An Answer Set Programming Encoding of Prioritized Removed Sets Revision: Application to GIS

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

An Answer Set Programming Encoding of Prioritized Removed Sets Revision: Application to GIS

Jonathan Ben-Naim[3], Salem Benferhat[2], Odile Papini[1], and Eric Würbel[1]

[1] Laboratoire SIS, université du Sud Toulon -Var. BP 132. 83957 La Garde Cedex.
{papini, wurbel}@univ-tln.fr
[2] CRIL-CNRS, université d'Artois. Rue Jean Souvraz. 62307 Lens Cedex. France.
benferhat@cril.univ-artois.fr
[3] LIF-CNRS, CMI technopôle de Château Gombert. 13353. Marseille cedex 13. France.
jbennaim@lif.univ-mrs.fr

Abstract. Geographical information systems are one of the most important application areas of belief revision. Recently, Würbel and colleagues [32] 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.

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. Different strategies have been proposed for performing revision [30], [22]. Most of the revision approaches have been developed at the theoretical level, except few applications [31] and it turns out that in the general case the theoretical complexity of revision is high [8] [16]. An example of belief revision system is Removed Sets Revision which has been proposed in [21], [13], [15] 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 [32] 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 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 [10]. 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 [11] 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. Morever, several efficient systems have been developed [9], [4], [24], [20], [17]. 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 [33], [32] 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 [25] 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 [7]. 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 paper is organized as follows. Section 2 gives a refresher on Removed Sets Revision. Section 3 presents 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. In section 6 we perform an experimental study which illustrates the approach on a real 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. Section 7 concludes the paper.

2 Background

2.1 Removed Sets Revision

We briefly recall the Removed Sets Revision [32] which deals with the revision of a set of propositional formulas by a set of propositional formulas. Let K and A be finite sets of clauses. The Removed Sets Revision focuses on the minimal subsets of clauses to remove from K, called *removed sets*[21], in order to restore consistency of $K \cup A$. More formally:

Definition 1. *Let K and A be two sets of clauses such that $K \cup A$ is inconsistent. R a subset of clauses of K, is a* removed set *of $K \cup A$ iff (i) $(K \cup A)\backslash R$ is consistent; $ii)$ $\forall R' \subseteq K,\ R' \neq R$, if $(K \cup A)\backslash R'$ is consistent then $\mid R \mid < \mid R' \mid$*[1].

It can be checked that if R is a removed set then $(K \cup A)\backslash R$ is a so-called cardinality-based maximal consistent subbase of $(K \cup A)$ [1], [13], [15]. Würbel et al.[32] adapted the Reiter's algorithm for diagnosis stemming from hitting sets [25] and implemented this adaptation to efficiently compute the removed sets. Note also that there are several recent compilations and implementations of removed sets revision, like the ones proposed in [2], [18], [5]. However, in our application, we need to compute all preferred removed sets which is not possible with these recent implementations.

2.2 Removed Sets Revision Translated into a SAT Problem

The Removed Set Revision can be translated into a SAT problem using the transformation proposed by De Kleer for ATMS [14]. Each clause c of K is replaced by the formula $\phi_c \rightarrow c$, where ϕ_c is a new variable, called 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 $\mathcal{H}(K)$ be the transformed set of clauses. The Removed Set Revision of K by A corresponds to the problem of looking for models of the set of clauses $\mathcal{H}(K) \cup A$ which minimize the number of falsified hypothesis variables ϕ_c. This leads to the definition of a preference relation between interpretations. Let first introduce some notations. H_K denotes the set of hypothesis variables, i. e. $H_K = \{\phi_c \mid \phi_c \rightarrow c \in \mathcal{H}(K)\}$ and let ω be an interpretation, $NI(\omega)$ denotes the number of hypothesis variables that are falsified by the interpretation ω, i. e. $NI(\omega) = \mid \{\phi_c \in H_K \mid \omega \not\models \phi_c\} \mid$[2].

Definition 2. *Let ω be an interpretation, ω is a H_K-preferred model of $\mathcal{H}(K) \cup A$ iff (i) $\omega \in Mod(\mathcal{H}(K) \cup A)$; (ii) $\forall \omega' \in Mod(\mathcal{H}(K) \cup A)$, $\omega' \neq \omega$, $NI(\omega) \leq NI(\omega')$.*

[1] $\mid R \mid$ denotes the number of clauses of R.

[2] For the sake of simplicity, we identify an hypothesis variables and a propositional formula.

The link between removed sets and models of $\mathcal{H}(K) \cup A$ is made by the following definition which assign each removed set R a set of models of $\mathcal{H}(K) \cup A$ denoted by $\mathcal{M}_R$.

Definition 3. *Let R be a removed set of $K \cup A$ the set of models of $\mathcal{H}(K) \cup A$ generated by R, denoted by $\mathcal{M}_R$, is a subset of $Mod(\mathcal{H}(K) \cup A)$ such that (i) $\forall c \in R$, $M \not\models \phi_c$ where $M \in \mathcal{M}_R$; (ii) $\forall c \in \mathcal{H}(K) \cup A \backslash R$, $M \models \phi_c$ where $M \in \mathcal{M}_R$.*

And the following proposition holds:

Proposition 1. *$\forall M \in \mathcal{M}_R$. R is a removed set iff M is a H_K-preferred model of $\mathcal{H}(K) \cup A$.*

Performing Removed Set Revision of K by A amounts to looking for the H_K-preferred models of $\mathcal{H}(K) \cup A$. This can be achieved using a SAT-solver. In order to compare different implementations of Removed Set Revision we used the SAT-solver MiniSat[7] which is a simplified version of the solver SATZOO that won the last SAT 2003 competition.

3 Prioritized Removed Sets Revision

We now present the Prioritized Removed Set Revision (PRSR) which generalizes the Removed Set Revision presented in section 2.1 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. $K = K_1 \cup \ldots \cup K_n$, 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 are the most prioritary beliefs in K, and K_n contains the ones which are the least prioritary in K [1], see also [13].

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

Prioritized removed sets are now defined as follows:

Definition 5. *Let K be a consistent and prioritized finite set of clauses and let A be a consistent finite set of clauses such that $K \cup A$ is inconsistent. R, a subset of clauses of $K \cup A$, is a prioritized removed set iff (i) $R \subseteq K$; (ii) $(K \cup A) \backslash R$ is consistent; (iii) $\forall R' \subseteq K$, if $(K \cup A) \backslash R'$ is consistent then R' is not preferred to R.*

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 [19], 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 $(K\cup A)\backslash R$ is consistent. For each clause c of K, we introduce a new atom denoted by r_c and V denotes 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 $(K\cup A)\backslash R$ 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 $(K\cup A)\backslash R$. 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 $(K\cup A)\backslash C_i$ 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: $r_c \leftarrow b_0, \cdots, b_n, b'_{n+1}, \cdots, b'_m$.

The steps (i) and (ii) are very similar to the ones proposed by Niemela, but the third one (iii) is new and is introduced for revision.

We denote by R_K the set $R_K = \{r_c \mid c \in K\}$ and R_K^+ (resp. R_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, $\forall r_c \in R_K^+,\ CL(r_c) = c$. The following result holds.

Proposition 2. *Let K be a consistent and prioritized finite set of clauses and let A be a finite consistent set of clauses. S is an answer set of $P_{K\cup A}$ iff $(K\cup A)\backslash CL(S\cap R_K^+)$ is consistent.*

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 $K = K_1 \cup \ldots \cup K_n$. For $1 \leq i \leq n$, 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 6. *Let K be a consistent and prioritized finite set of clauses. Let S and S' two sets of literals. S is preferred to S' iff*

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

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

Definition 7. *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) $\forall S'$ an answer set of $P_{K \cup A}$, S' is not preferred to S .*

The following result generalizes proposition 2.

Proposition 3. *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 such that $CL(S \cap R^+_K) = R$.*

In order to get a one to one correspondence between preferred answer sets and prioritized removed sets, instead of computing the set of preferred answer sets of $P_{K \cup A}$ we compute $\mathcal{X}$ the set of subsets of literals of R_K which are interpretations of R_K and that lead to preferred answer sets. More formally: $\mathcal{X} = \{X$ an interpretation of $R_K\ |\ \exists S$, a preferred answer set, such that $X \cap R_K = S \cap R_K\}$.

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 [12], [20], [28]. This is achieved using two algorithms. The first algorithm, Prio, is an adaptation of the smodels system algorithm which computes 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. The second algorithm, Rens, computes the prioritized removed sets of $K \cup A$, applying the principle of minimal change defined in 5 for PRSR, that is, stratum by stratum.

5.1 Prio: An Adaptation of Smodels System

Let $K = K_1 \cup \ldots \cup K_n$. Consider the stratum k. Let L be a subset of literals which is an interpretation of $R_{K_1 \cup \ldots \cup K_{k-1}}$ leading to an answer set and let $\mathcal{X}$ be the set of subsets of literals which are interpretations of $R_{K_1 \cup \ldots \cup K_k}$ leading to an answer set and such that they remove the same number of clauses from K_k. More formally: $\forall X, Y \in \mathcal{X}$, $|\ X \cap R^+_{K_1 \cup \ldots \cup K_k}\ | = |\ Y \cap R^+_{K_1 \cup \ldots \cup K_k}\ |$. The algorithm $Prio(P_{K \cup A}, L, k, \mathcal{X})$ returns the sets of literals which are interpretations of $R_{K_1 \cup \ldots \cup K_k}$ 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 Prio algorithm 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 than 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 $L' \cap Lit(R_{K_1\cup\ldots\cup K_k})$ is returned else $L' \cap Lit(R_{K_1\cup\ldots\cup K_k})$ 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 $R_{K_1\cup\ldots\cup K_k}$ that minimize the number of clauses to remove from K_k and strarts again with $L' \cup \{\neg a\}$.

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

The main adaptations of the original smodels algorithm consist in: **(1)** avoiding all the subsets of literals of $R_{K_1\cup\ldots\cup K_k}$ leading to an answer set which removes more clauses from K_k than those in $\mathcal{X}$; **(2)** not computing several times the same subsets of literals of $R_{K_1\cup\ldots\cup K_k}$ leading to an answer set; **(3)** comparing each new subset of literals of $R_{K_1\cup\ldots\cup K_k}$ leading to an answer set with the elements of $\mathcal{X}$, if the new subset 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 algorithm which computes the prioritized removed sets of $K \cup A$. The idea is to proceed stratum by stratum using the 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 $R_{K_1 \cup \ldots \cup K_k}$ leading to an answer set which minimize the number of clauses to remove from K_k. The algorithm is the following:

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

And the following proposition holds.

Proposition 4. *Let K be a consistent and prioritized finite set of clauses and let A be a finite consistent set of clauses. R is prioritized removed set of $K \cup A$ iff $R \in Rens(P_{K \cup A})$.*

6 Application in the Framework of GIS

6.1 Description of the Application

The aim of the application is to assess water height at different locations in a flooded valley. The valley is segmented into compartments in which the water height can be considered as constant. We want to assess a minimum/maximum interval of water height for each compartment in the valley. We have two sources of information about these compartments (aside from the knowledge of their geographical layout), see figure 1.

The first source of information (S_2) is a set of hydraulic relations between neighbouring compartments. This source is incomplete (not all neighbouring compartments are connected) and quite certain. The second source of information (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 is uncertain. For more details see [23] and [32].

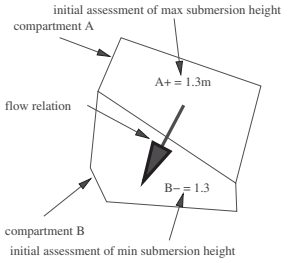

Fig. 1. Visual description of the sources of information in the flooding application.

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 heights. In the following, we denote by S_1 the set of clauses describing the initial assessments.

Concerning hydraulic relations, we have seen that they are expressed in terms of inequalities on the bounds of the water height. They are translated into binary negative clauses representing the exluded tuples of values. In the following, we denote by S_2 the set of clauses containing the clauses representing the hydraulic relations and the variable descriptions. S_1 is consistent, and S_2 is consistent, but $S_1 \cup S_2$ can be inconsistent. We want 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 1. 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 : (a) clauses describing the variables,

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

and (b) 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, $S_1 = \left\{A_2^+, B_3^+\right\}$. 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 section 4.1.

Example 2. Considering the previous example, the encoding is as follows: The generation rules for each propositional variables and each new atom r_c: (**i**) $A_1^+ \leftarrow not\ A_1^{'+}$, $A_1^{'+} \leftarrow not\ A_1^+$, etc. One rule for each clause of S_2. The translation of the set S_2 begins as follows : (**ii**) $false \leftarrow not\, A_1^+, not\ A_2^+, not\ A_3^+$, $false \leftarrow A_1^+, A_2^+$, 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: (**iii**) $r_{A_2^+} \leftarrow A_2^{'+}, r_{B_3^+} \leftarrow B_3^{'+}$

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 1GHz and equipped with 1GB of RAM.

Comparison between ASP encoding and REM algorithm. We first compare our ASP encoding to the REM algorithm presented by Würbel and al. in [32] which computes the removed sets by using a modification of Reiter's algorithm for the computation of minimal hitting sets.

Due to the lack of space we do not reproduce the tests provided in [32] by the REM algorithm, we just recall that REM was only able to handle 25 compartments (see [32] for more details). We compared ASP encoding to REM algorithm and we observed that until 20 compartments, the two approaches behave similarly. However, from 25 compartments, the answer set approach is significally better than REM approach. Moreover ASP encoding can handle the whole area of 120 compartments.

Comparison between ASP encoding and SAT encoding. We now compare the ASP encoding to a SAT encoding implemented with an efficient SAT-solver, MiniSat. The test deals with an increasing number of compartments from ten (210 variables, 2387 clauses) to sixty four compartments (1381 variables, 18782 clauses). The aim of this test is to compare the two approaches performances on the application and to identify their limits. Ten tests have been performed for a same number of compartments and an average running time on the ten tests is given.

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 significally 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. This is illustrated in figure 2.

Benefit of adding priorities. 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. Using a stratification of S_1 table 2 shows that Rens significally reduces the running time.

In the flooding application we have to deal with an area consisting of 120 compartments and the stratification mentioned above is useful to deal with the whole area. Using the stratification table 2 shows that Rens can deal with the whole area with a reasonable running time.

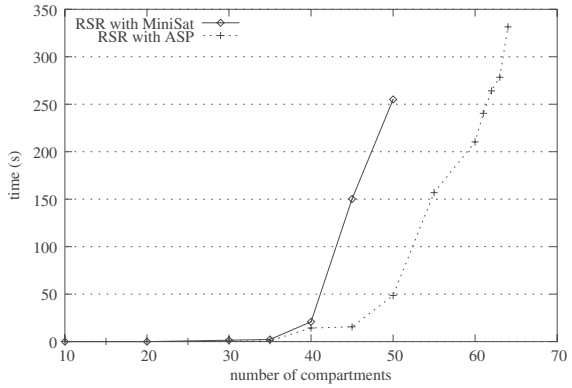

Fig. 2. Comparison between ASP encoding and SAT encoding in the flooding application.

Table 1. Gains induced by Rens.

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

Table 2. Gains induced by Rens on an area containing 120 compartments.

# 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 Concluding Discussion

This paper generalized Removed Sets Revision to prioritized belief bases (Prioritized Removed Sets Revision) and shows 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 order to compute answer sets corresponding to prioritized removed sets we introduced the notion of preferred answer set, an interesting question to investigate is how to directly encode this notion of preference in the logic program in order to get a direct one to one correspondence between answer sets and prioritized removed sets.

In [29], Extended language of smodels has been proposed as well as optimization statements for the smodels system. In PRSR we use cardinality constraints for encoding the data more compactly, however a first experimentation shows that the use of the

optimization statements like the minimize statement is not suitable because all answer sets are first generated then the required ones are filtered according to the minimize statement. The adaptation of smodels avoids the generation of all answer sets and the Rens algorithm partition the set of answer sets in classes such that only one answer set is computed by class.

A future work will compare our approach to other extensions of ASP. Among them, Prioritized Logic Programming (PLP) which deals with preferences among literals [26]. Answer Set Optimization approach (ASO) [3], which uses two different logic programs, the first one generates all the possible answer sets and the context dependent preferences are described in the second one. Several approaches have been proposed for dealing logic programs with preferences where the preference relation is a preference among rules like in [6]. A comparative study [27] has shown that these approaches can be characterized in terms of fixpoints, order preservation and translation into standard logic programs. 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.

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

References

1. S. Benferhat, Cayrol C, D. Dubois, J. Lang, and H. Prade. Inconsistency management and prioritized syntax-based entailment. In *Proceedings of IJCAI93*, pages 640–645, 1993.
2. S. Benferhat, S. Kaci, D. Leberre, and M. Williams. Weakening Conflicting Information for Iterated Revision and Knowledge Integration. In *Proceedings of Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 109–115, 2001.
3. G. Brewka, I. Niemelä, and M. Truszczynski. Answer Set Optimization. In *Proceedings of Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 867–872, 2003.
4. P. Cholewinski, V. Marek, A. Mikitiuk, and M. Truszczynski. Computing with default logic. *Artificial intelligence*, 112:105–146, 1999.
5. S. Coste-Marquis and P. Marquis. Knowledge compilation for circumscription and closed-world reasoning. *Logic and Computation*, 11:579–607, 2001.
6. J. P. Delgrande, T. Schaub, and H. Tompits. A Framework for Compiling Preferences in Logic Programs. *To appear in Theory and Practise of Logic Programming*, 2004.
7. N. Eén and N. Sörensson. An Extensible SAT-solver. In *Proceedings of 6th International Conference on Theory and Applications of Satisfiability Testing*, 2003.
8. T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactual. *Artificial Intelligence*, 57:227–270, 1992.
9. T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. the kr system dlv: progress report, comparison and benchmarks. In *Proceedings of KR'98*, pages 406–417, 1998.
10. R. Fagin, J. D. Ullman, and M. Y. Vardi. On The Semantic of Updates in Databases. In *Proceedings of the 2nd ACM Symp. on Principles of Data Base Systems*, pages 352–365, 1983.
11. P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. Bradford Books. MIT Press, Cambridge, 1988.

12. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings of the International Conference on Logic Programming*, pages 1070–1080, 1988.
13. De Kleer J. Using crude probability estimates to guide diagnosis. *Artificial Intelligenc*, 45:381–392, 1990.
14. J. De Kleer. An assumption-based TMS. *Artificial Intelligence*, 28:127–162, 1986.
15. D. Lehmann. Belief revision revisited. In *Proceedings of 14th Int. Joint Conference on Artificial Intelligence*, pages 1534–1539, 1995.
16. Paolo Liberatore and Marco Schaerf. The complexity of model checking for belief revision and update. In *AAAI'96*, pages 556–561, 1996.
17. T. Linke. More on nomore. In *Proceedings of NMR'02*, 2002.
18. P. Marquis and N. Porquet. Resource-bounded paraconsistent inference. *Annals of Mathematics and Artificial Intelligence*, 39:349–384, 2003.
19. I. Niemelä. Logic programs with stable semantics as a constraint programming paradigm. In *Proceedings of the workshop on computational Aspect of Non Monotonic Reasoning*, pages 72–79, 1998.
20. I. Niemelä and P. Simons. An implementation of stable model and well-founded semantics for normal logic programs. In *Proceedings of LPNMR'97*, pages 420–429, 1997.
21. Odile Papini. A complete revision function in propositionnal calculus. In B. Neumann, editor, *Proceedings of ECAI92*, pages 339–343. John Wiley and Sons. Ltd, 1992.
22. Odile Papini. Knowledge base revision. *The Knowledge Engineering Review*, 15(4):339–370, 2000.
23. D. Raclot and C. Puech. Photographies aériennes et inondation : globalisation d'informations floues par un système de contraintes pour définir les niveaux d'eau en zone inondée. *Revue internationale de géomatique*, 8(1):191–206, Février 1998.
24. P. Rao, K. Sagonas, Swift, D. S. Warren, and J. Friere. Xsb: A system for efficiently computing well-founded semantics. In *Proceedings of LPNMR'97*, pages 430–440, 1997.
25. Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.
26. C. Sakama and K. Inoue. Prioritized logic programming and its application to commonsense reasoning. *Artificial Intelligence*, 123(1-2):185–222, 2000.
27. T. Schaub and K. Wang. A Comparative Study of Logic Programs with Preference. In *Proceedings of Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 597–602, 2001.
28. P. Simons. *Extending and implementing the stable model semantics*. PhD thesis, Helsinky university of technology, 2000.
29. P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138(1-2):181–234, 2002.
30. Lea Sombe. *Revision and updating in knowledge bases*. International Journal of intelligent Systems. J. Wiley, New York, 1994.
31. M. A. Williams and D. Williams. A belief revision system for the world wide web. In *Proceedings of the IJCAI workshop of the Future of Artificial Intelligence and the Internet*, pages 39–51, 1997.
32. Eric Würbel, Robert Jeansoulin, and Odile Papini. Revision : An application in the framework of gis. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *Proceedings of the Seventh Internationnal Conference about Principles of Knowledge Representation and Reasoning, KR2000*, pages 505–516, Breckenridge, Colorado, USA, April 2000. KR, inc., Morgan Kaufmann.
33. Eric Würbel, Robert Jeansoulin, and Odile Papini. 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*, pages 454–465, Toulouse, France 2001. Springer Verlag.