Preference Revision via Declarative Debugging

Pierangelo Dell'Acqua*† and Luís Moniz Pereira†

* Department of Science and Technology - ITN
Linköping University, 601 74 Norrköping, Sweden
pier@itn.liu.se

† Centro de Inteligência Artificial - CENTRIA
Departamento de Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
lmp@di.fct.unl.pt

Abstract. Preference criteria are rarely static. Often they are subject to modification and aggregation. The resulting preference criteria may not satisfy the properties of the original ones and must therefore be revised. This paper investigates the problem of revising such preference criteria by means of declarative debugging techniques.

1 Motivation

Preference criteria are subject to be modified when new information is brought to the knowledge of the individual, or aggregated when we need to represent and reason about the simultaneous preferences of several individuals. As motivating example, suppose you invite three friends Karin, Leif and Osvald to go and see a movie. Karin prefers thrillers to action movies. Leif, on the other hand, prefers action movies to thrillers. Finally, Osvald is like Leif and prefers action movies to thrillers. Suppose you need to buy the tickets. Which movie do you choose?

Preference aggregation is an important problem and potential applications of this work include those where preference reasoning plays a role, e.g., in artificial intelligence, political science, and economics (cf. social choice and multi-criteria decision).

Typically, preference criteria must satisfy certain properties, e.g., those of strict partial order. When aggregating or updating preference criteria such properties might not be preserved, and therefore the need arises for a revision. In this paper, we consider any preference criteria expressible in the language of logic programs (LP), and investigate the problem of revising them by means of declarative debugging techniques for LP. In particular, we employ an adapted version of the contradiction removal method defined for the class of normal logic programs plus integrity constraints proposed in [10]. The resulting framework is flexible and general, and tailored neither to any specific preference criteria nor any specific method for preference aggregation, but rather to any method expressible in LP. The ability to express meta-information on the diagnoses of a revision problem gives us a further level of abstraction permitting to select the best diagnosis for the problem at hand.

2 Background

In this section we provide some logic programming fundamentals and few basic definitions regarding preference relations.

2.1 Language

Let $\mathcal{L}$ be a first order language. A literal in $\mathcal{L}$ is an atom A in $\mathcal{L}$ or its default negation $not\, A$. A normal logic program P over $\mathcal{L}$ (sometimes simply called program) is a set of rules and integrity constraints of the form:

$$A \leftarrow L_1, \ldots, L_n \quad (n \geq 0)$$

where A is an atom, $L_1, \ldots, L_n$ are literals in $\mathcal{L}$, where in integrity constraints A is $\bot$ (contradiction). A rule stands for all its ground instances with respect to $\mathcal{L}$. When $n = 0$ we write the rule as A. $\mathcal{L}_P$ denotes the language of P.

For normal logic programs we consider the Well Founded Semantics [7]. We write $P \models L$ whenever a literal L belongs to the well-founded model of a program P. P is contradictory if $P \models \bot$. Programs are liable to be contradictory because of the integrity constraints.

Example 1. Let $P = \{a \leftarrow not\, b; \bot \leftarrow a\}$. Since we have no rules for b, by Closed World Assumption (CWA), it is natural to accept $not\, b$ as true and therefore conclude a. Because of the integrity constraint, we conclude $\bot$ and thus engender a contradiction. ▯

2.2 Preference Relation

Given a set N, a preference relation $\succ$ is any binary relation on N. Given two elements a and b in N, $a \succ b$ means that a is preferred to b. We assume that N contains at least two elements.

We do not assume any property of $\succ$, although in many situations it will satisfy the properties of a strict partial order. Typical properties of $\succ$ include:

- irreflexivity: $\forall x.\ x \not\succ x$
- asymmetry: $\forall x\, \forall y.\ x \succ y \Rightarrow y \not\succ x$
- transitivity: $\forall x\, \forall y\, \forall z.\ (x \succ y \wedge y \succ z) \Rightarrow x \succ z$
- negative transitivity: $\forall x\, \forall y\, \forall z.\ (x \not\succ y \wedge y \not\succ z) \Rightarrow x \not\succ z$
- connectivity: $\forall x\, \forall y.\ x \succ y \vee y \succ x \vee x = y$

The relation $\succ$ is:

- a strict partial order if it is irreflexive and transitive (thus also asymmetric);
- a weak order if it is a negatively transitive strict partial order;
- a total order if it is a connected strict partial order.

Every preference relation $\succ$ induces an indifference relation $\sim$. Two elements a and b in N are indifferent $a \sim b$ if neither is preferred to the other one, that is, $a \not\succ b$ and $b \not\succ a$.

3 Diagnosis

In this section we present the notion of diagnosis adapted from [10] to handle preference relations. We illustrate the use of diagnoses with a number of examples. Given a contradictory program P, to revise its contradiction ($\bot$) we have to modify P by adding and removing rules. In this framework, the diagnostic process reduces to finding such rules. To specify which rules in P may be added or removed, we assume given a set C of predicate symbols in $\mathcal{L}_P$. C induces a partition of P into two disjoint parts: a changeable one P_c and a stable one P_s. P_c contains the rules in P defining predicate symbols in C, while P_s contains the rules in P defining predicate symbols not belonging to C. P_c is the part subject to the diagnosis process.

Definition 1. Let P be a program and C a set of predicate symbols in $\mathcal{L}_P$. Let D be a pair $\langle U, I\rangle$ where U is a set of atoms, whose predicate symbols are in C, and $I \subseteq P_c$. Then D is a *diagnosis* for P iff $(P - I) \cup U \not\models \bot$. The pair $\langle \{\}, \{\}\rangle$ is called *empty diagnosis*.

Intuitively, a diagnosis specifies the rules to be added and removed from the changeable part of P to revise its contradiction $\bot$. In order to minimize the number of changes we consider minimal diagnoses.

Definition 2. Let P be a program and $D = \langle U, I\rangle$ a diagnosis for P. Then, D is a *minimal diagnosis* for P iff there exists no diagnosis $D_2 = \langle U_2, I_2\rangle$ for P such that $(U_2 \cup I_2) \subseteq (U \cup I)$.

Preference relations can be composed in several ways, or updated to reflect changes in user preference criteria. Following [3] we distinguish between unidimensional and multidimensional composition. In unidimensional composition, a number of preference relations over the same domain are composed, producing another preference relation over the same domain. In contrast, in multidimensional composition, a number of preference relations defined over several domains are composed, producing a preference relation defined over the Cartesian product of those relations.

When composing preference relations, it is often the case (see [3] for a discussion) that the resulting preference relation does not satisfy some required property, and therefore needs revising.

Example 2. Consider a framework where preference relations are required to satisfy the properties of strict partial orders. Let $\succ_1$ and $\succ_2$ be two preference relations defined as: $a \succ_1 b$, and $b \succ_2 c$ and $b \succ_2 a$. Consider the boolean composition $\succ$ of $\succ_1$ and $\succ_2$ defined as $\succ = \succ_1 \cup \succ_2$. Clearly, $\succ$ is not a strict partial order being antisymmetric and transitivity not preserved. To revise $\succ$, we formalize both $\succ$ and the properties of strict partial order with program P. Assume we want to revise only the rules in P encoding $\succ_1$ and $\succ_2$.

$$P_s = \left\{ \begin{array}{l} \bot \leftarrow \mathrm{p(x,x)} \\ \bot \leftarrow \mathrm{p(x,y),\ p(y,x)} \\ \bot \leftarrow \mathrm{p(x,y),\ p(y,z),\ not\ p(x,z)} \\ \mathrm{p(x,y)} \leftarrow \mathrm{p_1(x,y)} \\ \mathrm{p(x,y)} \leftarrow \mathrm{p_2(x,y)} \end{array} \right\} \quad \text{and} \quad P_c = \left\{ \begin{array}{l} \mathrm{p_1(a,b)} \\ \mathrm{p_2(b,c)} \\ \mathrm{p_2(b,a)} \end{array} \right\}.$$

The integrity constraints in P_s state that if the preference relation $\succ$ (represented by p) is reflexive, symmetric or not transitive, then the program is contradictory. The last two rules in P_s define $\succ$ as the union of $\succ_1$ and $\succ_2$ (represented by p_1 and p_2). The rules in P_c formalizing the two original preference relations ($\succ_1$ and $\succ_2$) are those subject to diagnosis (that is, C=$\{\mathrm{p_1,p_2}\}$).

P is contradictory because its well-founded model M_P=$\{\mathrm{p_1(a,b)},\ \mathrm{p_2(b,c)},\ \mathrm{p_2(b,a)},\ \mathrm{p(a,b)},\ \mathrm{p(b,c)},\ \mathrm{p(b,a)},\ \bot\}$ contains $\bot$. According to Def. 1, P affords four minimal diagnoses:

$$D_1 = \langle\{\mathrm{p_2(a,c)}\}, \{\mathrm{p_2(b,a)}\}\rangle \qquad D_2 = \langle\{\mathrm{p_1(a,c)}\}, \{\mathrm{p_2(b,a)}\}\rangle$$
$$D_3 = \langle\{\}, \{\mathrm{p_2(b,a)},\ \mathrm{p_2(b,c)}\}\rangle \qquad D_4 = \langle\{\}, \{\mathrm{p_1(a,b)}\}\rangle$$

E.g., D_1 is a diagnosis since the well-founded model of $(P - \{\mathrm{p_2(b,a)}\}) \cup \{\mathrm{p_2(a,c)}\}$ is $M = \{\mathrm{p_1(a,b)},\ \mathrm{p_2(b,c)},\ \mathrm{p_2(a,c)},\ \mathrm{p(a,b)},\ \mathrm{p(b,c)},\ \mathrm{p(a,c)}\}$ and $M \not\models \bot$. ▯

This example illustrates the prioritized composition of conditional preference relations:

Example 3. Given the two preference relations $\succ_1$ and $\succ_2$, the prioritized composition $\succ$ of $\succ_1$ and $\succ_2$ is defined as: $x \succ y \ \equiv \ x \succ_1 y \vee (x \sim_1 y \wedge x \succ_2 y)$ where $\sim_1$ is the indifference relation induced by $\succ_1$, that is: $x \sim_1 y \ \equiv \ x \not\succ_1 y \wedge y \not\succ_1 x$. Let $\succ_1$ and $\succ_2$ be two preference relations defined as $a \succ_1 b$, and $b \succ_2 c$, $c \succ_2 a$ *if cond*, and $b \succ_2 a$. Conditional preference $c \succ_2 a$ *if cond* states c is preferred to a if *cond* holds. Suppose $\succ$ is required to be a strict partial order. Let *cond* denote some condition that cannot be revised, and assume *cond* true. This situation can be formalized with a program $P = P_s \cup P_c$. Suppose we wish to revise only preference relation $\succ_2$ (and not $\succ_1$) because $\succ_1$ has priority over $\succ_2$. To do so, we place the rules defining $\succ_1$ in P_s and the rules defining $\succ_2$ in P_c.

$$P_s = \left\{ \begin{array}{l} \bot \leftarrow \mathrm{p(x,x)} \\ \bot \leftarrow \mathrm{p(x,y),\ p(y,x)} \\ \bot \leftarrow \mathrm{p(x,y),\ p(y,z),\ not\ p(x,z)} \\ \mathrm{p(x,y)} \leftarrow \mathrm{p_1(x,y)} \\ \mathrm{p(x,y)} \leftarrow \mathrm{ind_1(x,y),\ p_2(x,y)} \\ \mathrm{ind_1(x,y)} \leftarrow \mathrm{not\ p_1(x,y),\ not\ p_1(y,x)} \\ \mathrm{p_1(a,b)} \\ \mathrm{cond} \end{array} \right\} \quad \text{and} \quad P_c = \left\{ \begin{array}{l} \mathrm{p_2(b,c)} \\ \mathrm{p_2(c,a)} \leftarrow \mathrm{cond} \\ \mathrm{p_2(b,a)} \end{array} \right\}.$$

It is easy to see that $\succ$ is not a strict partial order. The well-founded model of P is $M_P = \{\mathrm{p}_1(\mathrm{a,b}),\ \mathrm{p}_2(\mathrm{b,c}),\ \mathrm{p}_2(\mathrm{c,a}),\ \mathrm{p}_2(\mathrm{b,a}),\ \mathrm{p(a,b)},\ \mathrm{p(b,c)},\ \mathrm{p(c,a)}, \bot\}$. P admits three minimal diagnoses:

$$D_1 = \langle\{\mathrm{p}_2(\mathrm{a,c})\}, \{\mathrm{p}_2(\mathrm{c,a}) \leftarrow \mathrm{cond}\}\rangle \qquad D_2 = \langle\{\mathrm{p}_2(\mathrm{c,b})\}, \{\mathrm{p}_2(\mathrm{b,c})\}\rangle$$
$$D_3 = \langle\{\}, \{\mathrm{p}_2(\mathrm{b,c}),\ \mathrm{p}_2(\mathrm{c,a}) \leftarrow \mathrm{cond}\}\rangle$$

□

The next example exhibits a situation of multidimensional composition.

Example 4. Given the two preference relations $\succ_1$ and $\succ_2$, the Pareto composition $\succ$ of $\succ_1$ and $\succ_2$ is defined as:

$$(x, x_2) \succ (y, y_2) \ \equiv\ x \succeq_1 y \wedge x_2 \succeq_2 y_2 \wedge (x \succ_1 y \vee x_2 \succ_2 y_2)$$

where $\quad x \sim_i y \ \equiv\ x \not\succ_i y \wedge y \not\succ_i x \quad$ and $\quad x \succeq_i y \ \equiv\ x \succ_i y \vee x \sim_i y$ with $i = 1, 2$. Let $\succ_1$ and $\succ_2$ be:

$$a \succ_1 b$$
$$a_2 \succ_2 b_2 \quad b_2 \succ_2 c_2 \quad a_2 \succ_2 c_2$$

The Pareto composition $\succ$ does not preserve the properties of total order. In fact, the tuples (b, a_2) and (a, b_2) are indifferent to one another, and hence $\succ$ does not preserve connectivity.

The Pareto composition of $\succ_1$ and $\succ_2$ can be formalized by the program $P = P_s \cup P_c$:

$$P_s = \left\{ \begin{array}{r} \bot \leftarrow \mathrm{p(x,x)} \\ \bot \leftarrow \mathrm{p(x,y)},\ \mathrm{p(y,x)} \\ \bot \leftarrow \mathrm{p(x,y)},\ \mathrm{p(y,z)},\ \mathrm{not\ p(x,z)} \\ \bot \leftarrow \mathrm{notConnected} \\ \mathrm{p((x,x_2),(y,y_2))} \leftarrow \mathrm{p}_1\mathrm{(x,y)},\ \mathrm{peq}_2\mathrm{(x_2,y_2)} \\ \mathrm{p((x,x_2),(y,y_2))} \leftarrow \mathrm{peq}_1\mathrm{(x,y)},\ \mathrm{p}_2\mathrm{(x_2,y_2)} \\ \mathrm{peq}_1\mathrm{(x,y)} \leftarrow \mathrm{p}_1\mathrm{(x,y)} \\ \mathrm{peq}_1\mathrm{(x,y)} \leftarrow \mathrm{ind}_1\mathrm{(x,y)} \\ \mathrm{ind}_1\mathrm{(x,y)} \leftarrow \mathrm{not\ p}_1\mathrm{(x,y)},\ \mathrm{not\ p}_1\mathrm{(y,x)} \\ \mathrm{peq}_2\mathrm{(x,y)} \leftarrow \mathrm{p}_2\mathrm{(x,y)} \\ \mathrm{peq}_2\mathrm{(x,y)} \leftarrow \mathrm{ind}_2\mathrm{(x,y)} \\ \mathrm{ind}_2\mathrm{(x,y)} \leftarrow \mathrm{not\ p}_2\mathrm{(x,y)},\ \mathrm{not\ p}_2\mathrm{(y,x)} \\ \mathrm{notConnected} \leftarrow \mathrm{not\ p((x,x_2),(y,y_2))},\ \mathrm{not\ p((y,y_2),(x,x_2))} \end{array} \right\}$$

and

$$P_c = \left\{ \begin{array}{l} \mathrm{p}_1\mathrm{(a,b)} \\ \mathrm{p}_2\mathrm{(a_2,b_2)} \\ \mathrm{p}_2\mathrm{(b_2,c_2)} \\ \mathrm{p}_2\mathrm{(a_2,c_2)} \end{array} \right\}.$$

P is contradictory because there exist two tuples that are not connected (being indifferent to one another), i.e. the tuples (b,a_2) and (a,b_2). Thus, by the last rule in P_s notConnected holds and by the last integrity constraint $\bot$ holds as well. The following property generalizes this specific example by stating that the Pareto composition $\succ$ cannot be a total order.

Property 1. Let $\succ_x$ and $\succ_y$ be two preference relations whose domains contain at least two elements. If $\succ_x$ and $\succ_y$ are strict partial orders, then the Pareto composition $\succ$ of $\succ_x$ and $\succ_y$ cannot be a total order.

Proof. Let $N_x = \{x_1, x_2\}$ and $N_y = \{y_1, y_2\}$ be the domains of $\succ_x$ and $\succ_y$. Consider the tuples (x_1, y_1) and (x_1, y_2), and suppose that the first one is preferred to the second, i.e. $(x_1, y_1) \succ (x_1, y_2)$. Then, by definition of Pareto composition it must hold that $y_1 \succ_y y_2$. Consider now the tuples (x_1, y_1) and (x_2, y_1), and assume that $(x_1, y_1) \succ (x_2, y_1)$. Clearly, we must have that $x_1 \succ_x x_2$. Since $\succ_x$ and $\succ_y$ are strict partial orders, it follows that $(x_2, y_1) \sim (x_1, y_2)$. Hence, $\succ$ cannot be a total order.

The impossibility of $\succ$ of being a total order is reflected in the fact that there exists no diagnosis that makes P non-contradictory. ∎

4 Computing Minimal Diagnosis

To compute minimal diagnoses of a contradictory program, we employ the contradiction removal method presented in [10], adapted here to handle preference relations. Consider again Example 1. It is arguable that CWA may not hold of atom b as it leads to contradiction. The contradiction removal method is based on the idea of revising (to false) some of the default atoms *not* A true by CWA. A default atom *not* A can be revised to false by simply adding A to P. According to [10] the default literals *not* A true by CWA that are allowed to change their truth value are those for which there exists no rule in P defining A. Such literals are called revisable.

Definition 3. The *revisables* of a program P is a subset of that set of atoms A (with $A \neq \bot$) for which there are no rules defining A in P.

Definition 4. Let P be a program and V a set of revisables of P. A set $Z \subseteq V$ is a *revision* of P wrt. V iff $P \cup Z \not\models \bot$.

Example 5. Consider the contradictory program $P = P_s \cup P_c$:

$$P_s = \left\{ \begin{array}{l} \bot \leftarrow a, a' \\ \bot \leftarrow b \\ \bot \leftarrow d, not\ f \end{array} \right\} \quad \text{and} \quad P_c = \left\{ \begin{array}{l} a \leftarrow not\ b, not\ c \\ a' \leftarrow not\ d \\ c \leftarrow e \end{array} \right\}$$

with revisables $V = \{b, d, e, f\}$. Intuitively the literals *not* b, *not* d and *not* e are true by CWA, entailing a and a', and hence $\bot$ via the first integrity constraint. The revisions of P are $\{e\}$, $\{d, f\}$, $\{e, f\}$ and $\{d, e, f\}$, where the first two are minimal. ∎

The following transformation maps programs into equivalent programs that are suitable for contradiction removal.

Definition 5. Let P be a program and C a set of predicate symbols in $\mathcal{L}_P$. The transformation Γ that maps P into a program P' is obtained by applying to P the following two operations:

- Add *not incorrect*$(A \leftarrow Body)$ to the body of each rule $A \leftarrow Body$ in P_c.
- Add the rule $p(x_1, \ldots, x_n) \leftarrow$ *uncovered*$(p(x_1, \ldots, x_n))$ for each predicate p with arity n in C.

We assume the predicate symbols *incorrect* and *uncovered* do not belong to the language of P. The transformation Γ preserves the truths of program P.

Property 2. Let P be a program and L a literal. Then $P \models L$ iff $\Gamma(P) \models L$.

Proof. The claim follows immediately by noting that *not incorrect*(.) and *uncovered*(.) are true and false in $\Gamma(P)$ because *incorrect* and *uncovered* do not belong to $\mathcal{L}_P$.

Example 6. Let P be the program of Example 2. Then, the program $\Gamma(P)$ is:

$$\Gamma(P) = \left\{ \begin{array}{l} \bot \leftarrow \text{p(x,x)} \\ \bot \leftarrow \text{p(x,y), p(y,x)} \\ \bot \leftarrow \text{p(x,y), p(y,z), not p(x,z)} \\ \\ \text{p(x,y)} \leftarrow \text{p}_1\text{(x,y)} \\ \text{p(x,y)} \leftarrow \text{p}_2\text{(x,y)} \\ \\ \text{p}_1\text{(a,b)} \leftarrow \text{not incorrect(p}_1\text{(a,b))} \\ \text{p}_2\text{(b,c)} \leftarrow \text{not incorrect(p}_2\text{(b,c))} \\ \text{p}_2\text{(b,a)} \leftarrow \text{not incorrect(p}_2\text{(b,a))} \\ \\ \text{p}_1\text{(x,y)} \leftarrow \text{uncovered(p}_1\text{(x,y))} \\ \text{p}_2\text{(x,y)} \leftarrow \text{uncovered(p}_2\text{(x,y))} \end{array} \right\}$$

The minimal revisions of $\Gamma(P)$ wrt. the revisables of the form incorrect(.) and uncovered(.) are:

$$Z_1 = \{\text{uncovered(p}_2\text{(a,c))},\text{incorrect(p}_2\text{(b,a))}\}$$
$$Z_2 = \{\text{uncovered(p}_1\text{(a,c))},\text{incorrect(p}_2\text{(b,a))}\}$$
$$Z_3 = \{\text{incorrect(p}_2\text{(b,a))},\text{incorrect(p}_2\text{(b,c))}\}$$
$$Z_4 = \{\text{incorrect(p}_1\text{(a,b))}\}$$

It is easy to see that Z_1, for instance, is a revision since the well-founded model M of $P \cup Z_1$ is $M = \{\text{p}_1\text{(a,b)}, \text{p}_2\text{(b,c)}, \text{p(a,b)}, \text{p(b,c)}, \text{p}_2\text{(a,c)}, \text{p(a,c)}, \text{uncovered(p}_2\text{(a,c))}, \text{incorrect(p}_2\text{(b,a))}\}$ and $M \not\models \bot$. ∎

The following result relates the minimal diagnoses of a program P with the minimal revisions of $\Gamma(P)$.

Property 3. Let P be a program. The pair $D = \langle U, I \rangle$ is a diagnosis for P iff

$$Z = \{uncovered(A) : A \in U\} \cup \{incorrect(A \leftarrow Body) : A \leftarrow Body \in I\}$$

is a revision of $\Gamma(P)$, where the revisables are all the literals of the form $incorrect(.)$ and $uncovered(.)$. Furthermore, D is a minimal diagnosis iff Z is a minimal revision.

Proof. It follows immediately by noting that the programs $P - I \cup U$ and $P \cup Z$ are equivalent, that is, for every literal L with $L \neq uncovered(.)$ and $L \neq incorrect(.)$ it holds that $P - I \cup U \models L$ iff $P \cup Z \models L$.

To compute the minimal diagnosis of a program P we consider the transformed program $\Gamma(P)$ and compute its minimal revisions. An algorithm for computing minimal revisions is given in [10].

5 Selecting Minimal Diagnosis

Typically in a preference revision problem, we only consider minimal diagnoses (wrt. set inclusion) and the problem that naturally arises is how to select the best ones. In some situations, we may ask the user for more information about his or her preferences in order to narrow down the alternatives. In other situations, we require a completely automatized approach. Thus, we need a selection function $f(X) \subseteq X$ where X is a set of minimal diagnoses. Ideally $f(X)$ is a singleton, otherwise we must arbitrarily choose one diagnosis from it. The selection function can be defined by using meta-preference information:

- Temporal information: in case of conflict keep more recent/older preferences.
- Weights can be associated to preferences so that one can compute the total weight of a diagnosis.
- The preference relation can be revised for each minimal diagnosis and shown to the user. The user by choosing one answer over others makes the system infer the preferred diagnosis. Thus, consequences of preferences can be used to revise the preferences themselves. (Typically, preferences are revisable.)
- One may want to make the smallest number of changes. Thus, one will prefer D_4 in Example 2 to the other minimal diagnoses. In contrast, one may prefer adding preferences rather than removing them. In this case, one prefers D_2 to D_3.
- In multi-agent scenarios, it is often the case that one wants to select a fair revision, for example, and not to reject all the preferences of one agent while maintaining the preferences of another agent.
- Preferences can be associated with a domain that can be employed to select diagnoses. For instance, regarding wine one gives priority to the preferences of Carlo, who is a wine producer, rather than to the preferences of John.
- Diagnoses containing more specific preferences can be selected. For example, given the two diagnoses $\langle \{\}, \{moscato \succ chianti\} \rangle$ and $\langle \{\}, \{redWine \succ whiteWine\} \rangle$ the first is selected since moscato is a white wine and chianti is a red wine.

6 Related Work

Recently, several approaches have been proposed in literature for aggregating preference criteria. These approaches tackle the problem from different perspectives. One line of research studies which properties are preserved by different methods of preference criteria aggregation. In contrast, another line investigates how to reconcile (a posteriori) preference criteria once they are aggregated.

In [3, 9], the authors consider preference relations definable by first-order formulas, and study the preservation of properties by different composition operators. For example, Chomicki [3] shows that prioritized composition preserves the properties of weak orders, while it does not preserve the ones of strict partial orders. Furthermore, he studies the problem of preference refinement [4], a special case of preference revision in which the original preferences are augmented with new preferences. In this working paper the authors do not consider conflicting preferences, but instead assume the old and new preference relations are compatible.

A new method for preference aggregation is studied in [8] by Grosof. This method generalizes the lexicographic combination method and is applicable to conflict management problems as a qualitative "weak method".

The problem of fairness of preference aggregation systems is studied by Rossi et al [11]. They generalize Arrow's impossibility theorem for combining total orders [2] to the case of partial orders.

Yager [13] and Rossi et al [12] investigated the problem of preference aggregation in the context of multi-agent systems. The approach outlined by Yager supports different types of relationship with respect to the importance of the interacting agents (e.g., total and weak order). He studied also the aggregation of fuzzy sets representing individual agent preferences. Rossi et al proposed mCP nets, a formalism that extends CP nets to handle preferences of several agents. They provided a number of different semantics for reasoning with mCP nets based on the notion of voting.

In contrast to the above mentioned approaches, we followed the second line of research. Basically, two distinct approaches have been proposed in literature to tackle the problem of amalgamating distributed data:

- paraconsistent formalisms, in which the amalgamated data may remain inconsistent. The idea is to answer queries in a consistent way without computing the revision of the amalgamated data.
- coherent methods, in which the amalgamated data is revised to restore consistency.

We have considered a coherent method to handle the problem of revising (a posteriori) preference criteria. Our approach is flexible and general, and is tailored neither to any specific preference criteria nor to any specific method for preference aggregation. Furthermore, the ability to express meta-information on the diagnoses of a revision problem gives us a further level of abstraction allowing us to select the best diagnosis for the problem at hand. Our approach shares the principle of minimal change with classical belief revision [6]. However, the basic

theoretical setting is different belief revision being concerned with the revision of finite theories.

We have implemented an algorithm to compute minimal diagnoses by using a version of ABDUAL [1] for constructive negation with abducibles. ABDUAL extends Well-Founded semantics with abducibles and integrity constraints. By means of the latter it can also compute Stable Models, where default literals are envisaged as constrained abducibles. Alternatively, ABDUAL is implemented in XSB-Prolog and so, by means of its XASP version that connects it to the Smodels implementation, stable models can be computed where relevant abducibles for a query are coded into even loops over default negation. The role of the ABDUAL is then to identify the relevant abducibles and send the residue program plus the so coded abducubles to Smodels.

Furthermore, when there exist no abducubles, ABDUAL is polynomial on the size of the program. When there are abducibles, because of its program transformation, ABDUAL computes abductive solutions depth-first, one at a time, via backtracking. So, if solution minimality is not required, a satisfying solution for preferences revision may found with no need to compute them all first. A study of the complexity of ABDUAL can be found in [1].

In the near future, we plan to combine our work on preference updating [5] with preference revision.

References

1. J. J. Alferes, L. M. Pereira, and T. Swift. Abduction in Well-Founded Semantics and Generalized Stable Models via Tabled Dual Programs. *Theory and Practice of Logic Programming*, 4(4):383–428, 2004.
2. K. Arrow. *Social Choice and Individual Values*. John Wiley and Sons, 1951.
3. Jan Chomicki. Preference Formulas in Relational Queries. *ACM Transactions on Database Systems*, 28(4):427–466, 2003.
4. Jan Chomicki and Joyce Song. On Preference Refinement. Paper in progress, 2004.
5. P. Dell'Acqua and L. M. Pereira. Preferring and updating in logic-based agents. In O. Bartenstein, U. Geske, M. Hannebauer, and O. Yoshie (eds.), *Web-Knowledge Management and Decision Support. Selected Papers from the 14th Int. Conf. on Applications of Prolog (INAP)*, LNAI 2543, pp. 70–85, 2003.
6. P. Gärdenfors and H. Rott. Belief Revision. In D. M Gabbay, C. J. Hogger, and J. A. Robinson (eds.), *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 4, pp. 35–132. Oxford University Press, 1995.
7. A. V. Gelder, K. A. Ross, and J. S. Schlipf. The Well-Founded Semantics for General Logic Programs. *J. ACM*, 38(3):620–650, 1991.
8. B. N. Grosof. New Prioritization Methods for Conflict Management. In M. Klein (ed.), *Proc. IJCAI-93 W. on Computational Models of Conflict Management in Cooperative Problem-Solving*. Int. Joint Conf. on Artificial Intelligence, 1993.
9. Andréka H., M. Ryan, and P. Y. Schobbens. Operators and Laws for Combining Preference Relations. *J. of Logic and Computation*, 12(1):13–53, 2002.
10. L. M. Pereira, C. Damásio, and J. J. Alferes. Debugging by Diagnosing Assumptions. In P. Fritzson (ed.), *1st Int. Ws. on Automatic Algorithmic Debugging, AADEBUG'93*, LNCS 749, pp. 58–74. Preproceedings by Linköping Univ., 1993.

11. F. Rossi, K. B. Venable, and T. Walsh. Aggregating Preference Cannot be Fair. Preprint n. 12-2004, Dept. of Pure and Applied Mathematics, Univ. of Padova, Italy, 2004.
12. F. Rossi, K. B. Venable, and T. Walsh. mCP nets: Representing and Reasoning with Preferences of Multiple Agents. In D. L. McGuinness and G. Ferguson (eds.), *Proc. 19th Conf. on Artificial Intelligence, 16th Conf. on Innovative Applications of Artificial Intelligence*, LNCS 749, pp. 729–734. AAAI Press, 2004.
13. Ronald R. Yager. Fusion of Multi-Agent Preference Ordering. *Fuzzy Sets and Systems*, 117:1–12, 2001.