\subseteq)$, since $CR(P,Q) = \bigcup_{m \in Mod(Q)} CR(P,Q,m)$, there exists $m \in Mod(Q)$ s.t. $X \in CR(P,Q,m)$. By Definition 8

  - $X = Fal(P,m)$ thus $X \subseteq P$.
  - $m \in AS((P \setminus X) \cup Q)$ thus $AS((P \setminus X) \cup Q) \neq \emptyset$.

  We show by contradiction that for each $X' \subset X$, $(P \setminus X') \cup Q$ is inconsistent. Let us suppose that there exists some $X' \subset X$ such that $(P \setminus X') \cup Q$ is consistent. Let X'_0 be one least (with respect to inclusion) such X'. Then $(P \setminus X'_0) \cup Q$ is consistent and $\forall Y \subset X'_0$, $(P \setminus Y) \cup Q$ is inconsistent. Therefore by Definition 4, $X'_0 \in \mathscr{PR}(P,Q)$ and, by the first part of the proof of Theorem 2, $X'_0 \in Min(CR(P,Q),\subseteq)$. Now $X'_0 \subset X$, thus $X \notin Min(CR(P,Q),\subseteq)$. It contradicts the hypothesis. We can conclude that for each $X' \subset X$, $(P \setminus X') \cup Q$ is inconsistent.
  Therefore $X \in \mathscr{PR}(P,Q)$.

□

Proof of Theorem 5. Let P be a logic program. Let us suppose that P is m-consistent: $\exists m \in Mod(P)$, thus for all $r \in P$, $body^+(r) \not\subseteq m$ or $body^-(r) \cap m \neq \emptyset$ or $head(r) \in m$. Then, $\forall r \in GR(P,m)$, $head(r) \in m$, and $Head(GR(P,m)) \subseteq m$.
Let $Y = m \setminus Cn(GR(P,m)^+)$ (thus $fact(Y) = Nded(m,P)$). It follows that $m = Cn(GR(P,m)^+) \cup Y$.
Since we have:
(1) $Y \subseteq m$, thus $fact(Y) \subseteq GR(P \cup fact(Y),m)$ and $Y \subseteq Cn(GR(P \cup fact(Y),m)^+)$,
(2) Cn and GR are increasing, thus $Cn(GR(P,m)^+) \subseteq Cn(GR(P \cup fact(Y),m)^+)$,
(3) $Head(GR(P,m)) \subseteq m$, $Y = Head(GR(fact(Y),m)) \subseteq m$ and $GR(P \cup fact(Y),m) = GR(P,m) \cup GR(fact(Y),m)$ thus $Head(GR(P \cup fact(Y),m)) \subseteq m$ and $Cn(GR(P \cup fact(Y),m)^+) \subseteq Head(GR(P \cup fact(Y),m))$ thus $Cn(GR(P \cup fact(Y),m)^+) \subseteq m$,
we have $m = Cn(GR(P,m)^+) \cup Y \subseteq Cn(GR(P \cup fact(Y),m)^+) \subseteq m$ and it follows that $m = Cn(GR(P \cup fact(Y),m)^+)$. By definition, m is an answer set of $P \cup fact(Y)$: $m \in AS(P \cup Nded(m,P))$

□

Lemma 3. *Let P and Q be logic programs and Y be a set of facts.*
$\forall m \in AS(P \cup Q \cup Y)$, if $Y \in \mathscr{PA}(P,Q)$ then $Y \subseteq Nded(m,P \cup Q)$.

Proof of Lemma 3. Let $Y \in \mathscr{PA}(P,Q)$ and $m \in AS(P \cup Q \cup Y)$. Y is a set of facts, so $Y \subseteq GR(P \cup Q \cup Y,m)$ and $atom(Y) \subseteq m$.
We show that $atom(Y) \cap Cn(GR(P \cup Q,m)^+) = \emptyset$ and thus that $Y \subseteq Nded(m,P \cup Q)$. Let a be an

atom such that $fact(a) \in Y$. Let us suppose that $a \in Cn(GR(P \cup Q, m)^+)$. Then $a \in Cn(GR(P \cup Q \cup (Y \setminus \{fact(a)\}), m)^+)$ (since GR and Cn are increasing when the program increases). Thus $Cn(GR(P \cup Q \cup (Y \setminus \{fact(a)\}), m)^+) = Cn(GR(P \cup Q \cup Y, m)^+) = m$ and $\exists Y' = Y \setminus \{fact(a)\} \subset Y$ such that $P \cup Y' \cup Q$ is consistent. It contradicts the hypothesis (Y is a potential added set). Therefore $a \notin Cn(GR(P \cup Q, m)^+)$ and $Y \subseteq Nded(m, P \cup Q)$.

□

Proof of Theorem 6. Let P and Q be two logic programs.

- We first show that $\mathscr{PA}(P,Q) \subseteq Min(CA(P,Q), \subseteq)$.
  Let $X \in \mathscr{PA}(P,Q)$. By Definition 10, $P \cup X \cup Q$ is consistent, that is, $\exists m \in AS(P \cup Q \cup X)$, and, for each $X' \subset X$, $P \cup X' \cup Q$ is inconsistent.
  Let $m \in AS(P \cup Q \cup X)$. We have to show: (1) $m \in Mod(P \cup Q)$, (2) $X \in CA(P,Q,m)$, and (3) $X \in Min(CA(P,Q), \subseteq)$.
  (1) By Theorem 1, since $m \in AS(P \cup Q \cup X)$, $m \in Mod(P \cup Q)$.
  (2) By Lemma 3, $X \subseteq Nded(m, P \cup Q)$. Now, by hypothesis, $m \in AS(P \cup Q \cup X)$ and for each $X' \subset X$, $P \cup X' \cup Q$ is inconsistent. Thus $X \in CA(P,Q,m) \subseteq CA(P,Q)$.
  (3) Let us show by contradiction that $X \in Min(CA(P,Q), \subseteq)$. Let $X' \subseteq X$ such that $X' \in Min(CA(P,Q), \subseteq)$. $X' \in CA(P,Q)$ then $\exists m'$ s.t. $X' \subseteq Nded(m', P \cup Q)$ and $m' \in AS(P \cup Q \cup X')$. So m' is an answer set of $P \cup Q \cup X'$ and if $X' \subset X$, $X \notin \mathscr{PA}(P,Q)$. Therefore $X = X'$ and $X \in Min(CA(P,Q), \subseteq)$.

- We now show the other direction: $Min(CA(P,Q), \subseteq) \subseteq \mathscr{PA}(P,Q)$.
  Let $X \in Min(CA(P,Q), \subseteq)$. By Definition 14, $P \cup Q \cup X$ is trivially consistent. We show by contradiction that for each $X' \subset X$, $P \cup Q \cup X'$ is inconsistent. Let us suppose that there exists some $X' \subset X$ such that $P \cup Q \cup X'$ is consistent. Let X'_0 be one least (with respect to inclusion) such X'. Then $P \cup Q \cup X'_0$ is consistent and $\forall Y \subset X'_0$, $P \cup Q \cup Y$ is inconsistent. Therefore $X'_0 \in \mathscr{PA}(P,Q)$ and, by the first part of the proof of Theorem 6, $X'_0 \in Min(CA(P,Q), \subseteq)$. Now $X'_0 \subset X$, thus $X \notin Min(CA(P,Q), \subseteq)$. It contradicts the hypothesis. We can conclude that for each $X' \subset X$, $P \cup Q \cup X'$ is inconsistent. Therefore $X \in \mathscr{PA}(P,Q)$.

□

Proof of Theorem 9. Let P and Q be logic programs. By Definition 12, there exists a selection function f such that $P \star_{ASR(f)} Q$ is defined if and only if $\mathscr{A}(P,Q) \neq \emptyset$.

If $\mathscr{A}(P,Q) \neq \emptyset$ then, by Definitions 10 and 11, there exists some Y such that $P \cup Q \cup Y$ is consistent and, by Theorem 1, $P \cup Q$ is m-consistent.

For the other direction, let us suppose that $P \cup Q$ is m-consistent: $\exists m \in Mod(P \cup Q)$. By Theorem 5, m is an answer set of $P \cup Q \cup Nded(m, P \cup Q)$. Since $\exists Y$, $P \cup Q \cup Y$ is consistent, $\mathscr{A}(P,Q) \neq \emptyset$ and $P \star_{ASR(f)} Q$ is defined. □

Lemma 4. *Let P, Q be two logic programs, and X, Y be two sets of rules. If $(X,Y) \in \mathscr{PM}(P,Q)$ then $X \in \mathscr{PR}(P, Q \cup Y)$ and $Y \in \mathscr{PA}(P \setminus X, Q)$.*

Proof of Lemma 4. Let $(X,Y) \in \mathscr{PM}(P,Q)$. By Definition 17, $X \subseteq P$, $(P \backslash X) \cup Y \cup Q$ is consistent and, for each (X',Y') such that $(X',Y') \subset (X,Y)$, $(P \backslash X') \cup Y' \cup Q$ is inconsistent. By Definition 16, for each $X' \subset X$, $(P \backslash X') \cup Y \cup Q$ is inconsistent and, for each $Y' \subset Y$, $(P \backslash X) \cup Y' \cup Q$ is inconsistent. Thus, by Definition 4, $X \in \mathscr{PR}(P, Q \cup Y)$ and, by Definition 10, $Y \in \mathscr{PA}(P \setminus X, Q)$. □

Proof of Theorem 10. Let P and Q be two logic programs.

- We first prove $\mathscr{PM}(P,Q) \subseteq Min(CM(P,Q),\subseteq)$.
  Let $(X,Y) \in \mathscr{PM}(P,Q)$. By Lemma 4, $X \in \mathscr{PR}(P,Q \cup Y)$ and $Y \in \mathscr{PA}(P \setminus X,Q)$. And by Definition 17, $(P \setminus X) \cup Q \cup Y$ is consistent. Let $m \in AS((P \setminus X) \cup Q \cup Y)$. By Lemma 2, $X = Fal(P,m)$. And by Lemma 3, $Y \subseteq Nded(m,(P \setminus X) \cup Q)$. For the minimality, we have $(X,Y) \in \mathscr{PM}(P,Q)$ thus, by Definition 17, for each (X',Y') such that $(X',Y') \subset (X,Y)$, $(P \setminus X') \cup Y' \cup Q$ is inconsistent.Thus $(X',Y') \notin CM(P,Q)$. And $(X,Y) \in Min(CM(P,Q),\subseteq)$.

- We now prove that $Min(CM(P,Q),\subseteq) \subseteq \mathscr{PM}(P,Q)$.
  Let $(X,Y) \in Min(CM(P,Q),\subseteq)$. By Definition 20, there exists $m \in Mod(Q)$ s.t. $X = Fal(P,m)$. By definition, $Fal(P,m) \subseteq P$, thus $X \subseteq P$.
  By Definition 20, $(P \setminus X) \cup Q \cup Y$ is consistent. We show by the absurd that for each $(X',Y') \subset (X,Y)$, $(P \setminus X') \cup Q \cup Y'$ is inconsistent. Let us suppose that there exists $(X',Y') \subset (X,Y)$ s.t. $(P \setminus X') \cup Q \cup Y'$ is consistent. Let (X'_0,Y'_0) be one least (with respect to inclusion) such (X',Y'). Then $(P \setminus X'_0) \cup Q \cup Y'_0$ is consistent and for each $(A,B) \subset (X'_0,Y'_0)$, $(P \setminus A) \cup Q \cup B$ is inconsistent. Therefore $(X'_0,Y'_0) \in \mathscr{PM}(P,Q)$ and, by the first part of the proof of Theorem 10, $(X'_0,Y'_0) \in Min(CM(P,Q),\subseteq)$. Now $(X'_0,Y'_0) \subset (X,Y)$, thus $(X,Y) \notin Min(CM(P,Q),\subseteq)$. It contradicts the hypothesis. We can conclude that for each $(X',Y') \subset (X,Y)$, $(P \setminus X') \cup Q \cup Y'$ is inconsistent.
  Therefore $(X,Y) \in \mathscr{PM}(P,Q)$.

□

Proof of Theorem 13. Let P and Q be logic programs. By Definition 19, there exists a selection function f such that $P \star_{MSR(f)} Q$ is defined if and only if $\mathscr{M}(P,Q) \neq \emptyset$.

If $\mathscr{M}(P,Q) \neq \emptyset$ then, by Definitions 17 and 18, there exists some (X,Y) such that $(P \setminus X) \cup Q \cup Y$ is consistent and, by Theorem 1, Q is m-consistent.

For the other direction, let us suppose that Q is m-consistent: $\exists m \in Mod(Q)$. By Theorem 5, m is an answer set of $Q \cup Nded(m,Q) = (P \setminus P) \cup Q \cup Nded(m,Q)$. Since $\exists (X,Y)$ with $X = P$ and $Y = Nded(m,Q)$ such that $(P \setminus X) \cup Q \cup Y$ is consistent, $\mathscr{M}(P,Q) \neq \emptyset$ and $P \star_{MSR(f)} Q$ is defined. □

Proof of Theorem 14. We consider the three revision operators:

- We first show that for any selection function f, if $\star_{RSR(f)}$ is defined, $\star_{RSR(f)}$ satisfies *Success*, *Consistency*, *Inclusion*, *Vacuity*, $R-Relevance$ and $Uniformity$.

  By definition, $\star_{RSR(f)}$ satisfies, *Success*, $m-Consistency$, *Inclusion*, and *Vacuity*.
  $R-Relevance$: for any selection function f, $\forall D \subseteq P \setminus (P \star_{RSR(f)} Q)$ we have $D \subseteq f(\mathscr{R}(P,Q))$, since $f(\mathscr{R}(P,Q))$ is a removed set, by Definition 5 we have $(P \star_{RSR(f)} Q) \cup D$ is inconsistent.
  $Uniformity$: In case of $\star_{RSR(f)}$ we only consider the removed rules, thus $E = \emptyset$ and this postulate may be reformulated as follows:
  If for all subsets D of P, $D \cup Q$ is consistent if and only if $D \cup R$ is consistent then $P \setminus (P \star Q) = P \setminus (P \star R)$.

  We first prove that if for all subsets of D of P, $D \cup Q$ is consistent if and only if $D \cup R$ is consistent, then $\mathscr{R}(P,Q) = \mathscr{R}(P,R)$. We have $\mathscr{PR}(P,Q) = \{S \mid S \subseteq P, (P \setminus S) \cup Q$ is consistent, and for each $S' \subset S, (P \setminus S') \cup Q$ is inconsistent$\}$ and $\mathscr{PR}(P,R) = \{S \mid S \subseteq P, (P \setminus S) \cup R$ is consistent and for each $S' \subset S, (P \setminus S') \cup R$ is inconsistent$\}$. If $\mathscr{PR}(P,Q) \neq$

$\mathscr{PR}(P,R)$ by Definition 4 there is a contradiction thus $\mathscr{PR}(P,Q) = \mathscr{PR}(P,R)$ therefore $\mathscr{R}(P,Q) = \mathscr{R}(P,R)$. Hence for any selection function f we have $P \setminus (P \star_{RSR(f)} Q) = P \setminus (P \star_{RSR(f)} R)$.

- We then show that for any selection function f, if $\star_{ASR(f)}$ is defined, $\star_{ASR(f)}$ satisfies *Success*, *Consistency*, $Inclusion^{-}$, *Vacuity*, $A-Relevance$ and $Uniformity$.

  By definition, $\star_{ASR(f)}$ satisfies *Success*, *Consistency*, $Inclusion^{-}$ and *Vacuity*.
  $A-Relevance$: for any selection function f, $\forall E \subseteq P \star_{ASR(f)} Q) \setminus (P \cup Q)$ we have $E \subseteq f(\mathscr{A}(P,Q))$, since $f(\mathscr{A}(P,Q))$ is an added set, by Definition 11 we have $(P \star_{ASR(f)} R) \setminus E$ is inconsistent.
  $Uniformity$: in case of $\star_{ASR(f)}$ we only consider the added rules, thus $D = \emptyset$ and this postulate may be reformulated as follows:
  If for all supersets E of P, $E \cup Q$ is consistent if and only if $E \cup R$ is consistent then $(P \star Q) \setminus (P \cup Q) = (P \star R) \setminus (P \cup R)$.

  We first prove that if for all supersets E of P, $E \cup Q$ is consistent if and only if $E \cup R$ is consistent then $\mathscr{A}(P,Q) = \mathscr{A}(P,R)$. Let U and V be sets of rules made from the vocabulary of P and Q and P and R respectively, we have $\mathscr{PA}(P,Q) = \{U \mid (P \cup U) \cup Q$ is consistent, and for each $U' \subset U, (P \cup U') \cup Q$ is inconsistent.$\}$ $\mathscr{PA}(P,R) = \{V \mid (P \cup V) \cup R$ is consistent, and for each $V' \subset V, (P \cup V') \cup R$ is inconsistent.$\}$ If $\mathscr{PA}(P,Q) \neq \mathscr{PA}(P,R)$ by Definition 10 there is a contradiction, thus $\mathscr{PA}(P,Q) = \mathscr{PA}(P,R)$ therefore $\mathscr{A}(P,Q) = \mathscr{A}(P,R)$. Hence for any selection function f we have $(P \star_{ASR(f)} Q) \setminus (P \cup Q) = (P \star_{ASR(f)} R) \setminus (P \cup R)$.

- Finally, we show that for any selection function f, if $\star_{MSR(f)}$ is defined, $\star_{MSR(f)}$ satisfies *Success*, *Consistency*, *Vacuity*, $M-Relevance$ and $Uniformity$.

  By definition $\star_{MSR(f)}$ satisfies *Success*, *Consistency* and *Vacuity*.
  $M-Relevance$: for any selection function f, for all $(D,E) \neq (\emptyset,\emptyset)$ such that $D \subseteq P \setminus (P \star_{MSR(f)} Q)$ and $E \subseteq (P \star_{MSR(f)} Q) \setminus (P \cup Q)$ we have $(D,E) \subseteq f(\mathscr{M}(P,Q))$, since $f(\mathscr{M}(P,Q))$ is a modified set, by Definition 18 we have then $((P \star_{MSR(f)} Q) \cup E) \setminus D$ is inconsistent.
  $Uniformity$: we first prove that if for all subsets of D of P, $D \cup Q$ is consistent if and only if $D \cup R$ is consistent and if for all supersets E of P, $E \cup Q$ is consistent if and only if $E \cup R$ is consistent then $\mathscr{M}(P,Q) = \mathscr{M}(P,R)$. If $\mathscr{M}(P,Q) \neq \mathscr{M}(P,R)$ by Definition 17 there is a contradiction thus $\mathscr{M}(P,Q) = \mathscr{M}(P,R)$. Hence for any selection function f we have $P \setminus (P \star_{MSR(f)} Q) = P \setminus (P \star_{MSR(f)} R)$ and $(P \star_{MSR(f)} Q) \setminus (P \cup Q) = (P \star_{MSR(f)} R) \setminus (P \cup R)$.

□

Lemma 5. ASPMODELCHECKING(RS) *is in* **DP**.

Proof of Lemma 5. We recall that a language L is in the class **DP** if and only if there are two languages $L_1 \in$ **NP** and $L_2 \in$ **coNP** such that $L = L_1 \cap L_2$ (Papadimitriou, 1994).

Finding such a set R can be broken down as follows:

1. Is there a set $R \subseteq P$ such that $X \in AS((P \setminus R) \cup Q)$?
   (a) Guess a set of rules R
   (b) Check that $R \subseteq P$

(c) Check that $X \in AS((P \setminus R) \cup Q)$

2. Compute $k_R = |R|$
3. Check that $R \in \mathscr{R}(P,Q)$:
   (a) Guess a set of rules R_0 and a set of atoms X_0
   (b) Check that $R_0 \subseteq P$
   (c) Check that $|R_0| < k_R$
   (d) Check that $X_0 \in AS((P \setminus R_0) \cup Q)$

The algorithmic difficulty concentrates in points 1 and 3. The algorithm described in point 1 can be solved in polynomial time on a non-deterministic Turing machine, and R is a certificate, because it succinctly proves that $X \in AS((P \setminus R) \cup Q)$. Thus, this subproblem is in **NP**. In the algorithm described in point 3, (R_0, X_0) is a succinct disqualification, that is, it proves that $X \notin \mathscr{R}(P,Q)$. This algorithm can run in polynomial time on a non deterministic Turing machine. Thus, it is in **coNP**. □

Lemma 6. ASPMODELCHECKING(RS) *is* **DP**-*complete.*

Proof of Lemma 6. The first step consists in building a transformation of the problem EXACTINDEPENDENTSET into ASPMODELCHECKING(RS). The problem EXACTINDEPENDENTSET is known to be **DP**-complete (Papadimitriou, 1994). It is defined as follows :
Name : EXACTINDEPENDENTSET
Input : a graph $G = (V,E)$, a positive integer $k \leq |V|$.
Question : Does G contain an independent set of size k, that is a subset $V' \subseteq V$ such that $|V'| = k$ and such that no two vertices in V' are joined by an edge in E, such that there is no other independent set with a size $k' > k$.
From a given graph $G = (V,E)$ with $V = \{x_1, \ldots, x_n\}$ we define a transformation $\tau(G) = (P,Q)$, which builds two logic programs P and Q as follows. We define $Q = Q_1 \cup Q_2 \cup Q_3 \cup Q_4$ with:

$$Q_1 = \{e(x,y). \mid (x,y) \in E\} \tag{1}$$

This describes the edges of G.
Atoms $is(x)$ reflects the presence of a vertex x in an independent set.

$$Q_2 = \{\leftarrow is(x), is(y), e(x,y). \mid \forall (x,y) \in E\} \tag{2}$$

This eliminates candidate vertices which are joined by an edge in E.

$$Q_3 = \{\leftarrow not\ is(x_1), \ldots, not\ is(x_n).\} \tag{3}$$

This eliminates the empty set of vertices as a solution.
The program P contains the following rules:

$$P = \{is(x). \mid x \in V\} \tag{4}$$

The rules of P states that V is an independent set.

Now we define a transformation γ of a set of vertices $V' \subseteq V$ into a set of atoms $\gamma(V')$ as follows:

$$\begin{array}{rcl} X_1 & = & \{is(x) \mid x \in V'\} \\ X_2 & = & \{e(x,y) \mid (x,y) \in E\} \\ \gamma(V') & = & X_1 \cup X_2 \end{array}$$

Then, given a graph G, and its transformation $\tau(G) = (P,Q)$, we show that G contains a maximum independent set S of size k, if and only if S corresponds to a set of atoms $X = \gamma(S)$ such that there exists a set $R \in \mathscr{R}(P,Q)$ such that $X \in AS((P \setminus R) \cup Q)$.

We first prove that if G contains a maximum independent set S of size k, then S corresponds to a set of atoms $X = \gamma(S)$ such that there exists a set $R \in \mathscr{R}(P,Q)$ such that $X \in AS((P \setminus R) \cup Q)$.

We suppose that G contains a maximum independent set of size k. Let us denote this set by S.

Consider the transformations $\tau(G) = (P,Q)$ and $\gamma(S) = X$, and consider the algorithm given in the proof of Lemma 5. Suppose, without loss of generality, that the set of rules R guessed in step 1(a) is such that $R = \{r \mid r \in P, head(r) \notin X_1\}$. $R \subseteq P$ by construction, so step 1(b) is verified. In order to check that $X \in AS((P \setminus R) \cup Q)$ (step 1(c)), we first compute $((P \setminus R) \cup Q)^X$. This set contains:

- all rules of Q_1 and Q_2 because they have an empty negative body.
- no rule of Q_3 because S is not empty, so there exists at least one atom $is(x), x \in S$ in $X = \gamma(S)$.
- $\{r \mid r \in P \setminus R\}$

Thus, $Cn(((P \setminus R) \cup Q)^X)$ contains:

- all atoms $e(x,y)$ such that $(x,y) \in E$, that is X_2
- all atoms $is(x)$ such that $is(x) \in X_1$, that is X_1.

By construction of X_1 and X_2, none of the constraints in Q_2 is satisfied, because by construction, all atoms $is(x) \in X_1$ correspond to vertices of an independent set of G, so, given any $(is(x), is(y)) \in X_1^2$, there is no corresponding $e(x,y) \in X_2$.

Thus $Cn(((P \setminus R) \cup Q)^X) = X_1 \cup X_2 = X$. This proves that $X \in AS((P \setminus R) \cup Q)$.

The second step of our algorithm computes the size k_R of the set R. Note that at this time we can conclude that S is an independent set of G, and thus G admits an independent set of size at least $k = |P| - k_R$.

Then, the third step looks for a pair (R_0, X_0), $R_0 \subseteq P$ being a set of rules and X_0 a set of atoms, such that $|R_0| < k_R$.

We examine the two cases: either we find such a pair, either we do not find it:

- Suppose that we find such a pair (R_0, X_0). This means R is not minimal with respect to cardinality, and consequently the number of atoms $is(x) \in X_0$ will be larger. If $X_0 \in AS((P \setminus R_0) \cup Q)$, then X_0 corresponds to an independent set S_0 such that $|S_0| > k$, because this means that none of the constraints in Q_2 are satisfied, that is, given any $(is(x), is(y)) \in X_0$, there is no corresponding $e(x,y) \in X_0$. Moreover, as $|R_0| < k_R$, we have $|\{is(x) | is(x) \in X_0\}| > k$, which contradicts the hypothesis.
- If we do not find such a pair, this means that there is no independent set which is larger than $k = |P \setminus R|$, and R is a removed set.

We can conclude that if S is a maximal independent set of size k, then there exists $R \in \mathscr{R}(P,Q)$, with $\tau(G) = (P,Q)$, such that $\gamma(S) \in AS((P \setminus R) \cup Q)$.

We now show that, if $S \subseteq V$ with $X = \gamma(S)$ is such that there exists a set $R \in \mathscr{R}(P,Q)$ such that $X \in AS((P \setminus R) \cup Q)$, then S is a maximum independent set of G of size k.

Let $\tau(G) = (P,Q)$. We suppose that there is a $S \subseteq V$ such that there exists $R \in \mathscr{R}(P,Q)$ such that $\gamma(S) \in AS((P \setminus R) \cup Q)$. We prove that S is a maximum independent set of size $k = |P \setminus R|$.

Let us compute $((P \setminus R) \cup Q)^{\gamma(S)}$. This program contains the following rules:

- all rules of Q_1 and Q_2, since they do not have a negative body.
- no rule from Q_3, as $\gamma(S)$ contains at least one atom $is(x), x \in S$.
- all rules of $P \setminus R$, that is, all rules $(is(x).)$ such that $(is(x).) \notin R$. But note that, as $\gamma(S) \in AS((P \setminus R) \cup Q)$, these rules $(is(x).)$ are such that $is(x) \in \gamma(S)$, and thus $x \in S$.

As $\gamma(S) \in AS((P \setminus R) \cup Q)$ by hypothesis, then $\forall is(x)., is(y). \in (P \setminus R)$ there is no rule $(e(x,y).) \in Q_1$, because if it was the case, there would exist a constraint $(\leftarrow is(x), is(y), e(x,y).)$ in Q_2 which would prohibit $\gamma(S)$ to be an answer set of $(P \setminus R) \cup Q$.

This proves that $\forall x,y \in S$, $(x,y) \notin E$, as no constraint of Q_2 is satisfied. So S is an independent set of G of size $k = |P \setminus R|$.

Moreover, as R is a removed set, there is no $R_0 \subseteq P$ with $|R_0| < |R|$ such that $AS((P \setminus R_0) \cup Q) \neq \emptyset$. This means that for any such set R_0, there exists a pair of rules $(is(x)., is(y).)$ in $P \setminus R_0$ such that there exist a rule $(e(x,y).) \in Q_1$, which means that $\{x \mid (is(x).) \in P \setminus R_0\}$ is not an independent set of G, and thus there is no independent set of G with a size larger than $k = |P \setminus R|$.

Thus S is a cardinality-maximal independent set of G of size $k = |P \setminus R|$.

Finally, note that the transformation τ is linear in the size of G, and γ is linear in the size of S.

We can conclude that ASPMODELCHECKING(RS) is **DP**-complete. □

Proof of Theorem 15. Directly from Lemmas 5 and 6. □

Lemma 7. ASPMODELCHECKING(AS) *is in* **DP**.

Proof of Lemma 7. Finding such a set Y can be done with the following algorithm:

1. Is there a set of facts Y, $atom(Y) \subseteq atom(P \cup Q)$ such that $X \in AS(P \cup Y \cup Q)$:
   (a) Guess a set of facts Y
   (b) Check that $atom(Y) \subseteq atom(P \cup Q)$
   (c) Check that $X \in AS(P \cup Y \cup Q)$
2. Compute $k_Y = |Y|$
3. Check that $Y \in \mathscr{A}(P,Q)$:
   (a) Guess a set of facts Y_0 and a set of atoms X_0
   (b) Check that $atom(Y_0) \subseteq atom(P \cup Q)$
   (c) Check that $|Y_0| < k_Y$
   (d) Check that $X_0 \in AS(P \cup Y_0 \cup Q)$

The algorithmic difficulty concentrates in items 1 and 3. Item 1 can be solved in polynomial time using a non deterministic Turing machine, and Y is a certificate, because it succinctly proves that $X \in AS(P \cup Y \cup Q)$. Thus, this subproblem is in **NP**. For item 3, (Y_0, X_0) is a succinct disqualification, as it proves that $Y \notin \mathscr{A}(P,Q)$. This algorithm runs in polynomial time on a non deterministic Turing machine. Thus, it is in **coNP**, and the whole problem is in **DP**. □

Lemma 8. ASPMODELCHECKING(AS) *is* **DP**-*complete.*

Proof of Lemma 8. The first step consists in building a transformation of the problem EXACTINDEPENDENTSET into ASPMODELCHECKING(AS). From a given graph $G = (V,E)$ with $V = \{x_1, \ldots, x_n\}$ we define a transformation $\tau(G) = (P,Q)$ as follows. We define $Q = Q_1 \cup Q_2 \cup Q_3$ with:

$$Q_1 = \{e(x,y). \mid (x,y) \in E\} \tag{5}$$

This set describes the edges of G.

$$Q_2 = \{\leftarrow is(x), is(y), e(x,y). \mid (x,y) \in E\} \tag{6}$$

Atoms $is(x)$ reflect the presence of a vertex x in an independent set. These rules eliminate the candidate vertices which are joined by an edge in E.

$$Q_3 = \{\leftarrow is'(x_1), \ldots, is'(x_n). \mid V = \{x_1, \ldots, x_n\}\} \tag{7}$$

Atoms $is'(x)$ reflect the absence of a vertex x in an independent set. This rule eliminates the empty set of vertices as a solution.

The program P contains the following rules:

$$P = \{is(x) \leftarrow not\ is'(x). \mid x \in V\} \tag{8}$$

The meaning of these rules is that a vertex x is candidate in an independent set if its absence is not stated.

Now we define a transformation γ of a set of vertices $V' \subseteq V$ into a set of atoms $\gamma(V')$ as follows:

$$\begin{aligned} X_1 &= \{is(x) \mid x \in V'\}, \\ X_2 &= \{is'(x) \mid x \in V \setminus V'\}, \\ X_3 &= \{e(x,y) \mid (x,y) \in E\}, \\ \gamma(V') &= X_1 \cup X_2 \cup X_3. \end{aligned}$$

As for the proof of Lemma 6, X_1 represents the vertices of G contained in V', while X_2 represents the vertices of G not contained in V', and $X_1 \cap \{is(x) \mid is'(x) \in X_2\} = \emptyset$.

Then, given a graph $G = (V,E)$ and its transformation $\tau(G) = (P,Q)$, we show that G contains a maximum independent set S of size k if and only if S corresponds to a set of atoms $X = \gamma(S)$ such that there exists a set of facts $Y \in \mathscr{A}(P,Q)$ such that $X \in AS(P \cup Y \cup Q)$.

We show first that if G contains a maximum independent set S of size k then S corresponds to a set of atoms $X = \gamma(S)$ such that there exists a set of facts $Y \in \mathscr{A}(P,Q)$ such that $X \in AS(P \cup Y \cup Q)$.

Suppose that G contains a maximum independent set of size k. Let us denote this set by S. Consider the transformations $\tau(G) = (P,Q)$ and $\gamma(S) = X$, and consider the algorithm given in the proof of Lemma 7. Suppose, without loss of generality, that the set of facts Y guessed in step 1(a) is

such that $Y = \{is'(x). \mid is'(x) \in X_2\}$. By construction we have $atom(Y) \subseteq atom(P \cup Q)$, so step 1(b) is verified. In order to check that $X \in AS(P \cup Y \cup Q)$ (step 1(c)), we first compute $(P \cup Y \cup Q)^X$. This set contains :

- All rules in Q, because they have an empty negative body.
- $\{head(r). \mid r \in P, is'(x) \notin X_2\}$
- All rules in Y, because they have an empty negative body.

Thus, $Cn((P \cup Y \cup Q)^X)$ contains :

- all atoms $e(x,y)$ such that $(x,y) \in E$, that is X_3 ;
- all atoms $is'(x)$ such that $is'(x) \in X_2$;
- all atoms $is(x)$ such that $is'(x) \notin X_2$, that is X_1.

Moreover, by construction of X_1, X_2 and X_3, none of the constraints in Q_2 are verified, because by construction all atoms $is(x) \in X_1$ correspond to the independent set S, thus, given any $(is(x), is(y)) \in X_1^2$, there is no corresponding $e(x,y) \in X_3$.

Finally, the constraint in Q_3 is not satisfied as soon as $Y \neq \emptyset$. Thus $Cn((P \cup Y \cup Q)^X) = X_1 \cup X_2 \cup X_3 = X$. This proves that $X \in AS(P \cup Y \cup Q)$.

The second step of the algorithm computes the size k_Y of the set Y. After this step, we can conclude that S is an independent set of G, and so G have an independent set of size at least $k = |V| - k_Y$.

Then, step 3 looks for a pair (Y_0, X_0), Y_0 being a set of facts, $atom(Y_0) \subseteq atom(P \cup Q)$, and X_0 being a set of atoms such that $|Y_0| < k_Y$. We consider two cases: either we find such a pair, either we do not find it :

- Suppose that we find such a pair (Y_0, X_0). This means that Y is not minimal with respect to cardinality, and consequently the number of atoms $is(x) \in X_0$ will be larger than the number of atoms $is(x) \in X$. If $X_0 \in AS(P \cup Y_0 \cup Q)$, then X_0 corresponds to an independent set S_0 such that $|S_0| > k$, because none of the constraints in Q_2 are satisfied, that is, $\forall (is(x), is(y)) \in X_0^2$, there is no corresponding $e(x,y) \in X_0$. Moreover, as $|Y_0| < k_Y$, we have $|\{is(x) \mid is(x) \in X_0\}| > k$, which contradicts the hypothesis.
- If we do not find such a pair, this means that there is no independent set which is larger than $k = |V| - k_Y$, and then $Y \in \mathscr{A}(P,Q)$.

We can conclude that if S is a maximal independent set of size k, then there exists a set of facts $Y \in \mathscr{A}(P,Q)$, with $\tau(G) = (P,Q)$, such that $\gamma(S) \in AS(P \cup Y \cup Q)$.

Now we show that if $S \subseteq V$ and $X = \gamma(S)$ such that there exists a set of facts $Y \in \mathscr{A}(P,Q)$ such that $X \in AS(P \cup Y \cup Q)$, then S is a maximum independent set of G with size k.

Let $\tau(G) = (P,Q)$. We suppose that there is a set $S \subseteq V$ such that there exists $Y \in \mathscr{A}(P,Q)$ such that $\gamma(S) \in AS(P \cup Y \cup Q)$. We prove that S is a maximum independent set of size $k = |V| - |Y|$.

Let us compute $(P \cup Y \cup Q)^{\gamma(S)}$. This program contains the following rules :

- All rules of Q, since they do not have a negative body,

- all rules $(is(x).)$ of P such that $is'(x) \notin \gamma(S)$, that is, such that $x \notin S$,
- all rules of Y, whose content is $Y = \{is'(x). \mid is'(x) \in \gamma(S)\}$, because $\gamma(S) \in AS(P \cup Y \cup Q)$.

As $\gamma(S) \in AS(P \cup Y \cup Q)$, $\forall (is(x), is(y)) \in \gamma(S)$ there is no fact $(e(x,y).)$ in Q_1, because if that was the case, there would exist a constraint $(\leftarrow is(x), is(y), e(x,y).)$ in Q_2, prohibiting $\gamma(S)$ from being an answer set.

Additionally, the constraint in Q_3 is never satisfied because Y is not empty. This proves that $\forall x, y \in S$, $(x,y) \notin E$. So S is an independent set of G of size $k = |V| - |Y|$.

Moreover, as Y is an added set, there is no set of facts Y_0 with $atom(Y) \subseteq atom(P \cup Q)$ and $|Y_0| < |Y|$ such that $AS(P \cup Y_0 \cup Q) \neq \emptyset$. This means that for any such set Y_0, there exist a pair of rules $(is(x) \leftarrow not\ is'(x).)$, $(is(y) \leftarrow not\ is'(y).)$, with $(is'(x).), (is'(y).) \notin Y_0$ such that there exists a rule $(e(x,y).) \in Q$. This means in turn that $\{x \mid (is'(x).) \notin Y\}$ is not an independent set of G, and thus there is no independent set of G with a size larger than $k = |V| - |Y|$.

Consequently S is a cardinality-maximal independent set of G of size $k = |V| - |Y|$.

Finally, note that the transformation τ is linear in the size of G, and γ is linear in the size of S.

From what precedes we can conclude that ASPMODELCHECKING(AS) is **DP**-complete. □

Proof of Theorem 16. Directly from Lemmas 7 and 8. □

Lemma 9. ASPMODELCHECKING(MS) *is in* **DP**.

Proof of Lemma 9. Finding such a pair (R,Y) can be performed by the following algorithm:

1. Is there a set of rules $R \subseteq P$ and a set of facts Y such that $atom(Y) \subseteq atom(P \cup Q)$, such that $X \in AS((P \setminus R) \cup Y \cup Q)$:
   (a) Guess a set of facts Y and a set of rules R
   (b) Check that $R \subseteq P$
   (c) Check that $atom(Y) \subseteq atom(P \cup Q)$
   (d) Check that $X \in AS((P \setminus R) \cup Y \cup Q)$
2. Compute $k_{RY} = |R \cup Y|$
3. Check that $(R,Y) \in \mathscr{M}(P,Q)$:
   (a) Guess a set of facts Y_0, a set of rules R_0 and a set of atoms X_0
   (b) Check that $R_0 \subseteq P$
   (c) Check that $atom(Y_0) \subseteq atom(P \cup Q)$
   (d) Check that $|R_0 \cup Y_0| < k_{RY}$
   (e) Check that $X_0 \in AS((P \setminus R_0) \cup Y_0 \cup Q)$

The algorithmic difficulty concentrates in items 1 and 3. Item 1 can be solved in polynomial time using a non deterministic Turing Machine, and (R,Y) is a certificate, because it succinctly proves that $X \in AS((P \setminus R) \cup Y \cup Q)$. Thus this subproblem is **NP**. For item 3, (R_0, Y_0, X_0) is a succinct disqualification, as it proves that $(R,Y) \notin \mathscr{M}(P,Q)$. This algorithm runs in polynomial time on a non deterministic Turing machine. Thus, it is in **coNP**, and the whole problem is in **DP**. □

Lemma 10. ASPMODELCHECKING(MS) *is* **DP**-*complete.*

Proof of Lemma 10. Let us define a transformation of the problem EXACTINDEPENDENTSET into ASPMODELCHECKING(MS). Let $G = (V,E)$ be a graph with $V = \{x_1,\ldots,x_n\}$. We define the transformation $\tau(G) = (P,Q)$ as follows. Let $Q = Q_1 \cup Q_2 \cup Q_3 \cup Q_4 \cup Q_5 \cup Q_6$ with :

$$Q_1 = \{e(x,y). \mid (x,y) \in E\} \tag{9}$$

Q_1 describes the edges of G.

$$Q_2 = \{\leftarrow is_1(x), is_1(y), e(x,y)., \leftarrow is_2(x), is_2(y), e(x,y). \mid (x,y) \in E\} \tag{10}$$

the intuitive meaning of $is_1(x)$ and $is_2(x)$ is as follows. $is_1(x)$ represents the presence of a vertex x in an independent set built by removal. $is_2(x)$ represent the presence of a vertex x in an independent set built by addition. The rules in Q_2 state that two vertices cannot be in an independent set if they are linked by an edge.

$$Q_3 = \{\leftarrow not\ is_1(x_1), \ldots, not\ is_1(x_n), not\ is_2(x_1), \ldots, not\ is_2(x_n).\} \tag{11}$$

Q_3 states that the empty set is not an independent set.

$$Q_4 = \{is_2(x) \leftarrow not\ is'_2(x). \mid x \in V\} \tag{12}$$

Q_4 states that if a vertex is not discarded ($is'_2(x)$) it must be in an independent set.

$$Q_5 = \{is(x) \leftarrow is_1(x)., is(x) \leftarrow is_2(x). \mid x \in V\} \tag{13}$$

Each atom $is(x)$ represents the situation where x is in an independent set generated by removal ($is_1(x)$) or by addition ($is_2(x)$).

$$Q_6 = \{\leftarrow is_1(x), not\ is_2(x)., \leftarrow is_2(x), not\ is_1(x). \mid x \in V\} \tag{14}$$

Q_6 eliminates independent sets which are not generated both by addition and by removal.

The program P contains the following rules :

$$P = \{is_1(x). \mid x \in V\} \tag{15}$$

These rules state that V is an independent set.

Now we define a transformation γ of a set of vertices $V' \subseteq V$ into a set of atoms $\gamma(V')$ as follows:

$$\begin{aligned} X_1 &= \{is_1(x) \mid x \in V'\} \\ X_2 &= \{is_2(x) \mid x \in V'\} \\ X_3 &= \{is(x) \mid x \in V'\} \\ X_4 &= \{is'2(x) \mid x \notin V'\} \\ X_5 &= \{e(x,y) \mid (x,y) \in E\} \\ \gamma(V') &= X_1 \cup X_2 \cup X_3 \cup X_4 \cup X_5 \cup X_6 \end{aligned}$$

X_1, X_2 and X_3 represent together the vertices of G contained in V', while X_4 represent the vertices not contained in V'. Note that $X_2 \cap X_4 = \emptyset$.

Then, given a graph $G = (V,E)$ and its transformation $\tau(G) = (P,Q)$ we show that G contains a maximum independent set S of size k if and only if the set of atoms $X = \gamma(S)$ is such that there exist a set of facts Y and a set of rules R, $(Y,R) \in \mathscr{M}(P,Q)$, such that $X \in AS((P \setminus R) \cup Y \cup Q)$.

We first show that if G contains a maximum independent set S of size k then the set of atoms $X = \gamma(S)$ is such that there exist a set of facts Y and a set of rules R, $(Y,R) \in \mathscr{M}(P,Q)$, such that $X \in AS((P \setminus R) \cup Y \cup Q)$.

Suppose that G contains a maximum independent set S of size k. Let $\tau(G) = (P,Q)$ and $\gamma(S) = X$, and consider the algorithm given in the proof of Lemma 9. Suppose, without loss of generality, that the set of facts Y and the set of rules R guessed in step 1(a) are such that $Y = \{(is'_2(x).) \mid is'_2(x) \in X_4$, and $R = \{r \mid r \in P, head(r) \notin X_1\}$. We have : $R \subseteq P$ by construction, thus step 1(b) is verified, and $atom(Y) \subseteq atom(P \cup Q)$, thus step 1(c) is verified. In order to check that $X \in AS((P \setminus R) \cup Y \cup Q)$ (step 1(d)), we compute $((P \setminus R) \cup Y \cup Q)^X$. This set contains :

- All rules in Q_1, Q_2 and Q_5, because they have an empty negative body.
- no rule of Q_3, because $S \neq \emptyset$, so $X_1 \neq \emptyset$ and $X_2 \neq \emptyset$.
- $\{head(r). \mid r \in Q_4, is'_2(x) \notin X_4\}$, that is, facts $is_2(x)$ such that $x \in S$.
- no rule in Q_6, because $\forall x \in S$, $is_1(x) \in X$ and $is_2(x) \in X$.
- all rules in $P \setminus R$, that is, $\{(is_1(x).) \mid x \in S\}$.
- all rules in Y.

Thus $Cn(((P \setminus R) \cup Y \cup Q)^X)$ contains :

- all atoms $e(x,y)$ such that $(x,y) \in E$, that is, X_5.
- all atoms $is_2(x)$ such that $is'_2(x) \notin X_4$ (that is, such that $x \in S$), that is, X_2.
- all atoms $is_1(x)$ such that $x \in S$, that is, X_1.
- all atoms $is(x)$ such that $x \in S$, because of rules in Q_5, that is, X_3,
- all atoms $is'_2(x)$ such that $(is'_2(x).) \in Y$, that is, such that $x \notin S$, that is, X_4.

Moreover, by construction of X_1, X_2 and X_5, none of the constraints in Q_2 are satisfied, because by construction all atoms $is_1(x) \in X_1$ and $is_2(x) \in X_2$ correspond to the independent set S, so, given any $(is_1(x), is_1(y)) \in X_1^2$ (resp. $(is_2(x), is_2(y)) \in X_2^2$), there is no corresponding $e(x,y) \in X_5$.

Thus $Cn(((P \setminus R) \cup Y \cup Q)^X) = X_1 \cup X_2 \cup X_3 \cup X_4 \cup X_5 = X$. This proves that $X \in AS((P \setminus R) \cup Y \cup Q)$.

The second step of the algorithm computes $k_{RY} = |R \cup Y|$. By construction, $|R|$ is the number of vertices not in S. The same applies for $|Y|$. After this step, we can conclude that S is an independent set of G, and thus G have an independent set of size at least $k = |P| - (k_{RY}/2)$.

Then, the third step looks for a triplet (Y_0, R_0, X_0), R_0 being a subset of rules of P, Y_0 being a set of facts such that $atom(Y_0) \subseteq atom(P \cup Q)$, and $|R_0 \cup Y_0| < k_{RY}$. X_0 is a set of atoms such that $X_0 \in AS((P \setminus R_0) \cup Y_0 \cup Q)$. We consider two cases: either we find such a triplet, either we do not find it.

- Suppose that we find such a triplet (Y_0, R_0, X_0). This means that $|R \cup Y|$ is not minimal, and consequently the number of atoms $is(x) \in X_0$ will be larger than the number of atoms $is(x) \in X$. If $X_O \in AS((P \setminus R_0) \cup Y_0 \cup Q)$, then X_0 corresponds to an independent set S_0 such that $|S_0| > |P| - (k_{RY}/2)$, because none of the constraints in Q_2 are satisfied, that is, $\forall (is_1(x), is_1(y)) \in X_0^2$, $\forall (is_2(x), is_2(y)) \in X_0^2$, there is no corresponding $e(x,y) \in X_0$. Moreover, as $|R_0 \cup Y_0| < k_{RY}$, we have $|\{is(x) \mid is(x) \in X_0\}| > k$, which contradicts the hypothesis.

- If we do not find such a triplet, this means that there is no independent set which is larger than $k = |P| - (k_{RY}/2)$, and then $(Y,R) \in \mathscr{M}(P,Q)$.

We can conclude that if S is a maximal independent set of size k, then there exist $(Y,R) \in \mathscr{M}(P,Q)$, with $\tau(G) = (P,Q)$, such that $\gamma(S) \in AS((P \setminus R) \cup Y \cup Q)$.

We now show that G contains a maximum independent set S of size k only if the set of atoms $X = \gamma(S)$ is such that there exist a set of facts Y and a set of rules R, $(Y,R) \in \mathscr{M}(P,Q)$, such that $X \in AS((P \setminus R) \cup Y \cup Q)$.

Let $\tau(G) = (P,Q)$. We suppose that there is a set $S \subseteq V$ such that there exists $(Y,R) \in \mathscr{M}(P,Q)$ such that $\gamma(S) \in AS((P \setminus R) \cup Y \cup Q)$. We prove that S is a maximal independent set of size $k = |V| - (k_{RY}/2)$.

Let us compute $((P \setminus R) \cup Y \cup Q)^{\gamma(S)}$. This program contains the following rules :

- All rules in Q_1, Q_2, Q_5 since they do not have any negative body ;

- no rule from Q_3, as there exist at least one $is_1(x) \in \gamma(S)$;

- $\{(head(r).) \mid r \in Q_4, is_2'(x) \in body^-(r), is_2'(x) \notin \gamma(S)\}$;

- $\{\leftarrow body^+(r) \mid r \in Q_6, is_2(x) \in body^-(r), is_2(x) \notin \gamma(S)\}$;

- $\{\leftarrow body^+(r) \mid r \in Q_6, is_1(x) \in body^-(r), is_1(x) \notin \gamma(S)\}$;

- All rules in $P \setminus R$. Note that, as $\gamma(S) \in AS((P \setminus R) \cup Y \cup Q)$, these rules are such that $is_1(x) \in \gamma(S)$, and thus $x \in S$.

- all rules in Y, whose content is $Y = \{(is_2'(x).) \mid is_2'(x) \in \gamma(S)\}$, because $\gamma(S) \in AS((P \setminus R) \cup Y \cup Q)$.

As $\gamma(S) \in AS((P \setminus R) \cup Y \cup Q)$ by hypothesis, $\forall (is_1(x), is_1(y)) \in (P \setminus R)^2$, there is no fact $(e(x,y).) \in Q_1$ because if it was the case, there would exist a constraint $(\leftarrow is_1(x), is_1(y), e(x,y).) \in Q_2$ which would prohibit $\gamma(S)$ to be an answer set of $(P \setminus R) \cup Y \cup Q$. Similarly, $\forall (is_2(x), is_2(y)) \in \gamma(S)^2$, there is no fact $(e(x,y).) \in Q_1$ because if it was the case, there would exist a constraint $(\leftarrow is_2(x), is_2(y), e(x,y).) \in Q_2$ which would prohibit $\gamma(S)$ to be an answer set of $(P \setminus R) \cup Y \cup Q$. This proves that for all $(x,y) \in S^2$, $(x,y) \notin E$. Thus S is an independent set of size $k = |V| - |Y| = |P \setminus R|$.

Moreover, as $(Y,R) \in \mathscr{M}(P,Q)$, there is no (Y_0, R_0), with $R_0 \subseteq P$, $atom(Y_0) \subseteq atom(P \cup Q)$, $|R_0 \cup Y_0| < |R \cup Y|$ such that $AS((P \setminus R_0) \cup Y_0 \cup Q) \neq \emptyset$. This means that:

- for any such set Y_0 there exists a pair of rules $(is_2(x) \leftarrow not\ is_2'(x)., is_2(y) \leftarrow not\ is_2'(y))$. with $(is_2'(x).), (is_2'(y).) \notin Y_0$ such that there exist a rule $(e(x,y).) \in Q$, which means in turn that $\{x \mid is_2'(x) \in Y_0\}$ is not an independent set of G, or,

- for any such set R_0, there exists a pair of rules $(is_1(x).,is_1(y).)$ in $P \setminus R_0$ such that there is a rule $(e(x,y).) \in Q$, which means in turn that $\{x \mid (is_1(x).) \in P \setminus R_0\}$ is not an independent set of G.

Thus S is a cardinality-maximal independent set of G of size $k = |V| - |Y| = |P \setminus R|$.

Finally, note that the transformation τ is linear in the size of G, and γ is linear in the size of S. This allows us to conclude that ASPMODELCHECKING(MS) is **DP**-complete. □

Proof of Theorem 17. Directly from Lemmas 9 and 10. ASPMODELCHECKING(MS) is **DP**-complete. □

References

Alchourrón, C., & Makinson, D. (1985). On the logic of theory change : Safe contraction. *Studia Logica*, *44*(4), 14–37.

Alferes, J., Leite, J., Pereira, L., Przymusinska, H., & Przymusinski, T. (2000). Dynamic updates of non-monotonic knowledge bases. *Journal of Logic Programming*, *45*(1-3), 43–70.

Baral, C. (2003). *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.

Bauters, K., Schockaert, S., Cock, M. D., & Vermeir, D. (2015). Characterizing and extending answer set semantics using possibility theory. *Theory and Practice of Logic Programming*, *15*(1), 79–116.

Benferhat, S., Bennaim, J., Papini, O., & Würbel, E. (2010). Answer set programming encoding of prioritized removed sets revision: Application to GIS. *Applied Intelligence*, *32*, 60–87.

Benferhat, S., Cayrol, C., Dubois, D., Lang, J., & Prade, H. (1993). Inconsistency management and prioritized syntax-based entailment. In *Proceedings of the* 13^{th} *International Joint Conference on Artificial Intelligence (IJCAI'93)*, pp. 640–645.

Benferhat, S., Dubois, D., & Prade, H. (1995). How to infer from inconsisent beliefs without revising?. In *Proceedings of the* 14^{th} *International Joint Conference on Artificial Intelligence (IJCAI'95)*, pp. 1449–1457.

Binnewies, S., Zhuang, Z., & Wang, K. (2015). Partial meet revision and contraction in logic programs. In *Proceedings of the* 29^{th} *AAAI Conference on Artificial Intelligence (AAAI'15)*, pp. 1439–1445.

Delgrande, J. P., Peppas, P., & Woltran, S. (2013). AGM-style belief revision of logic programs under answer set semantics. In *Proceedings of the* 12^{th} *Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'13)*, pp. 264–276.

Delgrande, J. P., Schaub, T., Tompits, H., & Woltran, S. (2008). Belief Revision of Logic Programs under Answer Set Semantics. In *Proceedings of the* 11^{th} *International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*, pp. 411–421.

Delgrande, J. P., Schaub, T., Tompits, H., & Woltran, S. (2009). Merging logic programs under answer set semantics. In *Proceedings of the* 25^{th} *International Conference on Logic Programming (ICLP'09)*, pp. 160–174.

Delgrande, J. P., Schaub, T., Tompits, H., & Woltran, S. (2013). A model-theoretic approach to belief change in answer set programming. *ACM Transaction on Computational Logic*, *14*(2), 1–46.

Eiter, T., Fink, M., Sabbatini, G., & Tompits, H. (2002). On properties of update sequences based on causal rejection. *Theory and Practice of Logic Programming*, *2*(6), 711–767.

Garcia, L., Lefèvre, C., Papini, O., Stéphan, I., & Würbel, E. (2017). A semantic characterization for ASP base revision. In *Proceedings of the* 11^{th} *International Conference on Scalable Uncertainty Management (SUM'17)*, pp. 334–347.

Garcia, L., Lefèvre, C., Papini, O., Stéphan, I., & Würbel, E. (2018). Possibilistic asp base revision by certain input. In *Proceedings of the* 27^{th} *International Joint Conference on Artificial Intelligence (IJCAI'18)*, pp. 1824–1830.

Gebser, M., Kaufmann, B., & Schaub, T. (2012). Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence*, *187*, 52–89.

Gelfond, M., & Lifschitz, V. (1988). The stable model semantics for logic programming. In *Proceedings of the* 5^{th} *International Conference on Logic Programming (ICLP'88)*, pp. 1070–1080.

Hansson, S. O. (1997). Semi-revision. *Journal of Applied Non-classical Logics*, *7*, 151–175.

Hansson, S. O. (1999). A text of belief dynamics theory change and database updating. *Applied Logic Series*, *11*.

Hansson, S. O., & Wassermann, R. (2002). Local change. *Studia Logica*, *70*(1), 49–76.

Hué, J., Papini, O., & Würbel, E. (2008). Removed Sets Fusion: Performing Off the Shelf. In *Proceedings of the* 18^{th} *European Conference on Artificial Intelligence (ECAI'08)*, pp. 94–98.

Hué, J., Papini, O., & Würbel, E. (2009). Merging belief bases represented by logic programs. In *Proceedings of the* 10^{th} *European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'09)*, pp. 371–382.

Hué, J., Papini, O., & Würbel, E. (2013). Extending belief base change to logic programs with ASP. In E. L. Ferme, D. M. Gabbay, G. R. S. (Ed.), *Trends in Belief Revision and Argumentation Dynamics*, No. 48 in Studies in Logic. College Publications.

Katsuno, H., & Mendelzon, A. (1991). Propositional knowledge base revision and minimal change. *Artificial Intelligence*, *52*(3), 263–294.

Krümpelmann, P., & Kern-Isberner, G. (2012). Belief Base Change Operations for Answer Set Programming. In *Proceedings of the* 13^{th} *European Conference on Logics in Artificial Intelligence (JELIA'12)*, pp. 294–306.

Lehmann, D. (1995). Belief revision, revised. In *Proceedings of the* 14^{th} *International Joint Conference on Artificial Intelligence (IJCAI'95)*, pp. 1534–1540.

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., & Scarcello, F. (2006). The DLV system for knowledge representation and reasoning. *ACM Transaction on Computational Logic*, *7*(3), 499–562.

Nicolas, P., Garcia, L., Stéphan, I., & Lefèvre, C. (2006). Possibilistic uncertainty handling for answer set programming. *Annals of Mathematics and Artificial Intelligence*, *47*(1–2), 139–181.

Niemelä, I. (1999). Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, *25*(3-4), 241–273.

Papadimitriou, C. H. (1994). *Computational Complexity*, chap. 17, p. 412. Addison-Wesley.

Papini, O. (1992). A Complete Revision Function in Propositional Calculus. In *Proceedings of the* 10^{th} *European Conference on Artificial Intelligence (ECAI'92)*, pp. 339–343.

Reiter, R. (1980). A logic for default reasoning. *Artificial Intelligence*, *13*(1-2), 81–132.

Sakama, C., & Inoue, K. (1999). Updating Extended Logic Programs through Abduction. In *Proceedings of the* 5^{th} *Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'99)*, pp. 147–161.

Schaub, T., & Woltran, S. (2018). Answer set programming unleashed!. *KI*, *32*(2-3), 105–108.

Schwind, N., & Inoue, K. (2016). Characterization of logic program revision as an extension of propositional revision. *Theory and Practice of Logic Programming*, *16*(1), 111–138.

Slota, M., & Leite, J. (2010). On Semantic Update Operators for Answer-Set Programs. In *Proceedings of the* 19^{th} *European Conference on Artificial Intelligence (ECAI'10)*, pp. 957–962.

Slota, M., & Leite, J. (2014). The rise and fall of semantic rule updates based on se-models. *Theory and Practice of Logic Programming*, *14*(6), 869–907.

Turner, H. (2003). Strong equivalence made easy: nested expressions and weight constraints. *Theory and Practice of Logic Programming*, *3*, 609–622.

Würbel, E., Jeansoulin, R., & Papini, O. (2000). Revision : An application in the framework of GIS. In *Proceedings of the* 7^{th} *International Conference on Principles of Knowledge Representation and Reasoning (KR'00)*, pp. 505–518.

Zhang, Y., & Foo, N. Y. (1998). Updating Logic Programs. In *Proceedings of the* 13^{th} *European Conference on Artificial Intelligence (ECAI'98)*, pp. 403–407.

Zhuang, Z., Delgrande, J., Nayak, A., & Sattar, A. (2016a). A New Approach for Revising Logic Programs. In *Proceedings of the* 16^{th} *International Workshop on Non-Monotonic Reasoning (NMR'16)*, pp. 171–176.

Zhuang, Z., Delgrande, J., Nayak, A., & Sattar, A. (2016b). Reconsidering AGM-style belief revision in the context of logic programs. In *Proceedings of the* 22^{th} *European Conference on Artificial Intelligence (ECAI'16)*, pp. 671–679.