Revision Programming with Preferences

by

Inna Pivkina, Enrico Pontelli, Tran Cao Son
Outline

1. Basic concepts of revision programming (RP).

2. Two approaches to express preferences
   (a) Control program
      - Example
      - Syntax and Semantics
      - Properties
   (b) Soft revision rules with weights
      - Examples
      - Definitions
      - Implementation
Formalism for describing and enforcing constraints on databases

Database - a collection of atomic facts from some universe.

Revision rules
- specify constraints on a database,
- specify a preferred way to satisfy constraints.

Arbitrary initial database.

Justified revisions
- satisfy all constraints,
- all changes are justified by revision rules.
Basic concepts

- Revision literals: $in(a), out(a)$ ($a \in U$).
- Revision rules:

  $$in(a) \leftarrow in(a_1), \ldots, in(a_m), out(b_1), \ldots, out(b_n),$$  
  \hfill \text{(in-rule)}

  $$out(a) \leftarrow in(a_1), \ldots, in(a_m), out(b_1), \ldots, out(b_n),$$  
  \hfill \text{(out-rule)}

  where $a, a_i, b_i \in U$ ($1 \leq i \leq n$).
- Revision program - collection of revision rules.
Necessary change

- $\alpha^D$ - dual of a literal $\alpha$. $\text{in}(a)^D = \text{out}(a)$, $\text{out}(a)^D = \text{in}(a)$.
- A set of literals is coherent if it does not contain a pair of dual literals.
- $P$ – a revision program. The necessary change of $P$, $NC(P)$, is the least model of $P$, treated as a Horn program built of independent propositional atoms of the form $\text{in}(a)$ and $\text{out}(b)$.
- Coherent $NC(P)$ specifies a revision.

**Example.**

$P : \text{in}(Ann) \quad \leftarrow \quad NC(P) = \{\text{in}(Ann), \text{out}(Bob)\}$

$\text{out}(Bob) \quad \leftarrow \quad \text{in}(Ann)$

$\text{out}(Tom) \quad \leftarrow \quad \text{out}(Ann)$
Justified revisions

- Given a database $I$ and a coherent set of literals $L$, define
  \[ I \oplus L = (I \cup \{a: \text{in}(a) \in L\}) \setminus \{a: \text{out}(a) \in L\}. \]

- Inertia set for databases $I$, $R$:
  \[ I(I, R) = \{\text{in}(a) : a \in I \cap R\} \cup \{\text{out}(a) : a \notin I \cup R\}. \]

- Reduct of $P$ with respect to $(I, R)$ (denoted $P_{I,R}$) – the revision program obtained from $P$ by eliminating from the body of each rule in $P$ all literals in $I(I, R)$.

- $P$ - a revision program, $I$ and $R$ - databases. $R$ is called a $P$-justified revision of $I$ if $NC(P_{I,R})$ is coherent and $R = I \oplus NC(P_{I,R})$. 
Example

\[ P : \]

\begin{align*}
& \text{in}(Ann) \leftarrow \text{out}(Bob) \\
& \text{in}(Bob) \leftarrow \text{out}(Ann) \\
& \text{in}(David) \leftarrow \text{in}(Tom) \\
& \text{out}(Tom) \leftarrow \text{out}(David) \\
& \text{out}(Ann) \leftarrow \text{in}(David) \\
& \text{out}(David) \leftarrow \text{in}(Bob)
\end{align*}

\[ P_{I,R} : \]

\begin{align*}
& \text{in}(Ann) \leftarrow \text{out}(Bob) \\
& \text{in}(Bob) \leftarrow \\
& \text{in}(David) \leftarrow \text{in}(Tom) \\
& \text{out}(Tom) \leftarrow \text{out}(David) \\
& \text{out}(Ann) \leftarrow \text{in}(David) \\
& \text{out}(David) \leftarrow \text{in}(Bob)
\end{align*}

Initial database: \( I = \{David, Tom\}. \)

Revision: \( R = \{Bob\}. \)

Inertia (no justification is needed): \( \text{out}(Ann). \)

Necessary change: \( \text{in}(Bob), \text{out}(David), \text{out}(Tom). \)

Updating \( I: \) \( (I \cup \{Bob\}) \setminus \{David, Tom\}. \)
Basic properties

1. If a database $R$ is a $P$-justified revision of $I$, then $R$ is a model of $P$.

2. If a database $B$ satisfies a revision program $P$ then $B$ is a unique $P$-justified revision of itself.

3. If $R$ is a $P$-justified revision of $I$, then $R \div I$ is minimal in the family $\{B \div I : B$ is a model of $P\}$.
Relation to Logic Programming

$P$-justified revisions of $\emptyset$ coinside with stable models of the logic program with constraints, $lp(P)$, obtained from $P$ by replacing revision rules of the form

\[ in(a) \leftarrow in(a_1), \ldots, in(a_m), out(b_1), \ldots, out(b_n) \]

by

\[ a \leftarrow a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_n \]

and replacing revision rules of the form

\[ out(a) \leftarrow in(a_1), \ldots, in(a_m), out(b_1), \ldots, out(b_n) \]

by constraints

\[ \leftarrow a, a_1, \ldots, a_m, \neg b_1, \ldots, \neg b_n. \]
Shifting

$I \subseteq J$

$W = I \div J = (I \setminus J) \cup (J \setminus I)$

$W$ - a set of atoms that change status

Define a $W$-transformation (shift) as follows.

For a literal $\alpha$  ($\alpha = in(a)$ or $\alpha = out(a)$), $T_W(\alpha) = \begin{cases} \alpha^D, & \text{when } a \in W \\ \alpha, & \text{when } a \notin W. \end{cases}$

For a set of literals $L$, $T_W(L) = \{ T_W(\alpha) : \alpha \in L \}$.

For a set of atoms $X$, $T_W(X) = \{ a : in(a) \in T_W(\{ in(b) : b \in X \} \cup \{ out(b) : b \notin X \}) \}$.

For a revision program $P$, $T_W(P)$ is obtained from $P$ by applying $T_W$ to each literal in $P$. 
Shifting theorem

For any databases $I_1$ and $I_2$, database $R$ is a $P$-justified revision of $I_1$ if and only if $T_{I_1} \div I_2(R)$ is a $T_{I_1} \div I_2(P)$-justified revision of $I_2$.

**Corollary.** For each $I$ and $R$, $R$ is $P$-justified revision of $I$ if and only if $T_I(R)$ is $T_I(P)$-justified revision of $\emptyset$. 
Computing justified revisions

by means of LP

1. Given $P$ and $I$, apply $T_I$ to obtain $T_I(P)$ and $\emptyset$.
2. Convert $T_I(P)$ into the logic program $lp(T_I(P))$.
3. Compute its answer sets.
4. Apply $T_I$ to the answer sets to obtain the $P$-justified revisions of $I$. 
A robot is equipped with sensors which provide observations: 

\[ \text{observation}(Par, Value, Sensor) \]

View of the world has exactly one value for each parameter: 

\[ \text{world}(Par, Value, Sensor) \]

RP updates the view of the world, consists of rules of types:

\[ \text{in}(\text{observation}(Par, Value, Sensor)) \leftarrow \]

\[ \text{in}(\text{world}(Par, Value, Sensor)) \leftarrow \text{in}(\text{observation}(Par, Value, Sensor)). \]

\[ \text{out}(\text{world}(Par, Value, Sensor)) \leftarrow \text{in}(\text{world}(Par, Value1, Sensor1)). \]

(\text{where Sensor} \neq \text{Sensor1} \text{ and/or Value} \neq \text{Value1})
An ordered revision program is a pair \((P, \mathcal{L})\) where \(\mathcal{L}\) is a function which assigns to revision rules in \(P\) unique labels. \(\mathcal{L}(P)\) - set of labels in \(P\).

\[
l : \alpha_0 \leftarrow \alpha_1, \ldots, \alpha_n
\]

A preference on rules in \((P, \mathcal{L})\) is an expression of the form

\[
\text{prefer}(l_1, l_2) \leftarrow \text{initially}(\alpha_1, \ldots, \alpha_k), \alpha_{k+1}, \ldots, \alpha_n,
\]

where \(l_i\) are labels, \(\alpha_j\) are revision literals.

A revision program with preferences is a triple \((P, \mathcal{L}, S)\), where \((P, \mathcal{L})\) is an ordered revision program and \(S\) is a set of preferences on rules in \((P, \mathcal{L})\).

\(S\) - the control program.
\((P, \mathcal{L}, S)\) is translated into ordinary RP:

\[ U^{\mathcal{L}(P)} = U \cup \{\text{ok}(l), \text{defeated}(l), \text{prefer}(l, l') : l, l' \in \mathcal{L}(P)\} \]

Define \(P^{S,I}\) over \(U^{\mathcal{L}(P)}\) to be a revision program consisting of rules:

- for each \(l \in \mathcal{L}(P)\)

  \[
  \begin{align*}
  \text{head}(l) & \leftarrow \text{body}(l), \text{in}(\text{ok}(l)) \\
  \text{in}(\text{ok}(l)) & \leftarrow \text{out}(\text{defeated}(l))
  \end{align*}
  \]

- for each preference

  \[
  \text{prefer}(l_1, l_2) \leftarrow \text{initially}(\alpha_1, \ldots, \alpha_k), \alpha_{k+1}, \ldots, \alpha_n,
  \]

  in \(S\) such that \(\alpha_1 \ldots, \alpha_k\) are satisfied by \(I\)

  \[
  \begin{align*}
  \text{in}(\text{prefer}(l_1, l_2)) & \leftarrow \alpha_{k+1}, \ldots, \alpha_n \\
  \text{in}(\text{defeated}(l_2)) & \leftarrow \text{body}(l_1), \text{in}(\text{prefer}(l_1, l_2))
  \end{align*}
  \]
(\(P, \mathcal{L}, S\))-justified revisions

A database \(R\) is a \((P, \mathcal{L}, S)\)-justified revision of \(I\) if there exists \(R' \subseteq U^\mathcal{L}(P)\) such that \(R'\) is a \(P^S,I\)-justified revision of \(I\), and \(R = R' \cap U\).
Properties

- Justified revision semantics for revision programs with preferences extends justified revision semantics for ordinary revision programs
- Shifting property holds
- Not every \((P, \mathcal{L}, S)\)-justified revision is a model of \(P\)
When \((P, L, S)\)-justified revisions are models of \(P\)?

Two rules \(r, r'\) of \(P\) are in conflict if one of the following conditions is satisfied:

1. \((\text{head}(r))^D \in \text{body}(r')\) and \((\text{head}(r'))^D \in \text{body}(r)\); or

2. \(\text{body}(r) \cup \text{body}(r')\) is incoherent.

A set of preferences is conflict-resolving if it contains only preferences between conflicting rules.

**Theorem.** Let \((P, L, S)\) be a revision program with preferences where \(S\) is a set of conflict-resolving preferences and is cycle-free. For every \((P, L, S)\)-justified revision \(R\) of \(I\), \(R\) is a model of \(P\).
**Soft revision rules with weights**

Revision program is divided into hard and soft rules: \( P = HR \cup SR \)

All hard rules must be satisfied.

Only a subset of soft rules may be satisfied.

The subset of soft rules that is satisfied is optimal with respect to some criteria.
Maximal number of rules

**Definition 1**  \( R \) is a \((HR, SR)\)-preferred justified revision of \( I \) if \( R \) is a \((HR \cup S)\) - justified revision of \( I \) for some \( S \subseteq SR \), and for all \( S' \) if \( S \subseteq S' \subseteq SR \), then there are no \((HR \cup S')\)-justified revisions of \( I \).
For each $I$, translate $P = HR \cup SR$ into an smodels program $lp(T_I(HR)) \cup lp'(T_I(SR))$.

$lp'$ translates a rule

$$in(a) \leftarrow in(p_1), \ldots, in(p_m), out(s_1), \ldots, out(s_n)$$

into the rules

$$\{rule_i\} : - p_1, \ldots, p_m, not s_1, \ldots, not s_n.$$  

$$a : - rule_i$$

$lp'$ translates a rule

$$out(a) \leftarrow in(p_1), \ldots, in(p_m), out(s_1), \ldots, out(s_n)$$

into the rules

$$\{rule_i\} : - p_1, \ldots, p_m, not s_1, \ldots, not s_n.$$  

$$: - rule_i, a.$$
Implementation, cont’d

smodels statement

\texttt{maximize\{rule_1, \ldots, rule_k\}.}

$(k$ is the number of rules in $SR)$

allows to compute one (not all) $(HR, SR)$-preferred justified revision, which has max size.
Weighted rules

Each \( r \in SR \) is assigned a weight, \( w(r) \) (its importance).

**Definition 2** \( R \) is called a rule-weighted \( (HR, SR) \)-justified revision of \( I \) if the following two conditions are satisfied:

1. there exists a set of rules \( S \subseteq SR \) such that \( R \) is a \( (HR \cup S) \)-justified revision of \( I \), and

2. for any set of rules \( S' \subseteq SR \), if \( R' \) is a \( (HR \cup S') \)-justified revision of \( I \), then the sum of weights of rules in \( S' \) is less than or equal than the sum of weights of rules in \( S \).
Implementation

Same translation of soft rules into smodels program, but different maximize statement:

\[
\text{maximize}[\text{rule}_1 = w(1), \text{rule}_2 = w(2), \ldots, \text{rule}_k = w(k)].
\]
Weighted atoms.

Each \( a \in U \) is assigned a weight \( w(a) \)
(the more the weight the less we want to change its status)

**Definition 3** \( R \) is called an atom-weighted \((HR, SR)\)-justified revision of \( I \) if
the following two conditions are satisfied:

1. there exists a set of rules \( S \subseteq SR \) such that \( R \) is a \((HR \cup S)\)-justified
   revision of \( I \), and

2. for any set of rules \( S' \subseteq SR \), if \( Q \) is a \((HR \cup S')\)-justified revision of \( I \),
   then the sum of weights of atoms in \( I \div Q \) is greater than or equal to the
   sum of weights of atoms in \( I \div R \).
Implementation

Same translation of soft rules into `smodels` program, but different `maximize` statement:

\[
\text{minimize}[a_1 = w(a_1), a_2 = w(a_2), \ldots, a_n = w(a_n)]
\]

where \(a_1, \ldots, a_n\) are all the atoms in \(U\).
Definition 4  \( R \) is called a minimal size difference \( P \)-justified revision of \( I \) if the following two conditions are satisfied:

1. \( R \) is a \( P \)-justified revision of \( I \), and

2. for any \( P \)-justified revision \( R' \), the number of atoms in \( R \div I \) is less than or equal to the number of atoms in \( R' \div I \).
Implementation

Use minimize statement:

\[
\text{minimize}\{a_1, a_2, \ldots, a_n\}
\]

where \(a_1, \ldots, a_n\) are all the atoms in \(U\).