Appl Intell
DOI 10.1007/s10489-008-0135-x

An answer set programming encoding of Prioritized Removed Sets Revision: application to GIS

Salem Benferhat · Jonathan Ben-Naim · Odile Papini · Eric Würbel

Abstract Geographical information systems are ones of the most important application areas of belief revision. Recently, Würbel and colleagues (Proceedings of the seventh international conference about principles of knowledge representation and reasoning, KR2000, pp. 505–516, 2000) have applied the so-called "removed sets revision" (RSR) to the problem of assessment of water heights in a flooded valley. The application was partially satisfactory since only a small part of the valley has been handled. This paper goes one step further, and proposes an extension of (RSR) called "Prioritized Removed Sets Revision" (PRSR). We show that (PRSR) performed using answer set programming makes possible to solve a practical revision problem provided by a real application in the framework of geographical information system (GIS). We first show how PRSR can be encoded into a logic program with answer set semantics, we then present an adaptation of the smodels system devoted to efficiently compute the answer sets in order to perform PRSR. The experimental study shows that the answer set programming approach gives better results than previous implementations of RSR and in particular it allows to handle the whole valley. Lastly, some experimental studies comparing our encoding with implementations based on SAT-solvers are also provided.

Keywords Belief revision · Answer set programming · Knowledge representation

This paper is an extended and a revision of the conference paper: "An answer set programming encoding of prioritized removed sets revision: application to GIS" presented at the JELIA'2004 conference.

S. Benferhat
CRIL-CNRS, Université d'Artois, Rue Jean Souvraz, 62307 Lens Cedex, France
e-mail: benferhat@cril.univ-artois.fr

J. Ben-Naim
IRIT-CNRS, Université Paul Sabatier de Toulouse, 118 Route de Narbonne, 31062 Toulouse Cedex 2, France
e-mail: bennaim@irit.fr

O. Papini
LSIS-CNRS, ESIL, Département Informatique, Université de la Méditerranée, Campus de Luminy, Case 925, 13288 Marseille Cedex 09, France
e-mail: Odile.Papini@esil.univmed.fr

E. Würbel (✉)
LSIS-CNRS, Université du Sud Toulon–Var, BP 132, 83957 La Garde Cedex, France
e-mail: wurbel@univ-tln.fr

1 Introduction

In many applications, intelligent agents face incomplete, uncertain and inaccurate information, and often need a revision operation in order to manage their beliefs change in presence of a new item of information. The agent's epistemic state represents his reasoning process and belief revision consists in modifying his initial epistemic state in order to maintain consistency, while keeping new information and removing the least possible previous information.

During the last 20 years, the question of how to perform revision gave rise to numerous works according to the representation of epistemic states and different strategies have been proposed (e.g., [1, 14, 35]). Most of the revision approaches have been developed at the theoretical level, except few applications [37] and it turns out that in the general case the theoretical complexity of revision is high [11, 20].

Among the belief revision approaches the so-called "Removed Sets Revision", (also known as a lexicographic-based

approach or cardinality based approach) has been proposed in [3, 18, 19, 27] for revising a set of propositional formulas. This approach stems from removing a minimal number of formulas, called removed set, to restore consistency.

Recently, Würbel and colleagues [38] have applied the "Removed Sets revision" (RSR) to geographical information, more precisely to the problem of assessment of water heights in a flooded valley. As it will be precised below, the result was partially satisfactory since only a small part of the valley has been handled.

This paper goes one step further in the application of belief revision in the geographical information. It considers a prioritized form of Removed Sets Revision, called Prioritized Removed Sets Revision (PRSR). It shows how the encoding of PRSR using answer set programming allows us to solve a practical revision problem coming from a real application in the framework of geographical information system. In particular this paper focuses on the following three issues:

- The notion of priority is very important in the study of knowledge-based systems [13]. When priorities attached to pieces of knowledge are available, the task of coping with inconsistency is greatly simplified, since conflicts have a better chance to be solved. Gärdenfors [14] has proved that upon arrival of a new piece of propositional information, any revision process of a belief set which satisfies natural requirements, is implicitly based on a priority ordering. In this paper we generalize the Removed Sets Revision, to revise prioritized belief bases, called Prioritized Removed Sets Revision.
- In the last decade, answer set programming is considered as one of convenient tools to handle non-monotonic reasoning systems. Logic programs with answer sets semantics can be equivalently described in terms of reducing logic programs to default logic, autoepistemic logic or circumpscription. Moreover, several efficient systems have been developed [7, 12, 21, 25, 29]. We propose to formalize the Prioritized Removed Sets Revision in terms of answer set programming and to adapt the smodels system in order to compute preferred answer sets which correspond to prioritized removed sets.
- When dealing with GIS we face incomplete and uncertain information. Since the data come from different sources characterized by various data qualities, they may conflict and require belief revision operations. Moreover, geographic information systems are characterized by a huge amount of data. In [38, 39] three different implementations of Removed Sets Revision have been experimented and compared on application on geographic information system concerning the flooding problem. The result was that an adaptation of Reiter's algorithm for diagnosis [30] gave the best results. Moreover, the Removed Sets Revision has been translated into a SAT problem and an implementation has been performed using an efficient SAT-solver MiniSat [10]. However, these approaches were not able to handle the whole geographical area (composed of 120 compartments) and only a part of it composed of 20 compartments for the adaptation of Reiter's algorithm and composed of 40 compartments for the SAT translation has been considered.

  In this paper we apply our answer sets programming encoding of PSRS to the framework of Geographic Information Systems. An experimental study shows that our approach gives better results than the adaptation of Reiter's algorithm for diagnosis and than an implementation based on a SAT-solver. These good results hold even if no priority is introduced. The introduction of priorities allows to handle the whole area.

The rest of this paper is organized as follows. Section 2 fixes the notations and gives a refresher on revision and on answer set programming. Section 3 presents the Removed Sets Revision and the Prioritized Removed Sets Revision. Section 4 shows how Prioritized Removed Sets Revision is encoded into logic programming with answer sets semantics. Section 5 presents an adaptation of the smodels system for computing answer sets for performing Prioritized Removed Sets Revision. Section 6 details an experimental study which illustrates the approach on a real scale application, *the flooding problem*, provided by CEMAGREF. We show that the answer set programming implementation gives better results than the one obtained using an adaptation of Reiter's algorithm and than an implementation based on a SAT-solver. In Sect. 7, we compare our approach with the existing ones. Finally, Sect. 8 concludes the paper.

2 Background

2.1 Notations

In this paper we use propositional calculus, denoted by $\mathcal{L}_{\mathcal{PC}}$, as knowledge representation language with the usual connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$. The symbols $\top$ and $\bot$ denote the constants *TRUE* and *FALSE* respectively. Lower case letters $a, b, c, \ldots$, are used to denote propositional variables, lower case Greek letters $\phi, \psi, \ldots$, to denote formulas, upper case letters A, B, C, denote sets of formulas and also formulas,[1] and upper case Greek letters $\Psi, \Phi, \ldots$, are used to denote epistemic states. We denote by $\mathcal{W}$ the set of interpretations and by $\text{Mod}(\psi)$ the set of models of ψ, that is $\text{Mod}(\psi) = \{\omega \in \mathcal{W}, \omega \models \psi\}$ where $\models$ denotes the inference relation used for drawing conclusions. Let ψ and ϕ be formulas and X be a set of formulas, $\psi \models \phi$ denotes that $\text{Mod}(\psi) \subseteq \text{Mod}(\phi)$ and $X \models \phi$ denotes that $\forall \psi \in X$,

[1] We maintain both notations for formulas used in literature in order to express some results in their original form.

$\text{Mod}(\psi) \subseteq \text{Mod}(\phi)$. The symbol $\equiv$ denotes logical equivalence, and $\psi \equiv \phi$ means that $\psi \models \phi$ and $\phi \models \psi$. A set of logical conclusions is such that, let X be a set of formulas, $Cn(X)$ denotes the set of consequences of X, $A \in Cn(X)$ iff $X \models A$. A theory, also called deductively closed set of formulas, is such that, let X be a set of formulas, X is a theory iff $Cn(X) = X$. Since propositional formulas are equivalent to their Conjunctive Normal Forms (CNF), in the following propositional formulas and sets of propositional formulas are considered as sets of clauses.

2.2 Revision

Belief revision is the study of the rational means an agent uses in order to modify his epistemic state in view of new information. In general, agent's epistemic state is a complex structure. It contains agent's current beliefs and also strategies to revise his beliefs. The epistemic state has to be changed in order to restore consistency, keeping the new item of information and removing the least possible amount of previous information. The first formalizations of revision arose from philosophical logic with the work of Gärdenfors 1978, where revision is interpreted as belief change. Alchourron, Gärdenfors and Makinson 1985 formulated postulates, called the AGM postulates [1], in order to characterize revision. These postulates stem from three principal ideas: (1) the consistency principle (a revision operation has to produce a consistent set of beliefs); (2) the principle of minimal change (a revision operation should change the smallest possible number of beliefs); (3) priority is given to the new item of information. These postulates focus on the logical structure of beliefs. They are based on the theory of consistency and do not take into account the justification of beliefs such as in TMS systems [9, 17].

2.2.1 AGM postulates

Alchourron, Gärdenfors and Makinson [1] proposed a formal framework in which revision is interpreted as belief change and an epistemic state is considered as a theory of $\mathcal{L}_{\mathcal{PC}}$ (a deductively closed set of formulas). Focusing on the logical structure of beliefs, they formulate eight postulates that a revised theory must verify. More formally.

Let T be a theory, and let A and B be formulas. $T \star A$ denotes the theory T revised by A. $T + A$ is the smallest deductively closed set of formulas containing both T and A. $T^{\perp}$ denotes the set of all formulas.

(G$\star$1) $T \star A$ is a theory.
(G$\star$2) $A \in T \star A$.
(G$\star$3) $T \star A \subseteq T + A$.
(G$\star$4) If $\neg A \notin T$ then $T \star A = T + A$.
(G$\star$5) $T \star A = T^{\perp}$ only if A is unsatisfiable.
(G$\star$6) If $A \equiv B$ then $T \star A = T \star B$.
(G$\star$7) $T \star (A \wedge B) \subseteq (T \star A) + B$.
(G$\star$8) If $\neg B \notin T \star A$ then $(T \star A) + B = T \star (A \wedge B)$.

The postulate (G$\star$1) expresses the fact that a theory revised by a formula is a theory. (G$\star$2) specifies that the formula A belongs to the revised theory. (G$\star$3) and (G$\star$4) give the result of the revision when A is consistent with T. (G$\star$5) is linked to the preservation and the restoration of consistency, (G$\star$6) specifies that the result of the revision must be independent from the syntactic form of the added formula. (G$\star$7) and (G$\star$8) point out that when A is consistent with T the change has to be minimal. Minimally revising T to include both A and B should reduce to an expansion of $T \star A$, as long as B does not contradict $T \star A$.

Katsuno and Mendelzon [16] proposed a reformulation of the AGM postulates when an epistemic state is represented by only one propositional formula ψ. They showed that a revision operation satisfying the AGM postulates is equivalent to a total pre-order on interpretations.

More formally, let $\mathcal{W}$ be the set of all interpretations and $\text{Mod}(\psi)$ be the set of models of ψ. A pre-order on $\mathcal{W}$, denoted $\leq_\psi$, is linked with ψ. The relation $<_\psi$ is defined from $\leq_\psi$ as usual: $\omega <_\psi \omega'$ iff $\omega \leq_\psi \omega'$ and $\omega' \not\leq_\psi \omega$. Katsuno and Mendelzon introduced the notion of faithful assignment. A function that maps an epistemic state ψ to a total pre-order $\leq_\psi$ is a *faithful* assignment if and only if it verifies the following conditions:

(1) if $\omega, \omega' \in \text{Mod}(\psi)$ then $\omega <_\psi \omega'$ does not hold;
(2) if $\omega \in \text{Mod}(\psi)$ and $\omega' \notin \text{Mod}(\psi)$ then $\omega <_\psi \omega'$ holds;
(3) $\psi \equiv \phi$ iff $\leq_\psi = \leq_\phi$.

A minimal interpretation may thus be defined as follows. Let $\mathcal{M} \subseteq \mathcal{W}$, ω is minimal in $\mathcal{M}$ according to $\leq_\psi$, if $\omega \in \mathcal{M}$ and there is no $\omega' \in \mathcal{M}$ such that $\omega' <_\psi \omega$. The set of minimal interpretations in $\mathcal{M}$ according to $\leq_\psi$ is denoted $\text{Min}(\mathcal{M}, \leq_\psi)$. Katsuno and Mendelzon proved that a revision operator $\circ$ satisfies postulates (G$\star$1)–(G$\star$8) iff there exists a faithful assignment which associates to each epistemic state ψ a total pre-order $\leq_\psi$ such that:

$$\text{Mod}(\psi \circ \mu) = \min(\text{Mod}(\mu), \leq_\psi).$$

2.3 Answer set programming

In the last decade, logic programs with answer sets semantics have been used to implement non-monotonic reasoning and Answer Set Programming (ASP) can be viewed as an effective knowledge representation tool. We briefly remind some definitions about Answer Set Programming.

2.3.1 Answer sets

We deal with normal logic programs which contain the symbol *not*, which stands for *negation as failure*. A *normal logic program* is a set of rules of the form $c \leftarrow$

$a_1, \ldots, a_n, not\, b_1, \ldots, not\, b_m$ where c, a_i $(1 \leq i \leq n)$, and b_j $(1 \leq j \leq m)$ are propositional atoms. A rule is a fact if $n = m = 0$, it is a basic rule if $m = 0$. For a rule r like above, we define $head(r) = c$ and $body(r) = \{a_1, \ldots, a_n, b_1, \ldots, b_m\}$. Furthermore, let $body^+(r) = \{a_1, \ldots, a_n\}$ denotes the set of positive body atoms and $body^-(r) = \{b_1, \ldots, b_m\}$ the set of negative body atoms. By extension, a *basic program* is a program containing only basic rules.

Let r be a rule, r^+ denotes the rule $head(r) \leftarrow body^+(r)$, obtained from r by deleting all negative body atoms in the body of r.

Definition 1 A set of atoms X is *closed under* a basic program P iff for any rule $r \in P$, $head(r) \in X$ whenever $body(r) \subseteq X$. The smallest set of atoms which is closed under a basic program P is denoted by $CN(P)$.

Definition 2 The *reduct* (or GL-transformation), P^X of a program P *relatively to* a set X of atoms is defined by $P^X = \{r^+ \mid r \in P \text{ and } body^-(r) \cap X = \emptyset\}$.

Definition 3 A set of atoms X is an *answer set* of a program P iff $CN(P^X) = X$.

An answer set of a program P intuitively represents a possible set of beliefs a rational agent may hold on the basis of the information expressed by rules of the program P. Efficient implementation systems have been proposed for answer set programming, smodels [25, 33], dlv [12], xsb [29], DeRes [7], noMoRe [21]. In the following we will use the smodels system to compute answer sets.

2.3.2 *smodels system*

We briefly describe the smodels system proposed by Niemela and Simons [25, 33] to compute answer sets of a logic program P, since we need to modify the answer sets computation function in order to compute preferred answer sets. We first need to introduce the following definitions. Let L be a set of literals. $Pos(L)$ (resp. $Neg(L)$) denotes the set of positive (resp. negative) literals of L.

Definition 4 Let L be a set of literals and A be a set of atoms. A agrees with L iff $Pos(L) \subseteq A$ and $Neg(L) \cap A = \emptyset$.

Definition 5 Let L be a set of literals and A be a set of atoms. L covers A iff $A \subseteq Atom(L)$.

Let L be a finite set of literals, the function $smodels(P, L)$ returns the set of answer sets of P which agree with L. To do so, properties of answer sets are used in order to construct a set of literals L' as big as possible such that an answer set agrees with L iff it agrees with L'. Moreover, $L \subseteq L' \subseteq Lit(P)$, where $Lit(P)$ denotes the set of literals occurring in P. Three cases hold:

1. if L' is inconsistent then there is no answer set that agrees with L', therefore there is no answer set which agrees with L;
2. if L' is consistent and covers $Atom(P)$ then $Pos(L')$ is an answer set. Since L' covers $Atom(P)$, $Pos(L')$ is the only answer set which agrees with L', therefore it is the only answer set which agrees with L;
3. if L' is consistent and does not cover $Atom(P)$ then using some heuristics a new atom $a \in Atom(P)$ such that $a \notin L'$ is selected and the algorithm constructs two new sets $L' \cup \{a\}$ and $L' \cup \{\neg a\}$.

The algorithm of the function $smodels(P, L)$ is described in Fig. 1.

In the original algorithm Simons [33] introduces three procedures. The first one, named $Expand(P, L)$, returns a set L' as big as possible such that an answer set agrees with L iff it agrees with L'. The second one is called

```
function SMODELS(P, L)          ▷ L and L' are sets of literals, a is an atom
  L' ← Expand(P, L)
  if L' is inconsistent then
    return ∅
  else
    if L' covers Atom(P) then
      return {Pos(L')}
    else
      a ← Heuristic(P, L')
      return smodels(P, L' ∪ {a}) ∪ smodels(P, L' ∪ {¬a})
    end if
  end if
end function
```

Fig. 1 The algorithm of the function $smodels(P, L)$

Lookahead(P, L) which tries to find a literal l such that an answer set agrees with L iff it agrees with $L \cup \{l\}$ and such that $L \cup \{l\}$ is consistent. If it succeeds it returns $L \cup \{l\}$, else it returns L. The third one is called *Heuristic*(P, L), and returns an atom $a \in Atom(P) \backslash L$ such that *Expand*$(P, L \cup \{a\})$ and *Expand*$(P, L \cup \{\neg a\})$ are as big as possible.

The meaning of the function *smodels*(P, L) is that, given a program P, the answers sets of P are the result of the call *smodels*$(P, \emptyset)$.

3 Removed Sets Revision

We now present the Removed Sets Revision (RSR) which is a cardinality based or lexicographic approach of revision.

Let Ψ be an epistemic state and $Bel(\Psi)$ its corresponding belief set which is a propositional formula. Since a propositional formula is equivalent to its conjunctive normal form, we denote by K the set of clauses such that $\text{Mod}(Bel(\Psi)) \equiv \text{Mod}(K)$. From now on, belief bases and formulas are considered like finite set of clauses. However, following [22], belief bases are not assumed to be deductively closed.

3.1 RSR syntactic approach

Most of the approaches dedicated to belief bases revision concentrated to the construction of maximal[2] consistent subbases while the intuition in our approach was to focus on minimal inconsistent subsets. More precisely, in the case where the added formula is inconsistent with the belief base, by the compacity theorem, there exists at least one minimal inconsistent subset. The strategy is to determine which minimal subsets of formulas from the initial belief base intersect with the inconsistent subsets. This leads to the notion of *removed sets* [27], that is the minimal number of clauses to remove from the initial belief base to restore consistency. More formally:

Definition 6 Let K and A be two consistent finite sets of clauses. R a subset of clauses of K, is a removed set of $K \cup A$ iff

(i) $(K \backslash R) \cup A$ is consistent;
(ii) $\forall R' \subseteq K$, if $(K \backslash R') \cup A$ is consistent then $|R| \leq |R'|$.[3]

Example 1 Let $K = \{\neg c, a \vee c, b \vee c\}$, and $A = \{\neg a, \neg b, d \vee e, \neg d \vee \neg e\}$. $K \cup A$ is inconsistent and there are 5 possible subsets of K to remove to restore consistency. Namely $R_1 = \{\neg c\}$, $R_2 = \{a \vee c, b \vee c\}$, $R_3 = \{\neg c, a \vee c\}$, $R_4 = \{\neg c, b \vee c\}$, $R_5 = \{\neg c, a \vee c, b \vee c\}$, but R_1 is the only removed set according to Definition 6.

[2] Maximality in terms of inclusion or cardinality.

[3] $|R|$ denotes the number of clauses of R.

Let us denote by $\mathcal{R}(K \cup A)$ the set of removed sets of $K \cup A$, the following proposition holds.

Proposition 1 *Let K and A be two consistent sets of clauses. If $K \cup A$ is inconsistent then $\mathcal{R}(K \cup A) \neq \emptyset$, otherwise $\mathcal{R}(K \cup A) = \{\emptyset\}$.*

The removed sets revision (RSR) is defined by a selection function, denoted by s, that selects a non-empty subset of $\mathcal{R}(K \cup A)$ provided $\mathcal{R}(K \cup A)$ is non empty, $\emptyset$ otherwise.

Definition 7 Removed Sets Revision (RSR) Let K and A be two consistent sets of clauses. The removed set revision is defined by:

$$K \circ_{RSR} A =_{def} \bigvee_{R \in s(\mathcal{R}(K \cup A))} Cn((K \backslash R) \cup A).$$

Example 2 Let $K = \{\neg c, a \vee c, b \vee c, \neg d, \neg e\}$, and $A = \{\neg a, \neg b, d \vee e, \neg d \vee \neg e\}$. $K \cup A$ is inconsistent and $\mathcal{R}(K \cup A) = \{\{\neg c, \neg d\}, \{\neg c, \neg e\}\}$. Therefore, depending on how the selection function s is defined, the result of the revision operation is

$$K \circ_{RSR} A = Cn(\{a \vee c, b \vee c, \neg d, \neg a, \neg b, d \vee e, \neg d \vee \neg e\})$$

or

$$K \circ_{RSR} A = Cn(\{a \vee c, b \vee c, \neg e, \neg a, \neg b, d \vee e, \neg d \vee \neg e\})$$

or

$$K \circ_{RSR} A = Cn(\{a \vee c, b \vee c, \neg d, \neg a, \neg b, d \vee e, \neg d \vee \neg e\}) \vee Cn(\{a \vee c, b \vee c, \neg e, \neg a, \neg b, d \vee e, \neg d \vee \neg e\}).$$

The Removed Sets Revision strategy first requires the computation of the removed sets, then amounts to select one or several of them according to the selection function s.

Proposition 2 *Let K and A be two consistent sets of clauses. When $s(\mathcal{R}(K \cup A)) = \mathcal{R}(K \cup A)$ the revision operator $\circ_{RSR}$ satisfies the postulates* (G$\star$1)–(G$\star$8).

The main scope of the paper focuses on the computation of the removed sets, not on the revision operation defined in terms of selection among the removed sets. So, in order to simplify the reading, and without loss of generality, we will consider from now on s to be such that $s(\mathcal{R}(K \cup A)) = \mathcal{R}(K \cup A)$.

Since most of the approaches dedicated to belief bases revision concentrated on the construction of maximal (*in terms of cardinality*) consistent subbases, we show that RSR can be defined in terms of maximal consistent subbases. The idea of selecting a subset of maximal consistent subsets using a cardinality criterion was used independently in diagnosis problems. In model-based diagnosis, the number of diagnosis (sets of faulty components, called also a hitting set in [30]) is very high in general. To select a subset of all possible diagnosis, De Kleer [18] proposes a probabilistic criterion where he assumes that each component has a very small probability to fail and that all components fail independently. De Kleer [18] shows that the selected diagnosis are those which contain a small number of failing components.

Let K be a consistent set of clauses, and A be a set of clauses. We write $MAXCONS_A(K)$ to denote the set of maximal consistent subsets of clauses of K consistent with A, called maximal consistent subsets of clauses of K relatively to A. We define $MAXCONS_A(K)$ defined as follows:

Definition 8 Let K and A be two consistent sets of clauses. K' is a maximal consistent subset of clauses of K relatively to A iff

- $K' \subseteq K$;
- $K' \cup A$ is consistent;
- $\forall K'' \subseteq K$, if $K'' \cup A$ is consistent then $|K''| \leq |K'|$.

The notion of removed set can be expressed in terms of maximal consistent subbases.

Proposition 3 *Let K and A be two consistent sets of clauses. Let R be a set of clauses,*

- *R is a removed set iff $(K \setminus R) \cup A \in MAXCONS_A(K) \cup A$;*
- *$K \circ_{RSR} A = \bigvee_{K' \in s(MAXCONS_A(K))} Cn(K' \cup A)$.*

3.2 RSR semantic characterization

Let ω be an interpretation, $\mathcal{NS}_K(\omega)$ denotes the set of clauses of K falsified by ω. A total pre-order on interpretations associated with K can be defined according to the number of falsified clauses of K as follows.

Definition 9 Let K be a finite set of clauses. $\forall \omega_i, \omega_j \in \mathcal{W}$,

$\omega_i \leq_K \omega_j$ iff $|\mathcal{NS}_K(\omega_i)| \leq |\mathcal{NS}_K(\omega_j)|$.

Proposition 4 *The function that assigns K the total pre-order $\leq_K$ is a faithful assignment.*

According to a semantic point of view, the Removed Sets Revision has to minimize the number of clauses falsified by the models of A.

Definition 10 Let K and A be two consistent sets of clauses.

$\text{Mod}(K \circ_{RSR_{sem}} A) = \min(\text{Mod}(A), \leq_K)$.

The equivalence between the semantic and the syntactic Removed Set Revision is given by the following proposition.

Proposition 5 *Let K and A be two consistent sets of clauses.*

$\text{Mod}(K \circ_{RSR} A) = \text{Mod}(K \circ_{RSR_{sem}} A)$.

3.3 Prioritized Removed Sets Revision

We now present the Prioritized Removed Set Revision (PRSR) which generalizes the Removed Set Revision presented in Sect. 3.1 to the case of prioritized belief bases. Let K be a prioritized finite set of clauses, where K is partitioned into n strata, i.e. $K = K_1 \cup \cdots \cup K_n$, such that clauses in K_i have the same level of priority and have higher priority than the ones in K_j where $j > i$. K_1 contains the clauses which have the highest priority in K, and K_n contains the ones which have the lowest priority in K [3], see also [18].

When K is prioritized in order to restore consistency the principle of minimal change stems from removing the minimum number of clauses from K_1, then the minimum number of clauses in K_2, and so on. We generalize the notion of removed set in order to perform Removed Sets Revision with prioritized sets of clauses. This generalization first requires the introduction of a preference relation between subsets of K.

Definition 11 Let K be a consistent and prioritized finite set of clauses. Let X and X' be two subsets of K. X is strictly preferred to X' iff (i) $\exists i,\ 1 \leq i \leq n,\ |X \cap K_i| < |X' \cap K_i|$; (ii) $\forall j,\ 1 \leq j < i,\ |X \cap K_j| = |X' \cap K_j|$.

Prioritized removed sets are now defined as follows:

Definition 12 Let K be a consistent and prioritized finite set of clauses and let A be a consistent finite set of clauses. R, a subset of clauses of K, is a prioritized removed set iff (i) $R \subseteq K$; (ii) $(K \setminus R) \cup A$ is consistent; (iii) $\forall R' \subseteq K$, if $(K \setminus R') \cup A$ is consistent then R' is not strictly preferred to R.

Let denote by $\mathcal{PR}(K \cup A)$ the set of prioritized removed sets of $K \cup A$, the prioritized removed sets revision (PRSR) is defined by a selection function denoted by s that selects a non empty subset of $\mathcal{PR}(K \cup A)$ provided $\mathcal{PR}(K \cup A)$ is not empty, $\emptyset$ otherwise.

Definition 13 Prioritized Removed Sets Revision (PRSR) Let K and A be two consistent sets of clauses. The prioritized removed set revision is defined by:

$$K \circ_{PRSR} A =_{def} \bigvee_{R \in s(\mathcal{PR}(K \cup A))} Cn((K \setminus R) \cup A).$$

The theoretical computational complexity of the decision problem: "is a clause C a consequence of $K \circ_{PRSR} A$?" has been studied in the literature. It is in Δ_2^p (see [6, 23] for more details). The proof is based on the following decision problem that we denote D: "given K possibly inconsistent, is there a consistent subset A of K and an integer i, such that $|A| = i$?" The decision problem D is *NP*-hard. Hence, the proof is based on providing an algorithm that needs n calls to the decision problem D, where n is the number of clauses in K.

Example 3 Let $K = \{\neg c, a \vee c, b \vee c, \neg d, \neg e\}$ such that $K_1 = \{\neg c\}$, $K_2 = \{a \vee c, b \vee c\}$, $K_3 = \{\neg d, \neg e\}$, and $A = \{\neg a, \neg b, d \vee e, \neg d \vee \neg e\}$. $K \cup A$ is inconsistent, and the possible subsets of K to remove in order to restore consistency are: $R_1 = \{\neg c, \neg d\}$, $R_2 = \{\neg c, \neg e\}$, $R_3 = \{a \vee c, b \vee c, \neg d\}$, $R_4 = \{a \vee c, b \vee c, \neg e\}$, $R_5 = \{\neg c, a \vee c, \neg d\}$, $R_6 = \{\neg c, a \vee c, \neg e\}$, $R_7 = \{\neg c, b \vee c, \neg d\}$, $R_8 = \{\neg c, b \vee c, \neg e\}$, $R_9 = \{\neg c, a \vee c, b \vee c, \neg d\}$, $R_{10} = \{\neg c, a \vee c, b \vee c, \neg e\}$. Among those sets, R_3 and R_4 are the only removed sets satisfying Definitions 11 and 12, so $\mathcal{PR}(K \cup A) = \{\{a \vee c, b \vee c, \neg d\}, \{a \vee c, b \vee c, \neg e\}\}$. The result of $K \circ_{PRSR} A$ will depend on the selection function s. Namely:

$$K \circ_{PRSR} A = Cn(\{\neg a, \neg b, \neg c, \neg d, d \vee e, \neg d \vee \neg e\})$$

or

$$K \circ_{PRSR} A = Cn(\{\neg a, \neg b, \neg c, \neg e, d \vee e, \neg d \vee \neg e\})$$

or

$$K \circ_{PRSR} A = Cn(\{\neg a, \neg b, \neg c, \neg d, d \vee e, \neg d \vee \neg e\}) \vee Cn(\{\neg a, \neg b, \neg c, \neg e, d \vee e, \neg d \vee \neg e\}).$$

Like RSR, the generalization PRSR can be defined in terms of maximal consistent subbases, and can be equivalently defined from a semantic point of view. Moreover, it satisfies all AGM postulates.

Relying on the same arguments as in Sect. 3.1, from now on we will omit the selection function s, and assume that $s(\mathcal{PR}(K \cup A)) = \mathcal{PR}(K \cup A)$.

4 Encoding PRSR in answer set programming

We now show how we construct a logic program, denoted by $P_{K \cup A}$, such that the preferred answer sets of $P_{K \cup A}$ correspond to the prioritized removed sets of $K \cup A$. We first construct a logic program in the same spirit of Niemelä in [24], and then define the notion of preferred answer set in order to perform PRSR.

4.1 Translation into a logic program

Our aim in this subsection is to construct a logic program $P_{K \cup A}$ such that the answer sets of $P_{K \cup A}$ correspond to subsets R of K such that $(K \cup A) \setminus R$ is consistent. For each clause c of K, we introduce two new atoms denoted by r_c and r'_c, and for each atom $a \in Atom(K \cup A)$ we introduce a new atom a'. By V we denote the set of atoms such that $V = V^+ \cup V^-$, with $V^+ = Atom(K \cup A) \cup \{r_c \mid c \in K\}$ and $V^- = \{a' \mid a \in Atom(K \cup A) \cup \{r'_c \mid c \in K\}$ where $Atom(K \cup A)$ denotes the set of atoms occurring in $K \cup A$. The construction of $P_{K \cup A}$ stems from the enumeration of interpretations of V and the progressive elimination of interpretations which are not models of $(K \cup A) \setminus R$ with $R = \{c \in K \mid r_c \text{ is satisfied}\}$. This construction requires 3 steps: the first step introduces rules such that the answer sets of $P_{K \cup A}$ correspond to the interpretations of the propositional variables occurring in V^+, the second step introduces rules that constraint the answer sets of $P_{K \cup A}$ to correspond to models of A, the third step introduces rules such that answer sets of $P_{K \cup A}$ correspond to models of $(K \cup A) \setminus R$. More precisely:

(i) The first step introduces rules in order to build a one to one correspondence between answer sets of $P_{K \cup A}$ and interpretations of V^+. For each atom $a \in V^+$ we introduce two rules: $a \leftarrow not\ a'$ and $a' \leftarrow not\ a$ where $a' \in V^-$ is the negative atom corresponding to a.

(ii) The second step rules out answer sets of $P_{K \cup A}$ which correspond to interpretations which are not models of A. For each clause $c \in A$ such that $c = \neg a_0 \vee \cdots \vee \neg a_n \vee a_{n+1} \vee \cdots \vee a_m$, the following rule is introduced: $false \leftarrow a_0, \cdots, a_n, a'_{n+1}, \cdots, a'_m$ and in order to rule out *false* from the models of A: $contradiction \leftarrow false$, $not\ contradiction$.

(iii) The third step excludes answer sets S which correspond to interpretations which are not models of $(K \cup A) \setminus C_i$ with $C_i = \{c \mid r_c \in S\}$. For each clause c of K such that $c = \neg b_0 \vee \cdots \vee \neg b_n \vee b_{n+1} \vee \cdots \vee b_m$, we introduce the following rule: $r_c \leftarrow b_0, \ldots, b_n, b'_{n+1}, \ldots, b'_m$.

The steps (i) and (ii) are very similar to the ones proposed by Niemelä, but the third one (iii) is new and is introduced for revision. Note that (ii) is similar to (iii), and more precisely (ii) deals with clauses of A, while (iii) deals with clauses of K. Intuitively, (ii) states that clauses of A should be accepted, while (iii) means that clauses of K can be removed.

Example 4 Let $K = \{\neg c, a \vee c, b \vee c\}$ and $A = \{\neg a, \neg b, d \vee e, \neg d \vee \neg e\}$. We have

$$V = \{a, b, c, d, e, a', b', c', d', e', r_{a\vee c}, r_{b\vee c}, r_{\neg c}, r'_{a\vee c}, r'_{b\vee c}, r'_{\neg c}\},$$

with $V^+ = \{a, b, c, d, e, r_{a\vee c}, r_{b\vee c}, r_{\neg c}\}$ and $V^- = \{a', b', c', d', e', r'_{a\vee c}, r'_{b\vee c}, r'_{\neg c}\}$. The logic program $P_{K\cup A}$ is the following:

(i)
$$\begin{cases} a \leftarrow not\ a' \quad a' \leftarrow not\ a \\ b \leftarrow not\ b' \quad b' \leftarrow not\ b \\ c \leftarrow not\ c' \quad c' \leftarrow not\ c \\ d \leftarrow not\ d' \quad d' \leftarrow not\ d \\ e \leftarrow not\ e' \quad e' \leftarrow not\ e \\ r_{\neg c} \leftarrow not\ r'_{\neg c} \quad r'_{\neg c} \leftarrow not\ r_{\neg c} \\ r_{a\vee c} \leftarrow not\ r'_{a\vee c} \quad r'_{a\vee c} \leftarrow not\ r_{a\vee c} \\ r_{b\vee c} \leftarrow not\ r'_{b\vee c} \quad r'_{b\vee c} \leftarrow not\ r_{b\vee c} \end{cases}$$

(ii)
$$\begin{cases} false \leftarrow a \quad false \leftarrow d, e \\ false \leftarrow b \quad false \leftarrow d, e \\ contradiction \leftarrow false, not\ contradiction \end{cases}$$

(iii) $\{r_{a\vee c} \leftarrow a', c' \quad r_{b\vee c} \leftarrow b', c' \quad r_{\neg c} \leftarrow c.$

We denote by R_K the set $R_K = \{r_c \mid c \in K\} \cup \{r'_c \mid c \in K\}$ and $R_K^+ = \{r_c \mid c \in K\}$ (resp. $R_K^- = \{r'_c \mid c \in K\}$) denotes the positive (resp. negative) atoms of R_K. We denote by CL the mapping from R_K^+ to K which associates to each atom of R_K^+ the corresponding clause in K. More formally, $\forall r_c \in R_K^+,\ CL(r_c) = c$. Let S be a set of atoms, we define I_S such that $I_S = \{a \mid a \in S\} \cup \{\neg a \mid a' \in S\}$ and the following result holds.

Proposition 6 *Let K be a consistent finite set of clauses and let A be a finite consistent set of clauses. Let $S \subseteq V$ be a set of atoms. S is an answer set of $P_{K\cup A}$ iff I_S is an interpretation of V^+ which satisfies $(K \setminus CL(S \cap R_K^+)) \cup A$.*

Example 5 Coming back to the previous example, $K = \{\neg c, a \vee c, b \vee c\}$ and $A = \{\neg a, \neg b, d \vee e, \neg d \vee \neg e\}$. Let $S = \{d, e', a', b', c, r'_{a\vee c}, r'_{b\vee c}, r_{\neg c}\}$, it can be checked that S is an answer sets of $P_{K\cup A}$. Since $P_{K\cup A}$ is a normal logic program, I_S is constructed by replacing the negative atoms of V^- by negated atoms of V^+. Therefore $I_S = \{d, \neg e, \neg a, \neg b, c\}$ which is a model of $(K \setminus CL(S \cap R_K^+)) \cup A$.

Moreover, as a consequence of Proposition 6, the following result holds.

Proposition 7 *Let $R \subseteq K$. $(K \setminus R) \cup A$ is consistent iff there exists an answer set S of $P_{K\cup A}$ such that $CL(S \cap R_K^+) = R$.*

Example 6 Coming back to the previous example, the 12 answer sets of $P_{K\cup A}$ are the following:

$$S_1 = \{d, e', a', b', c, r'_{a\vee c}, r'_{b\vee c}, r_{\neg c}\},$$
$$S_2 = \{d, e', a', b', c, r_{a\vee c}, r_{b\vee c}, r'_{\neg c}\},$$
$$S_3 = \{d, e', a', b', c, r_{a\vee c}, r'_{b\vee c}, r_{\neg c}\},$$
$$S_4 = \{d, e', a', b', c, r'_{a\vee c}, r_{b\vee c}, r_{\neg c}\},$$
$$S_5 = \{d, e', a', b', c, r_{a\vee c}, r_{b\vee c}, r_{\neg c}\},$$
$$S_6 = \{d, e', a', b', c', r_{a\vee c}, r_{b\vee c}, r_{\neg c}\},$$
$$S_7 = \{d', e, a', b', c, r'_{a\vee c}, r'_{b\vee c}, r_{\neg c}\},$$
$$S_8 = \{d', e, a', b', c', r_{a\vee c}, r_{b\vee c}, r'_{\neg c}\},$$
$$S_9 = \{d', e, a', b', c, r_{a\vee c}, r'_{b\vee c}, r_{\neg c}\},$$
$$S_{10} = \{d', e, a', b', c, r'_{a\vee c}, r_{b\vee c}, r_{\neg c}\},$$
$$S_{11} = \{d', e, a', b', c, r_{a\vee c}, r_{b\vee c}, r_{\neg c}\},$$
$$S_{12} = \{d', e, a', b', c', r_{a\vee c}, r_{b\vee c}, r_{\neg c}\},$$

since $R_K^+ = \{r_{a\vee c}, r_{b\vee c}, r_{\neg c}\}$ we have

$$CL(S_1 \cap R_K^+) = CL(S_7 \cap R_K^+) = R_1 = \{\neg c\},$$
$$CL(S_2 \cap R_K^+) = CL(S_8 \cap R_K^+) = R_2 = \{a \vee c, b \vee c\},$$
$$CL(S_3 \cap R_K^+) = CL(S_9 \cap R_K^+) = R_3 = \{\neg c, a \vee c\},$$
$$CL(S_4 \cap R_K^+) = CL(S_{10} \cap R_K^+) = R_4 = \{\neg c, b \vee c\},$$
$$CL(S_5 \cap R_K^+) = CL(S_6 \cap R_K^+) = CL(S_{11} \cap R_K^+) = CL(S_{12} \cap R_K^+) = R_5 = \{\neg c, a \vee c, b \vee c\}.$$

We find the 5 subsets of clauses to remove from K to restore consistency.

In order to compute the answer sets corresponding to prioritized removed sets we introduce the notion of preferred answer set.

4.2 Preferred answer sets

Let $K = K_1 \cup \cdots \cup K_n$. For $1 \le i \le n$, the set R_{K_i} denotes $R_{K_i} = \{r_c \mid r_c \in R_K, \text{ and } c \in K_i\} \cup \{r'_c \mid r'_c \in R_K, \text{ and } c \in K_i\}$. The positive and the negative part of R_{K_i} are respectively denoted by $R_{K_i}^+ = \{r_c \mid r_c \in R_K \text{ and } c \in K_i\}$ and $R_{K_i}^- = \{r'_c \mid r'_c \in R_K \text{ and } c \in K_i\}$.

Definition 14 Let K be a consistent and prioritized finite set of clauses. Let S and S' be two sets of atoms. S is preferred to S' iff

(i) $\exists i,\ 1 \le i \le n,\ |S \cap R_{K_i}^+| < |S' \cap R_{K_i}^+|$;
(ii) $\forall j,\ 1 \le j < i,\ |S \cap R_{K_j}^+| = |S' \cap R_{K_j}^+|$.

We are now able to define the notion of preferred answer set.

Definition 15 Let S be a set of atoms. S is a preferred answer set of $P_{K\cup A}$ iff

(i) S is an answer set of $P_{K\cup A}$;
(ii) for every answer set S' of $P_{K\cup A}$, S' is not preferred to S.

The following result generalizes Proposition 7.

Proposition 8 *Let K be a consistent and prioritized finite set of clauses and let A be a finite consistent set of clauses. R is a prioritized removed set of $K \cup A$ iff there exists a preferred answer set S of $P_{K\cup A}$ such that $CL(S \cap R_K^+) = R$.*

The proof of Proposition 8 (provided in the Appendix) is based on exhibiting for each prioritized answer set its associated prioritized removed set, and conversely. More precisely, if S is a preferred answer set of $P_{K\cup A}$ then its associated prioritized removed set is simply defined by:

$$R = CL(S \cap R_K^+).$$

Now if R be a prioritized removed set of $K \cup A$, then select a model m of $(K \setminus R) \cup A$, and define S as:

$$S = \{a \in V : a \in m\} \cup \{a' \in V : \neg a \in m\} \cup \{r_c : c \in R\} \cup \{r'_c : c \in K \setminus R\}.$$

Then we show in appendix that S is indeed a prioritized preferred asnwer set.

Example 7 Let us continue again our previous examples (Examples 5 and 6). Recall that: $K = \{\neg c,\ a \vee c,\ b \vee c, \neg d, \neg e\}$, and $A = \{\neg a, \neg b, d \vee e, \neg d \vee \neg e\}$.
Assume that we have the knowledge base is prioritized as follows:

$$K_1 = \{\neg c\}, \qquad K_2 = \{a \vee c, b \vee c\}.$$

It is easy to check that there exists exactly one prioritized removed set:

$$R = \{a \vee c, a \vee b\}.$$

Moreover, from the previous example (Example 6), one can easily check that the only preferred answer sets are:

$$S_2 = \{d, e', a', b', c, r_{a\vee c}, r_{b\vee c}, r'_{\neg c}\},$$
$$S_8 = \{d', e, a', b', c', r_{a\vee c}, r_{b\vee c}, r'_{\neg c}\}.$$

First, we can check that: $R = CL(S \cap R_K^+)$. Indeed, recall that:

$$R_K^+ = \{r_{a\vee c}, r_{b\vee c}, r_{\neg c}\}.$$

Then we have: $S_2 \cap R_K^+ = S_8 \cap R_K^+ = \{r_{a\vee c}, r_{b\vee c}\}$, from which $CL(S_2 \cap R_K^+) = CL(S_8 \cap R_K^+) = \{a \vee c, a \vee b\}$ gives the prioritized set R.

Now let us show the converse. Namely, we have:

$$(K \setminus R) \cup A = \{\neg c, \neg a, \neg b, d \vee e, \neg d \vee \neg e\}.$$

A model of $(K \setminus R) \cup A$ is $m = \{\neg c, \neg a, \neg b, \neg d, e\}$. Defining:

$$\begin{aligned} S &= \{a \in V : a \in m\} \cup \{a' \in V : \neg a \in m\} \\ &\quad \cup \{r_c : c \in R\} \cup \{r'_c : c \in K \setminus R\} \\ &= \{e\} \cup \{c', a', b', d'\} \cup \{r_{a\vee c}, r_{b\vee c}\} \cup \{r'_{\neg c}\} \\ &= \{e, c', a', b', d', r_{a\vee c}, r_{b\vee c}, r'_{\neg c}\}. \end{aligned}$$

Clearly, $S = S_2$ which is a prioritized answer set.

In this paper, we consider a cardinality-based criterion to select preferred answer sets (respectively prioritized removed set). Our method can also be adapted to consider inclusion-based criterion instead of cardinality-based criterion. It is well-known (see [6, 23]) that the computational complexity of inclusion-based revision (which is in Π_p^2) is higher that the cardinality-based revision (which is in Δ_p^2). This has its importance in our application context which deals with a large amount of data. In fact, the number of inclusion-based preferred removed sets is generally larger than the number of cardinality-based preferred removed sets. Moreover, the cardinality-based revision satisfies the whole set of AGM rational postulates (which is not the case with inclusion-based revision).

5 Adaptation of smodels for PRSR

We now present the computation of Prioritized Removed Sets Revision based on the adaptation of the smodels system; for more details see [15, 25, 33]. This is achieved using two algorithms. The first algorithm, Prio, is an adaptation of the smodels system algorithm which computes the set of sets of literals of R_K which lead to preferred answer sets and which minimize the number of clauses to remove from each stratum. The second algorithm, Rens, computes the prioritized removed sets of $K \cup A$, applying the principle of minimal change defined in Sect. 3.3 for PRSR, that is, stratum by stratum.

Propositions 7 and 8 give us a basis for providing an algorithm for the computation of Prioritized removed sets revision. However, we need to adapt smodels algorithms, so that rather than computing all answer sets we compute preferred answer sets. The idea in our adaptation is that rather than producing all answer sets, we *only* provide preferred ones. This is done iteratively, stratum by stratum. Our algorithm is described in the two following sections.

function PRIO($P_{K\cup A}, L, k, \mathcal{X}$) ▷ L and L' are sets of literals, a is an atom
  $L' \leftarrow Expand(P_{K\cup A}, L)$
    if L' is inconsistent **then**
      return $\mathcal{X}$
    else if (1) $\exists X \in \mathcal{X},\ |L' \cap R^+_{K_k}| > |X \cap R^+_{K_k}|$ **then**
      return $\mathcal{X}$
    else if (2) $L' \cap Lit(R_{K_1\cup\cdots\cup K_k}) \in \mathcal{X}$ **then**
      return $\mathcal{X}$
    else if L' covers $Atom(P_{K\cup A})$ **then**
      if (3) $\exists X \in \mathcal{X},\ |L' \cap R^+_{K_k}| < |X \cap R^+_{K_k}|$ **then**
        return $\{L' \cap Lit(R_{K_1\cup\cdots\cup K_k})\}$
      else
        return $\mathcal{X} \cup \{L' \cap Lit(R_{K_1\cup\ldots\cup K_k})\}$
      end if
    else
      $a \leftarrow Heuristic(P_{K\cup A}, L')$
      $\mathcal{X}' \leftarrow Prio(P_{K\cup A}, L' \cup \{a\}, k, \mathcal{X})$
      return $Prio(P_{K\cup A}, L' \cup \{\neg a\}, k, \mathcal{X}')$
    end if
end function

Fig. 2 The Prio algorithm

5.1 Prio: an adaptation of smodels system

Let $K = K_1 \cup \cdots \cup K_n$. Consider the stratum k. Let L be a set of literals which is an interpretation of $R_{K_1\cup\cdots\cup K_{k-1}}$ leading to an answer set and let $\mathcal{X}$ be the set of sets of literals which are interpretations of $R_{K_1\cup\cdots\cup K_k}$ leading to an answer set and such that they remove the same number of clauses from K_k. More formally: $\forall X, Y \in \mathcal{X}$, $|X \cap R^+_{K_k}| = |Y \cap R^+_{K_k}|$. The algorithm $Prio(P_{K\cup A}, L, k, \mathcal{X})$ returns the sets of literals which are interpretations of $R_{K_1\cup\cdots\cup K_k}$ that either contain L or belong to $\mathcal{X}$ and that minimize the number of clauses to remove from K_k, that is the number of r_c such that $c \in K_k$.

The function *Prio*, presented in Fig. 2, constructs a set of literals L' from L where, as in the construction of smodels, several cases hold:

(i) if L' is inconsistent then L' does not lead to an answer set therefore $\mathcal{X}$ is returned.
(ii) if L' is consistent then again several cases hold:
  (1) if L' removes more clauses from K_k than an element of $\mathcal{X}$ then $\mathcal{X}$ is returned.
  (2) if L' leads to the same answer set as an element of $\mathcal{X}$ then $\mathcal{X}$ is returned.
(iii) if L' is consistent and covers $Atom(P_{K\cup A})$ then
  (3) if L' removes less clauses from K_k than any element of $\mathcal{X}$ then $\mathcal{X}$ is cancelled and $L' \cap Lit(R_{K_1\cup\cdots\cup K_k})$ is returned else $L' \cap Lit(R_{K_1\cup\cdots\cup K_k})$ is added to $\mathcal{X}$
(iv) if L' is consistent and does not cover $Atom(P_{K\cup A})$ then using some heuristics a new atom $a \in Atom(P_{K\cup A})$ is selected such that $a \notin L'$. The algorithm starts again with $L' \cup \{a\}$ and keeps in $\mathcal{X}'$ the sets of literals of $R_{K_1\cup\cdots\cup K_k}$ that minimize the number of clauses to remove from K_k and starts again with $L' \cup \{\neg a\}$.

The main adaptations of the original smodels algorithm consist in: (1) avoiding all the sets of literals of $R_{K_1\cup\cdots\cup K_k}$ leading to an answer set which removes more clauses from K_k than those in $\mathcal{X}$; (2) not computing several times the same sets of literals of $R_{K_1\cup\cdots\cup K_k}$ leading to an answer set; (3) comparing each new set of literals of $R_{K_1\cup\cdots\cup K_k}$ leading to an answer set with the elements of $\mathcal{X}$, if the new set removes less clauses from K_k than those in $\mathcal{X}$ then $\mathcal{X}$ is replaced by it.

5.2 Rens: an algorithm computing the prioritized removed sets

We finally present the function *Rens* which computes the prioritized removed sets of $K \cup A$. The idea is to proceed stratum by stratum using the function *Prio* algorithm defined in the previous subsection. We start with the empty set and we first compute, the subsets of literals of R_{K_1} leading to an answer set which minimize the number of clauses to remove from K_1, then among these subsets we compute the subsets of literals of $R_{K_1\cup K_2}$ leading to an answer set which minimize the number of clauses to remove from K_2, and so on. From a stratum to another, the algorithm *Prio* described in the previous subsection provides the subsets of literals of $R_{K_1\cup\cdots\cup K_k}$ leading to an answer set which minimize the

function RENS($P_{K\cup A}$) ▷ $\mathcal{X}$ and $\mathcal{Y}$ are two sets of sets of literals, k is an integer
  $k \leftarrow 1$
  $\mathcal{X} \leftarrow \{\emptyset\}$
  while $k \leq n$ **do**
    $\mathcal{Y} \leftarrow \{\emptyset\}$ **(B1)**
    while $\mathcal{X} \neq \emptyset$ **do**
      (I1) choose an element $X \in \mathcal{X}$ **(B2)**
      $\mathcal{Y} \leftarrow Prio(P_{K\cup A}, X, k, \mathcal{Y})$
      $\mathcal{X} \leftarrow \mathcal{X} \backslash \{X\}$
    end while
    $\mathcal{X} \leftarrow \mathcal{Y}$
    $k \leftarrow k + 1$
  end while
  return $\{CL(X \cap R^{+}_{K_1\cup\cdots\cup K_k}) \mid X \in \mathcal{X}\}$
end function

Fig. 3 The function *Rens*

number of clauses to remove from K_k. The function *Rens* is detailed in Fig. 3 and the following proposition holds.

Proposition 9 *Let K be a consistent and prioritized finite set of clauses and let A be a finite consistent set of clauses. R is a prioritized removed set of $K \cup A$ iff $R \in Rens(P_{K\cup A})$.*

In order to illustrate the mechanism of the functions *Rens* and *Prio*, we present an example.

Example 8 Consider the following sets of clauses:

$$K = K_1 \cup K_2,$$

$$K_1 = \{\neg a \vee b, \neg b \vee a\},$$

$$K_2 = \{\neg c \vee d, \neg d \vee c\}$$

$$A = \{a \vee b, \neg a \vee \neg b, c \vee d, \neg c \vee \neg d\}.$$

$P_{K\cup A}$ contains the following rules:

$$\text{(i)} \begin{cases} a \leftarrow not\ a' & a' \leftarrow not\ a \\ b \leftarrow not\ b' & b' \leftarrow not\ b \\ c \leftarrow not\ c' & c' \leftarrow not\ c \\ d \leftarrow not\ d' & d' \leftarrow not\ d \\ r_{\neg a\vee b} \leftarrow not\ r'_{\neg a\vee b} & r'_{\neg a\vee b} \leftarrow not\ r_{\neg a\vee b} \\ r_{\neg b\vee a} \leftarrow not\ r'_{\neg b\vee a} & r'_{\neg b\vee a} \leftarrow not\ r_{\neg b\vee a} \\ r_{\neg c\vee d} \leftarrow not\ r'_{\neg c\vee d} & r'_{\neg c\vee d} \leftarrow not\ r_{\neg c\vee d} \\ r_{\neg d\vee c} \leftarrow not\ r'_{\neg d\vee c} & r'_{\neg d\vee c} \leftarrow not\ r_{\neg d\vee c} \end{cases}$$

$$\text{(ii)} \begin{cases} false \leftarrow a', b' & false \leftarrow c', d' \\ false \leftarrow a, b & false \leftarrow c, d \\ contradiction \leftarrow false, notcontradiction. \end{cases}$$

$$\text{(iii)} \begin{cases} r_{\neg a\vee b} \leftarrow a, b' & r_{\neg b\vee a} \leftarrow b, a' \\ r_{\neg c\vee d} \leftarrow c, d' & r_{\neg d\vee c} \leftarrow d, c' \end{cases}$$

Figure 4 shows a possible *Prio* call tree, beginning with its first call from Rens.

The nodes of the tree correspond to the recursive calls to *Prio*. The following highlights the execution of *Prio* in each node:

- ① $Prio(P_{K\cup A}, \emptyset, 1, \emptyset)$
  - $L' \leftarrow \emptyset$ (*Expand* produces nothing);
  - L' is not inconsistent;
  - (1) is not verified;
  - (2) is not verified;
  - L' does not cover $Atom(P_{K\cup A})$;
  - The heuristic choice is $not\ a'$;
- ② $Prio(P_{K\cup A}, \{not\ a'\}, 1, \emptyset)$
  - $L' \leftarrow \{b', a, r_{\neg a\vee b}, not\ a', not\ b, not\ r'_{\neg a\vee b}\}$
  - L' is not inconsistent;
  - (1) is not verified;
  - (2) is not verified;
  - L' does not cover $Atoms(P_{K\cup A})$
  - The heuristic choice is $not\ r'_{\neg b\vee a}$;
- ③ $Prio(P_{K\cup A}, \{b', a, r_{\neg a\vee b}, not\ a', not\ b, not\ r'_{\neg a\vee b}, not\ r'_{\neg b\vee a}\}, 1, \emptyset)$
  - $L' \leftarrow \{b', a, r_{\neg a\vee b}, r_{\neg b\vee a}, not\ a', not\ b, not\ r'_{\neg a\vee b}, not\ r'_{\neg b\vee a}\}$
  - L' is not inconsistent;
  - (1) is not verified;
  - (2) is not verified;
  - L' does not cover $Atoms(P_{K\cup A})$;
  - The heuristic choice is $not\ c'$;
- ④ $Prio(P_{K\cup A}, \{b', a, r_{\neg a\vee b}, r_{\neg b\vee a}, not\ a', not\ b, not\ r'_{\neg a\vee b}, not\ r'_{\neg b\vee a}, not\ c'\}, 1, \emptyset)$
  - $L' \leftarrow \{b', a, r_{\neg a\vee b}, r_{\neg b\vee a}, d', c, r_{\neg c\vee d}, not\ a', not\ b, not\ r'_{\neg a\vee b}, not\ r'_{\neg b\vee a}, not\ c', not\ d, not\ r'_{\neg c\vee d}\}$
  - L' is not inconsistent;
  - (1) is not verified;
  - (2) is not verified;
  - L' does not cover $Atoms(P_{K\cup A})$;

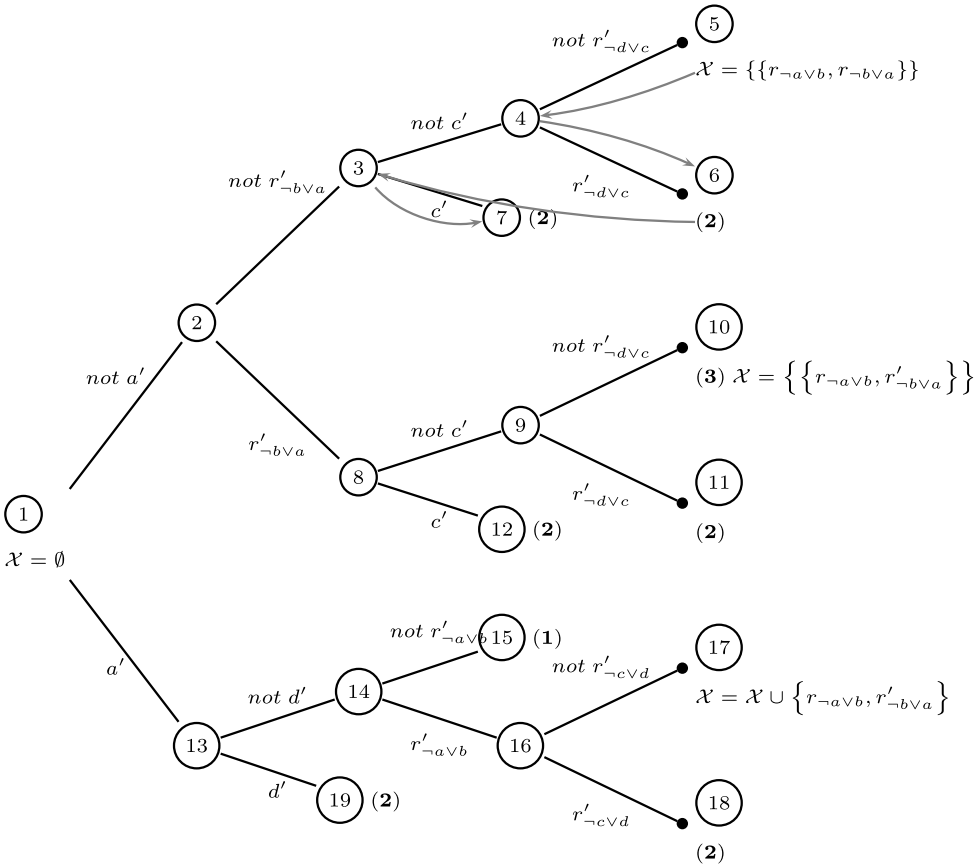

Fig. 4 Prio call tree on stratum 1. Each node has a n label, n being a node number corresponding to the tree walking order. The bold number in parentheses corresponds to the conditions of rejection of a partial model in *Prio*. See details about each node in example. The grey arrows show how $\mathcal{X}$ is propagated in the search tree

  - The heuristic choice is $not\ r'_{\neg d \vee c}$;
- ⑤ $Prio(P_{K \cup A}, \{b', a, r_{\neg a \vee b}, r_{\neg b \vee a}, d', c, r_{\neg c \vee d}, not\ a',$
  $not\ b, not\ r'_{\neg a \vee b}, not\ r'_{\neg b \vee a}, not\ c',$
  $not\ d, not\ r'_{\neg c \vee d}, not\ r'_{\neg d \vee c}\}, 2, \emptyset)$
  - $L' \leftarrow \{b', a, r_{\neg a \vee b}, r_{\neg b \vee a}, d', c, r_{\neg c \vee d}, r_{\neg d \vee c},$
    $not\ a', not\ b, not\ r'_{\neg a \vee b}, not\ r'_{\neg b \vee a},$
    $not\ c' not\ d, not\ r'_{\neg c \vee d}, not\ r'_{\neg d \vee c}\}$
  - L' is not inconsistent;
  - (1) is not verified;
  - (2) is not verified;
  - L' covers $Atoms(P_{K \cup A})$;
  - (3) is not verified;
  - we return a new $\mathcal{X} = \{\{r_{\neg a \vee b}, r_{\neg b \vee a}\}\}$.

The remaining of the branch $not\ r'_{\neg b \vee a}$ does not add any partial removed set. At nodes ⑥ and ⑦, the potential removed sets are rejected, because of condition (2), i.e. the set $\{r_{\neg a \vee b}, r_{\neg b \vee a}\}$ is already in $\mathcal{X}$ (we skip the details of the construction, but Fig. 4 sketches it). At this point, the only set in $\mathcal{X}$ is not optimal with respect to stratum 1. Then the algorithm explores the branch $r'_{\neg b \vee a}$. This branch produces in node ⑩ the potential removed set $\{r_{\neg a \vee b}, r'_{\neg b \vee a}\}$. This discards the previous element of $\mathcal{X}$, because of condition (3).

Finally, The branch a' is explored, producing minimal sets according to stratum 1 and containing $r_{\neg b \vee a}$. Note that at node ⑮, the tree is pruned because L' contains both $r_{\neg b \vee a}$ and $r_{\neg a \vee b}$, to it violates the cardinality condition (1), avoiding the unnecessary computation of useless models. At the end of execution of *Prio*, we have two partial removed sets in $\mathcal{X}$, minimizing the literals in $R^+_{K_1}$.

$$\mathcal{X} = \left\{ \begin{array}{l} \{r_{\neg a \vee b}, r'_{\neg b \vee a}\}, \\ \{r'_{\neg a \vee b}, r_{\neg b \vee a}\}, \end{array} \right\}.$$

The next iteration of loop B2 in *Rens* will complete and optimize $\mathcal{X}$, keeping only the sets which are minimal according to stratum 2. In our case, the first call to *Prio* will be $Prio(P_{K \cup A}, \{r_{\neg a \vee b}, r'_{\neg b \vee a}\}, 2, \emptyset)$. The search tree is depicted in Fig. 5.

- ① $Prio(P_{K \cup A}, \{r_{\neg a \vee b}, r'_{\neg b \vee a}\}, 2, \emptyset)$
  - $L' \leftarrow \{b', a, r'_{\neg b \vee a}, r_{\neg a \vee b}, not\ a', not\ b,$
    $not\ r_{\neg b \vee a}, not\ r'_{\neg a \vee b}\}$;
  - L' is not inconsistent;
  - (1) is not verified;
  - (2) is not verified;
  - L' does not cover $Atoms(P_{K \cup A})$;
  - The heuristic choice is $not\ d$;
- ② $Prio(P_{K \cup A}, \{b', a, r'_{\neg b \vee a}, r_{\neg a \vee b}, not\ a', not\ b,$
  $not\ r_{\neg b \vee a}, not\ r'_{\neg a \vee b}, not\ d\}, 2, \emptyset)$
  - $L' \leftarrow \{b', a, d', c, r_{\neg c \vee d}, r'_{\neg b \vee a}, r_{\neg a \vee b}, not\ a', not\ b,$
    $not\ c', not\ d, not\ r'_{\neg c \vee d}, not\ r_{\neg b \vee a}, not\ r'_{\neg a \vee b}\}$

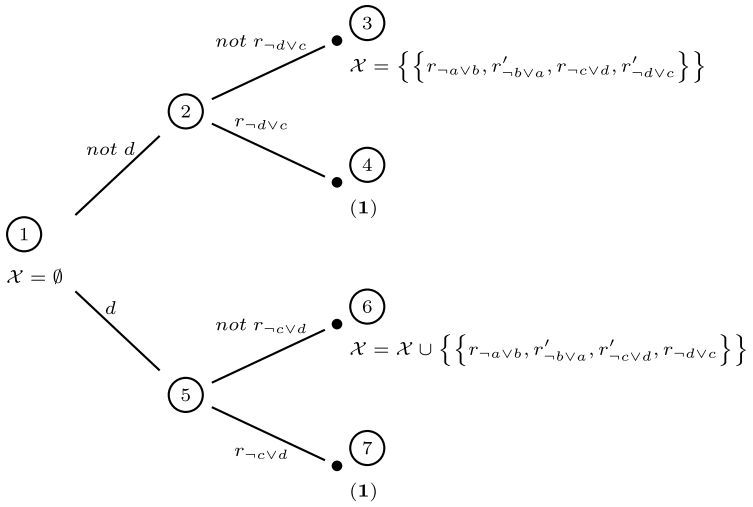

Fig. 5 Search tree of the second call to Prio

- L' is not inconsistent;
- (1) is not verified;
- (2) is not verified;
- L' does not cover $Atoms(P_{K \cup A})$;
- The heuristic choice is *not* $r_{\neg d \vee c}$;

– ③ $Prio(P_{K \cup A}, \{b', a, d', c, r_{\neg c \vee d}, r'_{\neg b \vee a}, r_{\neg a \vee b}, not\ a', not\ b, not\ c', not\ d, not\ r'_{\neg c \vee d}, not\ r_{\neg b \vee a}, not\ r'_{\neg a \vee b}, r_{\neg d \vee c}\}, 2, \emptyset)$

- $L' \leftarrow \{b', a, d', c, r_{\neg c \vee d}, r'_{\neg d \vee c} r'_{\neg b \vee a}, r_{\neg a \vee b}, not\ a', not\ b, not\ c', not\ d, not\ r'_{\neg c \vee d}, not\ r_{\neg b \vee a}, not\ r'_{\neg a \vee b}, r_{\neg d \vee c}\}$
- L' is not inconsistent;
- (1) is not verified;
- (2) is not verified;
- L' covers $Atoms(P_{K \cup A})$;
- (3) is not verified;
- we return a new $\mathcal{X} = \{\{r_{\neg a \vee b}, r'_{\neg b \vee a}, r_{\neg c \vee d}, r'_{\neg d \vee c}\}\}$.

For the remaining of the search tree, in node ④ L' is rejected because it contains both $r_{\neg d \vee c}$ and $r_{\neg c \vee d}$, so condition (1) rejects L'. Node ⑥ adds a new set to $\mathcal{X}$, and in node ⑦ L' is rejected, similarly to ④.

The remaining call to *Prio*, starting with $L = \{r'_{\neg a \vee b}, r_{\neg b \vee a}\}$ will produce the two remaining removed sets. Finally, the four removed sets are

$$\mathcal{X} = \left\{ \begin{array}{l} \{r_{\neg a \vee b}, r'_{\neg b \vee a}, r_{\neg c \vee d}, r'_{\neg d \vee c}\}, \{r_{\neg a \vee b}, r'_{\neg b \vee a}, r_{\neg d \vee c}, r'_{\neg c \vee d}\} \\ \{r'_{\neg a \vee b}, r_{\neg b \vee a}, r_{\neg c \vee d}, r'_{\neg d \vee c}\}, \{r'_{\neg a \vee b}, r_{\neg b \vee a}, r_{\neg d \vee c}, r'_{\neg c \vee d}\} \end{array} \right\}.$$

5.3 Comparing *Rens* with the `minimize` approach

Note that the computation of the prioritized removed sets could be done with the use of the `minimize` statement of smodels and a minor modification of smodels. We begin this section by briefly describing this approach. Next we discuss why our approach is more efficient than the use of the `minimize` statement.

Starting with the encoding of the knowledge described in the previous section, we can use the `minimize` statement of smodels to compute the prioritized removed sets. As explained in [34], multiple minimize statements in a single answer set program *"orders the stable models lexicographically according to the weights of the statements with the first statement being the most significant"* (practically, smodels works in reverse order, so the last statement is the most significant). In our case, we want to order models lexicographically according to the different strata, stratum 1 being the most significant. For us, the weight is just the cardinality of $R^+_{K_k}$, for each stratum k. In order to use `minimize` statements to compute the prioritized removed sets using this ordering of models, we can add the following statements to the previously defined encoding:

$\forall R_i = \{r_{c_1}, \ldots, r_{c_m}\}, \quad i \in \{1, \ldots, k\}$
`minimize` $\{r_{c_1}, \ldots, r_{c_m}\}$.

However, as noted in the same paper, smodels only finds one minimal model. This is because the test for the minimality of a model is done during the conflict detection in the current potential answer set being under construction, using the following algorithm:

▷ L is the current partial answer set, $weight(L)$ is the weight of the model
for each `mimimize` statement s **do**
  if $weight(L) >$ actual optimum for s **then**
    return conflict; ▷ L rejected
  else if $weight(L) <$ actual optimum for s **then**
    return no conflict ▷ L accepted
  else if there are no more `minimize` statements **then**
    return conflict
  end if
end for
return no conflict

In order to compute *all* minimal models, all we have to do is to remove the last condition, because it rejects the models which have a weight equal to the currently optimal weight of the considered statement s. With this small modification, smodels is able to compute the prioritized removed sets.

The two previously described implementations of the computation of the prioritized removed sets are rather different. The one using the `minimize` statement tries to stick with the good practice now established in the ASP community, consisting of the translation of an extension of ASP into a standard ASP program. Even with this point in mind, we saw that we have to do some modifications in the code of the solver. It remains that this modification is minor.

The other approach, using the functions *Prio* and *Rens*, modifies more heavily the initial solver.[4]

Concerning the efficiency, our direct approach is more efficient than the use of the `minimize` statement, mainly because we only deal with one strata at a time, while the computations involved by the `minimize` statements examine all strata during each conflict detection phase, as shown by the above algorithm.

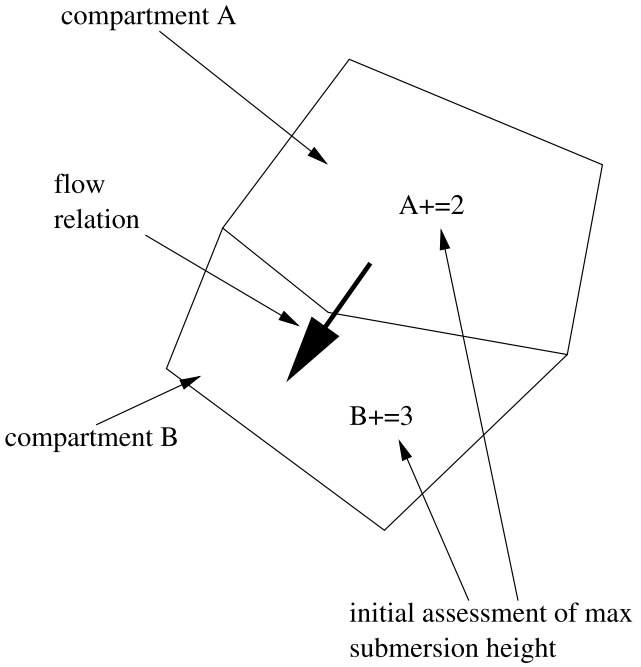

Fig. 6 Visual description of the sources of information in the flooding application

6 Application in the framework of GIS

6.1 Description of the application

We conducted tests based on the data of an application. The aim of that application is to assess water height at different locations in a flooded valley without building a complete hydrological model of the valley. The valley is segmented into 120 *compartments*, which are geographical entities in which the water height can be considered as constant. The goal is to assess a minimum/maximum interval of water height for each compartment in the valley. For this, we use two sources of information about these compartments (aside from the knowledge of their geographical layout), see Fig. 6.

The first source of information, denoted by S_2, is a set of hydraulic relations between neighbouring compartments. This source is incomplete (not all neighbouring compartments are connected) and considered quite certain by the experts. The second source of information, denoted by S_1, consists of a set of initial assessments of minimal and/or maximal submersion heights for *some* compartments (i.e. this source in incomplete). This information also is uncertain, and considered by the experts as being less reliable than the first source. For more details see [28] and [38].

A solution for this problem has been initially proposed by the CEMAGREF.[5] Their initial vision of the problem was to find a simple schema for the resolution of the constraints describing the problem. Using the oriented nature of the constraints, induced by the upstream to downstream flowing of water, they designed an algorithm of quadratic complexity allowing to stretch the minimal/maximal intervals of water heights, after the removal of the conflicts present in the initial data.

Our objective on this application concentrates on the detection of conflicts, as it has been done in e.g. [38]. As it has been said in the introduction, the huge amount of data led to insatisfactory results, which prevents the processing of the whole valley. Using stratification, we essentially want to reuse the upstream/downstream orientation of the initial data, and rethink about the initial assertion of the CEMAGREF people, which states that we can be more confident about the upstream initial assertions than on the downstream ones.

6.2 Representation with a logic program

The available knowledge is translated into a set of propositional formulas. The description of the variables (water heights) and their domains leads to n-ary positive clauses (enumeration of possible values) and binary negative clauses (mutual exclusion of the values). The initial assessments of water heights for some compartments are translated into a set of monoliteral clauses representing the assessed height values. In the sequel, by abuse of notation, we denote by S_1 the set of clauses describing the initial assessments.

[4] Even if this can be coded very cleanly, thanks to the well documented smodels API.

[5] French research institute for agriculture and environment engineering.

Concerning hydraulic relations, they are expressed in terms of inequalities on the bounds of the water height. They are translated into binary negative clauses representing the excluded tuples of values. By abuse of notation, in the sequel, we will denote by S_2 the set of clauses containing the clauses representing the hydraulic relations and the variable descriptions. We know that S_1 is consistent and S_2 is consistent, but $S_1 \cup S_2$ can be inconsistent. The goal is to drop out some of the initial assessments of S_1 in order to restore consistency. This leads to the revision of S_1 by S_2.

Example 9 Let A and B be two compartments, defining the following variables: A^+ and A^- for maximal and minimal submersion height for compartment A, and B^+ and B^- for the same counterparts for B. These variables are defined on a domain $D = \{1, 2, 3\}$. There is a flow pouring from A to B and there are assessments telling us that the maximum submersion height is 2 for A and 3 for B. The translation leads to a set S_2 containing:

1. Clauses describing the variables:

$$\left\{\begin{array}{l} A_1^+ \vee A_2^+ \vee A_3^+, \neg A_1^+ \vee \neg A_2^+, \neg A_1^+ \vee \neg A_3^+, \neg A_2^+ \vee \neg A_3^+, \\ A_1^- \vee A_2^- \vee A_3^-, \neg A_1^- \vee \neg A_2^-, \neg A_1^- \vee \neg A_3^-, \neg A_2^- \vee \neg A_3^-, \\ B_1^+ \vee B_2^+ \vee B_3^+, \neg B_1^+ \vee \neg B_2^+, \neg B_1^+ \vee \neg B_3^+, \neg B_2^+ \vee \neg B_3^+, \\ B_1^- \vee B_2^- \vee B_3^-, \neg B_1^- \vee \neg B_2^-, \neg B_1^- \vee \neg B_3^-, \neg B_2^- \vee \neg B_3^- \end{array}\right\},$$

2. The clauses describing the inequalities representing the flow relation (i.e. $A^+ \geq B^+$, $A^- \geq B^-$, $A^+ > B^-$):

$$\left\{\begin{array}{c} \neg A_1^+ \vee \neg B_2^+, \neg A_1^+ \vee \neg B_3^+, \neg A_2^+ \vee \neg B_3^+, \\ \neg A_1^- \vee \neg B_2^-, \neg A_1^- \vee \neg B_3^-, \neg A_2^- \vee \neg B_3^-, \\ \neg A_1^+ \vee \neg B_1^-, \neg A_1^+ \vee \neg B_2^-, \neg A_1^+ \vee \neg B_3^-, \ldots, \neg A_3^+ \vee \neg B_3^- \end{array}\right\}.$$

The set S_1 contains the initial assessments, that is, $S_1 = \{A_2^+, B_3^+\}$. In practice, of course, the problem is compactly encoded by means of cardinality constraints.

For each clause $c \in S_1$ we introduce a new atom r_c and we construct a logic program $P_{S_1 \cup S_2}$ according to the translation proposed in Sect. 4.1.

Example 10 Considering the previous example, the encoding is as follows:

- The generation rules for each propositional variables and each new atom r_c: $A_1^+ \leftarrow not\ A_1^{'+}$, $A_1^{'+} \leftarrow not\ A_1^+$, etc.
- One rule for each clause of S_2. The translation of the set S_2 begins as follows: $false \leftarrow not\ A_1^+, not\ A_2^+, not\ A_3^+$, $false \leftarrow A_1^+, A_2^+$, etc. and the contradiction detection rule: $contradiction \leftarrow not\ contradiction, false$.
- One rule for each clause of S_1. The translation of the set S_1 gives the following rules: $r_{A_2^+} \leftarrow A_2^{'+}$, $r_{B_3^+} \leftarrow B_3^{'+}$.

6.3 Experimental study and comparison

This subsection presents a summary of experimental results provided by our answer set programming (ASP) encoding of RSR and PRSR. The tests are conducted on a Pentium III cadenced at 1 GHz and equipped with 1 GB of RAM.

6.3.1 Comparing three implementations of RSR

First, we compare the results of RSR using three approaches:

- the ASP encoding of the problem presented in this article, with the modification of smodels also presented here.
- a direct encoding into a SAT problem, using a modified version of an efficient SAT-solver, MiniSat [2]. This encoding is presented in annex.
- the REM algorithm, presented by Würbel et al. in [38] which computes the removed sets by using a modification of Reiter's algorithm for the computation of minimal hitting sets.The encoding and the REM algorithm is presented in annex.

This comparative test deals with an increasing number of compartments from ten to sixty four compartments. The aim of this test is to compare the performance of the three approaches on the application and to identify their limits. The 64 compartments have been picked out of an area of 120 (randomly, but choosen compartments are neighbours). Ten tests have been performed for a same number of compartments and an average running time on the ten tests is given.

We observe that, after 30 compartments, the REM algorithm quickly arrives to its maximum capacity. Over 35 compartments, tests where aborted, reaching a time limit (10 hours) or a memory limit (500 Mb of memory).

Concerning the ASP encoding and the SAT encoding, we observe that until 35 compartments, the two approaches behave similarly. From 40 compartments, the ASP encoding begins to give better results, and from 45 compartments the ASP encoding is significantly better than the SAT encoding. From 50 compartments the SAT encoding reaches a limit in CPU time (10 hours). The ASP encoding can deal with 60 compartments with a reasonable running time (few minutes) and reaches a limit in CPU time around 64 compartments. The results of the test, described in Table 1, are illustrated graphically in Fig. 7.

6.3.2 Benefit of adding priorities

We use an implementation of Prioritized Removed Set Revision which relies on the ASP encoding and the associated algorithms presented in this article.

Table 1 Comparison between ASP encoding, SAT encoding, and REM algorithm. The last line (>64) states that we cannot handle these cases (computation time greater than one hour)

# of compartments	# of variables	# of clauses	Time ASP (s)	Time MiniSat (s)	Time REM (s)
10	210	2387	0.06	0.003	
15	316	3650	0.06	0.017	0.424
20	432	5602	0.06	0.041	1.403
25	530	6796	0.137	1.437	0.988
30	661	9233	0.403	2.294	168.63
35	764	10446	0.890	20.907	403.424
40	869	11998	14.296	150.275	–
45	983	13833	15.584	255.116	–
50	1092	15373	48.530	–	–
55	1177	15665	156.895	–	–
60	1302	18118	210.318	–	–
61	1337	18887	240.449	–	–
62	1329	17960	264.100	–	–
63	1368	18847	278.418	–	–
64	1381	18782	331.436	–	–
> 64	–	–	–	–	–

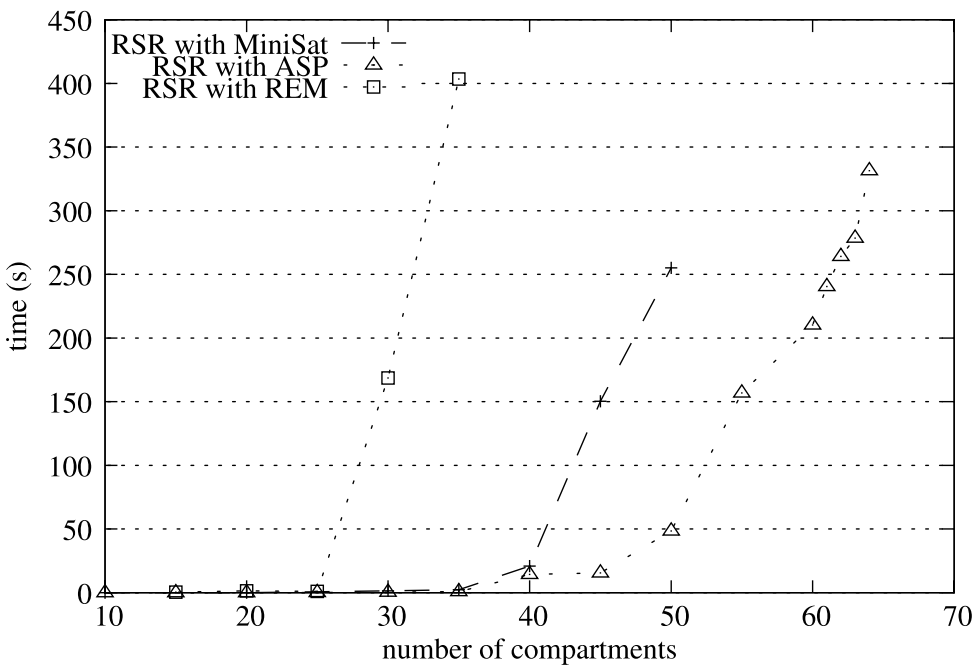

Fig. 7 Comparison between ASP encoding and SAT encoding in the flooding application

To estimate the gain induced by stratification, we performed a series of tests on the whole dataset, representing the 120 compartments of the valley.

Prioritized Removed Set Revision is performed with a stratification of S_1 induced from the geographic position of compartments. Compartments located in the north part of the valley are preferred to the compartments located in the south of the valley. We used an increasing number of strata from 2 to 5. For each number of strata, five tests were performed, using slightly different stratum content. Table 2 summarizes the results of the test. We can observe that Rens significantly reduces the running time.

In the flooding application we have to deal with an area consisting of 120 compartments and the stratification is useful to deal with the whole area. Using the stratification, Table 3 shows that Rens can deal with the whole area with a reasonable running time, even with only two strata.

Table 2 Gains induced by Rens

# of compartments	# strata	Time Rens (s)	Time ASP (s)	# of variables	# of clauses
64	2	55	331.436	1381	18782
64	3	21	331.436	1381	18782
64	4	24	331.436	1381	18782
64	5	19	331.436	1381	18782

Table 3 Gains induced by Rens on an area containing 120 compartments (direct ASP encoding cannot process this area)

# of compartments	# strata	time Rens (s)	# of variables	# of clauses
120	2	24132.49	2343	33751
120	3	3047.55	2343	33751
120	4	1698.67	2343	33751
120	5	424.62	2343	33751

7 Related works

Several approaches have been proposed to extend ASP to take into account preferences like Prioritized Logic Programming (PLP) [31], Answer Set Optimization approach (ASO) [4], and others that considers the preference is among rules like in [8] (see [32] for a comparative study).

Our approach departs from existing proposals regarding the following points:

1. In most existing approaches priorities are either attached to rules (e.g. [5, 40]) or to literals (atoms or negated atoms) (e.g. [31]). Moreover, priorities are encoded by means of rules of logic programs. In the Answer Set Optimization approach (ASO) [4], the ASO programs consist of two parts: a generating program, denoted by P_{gen} which produces answer sets and a preference program, denoted by P_{pref} which expresses user preferences. The answer sets generation and the answer set comparison is decoupled. All answer sets are first computed and they are compared according to the preference program P_{pref}. Similarly, the Prioritized Logic programming (PLP) [31] framework explicitly represents priorities in a logic program. PLP consists of a pair (P, Φ) where P is a general extended disjunctive program and Φ is a set of priorities over literals. The semantics is given in terms of preferred answer sets where the preference relation on answer sets comes from the preference relation over literals. Clearly, in our paper, priorities are attached to rules and more precisely to clauses of knowledge bases. However, there is no explicit encoding of priorities, namely in our approach we only consider a unique logic program and priorities are not encoded as additional rules in a logic program. In fact, they are integrated and taken into account in our adaptation of smodels algorithms.
2. Existing approaches do not modify algorithms to compute preferred answer sets (since priorities are encoded by means of additional rules). Many of existing algorithms first computes all answer sets, then proceeds to a selection of preferred ones on the basis of priority relations. For instance, within Prioritized Logic programming (PLP) the selection algorithm introduced in ([31], Sect. 4.1) requires the computation of every answer set in advance, which is not always reasonable. More precisely, in order to compute the preferred answer sets, the program P is first extended to a program P_Φ by the introduction of new atoms and new rules that represent the preferences over literals. The computation of preferred answer sets of P_Φ is achieved in two steps, all answer sets are first computed, then the answer sets that are not preferred according to the preference relation over literals are discarded. Our approach does not compute all answer sets, but directly computes the preferred answer sets by an adaptation of the s-model algorithm. Moreover, the computation of answer sets is done incrementally level by level.
3. The aim of existing approaches is to enrich the expressive power of ASP for instance by dealing with prioritized disjunctive logic programs or even dealing with general formulas. Some approaches deal with the update of logic programs [26] which stems from the extension of abductive logic programs. This work addresses update which is an operation that modifies logic programs. The aim of our approach is not to enrich the expressive power of ASP, but more to encode PRSR in order to be applied in the framework of GIS.
4. Most of existing approaches use inclusion-based criterion to define preferred answer sets. In the Answer Set Optimization approach (ASO) [4], a weaker criterion based on well-known Pareto-ordering has been used. In our paper, we adapted the cardinality-based criterion which extends the inclusion-set based criterion. Note that our approach can be easily adapted to produce preferred answer set with respect to inclusion-based crite-

rion. However, from computational point of view it is better to use a cardinality-based approach since the number of cardinality-based preferred answer sets is smaller than the number of inclusion-based preferred answer sets. This is particularly important when dealing GIS problems that involve a high number of formulas.

8 Concluding discussion

This paper generalized Removed Sets Revision to prioritized belief bases (Prioritized Removed Sets Revision) and showed that PRSR can be successfully encoded into answer set programming. An implementation stemming from smodels system is proposed and an experimental study in the framework of GIS shows that the answer set approach gives better results than the REM algorithm based on the adaptation of Reiter's algorithm for diagnosis and than an implementation based on an efficient SAT-solver MiniSat. Indeed, it first allows to deal with the whole area if priorities are provided, and even if there are no priorities, it can deal with 64 compartments, which is impossible with the REM algorithm nor with the SAT approach. It is important to note that both ASP encoding and SAT encoding introduce new variables (basically associated to clauses of the knowledge base).

In [34], the language of smodels has been extended and optimization statements for the smodels system have been proposed. In PRSR we use cardinality constraints for more compactly encoding the data. The computation of the prioritized removed sets could be done with the use of the `minimize` statement of smodels however we showed that our approach is more efficient than the use of the `minimize` statement.

Several approaches have been proposed to extend ASP to take into account preferences, like Prioritized Logic Programming (PLP) [31], Answer Set Optimization approach (ASO) [4], and others that considers the preference is among rules like in [8] (see [32] for a comparative study). Most of these approaches first generate all answer sets for a program then select the preferred ones. On contrast, we adapted the smodels algorithm in order to directly compute the preferred answer sets. Contrary to existing approaches, the first aim of the extension of ASP is not to enrich the expressive power of ASP, but more to encode PRSR in order to be applied in the framework of GIS.

Acknowledgement This work was supported by European Community project IST-1999-14189 REVIGIS.

Appendix A: Proofs

A.1 Proofs of Sect. 3

Proposition 1 *Let K and A be two consistent sets of clauses. If $K \cup A$ is inconsistent then $\mathcal{R}(K \cup A) \neq \emptyset$.*

Proof $K \cup A$ is inconsistent then, by the theorem of compacity of propositional calculus, there exists at least one minimal inconsistent subset of $K \cup A$. Since K and A are consistent, there exists at least one subset of clauses from K, denoted by R, which intersects each minimal inconsistent subset of $K \cup A$ such that $(K \backslash R) \cup A$ is consistent. In case of several sets R_i such that $(K \backslash R_i) \cup A$ is consistent, some are minimal in number of clauses and, according to the definition, they are removed sets therefore $\mathcal{R}(K \cup A) \neq \emptyset$. □

Proposition 2 *When $s(\mathcal{R}(K \cup A)) = \mathcal{R}(K \cup A)$, the revision operator $\circ_{RSR}$ satisfies the postulates* (G⋆1)–(G⋆8).

Proof

(G⋆1): This postulate is verified since $K \circ_{RSR} A$ is a belief base.

(G⋆2): By the definition of $\circ_{RSR}$, $A \in K \circ_{RSR} A$.

(G⋆3): follows from the fact that either

$$K \circ_{RSR} A = \bigvee_{R \in \mathcal{R}(K \cup A)} Cn((K \backslash R) \cup A)$$

or

$$K \circ_{RSR} A = Cn(K \cup A)$$

(G⋆4): if $K \not\models \neg A$ then, by definition, $K \circ_{RSR} A = Cn(K \cup A)$.

(G⋆5): By $(G \star 2)$, $A \in K \circ_{RSR} A$. If $\neg A$ is a tautology then $K \circ_{RSR} A$ is inconsistent. Conversely, since K is consistent and $A \in K \circ_{RSR} A$, if $K \circ_{RSR} A$ is inconsistent then $\neg A$ is a tautology.

(G⋆6): if $A \equiv B$ they share the same CNF, by the definition of $\circ_{RSR}$, $K \circ_{RSR} A = K \circ_{RSR} B$.

(G⋆7): By definition,

$$K \circ_{RSR} (A \wedge B) = \bigvee_{R \in \mathcal{R}(K \cup (A \wedge B))} Cn((K \backslash R) \cup (A \wedge B))$$

and

$$K \circ_{RSR} A = \bigvee_{R' \in \mathcal{R}(K \cup A)} Cn((K \backslash R') \cup A).$$

If $K \circ_{RSR} A + B \subset K \circ_{RSR} (A \wedge B)$ then $K \circ_{RSR} A + B$ removes more clauses from K than $K \circ_{RSR} (A \wedge B)$, therefore $|R| < |R'|$. By definition R and R' are subsets of clauses of K, and $(K \backslash R) \cup (A \wedge B) \equiv (K \backslash R) \cup \{A, B\}$ is consistent therefore R' is not a removed set of $\mathcal{R}(K \cup A)$ because $(K \backslash R) \cup A$ is consistent and $|R| < |R'|$. This leads to a contradiction.

$(G \star 8)$: By definition,

$$K \circ_{RSR} (A \wedge B) = \bigvee_{R \in \mathcal{R}(K \cup (A \wedge B))} Cn((K \backslash R) \cup \{A, B\}$$

and

$$A \in K \circ_{RSR} A = \bigvee_{R' \in \mathcal{R}(K \cup A)} Cn((K \backslash R') \cup A)$$

If $\neg B \notin K \circ_{RSR} A$ then $(K \backslash R') \cup (A \wedge B) \equiv (K \backslash R') \cup \{A, B\}$ is consistent, and by $(G \star 7)$, $|R'| \leq |R|$ holds. If $|R'| < |R|$ then R is not a removed set of $\mathcal{R}(K \cup (A \wedge B))$ since $(K \backslash R') \cup \{A, B\}$ is consistent. Therefore $|R'| = |R|$ and $K \circ_{RSR} A + B = K \circ_{RSR} (A \wedge B)$. □

Proposition 3 *Let R be a set of clauses,*

- *R is a removed set iff $(K \backslash R) \cup A \in MAXCONS_A(K) \cup A$;*
- *$K \circ_{RSR} A = \bigvee_{K' \in s(MAXCONS_A(K))} Cn(K' \cup A)$.*

Proof We recall that the selection function s is such that $s(\mathcal{R}(K \cup A)) = \mathcal{R}(K \cup A)$.

- We first show that if R is a removed set then $(K \backslash R) \cup A \in MAXCONS_A(K) \cup A$.
  If R is a removed set, by definition, $(K \backslash R) \cup A$ is consistent.
  If $(K \backslash R) \cup A \notin MAXCONS_A(K) \cup A$ then there exists a set M, such that $M \subset (K \cup A)$ and $|(K \backslash R) \cup A| < |M|$. Since $K \cup A$ is inconsistent there exists $R' \subset K$ such that $M = (K \backslash R') \cup A$ is consistent and $|R'| \leq |R|$ which contradicts the fact that R is a removed set.
- We now show that if $(K \backslash R) \cup A \in MAXCONS_A(K) \cup A$ then R is a removed set.
  If $(K \backslash R) \cup A \in MAXCONS_A(K) \cup A$, then $(K \backslash R) \cup A$ is consistent.
  If $(K \backslash R) \cup A \in MAXCONS_A(K) \cup A$, since $R \cap A = \emptyset$, then $K \backslash R \in MAXCONS_A(K)$, therefore, $K \backslash R \subseteq K$ and $(K \backslash R) \cup A$ is consistent. Since $K \backslash R \in MAXCONS_A(K)$, if there exists R', $R' \subseteq K$ and $(K \backslash R') \cup A$ is consistent such that $|R'| < |R|$ then $|K \backslash R| < |K \backslash R'|$ then $K \backslash R \notin MAXCONS_A(K)$.
- $K \circ_{RSR} A = \bigvee_{K' \in s(MAXCONS_A(K))} Cn(K' \cup A)$. The proof follows from the characterization of removed sets in terms of maximal consistent subsets and from the definition of RSR revision. □

Proposition 4 *The function that assigns K the total pre-order $\leq_K$ is a faithful assignment.*

Proof Let K be a finite set of clauses. A total pre-order between interpretations can be defined as follows: $\forall \omega_i, \omega_j \in \mathcal{W}4$, $\omega_i \leq_K \omega_j$ iff $|\mathcal{NS}_K(\omega_i)| \leq |\mathcal{NS}_K(\omega_j)|$. We show that the function that assigns K the total pre-order $\leq_K$ is a faithful assignment.

(1) If $\omega, \omega' \in \text{Mod}(K)$ then $\mathcal{NS}_K(\omega) = \mathcal{NS}_K(\omega') = \emptyset$ thus $|\mathcal{NS}_K(\omega)| = |\mathcal{NS}_K(\omega')| = 0$ therefore $\omega =_K \omega'$.

(2) If $\omega \in \text{Mod}(K)$ then $\mathcal{NS}_K(\omega) = \emptyset$ therefore $|\mathcal{NS}_K(\omega)| = 0$. If $\omega' \notin \text{Mod}(K)$ then $\mathcal{NS}_K(\omega) \neq \emptyset$ therefore $|\mathcal{NS}_K(\omega)| > 0$. By definition of $\leq_K$ we have $\omega <_K \omega'$.

(3) We first show that if $K \equiv L$ then $\leq_K = \leq_L$.
  Let K and L be two finite sets of clauses, $\forall \omega \in \mathcal{W}$, if $K \equiv L$ then $\mathcal{NS}_K(\omega) = \mathcal{NS}_L(\omega)$ and $|\mathcal{NS}_K(\omega)| = |\mathcal{NS}_L(\omega)|$.
  We now show If $\leq_K = \leq_L$ then $K \equiv L$.
  By contraposition we show that if $K \not\equiv L$ then $\leq_K \neq \leq_L$.
  If $K \not\equiv L$ then there exists $\omega \in \mathcal{W}$ such that $\omega \models K$ and $\omega \not\models L$ or such that $\omega \not\models K$ and $\omega \models L$.
  - if $\omega \models K$ and $\omega \not\models L$, then $|\mathcal{NS}_K(\omega)| = 0$ and $|\mathcal{NS}_L(\omega)| > 0$. Since L is an initial set of clauses, L is consistent and there exists ω' such that $\omega' \models L$ and $\mathcal{NS}_L(\omega') = \emptyset$ and $|\mathcal{NS}_L(\omega)| = 0$ thus $\omega <_L \omega'$. By consequence $\omega' = \min(\mathcal{W}, \leq_L)$ and $\omega = \min(\mathcal{W}, \leq_K)$ therefore $\leq_K \neq \leq_L$.
  - if $\omega \not\models K$ and $\omega \models L$, then $|\mathcal{NS}_K(\omega)| > 0$ and $|\mathcal{NS}_L(\omega)| = 0$. The proof is similar than the one above. □

Proposition 5 *Let $\mathcal{R}(K \cup A)$ be the set of removed sets of $(K \cup A)$.*

$$\text{Mod}(K \circ_{RSR} A) = \text{Mod}(K \circ_{RSR_{sem}} A).$$

Proof

- We first show that $\text{Mod}(K \circ_{RSR} A) \subseteq \min(\text{Mod}(A), \leq_K)$.
  $\forall \omega \in \text{Mod}(K \circ_{RSR} A)$ there exists at least one subset $R \in \mathcal{R}(K \cup A)$ such that $\omega \in \text{Mod}(((K \backslash R) \cup A)$. Let $\mathcal{NS}(\omega)$ be the set of clauses of K falsified by ω. Since R is a removed set, $\mathcal{NS}(\omega) = R$. Since $\mathcal{NS}(\omega) < R$, R is not a removed set holds. Therefore $\omega \not\models R$, $\omega \models A$ and $\omega \models K \backslash R$. We show that $\omega \in \min(\text{Mod}(A), \leq_K)$.
  If $\omega \notin \min(\text{Mod}(A), \leq_K)$ then there exists $\omega' \in \text{Mod}(A)$ such that $\mathcal{NS}(\omega') < \mathcal{NS}(\omega)$, since $\mathcal{NS}(\omega) = R$, then $\mathcal{NS}(\omega') < R$ this contradicts the fact that R is not a removed set. By consequence, $\text{Mod}(K \circ_{RSR} A) \subseteq \min(\text{Mod}(A), \leq_K)$.
- We now show that $\min(\text{Mod}(A), \leq_K) \subseteq \text{Mod}(K \circ_{RSR} A)$
  $\forall \omega \in \min(\text{Mod}(A)), \leq_K)$, let $\mathcal{NS}(\omega) \neq \emptyset$ because $K \cup A$ is inconsistent. Anyway $\mathcal{NS}(\omega) \subseteq K$ and $\omega \models A$ and $\omega \models K \backslash \mathcal{NS}(\omega)$, since $A \cap \mathcal{NS}(\omega) = \emptyset$, thus $\omega \models (K \backslash \mathcal{NS}(\omega)) \cup A$ therefore $(K \backslash \mathcal{NS}(\omega)) \cup A$ is consistent. We have to show that $\mathcal{NS}(\omega)$ is a removed set. We already shown that $\mathcal{NS}(\omega) \subseteq K$, then that $(K \backslash \mathcal{NS}(\omega)) \cup A$ is consistent.
  We show that $\forall \omega' \in \text{Mod}(A)$ we have $|\mathcal{NS}_K(\omega)| < |\mathcal{NS}_K(\omega')|$. If this is not the case then $\omega' \notin \min(\text{Mod}(A), \leq_K)$ which contradicts the hypothesis. By consequence $\mathcal{NS}(\omega)$ is a removed set and $\min(\text{Mod}(A), \leq_K) \subseteq \text{Mod}(K \circ_{RSR} A)$. □

A.2 Proofs of Sect. 4

In the following the interpretations are sets of atoms and for the sake of simplicity in the proofs, since we deal with normal logic programs, we identify the negative atoms $a' \in V^-$ with the negated atoms $\neg a$.

Proposition 6 *Let K be a consistent finite set of clauses and let A be a finite consistent set of clauses. Let $S \subseteq V$ be a set of atoms. S is an answer set of $P_{K\cup A}$ iff I_S is an interpretation of V^+ which satisfies $(K \setminus CL(S \cap R_K^+)) \cup A$.*

Proof We prove that if S is an answer set of $P_{K\cup A}$ then I_S is an interpretation of V^+ which satisfies $(K \setminus CL(S \cap R_K^+)) \cup A$.
We first prove that $I_S = \{a \mid a \in S\} \cup \{\neg a \mid a' \in S\}$ is an interpretation of V^+.

$\forall a \in V^+$, we show that either $a \in S$ or $a' \in S$. If it does not hold then there are two cases.

- Case 1: $a \notin S$ and $a' \notin S$. By the rule $a' \leftarrow not\ a$, we get $a' \in CN(P^S_{K\cup A})$. By the definition of answer sets $S = CN(P^S_{K\cup A})$, thus $a' \in S$, which is impossible.
- Case 2: $a \in S$ and $a' \in S$. Since the only rule with head a' is $a' \leftarrow not\ a$, we get $a' \notin CN(P^S_{K\cup A})$, but $S = CN(P^S_{K\cup A})$, thus $a' \notin S$, which is impossible.

We now show that *false* $\notin S$ and *contradiction* $\notin S$. If *false* $\in S$ then two cases hold.

- Case 1: *contradiction* $\in S$. As the only rule with head *contradiction* is *contradiction* $\leftarrow$ *false*, *not contradiction*, we get *contradiction* $\notin CN(P^S_{K\cup A}) = S$, which is impossible.
- Case 2: *contradiction* $\notin S$. By the rule *contradiction* $\leftarrow$ *false*, *not contradiction*, we get *contradiction* $\in CN(P^S_{K\cup A}) = S$, which is impossible. Now *contradiction* $\notin S$ follows from *false* $\notin S$ and from the fact that the only rule with head *contradiction* includes *false* in its body. Therefore I_S is an interpretation of V^+.

We now show that I_S satisfies $(K \setminus CL(S \cap R_K^+)) \cup A$.

We first show that I_S is a model of A. Suppose that I_S is not a model of A then $\exists\, c \in A$ such that $I_S \not\models c$. Clearly, by the rule introduced in (ii) that concerns c, *false* $\in CN(P^S_{K\cup A}) = S$, which is impossible since we have previously shown that *false* $\notin S$.

We now suppose that I_S is not a model of $(K \setminus CL(S \cap R_K^+)) \cup A$, then $\exists\, c \in (K \setminus CL(S \cap R_K^+)) \cup A$, such that $I_S \not\models c$. Since I_S is a model of A, thus $c \notin A$ therefore $c \in K$. However, by the rule introduced in (iii) that concerns c, $r_c \in CN(P^S_{K\cup A}) = S$, thus $c \in CL(S \cap R_K^+)$, which is impossible.
We now show that if I_S is an interpretation of V^+ which satisfies $(K \setminus CL(S \cap R_K^+)) \cup A$ then S is an answer set of $P_{K\cup A}$.

From I_S we construct a set S such that $S = \{a \mid a \in I_S\} \cup \{a' \mid \neg a \in I_S\}$.
We first show that $S \subseteq CN(P^S_{K\cup A})$.
Let $a \in S$. Two cases hold.

- Case 1: $a \in V^+$, since I_S is an interpretation of V^+ then $a' \notin S$. Thus, by the rule $a \leftarrow not\ a'$, we get $a \in CN(P^S_{K\cup A})$.
- Case 2: $a \in V^-$ is similar to the Case 1.

We now show that $CN(P^S_{K\cup A}) \subseteq S$.
Clearly $CN(P^S_{K\cup A}) \subseteq V \cup \{false, contradiction\}$.
We first show that $CN(P^S_{K\cup A}) \cap (V \setminus R_K) \subseteq S$.
Suppose that $\exists\, a \in CN(P^S_{K\cup A}) \cap (V \setminus R_K)$, $a \notin S$. Two cases hold.

- Case 1: $a \in V^+$. Then, $a' \in S$, since I_S is a an interpretation of V^+. Thus, since we have shown that $S \subseteq CN(P^S_{K\cup A})$, therefore $a' \in CN(P^S_{K\cup A})$. Thus, as the only rule with head a is $a \leftarrow not\ a'$, we get $a \notin CN(P^S_{K\cup A})$, which is impossible.
- Case 2: $a \in V^-$. Similar to the Case 1.

We show that $CN(P^S_{K\cup A}) \cap R_K \subseteq S$.
Suppose that $\exists a \in CN(P^S_{K\cup A}) \cap R_K$, $a \notin S$. Two cases hold.

- Case 1: $a \in R_K^-$. Then, $a = r'_c$ and $r_c \in S$, since I_S is an interpretation of V^+. Thus, since we have shown that $S \subseteq CN(P^S_{K\cup A})$, $r_c \in CN(P^S_{K\cup A})$. Therefore, as the only rule with head r'_c is $r'_c \leftarrow not\ r_c$, we get $r'_c \notin CN(P^S_{K\cup A})$, which is impossible.
- Case 2: $a \in R_K^+$. Then, $a = r_c$, thus $r'_c \in S$, I_S is an interpretation of V^+. Thus, the rule that concerns c introduced in (i) is not in P^S. Thus, the rule that concerns c introduced in (iii) is the only one with head r_c. Thus, its body is in $CN(P^S_{K\cup A})$ and thus, since we have shown that $CN(P^S_{K\cup A}) \cap (V \setminus R_K) \subseteq S$, its body is also in S, which implies that $I_S \not\models c$. But, I_S is a model of $K \setminus CL(S \cap R_K^+)$ and $r_c \notin S$, which is impossible.

We finally show that *false* $\notin CN(P^S_{K\cup A})$. Suppose that *false* $\in CN(P^S_{K\cup A})$, then, there is a rule introduced in (ii) such that its body is in $CN(P^S_{K\cup A})$, thus, by (8), its body is also in S. But, this means that some clause in A is not satisfied by I_S, which is impossible as I_S is a model of A. The roof of *contradiction* $\notin CN(P^S_{K\cup A})$ follows from the fact that *false* $\notin CN(P^S_{K\cup A})$.

Since $CN(P^S_{K\cup A}) \subseteq V \cup \{false, contradiction\}$ and we have proved that $CN(P^S_{K\cup A}) \cap (V \setminus R_K) \subseteq S$ and $CN(P^S_{K\cup A}) \cap R_K \subseteq S$ and *false* $\notin CN(P^S_{K\cup A})$, *contradiction* $\notin CN(P^S_{K\cup A})$ it follows that $CN(P^S_{K\cup A}) \subseteq S$. □

Proposition 7 *Let $R \subseteq K$. Then, $(K \setminus R) \cup A$ is consistent iff there exists an answer set S of $P_{K\cup A}$ such that $CL(S \cap R_K^+) = R$.*

Proof We first show that if $(K \setminus R) \cup A$ is consistent then there exists an answer set S of $P_{K \cup A}$ such that $CL(S \cap R_K^+) = R$.

Suppose $(K \setminus R) \cup A$ is consistent and let m be an interpretation of $Atom(K \cup A)$ which satisfies $(K \setminus R) \cup A$. Then, let $S = \{a \mid a \in m\} \cup \{a' \mid \neg a \in m\} \cup \{r_c \mid c \in R\} \cup \{r'_c \mid c \in K \setminus R\}$. Let I_S be such that $I_S = \{a \mid a \in S\} \cup \{\neg a \mid a' \in S\}$. Clearly, I_S is an interpretation of V^+ which is a model of $(K \setminus R) \cup A$, therefore by Proposition 6, S is an answer set of $P_{K \cup A}$.

Ne now show that if there exists an answer set S of $P_{K \cup A}$ such that $CL(S \cap R_K^+) = R$ then $(K \setminus R) \cup A$ is consistent.

The proof is obvious and follows from Proposition 6. □

In order to prove the Proposition 8, for the sake of readability, we introduce the preferred *T-generators* of $P_{K \cup A}$ which are the sets of literals which are interpretations of T that lead to a preferred answer set of $P_{K \cup A}$.

- Let $A \subseteq V$ and $L \subseteq lit(V)$. We say that A *agrees with* L iff $pos(L) \subseteq A$ and $\forall \neg a \in neg(L)$, $a \notin A$.
- Let $I \subseteq lit(R_K)$ and $T \subseteq R_K$, we say that I is a preferred *T-generator* of $P_{K \cup A}$ iff
  - I is an interpretation of T and
  - there exists a preferred answer set S of $P_{K \cup A}$ such that S agrees with I.

We first prove the following lemma.

Lemma 1 *Let $\mathcal{I}$ be the set of all preferred R_K-generators of $P_{K \cup A}$ and $\mathcal{R}$ be the set of all prioritized removed sets of $K \cup A$. Then,*

(i) $\{CL(I \cap R_K^+) : I \in \mathcal{I}\} = \mathcal{R}$;
(ii) $|\mathcal{R}| = |\mathcal{I}|$.

Proof (i) We first prove that $\{CL(I \cap R_K^+) : I \in \mathcal{I}\} \subseteq \mathcal{R}$.

Let I be a preferred R_K-generator. By definition, there exists a preferred answer set S which agrees with I. By Proposition 7, $(K \setminus CL(S \cap R_K^+)) \cup A$ is consistent and clearly $S \cap R_K = I \cap R_K$, thus $S \cap R_K^+ = I \cap R_K^+$. Therefore $(K \setminus CL(I \cap R_K^+)) \cup A$ is consistent. Moreover, if there is another I' such that $(K \setminus CL(I' \cap R_K^+)) \cup A$ is consistent then $I \cap R_K^+$ is preferred to $I' \cap R_K^+$, because if not, S is not anymore a preferred answer set. Therefore is a prioritized removed set.

We now prove that $\mathcal{R} \subseteq \{CL(I \cap R_K^+) : I \in \mathcal{I}\}$.

Let $R \subseteq K$ such that R is a prioritized removed set, then $(K \setminus R) \cup A$ is consistent. By Proposition 7, there is an answer set S such that $CL(S \cap R_K^+) = R$. Moreover S is a preferred answer, because if not, R is not a prioritized removed set. Let $I = \{a : a \in S \cap R_K\} \cup \{\neg a : a \in R_K \setminus S\}$. Clearly I is a preferred R_K-generator and $S \cap R_K^+ = I \cap R_K^+$. Therefore $CL(I \cap R_K^+) = R$.

(ii) In order to prove that $|\mathcal{R}| = |\mathcal{I}|$ we show that there is a one to one correspondence between $\mathcal{I}$ and $\mathcal{R}$.

By definition the function f such that $f(I) = CL(I \cap R_K^+)$ is surjective. We have now to show that it is injective, that is $\forall I, I' \in \mathcal{I}$, if $I \neq I'$ then $CL(I \cap R_K^+) \neq CL(I' \cap R_K^+)$. If $CL(I \cap R_K^+) = CL(I' \cap R_K^+)$ then by the definition of CL we have $I \cap R_K^+ = I' \cap R_K^+$. Since I and I' are preferred R_K-generators of $P_{K \cup A}$ then there exists two preferred answer sets S and S' such that S agrees with I and S' agrees with I' and it follows that $S \cap R_K = I \cap R_K$ and $S' \cap R_K = I' \cap R_K$, therefore $S \cap R_K^+ = S' \cap R_K^+$. We now show that $S \cap R_K = S' \cap R_K$. Let I_S (resp. I'_S) be the interpretation of V corresponding to S (resp. S'). If $S \cap R_K^- \neq S' \cap R_K^-$ then there exists $a' \in S \cap R_K^-$ and $a' \notin S' \cap R_K^-$ thus $\neg a \in I_S$ but $\neg a \notin I'_S$ which is impossible. Therefore $I \cap R_K = I' \cap R_K$ and by definition of R_K-generator, $I = I'$ which contradicts the hypothesis. □

We now prove the Proposition 8.

Proposition 8 *Let K be a consistent and prioritized finite set of clauses and let A be a finite consistent set of clauses. R is a prioritized removed set of $K \cup A$ iff there exists a preferred answer set S of $P_{K \cup A}$ such that $CL(S \cap R_K^+) = R$.*

Proof Let $\mathcal{S}$ be the set of all preferred answer sets of $P_{K \cup A}$ and $\mathcal{R}$ be the set of all prioritized removed sets of $K \cup A$. We show that $\{CL(S \cap R_K^+) : S \in \mathcal{S}\} = \mathcal{R}$.

We first show that $\{CL(S \cap R_K^+) : S \in \mathcal{S}\} \subseteq \mathcal{R}$.

Let S be a preferred answer set of $P_{K \cup A}$. Let $R = CL(S \cap R_K^+)$, since S is an answer set, by Proposition 6, $(K \setminus R) \cup A$ is consistent. Suppose that R is not a prioritized removed set, since $R \subseteq K$ and $(K \setminus R) \cup A$ is consistent there exists $R' \subseteq K$ such that $(K \setminus R') \cup A$ is consistent and R' is preferred to R. As $(K \setminus R') \cup A$ is consistent, there exists an interpretation m of $Atom(K \cup A)$ such that m satisfies $(K \setminus R') \cup A$. Let $S' = \{a \in V : a \in m\} \cup \{a' \in V : \neg a \in m\} \cup \{r_c : c \in R'\} \cup \{r'_c : c \in K \setminus R'\}$. Let I'_S be such that $I'_S = \{a \mid a \in S'\} \cup \{\neg a \mid a' \in S'\}$. Clearly, I'_S is an interpretation of V^+ which satisfies $(K \setminus R') \cup A$, besides $R' = CL(S' \cap R_K^+)$. Thus, by proposition 7, S' is an answer set. Moreover, $R' = CL(S' \cap R_K^+)$ is preferred to $R = CL(S \cap R_K^+)$. And, $\forall\, i,\ 1 \leq i \leq n$, $|CL(S' \cap R_K^+) \cap K_i| = |CL(S' \cap R_{K_i}^+)| = |S' \cap R_{K_i}^+|$. Similarly, $\forall i, 1 \leq i \leq n$, $|CL(S \cap R_K^+) \cap K_i| = |S \cap R_{K_i}^+|$. Thus, S' is preferred to S and thus S is not a preferred answer set,which contradicts the hypothesis. Therefore, R is a prioritized removed set.

We now show that $\mathcal{R} \subseteq \{CL(S \cap R_K^+) : S \in \mathcal{S}\}$.

Let R be a prioritized removed set of $K \cup A$. Then, $(K \setminus R) \cup A$ is consistent. thus, there is an interpretation m of $Atom(K \cup A)$ which satisfies $(K \setminus R) \cup A$. Let $S = \{a \in V : a \in m\} \cup \{a' \in V : \neg a \in m\} \cup \{r_c : c \in R\} \cup \{r'_c : c \in K \setminus R\}$. Let I_S be such that $I_S = \{a \mid a \in S\} \cup \{\neg a \mid a' \in S\}$. Clearly,

I_S is an interpretation of V^+ which satisfies $(K \setminus R) \cup A$. Besides $R = CL(S \cap R_K^+)$. Therefore, by Proposition 7, S is an answer set. Suppose that S is not a preferred answer set. Then, there exists an answer set S' such that S' is preferred to S. And, as seen above, $\forall i,\ 1 \leq i \leq n$, $|CL(S \cap R_K^+) \cap K_i| = |S \cap R_{K_i}^+|$ and $|CL(S' \cap R_K^+) \cap K_i| = |S' \cap R_{K_i}^+|$. Thus, $CL(S' \cap R_K^+)$ is preferred to $CL(S \cap R_K^+) = R$. But, $CL(S' \cap R_K^+) \subseteq K$ and $(K \setminus CL(S' \cap R_K^+)) \cup A$ is consistent. Thus, R is not a prioritized removed set, which contradicts the hypothesis. Therefore S is a preferred answer set. □

Proofs of Sect. 5

In order to prove the Proposition 9 we first prove 3 lemmas.

The first lemma characterizes the set of subsets of literals of R_K which lead to preferred answer sets and which minimize the number of clauses to remove from each stratum.

Lemma 2 *Let $k \in \mathbb{N}$, $1 \leq k \leq n$, let L be a $R_{K_1 \cup \cdots \cup K_{k-1}}$-generator, and let $\mathcal{X}$ be a finite set of $R_{K_1 \cup \cdots \cup K_k}$-generators such that $\forall X, Y \in \mathcal{X}$, $|X \cap R_{K_k}^+| = |Y \cap R_{K_k}^+|$. Then, $X \in Prio(P_{K \cup A}, L, k, \mathcal{X})$ iff*

(i) *X is a $R_{K_1 \cup \cdots \cup K_k}$-generator,*
(ii) *$L \subset X$ or $X \in \mathcal{X}$,*
(iii) *for all $R_{K_1 \cup \cdots \cup K_k}$-generator X' such that $L \subset X'$, we have $|X \cap R_{K_k}^+| \leq |X' \cap R_{K_k}^+|$,*
(iv) *$\forall X' \in \mathcal{X}$, $|X \cap R_{K_k}^+| \leq |X' \cap R_{K_k}^+|$.*

Proof Let $\mathcal{A}_{Sml}$ denote the tree run by $Smodels(P_{K \cup A}, L)$ and $\mathcal{A}_{Prio}$ run by $Prio(P_{K \cup A}, L, k, \mathcal{X})$.

Let $X \in Prio(P_{K \cup A}, L, k, \mathcal{X})$, we first show that X satisfies (i), (ii), (iii), and (iv).

If X is initially a member of $\mathcal{X}$, then X satisfies (i) and (ii), otherwise, there is a leaf F of $\mathcal{A}_{Prio}$ such that $X = F' \cap lit(R_{K_1 \cup \cdots \cup K_k})$ and $Pos(F')$ is an answer set. Thus, X is consistent and covers $R_{K_1 \cup \cdots \cup K_k}$ and $Pos(F')$ agrees with X. Thus, X is a $R_{K_1 \cup \cdots \cup K_k}$-generator, and as L is the root of $\mathcal{A}_{Prio}$, we get $L \subset F'$, thus $L \subset X$, thus X satisfies (i) and (ii).

Suppose that X does not satisfy (iv). Then, there is $X' \in \mathcal{X}$ such that $|X' \cap R_{K_k}^+| < |X \cap R_{K_k}^+|$. Thus, initially $X \notin \mathcal{X}$, thus there is a leaf F such that $X = F' \cap lit(R_{K_1 \cup \cdots \cup K_k})$ and F' satisfies Condition (1). Consider the time when F' is testing (1). Clearly, it is impossible that $X' \in \mathcal{X}$ at this time, thus X' has been rejected before, thus Condition (2) has been satisfied at least once, thus there is $X'' \in \mathcal{X}$ such that $|X'' \cap R_{K_k}^+| < |X' \cap R_{K_k}^+| < |X \cap R_{K_k}^+| = |F' \cap R_{K_k}^+|$, thus F' does not satisfy (1), which is impossible, thus X satisfies (iv).

Suppose that X does not satisfy (iii). Then, there is a $R_{K_1 \cup \cdots \cup K_k}$-generator X' such that $L \subset X'$ and $|X' \cap R_{K_k}^+| < |X \cap R_{K_k}^+|$.

Case 1: X' has been added to $\mathcal{X}$. By Condition (3), X and X' are added only once. Thus two cases arise:
Case 1.1: X has been added before X'. Consider the time when X' is added to $\mathcal{X}$. As $X \in \mathcal{X}$ and $|X' \cap R_{K_k}^+| < |X \cap R_{K_k}^+|$, we have that (3) is satisfied and thus X is rejected, which is impossible.
Case 1.2: X' has been added before X. Then, initially $X \notin \mathcal{X}$, thus there is a leaf F such that $X = F' \cap lit(R_{K_1 \cup \cdots \cup K_k})$ and F' satisfies (1). But, when F' is testing (1), we have $X' \notin \mathcal{X}$, thus X' has been rejected before, thus (2) has been satisfied, thus $\exists X'' \in \mathcal{X}$ such that $|X'' \cap R_{K_k}^+| < |X' \cap R_{K_k}^+| < |X \cap R_{K_k}^+| = |F' \cap R_{K_k}^+|$, thus F' does nor satisfy (1), which is impossible.

Case 2: X' has not been added to $\mathcal{X}$. Let S be an answer set that agrees with X'. As $L \subset X$, we have that S agrees with L, thus there is a leaf F of A_{Sml} such that $S = Pos(F)$, thus such that $X' = F' \cap lit(R_{K_1 \cup \cdots \cup K_k})$. As X' has not been added to $\mathcal{X}$, there is an ancestor N of F which does not satisfy (1). Thus, $\exists X'' \in \mathcal{X}, |X'' \cap R_{K_k}^+| < |N \cap R_{K_k}^+|$ But, $|N \cap R_{K_k}^+| \leq |F' \cap R_{K_k}^+| = |X' \cap R_{K_k}^+| < |X \cap R_{K_k}^+|$. If $L \subset X''$, then we fall down on Case 1, which is impossible. Otherwise, we have that initially $X'' \in \mathcal{X}$ and thus X does not satisfy (iv), which is impossible. Thus, X satisfies (iii).

Suppose that X satisfies (i), (ii), (iii), and (iv), we now show that $X \in Prio(P_{K \cup A}, L, k, \mathcal{X})$.

Case 1: $L \subset X$. As X is a $R_{K_1 \cup \cdots \cup K_k}$-generator, there is an answer set S that agrees with X and thus that agrees with L. Thus, there is a leaf F of A_{Sml} such that $Pos(F') = S$, thus such that $X = F' \cap lit(R_{K_1 \cup \cdots \cup K_k})$.
Case 1.1: F has not been pruned. Then, X has been added to $\mathcal{X}$. Suppose that X has been rejected. Then, Condition (2) has been satisfied despite the presence of X, which is impossible by (iii), thus X has not been rejected and thus $X \in Prio(P_{K \cup A}, L, k, \mathcal{X})$.
Case 1.2: an ancestor N of F has been pruned. Suppose that this is due to (1). Then, there is a $R_{K_1 \cup \cdots \cup K_k}$-generator $X' \in \mathcal{X}$ such that $|X' \cap R_{K_k}^+| < |N \cap R_{K_k}^+| \leq |F' \cap R_{K_k}^+| = |X \cap R_{K_k}^+|$. On the one hand, if $L \subset X'$, then we get a contradiction by (iii). On the other hand, if initially $X' \in \mathcal{X}$, then we get a contradiction by (iv). Thus, the pruning of F is not due to (1). Thus, it is due to (3). But, then X has been added before $\mathcal{X}$ and we have seen in Case 1.1 that X cannot then be rejected, thus $X \in Prio(P_{K \cup A}, L, k, \mathcal{X})$.

Case 2: initially $X \in \mathcal{X}$. If X has been rejected, then clearly we get a contradiction by (iii), thus $X \in Prio(P_{K \cup A}, L, k, \mathcal{X})$. □

We following lemma characterizes what is the content of $\mathcal{X}$ after the k-th execution of the first loop in the algorithm *Rens* and we denote by $\mathcal{X}_k$ the value of the variable $\mathcal{X}$ just after the k-th execution of the first loop in the algorithm *Rens*.

Lemma 3 *Let $k \in \mathbb{N}$, $1 \leq k \leq n$. Then $X \in \mathcal{X}_k$ iff*

- *X is a $R_{K_1 \cup \cdots \cup K_k}$-generator and*
- *$\exists Y \in \mathcal{X}_{k-1}$, $Y \subset X$ and*
- *for all $R_{K_1 \cup \cdots \cup K_k}$-generator X' such that $\exists Y \in \mathcal{X}_{k-1}$, $Y \subset X'$, we have $|X \cap R^+_{K_k}| \leq |X' \cap R^+_{K_k}|$.*

Proof Le I be an interpretation, we denote by $S(I)$ the set of all clauses of K satisfied by I and we denote by $NS(I)$ the set of all clauses of K not satisfied by I. We start by induction on k and $R_{K^0} = \emptyset$.

For $k = 1$, $\mathcal{X}_1 = Prio(P_{K \cup A}, \emptyset, 1, \emptyset)$. As A is consistent, there is a interpretation M of $Atom(K \cup A)$ such that M is a model of A. Then, let $S = \{a \in V : a \in M\} \cup \{a' \in V : \neg a \in M\} \cup \{r_c : c \in NS(M)\} \cup \{r'_c : c \in K \setminus NS(M)\}$. Clearly, I_S the interpretation corresponding to S is an interpretation of V^+ and a model of $(K \cup A) \setminus NS(M)$, and $NS(M) = CL(S \cup R^+_K)$, Thus, by Proposition 6, S is an answer set. And obviously, S agrees with $\emptyset$. Thus, $\emptyset$ is a R_{K^0}-generator. Thus, by Lemma 2 and by the fact that $\mathcal{X}_0 = \emptyset$, we get that $X \in \mathcal{X}_1$ iff

(i) X is a R_{K_1}-generator;
(ii) $\exists Y \in \mathcal{X}_0$, $Y \subset X$;
(iii) for all R_{K^1}-generator X' such that $\exists Y \in \mathcal{X}_0$, $Y \subset X'$, we have that $|R^+_{K_1} \cap X| \leq |R^+_{K_1} \cap X'|$.

Now, let k such that $1 \leq k \leq n-1$ and we make the assumption (H1): $X \in \mathcal{X}_k$ iff

(i) X is a $R_{K_1 \cup \cdots \cup K_k}$-generator,
(ii) $\exists Y \in \mathcal{X}_{k-1}$, $Y \subset X$ and
(iii) for all $R_{K_1 \cup \cdots \cup K_k}$-generator X' such that $\exists Y \in \mathcal{X}_{k-1}$, $Y \subset X'$, we have $|R^+_{K_k} \cap X| \leq |R^+_{K_k} \cap X'|$.

We need the following notations: $\mathcal{Y}_0 = \{\emptyset\}$ and $\forall i$, $1 \leq i \leq n_k$, $\mathcal{Y}_i$ is the value of $\mathcal{Y}$ in the algorithm *Rens* just after the i-th execution of (B2) during the $k+1$-th execution (B1). In addition, $\forall i$, $1 \leq i \leq n_k$, we denote by X_i is the chosen element in the i-th execution of (I1) during the $k+1$-th execution of (B1).

We start a second induction within the first one. Clearly $\mathcal{Y}_1 = Prio(P_{K \cup A}, X_1, k+1, \emptyset)$. But, by (H1), X_1 is a R_{K_1}-generator. Thus, by Lemma 2 and by the fact that $\{X_1\} = \{X_i : 1 \leq i \leq 1\}$, we get that $X \in \mathcal{Y}_1$ iff

(i) X is a $R_{K_1 \cup \cdots \cup K_{k+1}}$-generator;
(ii) $\exists Y \in \{X_i : 1 \leq i \leq 1\}$ such that $Y \subset X$;
(iii) for all $R_{K_1 \cup \cdots \cup K_{k+1}}$-generator X' such that $\exists Y \in \{X_i : 1 \leq i \leq 1\}$, $Y \subset X'$, we have that $|R^+_{K_{k+1}} \cap X| \leq |R^+_{K_{k+1}} \cap X'|$.

Let $r \in \mathbb{N}$, $1 \leq r \leq n_k - 1$ where $n_k = |\mathcal{X}_k|$ and we make the assumption (H2): $X \in \mathcal{Y}_r$ iff

(i) X is a $R_{K_1 \cup \cdots \cup K_{k+1}}$-generator;
(ii) $\exists Y \in \{X_i : 1 \leq i \leq r\}$ such that $Y \subset X$;
(iii) for all $R_{K_1 \cup \cdots \cup K_{k+1}}$-generator X' such that $\exists Y \in \{X_i : 1 \leq i \leq r\}$, $Y \subset X'$, we have that $|R^+_{K_{k+1}} \cap X| \leq |R^+_{K_{k+1}} \cap X'|$.

Clearly, $\mathcal{Y}_{r+1} = Prio(P_{K \cup A}, X_{r+1}, k+1, \mathcal{Y}_r)$, and by (H2), $\mathcal{Y}_r$ is a finite set of $R_{K_1 \cup \cdots \cup K_{k+1}}$-generators such that $\forall X, Y \in \mathcal{Y}_r$, $|R^+_{K_{k+1}} \cap X| = |R^+_{K_{k+1}} \cap Y|$, and by (H1), X_{r+1} is a $R_{K_1 \cup \cdots \cup K_k}$-generator. Thus, by Lemma 2, we get that $X \in \mathcal{Y}_{r+1}$ iff

(i) X is a $R_{K_1 \cup \cdots \cup K_{k+1}}$-generator;
(ii) $X_{r+1} \subset X \vee X \in \mathcal{Y}_r$;
(iii) for all $R_{K_1 \cup \cdots \cup K_{k+1}}$-generator X' such that $X_{r+1} \subset X'$, we have $|R^+_{K_{k+1}} \cap X| \leq |R^+_{K_{k+1}} \cap X'|$;
(iv) $\forall X' \in \mathcal{Y}_r$, $|R^+_{K_{k+1}} \cap X| \leq |R^+_{K_{k+1}} \cap X'|$.

We now show that this entails $X \in \mathcal{Y}_{r+1}$ iff

(i′) X is a $R_{K_1 \cup \cdots \cup K_{k+1}}$-generator;
(ii′) $\exists Y \in \{X_i : 1 \leq i \leq r+1\}$ such that $Y \subset X$;
(iii′) for all $R_{K_1 \cup \cdots \cup K_{k+1}}$-generator X' such that $\exists Y \in \{X_i : 1 \leq i \leq r+1\}$, $Y \subset X'$, we have $|R^+_{K_{k+1}} \cap X| \leq |R^+_{K_{k+1}} \cap X'|$.

This will ends the second induction.

We first show that the top conditions entail the bottom conditions. Suppose X satisfies (i), (ii), (iii) and (iv). Then, X clearly satisfies (i′). If $X \in \mathcal{Y}_r$, then $\exists Y \in \{X_i : 1 \leq i \leq r\}$ such that $Y \subset X$. Thus, (ii) entails $\exists Y \in \{X_i : 1 \leq i \leq r\}$, $Y \subset X$ or $X_{r+1} \subset X$. Thus, X satisfies (ii′).

Suppose by contradiction that X does not satisfy (iii).

Case 1: $\mathcal{Y}_k = \emptyset$ and X does not satisfy (iii′). Then, $\exists X'$ a $R_{K_1 \cup \cdots \cup K_{k+1}}$-generator such that $\exists Y \in \{X_i : 1 \leq i \leq r+1\}$, $Y \subset X'$ and $|R^+_{K_{k+1}} \cap X'| < |R^+_{K_{k+1}} \cap X|$.

Case 1.1: $X_{r+1} \subset X'$. By (iii), we get $|R^+_{K_{k+1}} \cap X| \leq |R^+_{K_{k+1}} \cap X'|$, which is impossible.

Case 1.2: $\exists Y \in \{X_i : 1 \leq i \leq r\}$, $Y \subset X'$. Let $\mathcal{Z} = \{Z : Z$ is a $R_{K_1 \cup \cdots \cup K_{k+1}}$-generator and $\exists Y \in \{X_i : 1 \leq i \leq r\}$, $Y \subset Z\}$. Clearly, $X' \in \mathcal{Z}$, thus $\mathcal{Z} \neq \emptyset$. Thus, $\exists Z$ such that Z is a $R_{K_1 \cup \cdots \cup K_{k+1}}$-generator and $\exists Y \in \{X_i : 1 \leq i \leq r\}$, $Y \subset Z$ and $\forall Z' \in \mathcal{Z}$, $|R^+_{K_{k+1}} \cap Z| \leq |R^+_{K_{k+1}} \cap Z'|$. But, clearly $Z \in \mathcal{Y}_r$, thus $\mathcal{Y}_r \neq \emptyset$, which is impossible.

Case 2: $\mathcal{Y}_k \neq \emptyset$ and X does not satisfy (iii′). As above, $\exists X'$ a $R_{K_1 \cup \cdots \cup K_{k+1}}$-generator such that $\exists Y \in \{X_i : 1 \leq i \leq r+1\}$, $Y \subset X'$ and $|R^+_{K_{k+1}} \cap X'| < |R^+_{K_{k+1}} \cap X|$.

Case 2.1: $X_{r+1} \subset X'$. As above, we get a contradiction.

Case 2.2: $\exists Y \in \{X_i : 1 \leq i \leq r\}$, $Y \subset X'$. As $\mathcal{Y}_k \neq \emptyset$, $\exists X''$, $X'' \in \mathcal{Y}_r$. Thus, by (H2), $|R^+_{K_{k+1}} \cap X''| \leq |R^+_{K_{k+1}} \cap X'|$. But, by (iv), $|R^+_{K_{k+1}} \cap X| \leq |R^+_{K_{k+1}} \cap X''|$. Thus, $|R^+_{K_{k+1}} \cap X| \leq |R^+_{K_{k+1}} \cap X'|$, which is impossible.

Now, we show that the bottom conditions entail the top conditions. Suppose that X satisfies (i′), (ii′), and (iii′). Then, X clearly satisfies (i). By (iii′), X satisfies (iii).

Suppose by contradiction that X does not satisfy (ii). By

$X_{r+1} \not\subset X$ and (ii′), we get $\exists Y \in \{X_i : 1 \leq i \leq r\}$ such that $Y \subset X$. Suppose that X' is a $R_{K_1 \cup \ldots \cup K_{k+1}}$-generator such that $\exists Y \in \{X_i : 1 \leq i \leq r\}$, $Y \subset X'$. As X satisfies (iii′), we get $|R^+_{K_{k+1}} \cap X| \leq |R^+_{K_{k+1}} \cap X'|$. Thus, by (H2), $X \in \mathcal{Y}_r$, which is impossible. Thus, X satisfies (ii).

Suppose by contradiction that X does not satisfy (iv). Then, $\exists X' \in \mathcal{Y}_r, |R^+_{K_{k+1}} \cap X'| < |R^+_{K_{k+1}} \cap X|$. By (H2), X' is a $R_{K_1 \cup \ldots \cup K_{k+1}}$-generator such that $\exists Y \in \{X_i : 1 \leq i \leq r+1\}$, $Y \subset X'$. Thus, by (iii′) $|R^+_{K_{k+1}} \cap X| \leq |R^+_{K_{k+1}} \cap X'|$, which is impossible. Thus, X satisfies (iv).

Now that the second induction is shown, we finish the first one.

Clearly, $\mathcal{X}_{k+1} = \mathcal{Y}_{n_k}$. Thus, $X \in \mathcal{X}_{k+1}$ iff

(i) X is a $R_{K_1 \cup \ldots \cup K_{k+1}}$-generator;
(ii) $\exists Y \in \mathcal{X}_k$ such that $Y \subset X$;
(iii) for all $R_{K_1 \cup \ldots \cup K_{k+1}}$-generator X' s.t. $\exists Y \in \mathcal{X}_k$, $Y \subset X'$, we have $|R^+_{K_{k+1}} \cap X| \leq |R^+_{K_{k+1}} \cap X'|$. □

Lemma 4 *Let $k \in \mathbb{N}$, $0 \leq k \leq n$, let X, Y be two elements of $\mathcal{X}_k$, S be a preferred answer set, T be an answer set, and Z be a $R_{K_1 \cup \ldots \cup K_k}$-generator. Then,*

(i) *if T agrees with Z, then $X \cap R_{K_1 \cup \ldots \cup K_k} = S \cap R_{K_1 \cup \ldots \cup K_k}$;*
(ii) *$\forall i, 0 \leq i \leq k$, $X \cap lit(R_{K_1 \cup \ldots \cup K_i}) \in \mathcal{X}_i$;*
(iii) *$\forall i, 0 \leq i \leq k$, $|X \cap R^+_{K_i}| = |Y \cap R^+_{K_i}|$;*
(iv) *$\forall i,\ 0 \leq i \leq n$, $\{a : a \in S \cap R_{K_1 \cup \ldots \cup K_i}\} \cup \{\neg a : a \in R_{K_1 \cup \ldots \cup K_i} \setminus S\} \in \mathcal{X}_i$;*
(v) *if $\forall i, 0 \leq i \leq k$, $|X \cap R^+_{K_i}| = |Z \cap R^+_{K_i}|$, then $Z \in \mathcal{X}_k$.*

Proof Proof of (i). Direction: "⊆". As $X \subset lit(R_{K_1 \cup \ldots \cup K_k})$, we have $Pos(X) = X \cap R_{K_1 \cup \ldots \cup K_k}$. As S agrees with X, we have $Pos(X) \subseteq S$. Thus, $X \cap R_{K_1 \cup \ldots \cup K_k} \subseteq S$, thus $X \cap R_{K_1 \cup \ldots \cup K_k} \subseteq S \cap R_{K_1 \cup \ldots \cup K_k}$.

Direction: "⊇". Suppose that $S \cap R_{K_1 \cup \ldots \cup K_k} \not\subseteq X \cap R_{K_1 \cup \ldots \cup K_k}$. Then, let x be an atom such that $x \in S \cap R_{K_1 \cup \ldots \cup K_k}$ and $x \notin X \cap R_{K_1 \cup \ldots \cup K_k}$. As $x \in R_{K_1 \cup \ldots \cup K_k}$, we have $x \notin X$. But, X covers $R_{K_1 \cup \ldots \cup K_k}$, thus $\neg x \in X$. But, as S agrees with X, we have $x \notin S$, which is impossible.

Proof of (ii) by induction. Clearly, $X \cap lit(R_{K_1 \cup \ldots \cup K_k}) \in \mathcal{X}_k$. Let r be an integer between 1 and k, and suppose that $X \cap lit(R_{K_1 \cup \ldots \cup K_r}) \in \mathcal{X}_r$. By lemma 3, we have $\exists Y \in \mathcal{X}_{r-1}$, $Y \subset X \cap lit(R_{K_1 \cup \ldots \cup K_r})$. We show $Y = X \cap lit(R_{K_1 \cup \ldots \cup K_{r-1}})$.

Direction: "⊆". As Y is a $R_{K_1 \cup \ldots \cup K_{r-1}}$-generator, we have $Y \subset lit(R_{K_1 \cup \ldots \cup K_{r-1}})$, and as $Y \subset X$, we have $Y \subseteq X \cap lit(R_{K_1 \cup \ldots \cup K_{r-1}})$.

Direction: "⊇". Suppose that $X \cap lit(R_{K_1 \cup \ldots \cup K_{r-1}}) \not\subseteq Y$. Then, let x be a literal such that $x \in X \cap lit(R_{K_1 \cup \ldots \cup K_{r-1}})$ and $x \notin Y$. As $x \in lit(R_{K_1 \cup \ldots \cup K_{r-1}})$ and Y covers $R_{K_1 \cup \ldots \cup K_{r-1}}$, we have that $\neg x \in Y$, and as $Y \subset X$, we have that $\neg x \in X$. But, $x \in X$, thus X is inconsistent and thus there is no answer set that agrees with it. But, X is a $R_{K_1 \cup \ldots \cup K_k}$-generator, which is impossible. Thus, $Y = X \cap lit(R_{K_1 \cup \ldots \cup K_{r-1}})$ and thus $X \cap lit(R_{K_1 \cup \ldots \cup K_{r-1}}) \in \mathcal{X}_{k-1}$.

Proof of (iii). Clearly, $|X \cap R^+_{K_0}| = |Y \cap R^+_{K_0}|$. Let r be an integer between 1 and k, and suppose that $|X \cap R^+_{K_r}| \neq |Y \cap R^+_{K_r}|$. As $X \in \mathcal{X}_k$, we get, by (ii), that $X \cap lit(R_{K_1 \cup \ldots \cup K_r}) \in \mathcal{X}_r$. Similarly, $Y \cap lit(R_{K_1 \cup \ldots \cup K_r}) \in \mathcal{X}_r$. To increase readability we will denote by X^r the set $X \cap lit(R_{K_1 \cup \ldots \cup K_r})$ and by Y^r the set $Y \cap lit(R_{K_1 \cup \ldots \cup K_r})$. Clearly, $|X \cap R^+_{K_r}| = |X^r \cap R^+_{K_r}|$. Similarly, $|Y \cap R^+_{K_r}| = |Y^r \cap R^+_{K_r}|$. Now, suppose $|X^r \cap R^+_{K_r}| < |Y^r \cap R^+_{K_r}|$. As X^r is a $R_{K_1 \cup \ldots \cup K_r}$-generator such that $\exists X' \in \mathcal{X}_{r-1}$, $X' \subset X$, we get, by Lemma 3, that $Y^r \notin \mathcal{X}_r$, which is impossible. Similarly, if $|Y^r \cap R^+_{K_r}| < |X^r \cap R^+_{K_r}|$, then $X^r \notin \mathcal{X}_r$.

Proof of (iv) by induction. $\forall i \in \mathbb{N}$, $0 \leq i \leq n$, we denote by X^i the following set: $\{a : a \in S \cap R_{K_1 \cup \ldots \cup K_i}\} \cup \{\neg a : a \in R_{K_1 \cup \ldots \cup K_i} \setminus S\}$. We have that $X^0 = \emptyset$, thus $X^0 \in \mathcal{X}_0$.

Let i be an integer between 0 and $n-1$ and suppose that $X^i \in \mathcal{X}_i$. As X^{i+1} is a interpretation of $R_{K_1 \cup \ldots \cup K_{i+1}}$ and S agrees with X^{i+1}, we have that X^{i+1} is a $R_{K_1 \cup \ldots \cup K_{i+1}}$-generator. In addition, clearly $X^i \subset X^{i+1}$. Now, suppose that X' is a $R_{K_1 \cup \ldots \cup K_{i+1}}$-generator such that $\exists Y' \in \mathcal{X}^i$, $Y' \subset X$. We show $|X^{i+1} \cap R^+_{K_{i+1}}| \leq |X' \cap R^+_{K_{i+1}}|$, which entails, by Lemma 3, that $X^{i+1} \in \mathcal{X}_{i+1}$.

As X' is a $R_{K_1 \cup \ldots \cup K_{i+1}}$-generator, there is an answer set S' such that S' agrees with X'. Thus, by (i), we have $X' \cap R_{K_1 \cup \ldots \cup K_{i+1}} = S' \cap R_{K_1 \cup \ldots \cup K_{i+1}}$. In addition, X' is a $R_{K_1 \cup \ldots \cup K_{i+1}}$-generator, Y' is a $R_{K_1 \cup \ldots \cup K_{i+1}}$-generator and $Y' \subset X$, thus, clearly $Y' = X' \cap lit(R_{K_1 \cup \ldots \cup K_i})$. Thus, $Y' = X' \cap lit(R_{K_1 \cup \ldots \cup K_{i+1}}) \cap lit(R_{K_1 \cup \ldots \cup K_i})$, thus $Y' = S' \cap lit(R_{K_1 \cup \ldots \cup K_{i+1}}) \cap lit(R_{K_1 \cup \ldots \cup K_i})$, thus $Y' = S' \cap lit(R_{K_1 \cup \ldots \cup K_i})$. Thus, we have $\forall j, 0 \leq j \leq i$, $|Y' \cap R^+_{K_j}| = |S' \cap R^+_{K_j}|$ and clearly $X^i \cap R_{K_1 \cup \ldots \cup K_i} = S \cap R_{K_1 \cup \ldots \cup K_i}$, thus $\forall j, 0 \leq j \leq i$, $|X^i \cap R^+_{K_j}| = |S \cap R^+_{K_j}|$. But, $X^i \in \mathcal{X}^i$ et $Y' \in \mathcal{X}^i$, thus, by (iii), we have $\forall j, 0 \leq j \leq i$, $|X^i \cap R^+_{K_j}| = |Y' \cap R^+_{K_j}|$. Thus, $\forall j, 0 \leq j \leq i$, $|S \cap R^+_{K_j}| = |S' \cap R^+_{K_j}|$. In addition, we have that $|X' \cap R^+_{K_{i+1}}| = |S' \cap R^+_{K_{i+1}}|$. And, clearly $|X^{i+1} \cap R^+_{K_{i+1}}| = |S \cap R^+_{K_{i+1}}|$. Now, suppose $|X' \cap R^+_{K_{i+1}}| < |X^{i+1} \cap R^+_{K_{i+1}}|$. Then, we get $|S' \cap R^+_{K_{i+1}}| < |S \cap R^+_{K_{i+1}}|$, thus S' is preferred to S and thus S is not a preferred answer set, which is impossible. Thus, $|X^{i+1} \cap R^+_{K_1 \cup \ldots \cup K_{i+1}}| \leq |X' \cap R^+_{K_1 \cup \ldots \cup K_{i+1}}|$, thus $X^{i+1} \in \mathcal{X}_{i+1}$.

Proof of (v) by induction. $\forall r \in \mathbb{N}$, $0 \leq r \leq k$, we denote by X^r the following set: $X \cap lit(R_{K_1 \cup \ldots \cup K_r})$ and by Z^r the following set: $Z \cap lit(R_{K_1 \cup \ldots \cup K_r})$.

Clearly, $Z^0 \in \mathcal{X}_0$.

Let r be an integer between 0 and $k-1$. Suppose $Z^r \in \mathcal{X}_r$. As Z is a $R_{K_1 \cup \ldots \cup K_k}$-generator, we have that Z^{r+1} is a $R_{K^{r+1}}$-generator. In addition, $Z^r \subseteq Z^{r+1}$. Now, suppose that X' is a $R_{K^{r+1}}$-generator such that $\exists Y' \in \mathcal{X}_r$, $Y' \subset Z$. We show $|Z^{r+1} \cap R^+_{K_{r+1}}| \leq |X' \cap R^+_{K_{r+1}}|$ which entails, by Lemma 3, that $Z^{r+1} \in \mathcal{X}_{r+1}$.

As $X \in \mathcal{X}_k$, we get, by (ii), that $X^{r+1} \in \mathcal{X}_{r+1}$. Thus, $|X^{r+1} \cap R^+_{K_{r+1}}| \leq |X' \cap R^+_{K_{r+1}}|$. But, $|X^{r+1} \cap R^+_{K_{r+1}}| = |X \cap R^+_{K_{r+1}}| = |Z \cap R^+_{K_{r+1}}| = |Z^{r+1} \cap R^+_{K_{r+1}}|$. Thus, $|Z^{r+1} \cap R^+_{K_{r+1}}| \leq |X' \cap R^+_{K_{r+1}}|$, which ends the induction. Thus, $Z^k = Z \in \mathcal{X}_k$. □

Proposition 9 *Let $R \subseteq K$. Then, R is prioritized removed set of $K \cup A$ iff $R \in Rens(P_{K \cup A})$.*

Proof Let $\mathcal{R}$ be the set of all (prioritized) removed sets of $K \cup A$. It suffices to show $Rens(P_{K \cup A}) = \mathcal{R}$.
Direction: "$\subseteq$". Suppose that I is a preferred R_K-generator. Then, there is a preferred answer set S that agrees with I. We denote by X the following set: $\{a : a \in S \cap R_K\} \cup \{\neg a : a \in R_K \setminus S\}$. By Lemme 4 (iv), $X \in \mathcal{X}_n$. We show that $I = X$. Suppose that $L \not\subseteq X$. Then, let $x \in I$ such that $x \notin X$. As $I \subset lit(R_K)$, we have $x \in lit(R_K)$. But, X covers R_K, thus $\neg x \in X$. But, S agrees with I and with X, and one contains x, while the other contains $\neg x$, which is impossible. Similarly, if $x \notin I$, then we will get a contradiction. Thus, $I \in \mathcal{X}_n$.
Direction: "$\supseteq$". Suppose that $X \in \mathcal{X}_n$. Then, X is a R_K-generator, thus there is an answer set S such that S agrees with X. Suppose that S is not a preferred answer set. Then, there is an answer set S' that is preferred to S, thus $\exists i$, $1 \leq i \leq n$, $|S' \cap R^+_{K_i}| < |S \cap R^+_{K_i}|$ and $\forall j$, $1 \leq j < i$, $|S \cap R^+_{K_j}| = |S' \cap R^+_{K_j}|$. Let Y be the following set: $\{a : a \in S' \cap R_K\} \cup \{\neg a : a \in R_K \setminus S'\}$.
$\forall k, 0 \leq k \leq n$, we denote by X^k the following set: $X \cap lit(R_{K_1 \cup \cdots \cup K_k})$ and by Y^k the following set: $Y \cap lit(R_{K_1 \cup \cdots \cup K_k})$. By Lemma 4 (i), we have $X \cap R_K = S \cap R_K$. Thus,

(i) $\forall i, 0 \leq j \leq i-1, |X^{i-1} \cap R^+_{K_j}| = |S \cap R^+_{K_j}|$
(ii) $|X^i \cap R^+_{K_i}| = |S \cap R^+_{K_i}|$.

In addition, clearly $Y \cap R_K = S' \cap R_K$. Thus,

(i′) $\forall i, 0 \leq j \leq i-1, |Y^{i-1} \cap R^+_{K_j}| = |S \cap R^+_{K_j}|$
(ii′) $|Y^i \cap R^+_{K_i}| = |S' \cap R^+_{K_i}|$.

Thus, by (i) and (i′), we have $\forall j$, $0 \leq j \leq i-1$, $|X^{i-1} \cap R^+_{K_j}| = |Y^{i-1} \cap R^+_{K_j}|$. But, $X \in \mathcal{X}_n$ thus, by Lemma 4 (ii), $X^{i-1} \in \mathcal{X}_{i-1}$, thus, by Lemma 4 (v), $Y^{i-1} \in \mathcal{X}_{i-1}$. But, Y^i is a $R_{K_1 \cup \cdots \cup K_i}$-generator and $Y^{i-1} \subset Y^i$ and by (ii) and (ii′), we have $|Y^i \cap R^+_{K_i}| < |X^i \cap R^+_{K_i}|$. Thus, by Lemma 3, we have $X^i \notin \mathcal{X}_i$. But, $X^i = X \cap lit(R_{K_1 \cup \cdots \cup K_i})$, thus, by Lemma 4 (ii), we have $X^i \in \mathcal{X}_i$, which is impossible. Thus, S is a preferred answer set and thus I is a preferred R_K-generator. Thus, $\mathcal{X}_n$ is the set of all preferred R_K-generator. Consequently, by Proposition 8, $Rens(P_{K \cup A}) = \mathcal{R}$. □

Appendix B

B.1 Representing RSR as a SAT problem

This section is a summary of the section concerning the SAT encoding of RSR described in [2].

In order to represent the problem of revising the set of clauses K by the set of clauses A using RSR as a SAT problem, we transform the set K as follows.We use the transformation proposed by De Kleer for ATMS [17]. Each clause c of K is replaced by the formula $\phi_c \rightarrow c$, where ϕ_c is a new variable, called an hypothesis variable. If ϕ_c is assigned true then $\phi_c \rightarrow c$ is true iff c is true, this enforces c. On contrast if ϕ_c is assigned false then $\phi_c \rightarrow c$ is true whatever the truth value of c, the clause c is ignored. Let us denote $\mathcal{H}(K)$ the transformed set. The revision of K by A using RSR corresponds to the satisfiability of the set of clauses $\mathcal{H}(K) \cup A$. The models which minimize the number of falsified hypothesis variables ϕ_c then corresponds to the removed sets. The implementation can be performed using a state of the art SAT solver, like, for example, MINISAT [10].

B.2 The REM algorithm and the associated problem representation

The following sections are summaries of the corresponding sections in [38].

Encoding Concerning the REM algorithm, there is no special encoding. The algorithm takes as input the two sets of clauses K and A.

The REM algorithm The computation of removed sets consists in removing a clause from each element of the collection of minimal inconsistent subsets of $K \cup A$ *without listing all elements of this collection*. This strategy stems from the notion of hitting sets.

Definition 16 Let F be a collection of sets.

- A *hitting set* of F is a set $H \subseteq \bigcup_{S \in F} S$ such that $\forall S \in F$, $H \cap S \neq \emptyset$.
- H is a minimal hitting set of F if and only if H is a hitting set of F and $\forall H' \subset H$ with $H' \neq \emptyset$, H' is not a hitting set of F.

$\mathcal{N}(F)$ denotes the collection of minimal hitting sets of F. In order to compute the removed sets of $K \cup A$, we compute the hitting sets of the collection of the inconsistent subsets of $K \cup A$ denoted by $\mathcal{I}(K \cup A)$.[6] More precisely, since removed sets are subsets of K, we limit the computation to

[6] For a justification of the choice of $\mathcal{I}(\mathcal{K} \cup \mathcal{A})$ in place of the collection of *minimal* inconsistent subsets, see [38]. Juste note that it doesn't change the results.

those hitting sets containing only clauses of K.We then establish the correspondence between removed sets and hitting sets as follows:

Theorem 1 *Let $R \subseteq K$. R is a removed set of $K \cup A$ if and only if R is a minimal hitting set of minimal cardinality of the collection $\mathcal{I}(K \cup A)$ of the inconsistent subsets of $K \cup A$.*

The algorithm proposed by Reiter [30] and modified by Grenier, Smith and Wilkerson [36] allows us to compute the minimal hitting sets of a collection of sets $\mathcal{F}$ *without the extensive knowledge of* $\mathcal{F}$. This algorithm is based on the construction of a n-ary tree in a width first order (more exactly, this is a directed acyclic graph). Each node is labeled by an element of $\mathcal{F}$ or by $\surd$, which means that we have exhausted the collection $\mathcal{F}$. The tree is built as follows:

Level 0 The root is labeled with an element of $F_0 \in \mathcal{F}$. If $\mathcal{F}$ is empty, the root is labeled by $\surd$ (in this case the algorithm stops, there are no minimal hitting sets).

Level 1 Starting from the root, we build as many branches as there are elements in F_0. For each branch labeled by a $f_0^i \in F_0$, we produce a new node. This node is labelled by an element $F_1^i \in \mathcal{F}$ such that $f_0^i \notin F_1^i$. If there is no such F_1^i, the new node is labeled by $\surd$.

Level 2 For each node i in level one, labeled by F_1^i, we build as many branches as there are elements in F_1^i. Each branch is labeled by an element of F_1^i. For each branch f_1^i, we build a new node labeled by an element $F_2^j \in \mathcal{F}$ such that $\{f_0^i, f_1^j\} \not\subseteq F_2^j$. If there is no such F_2^j, the new node is labeled by $\surd$.

We repeat the same process for subsequent levels, until all branches end up with a node labeled by $\surd$. More formally:

Definition 17 Let $\mathcal{F}$ be a collection of sets, T is an HS-tree if and only if it is the smallest tree verifying the following properties:

1. its root is labeled by an element of $\mathcal{F}$. If $\mathcal{F}$is empty, its root is labeled by $\surd$;
2. if n is an element of T, let $H(n)$ be the set of the labels of the branches of the path going from the root of T to n. If n is labeled by $\surd$, then it has no successor in T. If n is labeled by a set $F \in \mathcal{F}$, then for each $f \in F$, n has a successor n_f in T, linked with n by a branch labeled by f. The label of n_f is a set $F' \in \mathcal{F}$ such that $F' \cap H(n_f) = \emptyset$, if such a set exists, otherwise n_f is labeled by $\surd$.

Such trees have two fundamental properties:

– if a node n is labeled by $\surd$, then $H(n)$ is a hitting set of $\mathcal{F}$;
– for each minimal hitting set S of $\mathcal{F}$, there exists a node $n \in T$ labeled by $\surd$ such that $H(n) = S$.

From this properties, Reiter, and after him Grenier, Smith and Wilkerson added several techniques which allows us to re-use already computed node labels and to limit the computation to the *minimal* hitting sets:

1. The tree is generated in a breadth first order. This avoids the generation of nodes n such that there exists a node n' at the same level with $H(n) = H(n')$.
2. Node re-used: if a node n is labeled by a set $F \in \mathcal{F}$, and if n' is a node such that $H(n') \cap F = \emptyset$, then n' is labeled by F. With this property, we avoid unnecessary accesses to $\mathcal{F}$.
3. Tree pruning:
   (a) If a node n is labeled by $\surd$ and there exists a node n'such that $H(n) \subseteq H(n')$, we then close the node n' without computing its successors.
   (b) If we are ready to generate a node n and there exists a node n' such that $H(n) = H(n')$, then we just link the branch to the node n'.
   (c) if n and n' are two nodes respectively labeled by F and F' and such that $F' \subset F$, then, for each $f \in F \setminus F'$, delete the branches starting from F and labeled by f, until a node having more than one ancestor is reached.

In our case, we want to generate minimal hitting sets of $\mathcal{I}(K \cup A)$. The HS-tree algorithm presented above needs a procedure which can generate on demand an element of this collection. For this purpose, we use a modified version of the Davis and Putnam procedure.

References

1. Alchourron C, Gärdenfors P, Makinson D (1985) On the logic of theory change: partial meet functions for contraction and revision. J Symb Log 50(2):510–530
2. Ben-Naim J, Benferhat S, Papini O, Würbel E (2004) An answer set programming approach of prioritized removed sets revision: application to GIS. In: Alferes J, Leite J (eds) Proceedings of JELIA'04, Lisbon, Portugal, September 2004. Lecture notes in artificial intelligence. Logics for AI. Springer, Berlin, pp 604–616
3. Benferhat S, Cayrol C, Dubois D, Lang J, Prade H (1993) Inconsistency management and prioritized syntax-based entailment. In: Proceedings of IJCAI93, pp 640–645
4. Brewka G, Niemelä I, Truszczynski M (2003) Answer set optimization. In: Proceedings of eighteenth international joint conference on artificial intelligence (IJCAI'03), pp 867–872
5. Buccafurri F, Faber W, Leone N (1999) Disjunctive logic programs with inheritance. In: Proceedings of the 1997 international conference on logic programming. MIT Press, Cambridge, pp 79–93
6. Cayrol C, Lagasquie-Schiex M-C (1994) On the complexity of nonmonotonic entailment in syntax-based approaches. In: Proceedings of the ECAI94 workshop on algorithms, complexity and commonsense reasoning
7. Cholewinski P, Marek V, Mikitiuk A, Truszczynski M (1999) Computing with default logic. Artif Intell 112:105–146

8. Delgrande JP, Schaub T, Tompits H (2003) A framework for compiling preferences in logic programs. Theory Pract Log Program 3(2):129–187
9. Doyle J (1979) A truth maintenance system. Artif Intell 12:231–272
10. Eén N, Sörensson N (2003) An extensible SAT-solver. In: Proceedings of 6th international conference on theory and applications of satisfiability testing
11. Eiter T, Gottlob G (1992) On the complexity of propositional knowledge base revision, updates and counterfactual. Artif Intell 57:227–270
12. Eiter T, Leone N, Mateis C, Pfeifer G, Scarcello F (1998) The kr system dlv: progress report, comparison and benchmarks. In: Proceedings of KR'98, pp 406–417
13. Fagin R, Ullman JD, Vardi MY (1983) On the semantic of updates in databases. In: Proceedings of the 2nd ACM symp. on principles of data base systems, pp 352–365
14. Gärdenfors P (1988) Knowledge in flux: modeling the dynamics of epistemic states. Bradford books. MIT Press, Cambridge
15. Gelfond M, Lifschitz V (1988) The stable model semantics for logic programming. In: Proceedings of the international conference on logic programming, pp 1070–1080
16. Katsuno H, Mendelzon A (1991) Propositional knowledge base revision and minimal change. Artif Intell 52:263–294
17. De Kleer J (1986) An assumption-based TMS. Artif Intell 28:127–162
18. De Kleer J (1990) Using crude probability estimates to guide diagnosis. Artif Intell 45:381–392
19. Lehman D (1995) Belief revision revisited. In: Proceedings of 14th int. joint conference on artificial intelligence, pp 1534–1539
20. Liberatore P, Schaerf M (1996) The complexity of model checking for belief revision and update. In: AAAI'96, pp 556–561
21. Linke T (2002) More on nomore. In: Proceedings of NMR'02
22. Nebel B (1991) Belief revision and default reasoning: syntax-based approach. In: Proceedings of knowledge representation, pp 417–427
23. Nebel B (1998) How hard is to revise a belief base? In: Handbook of defeasible reasoning and uncertainty management systems, vol 3, pp 77–145
24. Niemelä I (1998) Logic programs with stable semantics as a constraint programming paradigm. In: Proceedings of the workshop on computational aspect of non monotonic reasoning, pp 72–79
25. Niemelä I, Simons P (1997) An implementation of stable model and well-founded semantics for normal logic programs. In: Proceedings of LPNMR'97, pp 420–429
26. Osorio M, Zepeda C (2007) Properties of update sequences. In: Proceedings of 4th international workshop on answer set programming (ASP2007)
27. Papini O (1992) A complete revision function in propositional calculus. In: Neumann B (ed) Proceedings of ECAI92. Wiley, New York, pp 339–343
28. Raclot D, Puech C (1998) Photographies aériennes et inondation: globalisation d'informations floues par un système de contraintes pour définir les niveaux d'eau en zone inondée. Rev Int Géomatique 8(1):191–206
29. Rao P, Sagonas K, Swift, Warren DS, Friere J (1997) Xsb: A system for efficiently computing well-founded semantics. In: Proceedings of LPNMR'97, pp 430–440
30. Reiter R (1987) A theory of diagnosis from first principles. Artif Intell 32:57–95
31. Sakama C, Inoue K (2000) Prioritized logic programming and its application to commonsense reasoning. Artif Intell 123(1–2):185–222
32. Schaub T, Wang K (2001) A comparative study of logic programs with preference. In: Proceedings of seventeenth international joint conference on artificial intelligence (IJCAI'01), pp 597–602
33. Simons P (2000) Extending and implementing the stable model semantics. PhD Thesis, Helsinki University of Technology
34. Simons P, Niemelä I, Soininen T (2002) Extending and implementing the stable model semantics. Artif Intell 138(1–2):181–234
35. Sombe L (1994) A glance at revision and updating in knowledge bases. Int J Intell Syst 9:1–27
36. Wilkerson RW, Greiner R, Smith BA (1989) A correction to the algorithm in Reiter's theory of diagnosis. Artif Intell 41:79–88
37. Williams MA, Williams D (1997) A belief revision system for the world wide web. In: Proceedings of the IJCAI workshop of the future of artificial intelligence and the Internet, pp 39–51
38. Würbel E, Jeansoulin R, Papini O (2000) Revision: an application in the framework of gis. In: Cohn AG, Giunchiglia F, Selman B (eds) Proceedings of the seventh international conference about principles of knowledge representation and reasoning, KR2000, Breckenridge, Colorado, USA, April 2000. Morgan Kaufmann, San Mateo, pp 505–516
39. Würbel E, Jeansoulin R, Papini O (2001) Spatial information revision: a comparison between 3 approaches. In: Proceedings of the sixth European conference on symbolic and quantitative approaches to reasoning with uncertainty, ECSQARU 2001, Toulouse, France. Springer, Berlin, pp 454–465
40. Zang Y, Foo N (1997) Answer sets for prioritized logic programs. In: Proceedings of the 1997 international logic symposium. MIT Press, Cambridge, pp 69–83

Salem Benferhat (CRIL-CNRS, Centre de Recherche en Informatique de Lens) is a professor at University of Artois, France. He received a PhD thesis degree (1994) and the "Habilitation à Diriger des Recherches" (2000) from University Paul Sabatier, Toulouse, France. He is the author or the co-author of a large number of technical papers on uncertainty modelling, knowledge representations and applications. He is in program committe of several international conferences such that AAAI, ECAI, KR, ECSQARU, UAI, etc. He also co-organized several national and international conferences such as ECSQARU'01 (Sixth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty), NMR'02 (ninth Intl. Workshop on Non-Monotonic Reasoning NMR'2002) or LFA'08 (Rencontres francophones sur la Logique Floue et ses Applications). His current research interests are in knowledge representation, uncertainty handling, data fusion, preference modelling, possibilistic networks, causality, non-classical logics, plausible and non-monotonic reasoning with applications to computer security.

Jonathan Ben-Naim (IRIT CNRS) got a PhD in Computer Science in 2006 at the University of Aix-Marseille 1. His thesis is in the areas of reasoning in the presence of uncertain and incomplete information. He characterized by syntactical properties certain families of non-monotonic or paraconsistent logics defined semantically. From October 2006 to September 2007, he was a post-doctoral researcher at the University of Luxembourg, where he was investigating the field of Trust and Reputation. Since October 2007, he is a CNRS researcher at the university of Toulouse 3, and is working on the development and the axiomatization of trust or reputation systems. He participated in the ESPRIT project REVIGIS IST-1999-14189 (2000–2004) (revision of geographical information), and he is currently participating in the ANR-SETIN project ForTrust (models of trust and reputation).

Odile Papini (LSIS UMR CNRS 6168) is professor in computer science at the school of engineering ESIL at the University "de la Méditerranée" in Marseilles (FRANCE). She defended a PhD in applied mathematics in 1984 at the University of Provence (Marseilles) and the "Habilitation à Diriger des Recherches" in computer science in 1997 at the University "de la Méditerranée". She is involved in research in Artificial Intelligence since more than 15 years and her works are regularly

published in national and international conferences and journals in Artificial Intelligence. She has been involved in the European projects REVIGIS IST-1999-14189 (2000–2004) and VENUS FP6-2005-IST-5 (2006–2009). She is member of the scientific committee of several national and international conferences in Artificial Intelligence. She co-managed the Pluridisciplinary Thematic Network (RTP 11) "information and intelligence: reasoning and deciding" of department STIC of CNRS from 2002 to 2004. Her research interests focus on knowledge representation, formalisation and implementation of belief change operations with specific application to spatial information.

Eric Würbel (LSIS UMR CNRS 6168) is assistant professor at the University of Toulon. He defended a PhD in computer science in 2000 at the University of Provence (Marseilles). He is involved in Artificial Intelligence research since then, concentrating on the fields of belief revision and belief fusion. He is particularly focusing on implementation problems of these operations. This leads him to also investigate in the field of Answer Set Programming. His work is regularly published in international conferences. He has been involved in the REVIGIS IST-1999-14189 (2000–2004) european project and actually participates to the VENUS FP6-2005-IST-5 (2006–2009) european project.