Abstract.
In this paper we develop a language for representing complex qualitative preferences among problem solutions. We use ranked knowledge bases to represent prioritized goals. A basic preference description, that is a ranked knowledge base together with a preference strategy, defines a preference relation on models which represent problem solutions. Our language allows us to express nested combinations of preference descriptions using various connectives. This gives the user the possibility to represent her preferences in a natural, concise and flexible manner.

1 Introduction

In this paper we develop a language for specifying complex, qualitative preferences among potential problem solutions. Preferences play a crucial role in many areas of AI: in soft constraint solving constraints may have different priority, in decision making or planning some goals may be more important than others, in configuration some properties of the system to be designed are more critical than others, and so on.

By a solution we mean an assignment of a certain value \( d \) to each variable \( v \) in given set of variables \( V \) such that \( d \) is taken from the finite domain of \( v \). Without loss of generality, we will restrict our discussion here to the boolean case where the values for each variable are \( \text{true} \) or \( \text{false} \). Solutions thus correspond to interpretations in the sense of classical propositional logic. Moreover, we also assume that background knowledge may be given in the form of a set of propositional formulas \( B \). This background knowledge further constrains the set of interpretations: only models of \( B \) are considered as potential solutions. We are thus looking for ways of specifying preferences among such models in a concise yet flexible way.

The number of models is exponential in the number of variables. For this reason it is, in general, impossible for a user to describe her preferences by enumerating all pairs of the preference relation among models. This is where logic comes into play.

Traditionally, logic is used for proving theorems. Here, we are not so much interested in logical consequence, we are interested in whether a model satisfies a formula or not. In the simplest case we can use a single formula \( f \), interpret it as a goal, and say a model \( m_1 \) is preferred to model \( m_2 \) (denoted \( m_1 \rightarrow m_2 \)) iff \( m_1 \models f \) and \( m_2 \not\models f \).

In the general case, a single formula is not sufficient and we need a set of formulas \( F \) rather than a single formula. We obviously may have more than one goal. Since it is often impossible to satisfy all of them, a preference relation among the elements of \( F \) is useful to distinguish important from less important goals.

To express the preferences among goals we will use ranked knowledge bases (RKBs) in this paper \([8, 3, 14, 12]\) which are sometimes also called stratified knowledge bases. Such knowledge bases have proven fruitful in a number of approaches. A brief introduction will be given in the next section. Intuitively, the rank \( \text{rank}(f) \) of a formula \( f \) in an RKB is an integer expressing its relative importance.

It is important to note that an RKB alone is not sufficient to determine the preference relation on models, even if all formulas are interpreted as goals. For instance, the RKB does not tell us whether a model satisfying 2 goals of the same high rank is better than a model satisfying only 1 such goal. We need in addition a recipe of how to use the RKB for this purpose, in other words, we need a preference strategy.

Although the use of integers is convenient, RKBs are often used in a purely qualitative way where the actual numbers are irrelevant. What counts is only the total preorder \( \geq \) on formulas represented through the integers, where \( f_1 \geq f_2 \iff \text{rank}(f_1) \geq \text{rank}(f_2) \).

Our focus in this paper will be entirely on these qualitative approaches. This excludes, for instance, approaches which consider ranks as rewards and maximize their sum, as is often done in soft constraint satisfaction \([15]\). For an excellent overview of some of these approaches see \([13]\). Numerical approaches certainly are highly interesting. Nevertheless, we believe that they are better treated in the realm of classical decision theory. The strength of RKBs lies in their potential for modeling qualitative preferences.

We will thus restrict our discussion to qualitative strategies which have been used in combination with an RKB. Different strategies reflect different meanings a user can associate with the importance ranks. Since there is no single best reading of such ranks, there is no single best strategy. We therefore believe it is important to give users the ability to choose and possibly combine different strategies in flexible ways.

Our main contribution is thus a language for defining complex preferences among models. Complex preferences may arise because a single agent uses different strategies for different aspects of a problem, which then must be combined. They may also arise because the preferences of multiple agents have to be combined. The basic building blocks of our language are pairs consisting of a strategy and an RKB. The language also allows for (nested) combinations of preference expressions using different combination methods.

Throughout the paper, the RKBs we use contain formulas representing goals or desires. Independently of the chosen strategy, making more formulas true can never decrease the quality of a model. Some authors have also investigated rejections, that is formulas which should be falsified \([4]\). It turns out that the rejection of \( p \) can be
modeled using the goal ρ, given an adequate strategy. Our choice of a goal based approach thus does not reduce generality.

The rest of the paper is organized as follows. In the next section we give a brief reminder on ranked knowledge bases. Section 3 then introduces basic preference expressions, consisting of an RKB together with one of 4 qualitative strategies. We also investigate their relationship. Section 4 defines our full preference description language. In this language, expressions can be combined using various operators. Section 5 illustrates our language using a movie selection example. Section 6 discusses related work and concludes.

2 Ranked Knowledge Bases

A ranked knowledge base (RKB), sometimes also called stratified knowledge base, is a set F of propositional formulas together with a total preorder ≥ on F. A preorder is a transitive and reflexive relation, totality means that for each f1, f2 ∈ F we have f1 ≥ f2 or f2 ≥ f1. Usually, RKBs are represented in one of the following ways:

1. as a sequence (F1, . . . , Fn) of sets of formulas such that f1 ≥ f2 iff for some i, j; f1 ∈ Fj, f2 ∈ Fi and i ≥ j.
2. as a set of ranked formulas (f1, r1), where f1 is a propositional formula and r1, the rank of f1, is a non-negative integer such that fj ≥ fjk ifrj ≥ rk.

The two representations of RKBs are clearly equivalent: the rank of a formula corresponds to the set index in the first formulation. For convenience we will mostly use the second one in this paper. Note that starting from a pair (F, ≥) one always gets a set of ranked formulas where each formula has a unique rank.2

Intuitively, we consider formulas with higher rank to be more important than those with lower rank.3 The exact meaning of the ranks depends on the chosen preference strategy.

Different ways of defining consequence relations for RKBs have been defined in the literature. In [8] an inclusion based method was used to define preferred maximal consistent subsets (called preferred subtheories in [8]) of the premises. A maximal subset S1 is strictly preferred to S2 iff there is a rank r such that the formulas of rank r in S1 are a proper superset of those in S2, and for all ranks higher than r, S1 and S2 agree on the contained formulas. Benferhat and colleagues [3] investigated ranked knowledge bases under a cardinality based criterion. To define preferred maximal consistent subsets, they take the number of formulas satisfied in a particular stratum into account. System Z [14, 12] generates a ranking from a knowledge base of rules which gives more importance to more specific rules. Intuitively, to determine whether a model M is preferred, the lowest rank r is considered for which M satisfies all rules of degree r and higher. A close connection between System Z and possibilistic logic was established in [5]. The major difference is that possibilistic logic uses reals in the unit interval rather than integers.

In a possibilistic setting, Benferhat and colleagues [4] investigated bipolar preferences based on the maximal degree of a satisfied goal (a model is better the higher the maximal degree) and the maximal degree of a satisfied rejection (a model is the better the smaller the maximal degree).

Since all of these strategies from the literature are of interest, the language to be developed in the next sections will allow the user to pick the one she has in mind when specifying preferences through a ranked knowledge base, and to combine them in a flexible manner.

3 Basic preference expressions

In this and the following section we define the language LPD for expressing complex preferences among models. We identify 4 basic qualitative strategies which we consider fundamental, given preferences among goals are specified using RKBs. In our language we use identifiers taken from the set

$$Strat = \{T, \kappa, \subseteq, \#\}.$$ 

for particular strategies. The meaning of these identifiers will be defined shortly.

Definition 1 A basic preference description is a pair (s, K) consisting of a basic strategy identifier s and an RKB K.

Rather than using pair notation (s, {(f1, r1), . . . , (fn, rn)}) for (s, K), we will often use a strategy identifier as an upper index for the RKB, that is, we write {((f1, r1), . . . , (fn, rn))}κ or Kκ, respectively.

A basic preference description defines a preorder ≥ (that is, a transitive and reflexive relation) on models. As usual, the preorder implicitly defines an associated strict partial order defined by m1 ≥ m2 iff m1 ≥ m2 and not m2 ≥ m1.

Let K = {((fi, vi))} be an RKB, s a basic strategy name. We use ≥κ to denote the preorder on models defined by (s, K). We first introduce the following notation and auxiliary definitions:

$$K^n(m) = \{f \mid (f, n) \in K, m \models f\}$$

$$\text{maxsat}^K(m) = -\infty \text{if } m \not\models f_i \text{ for all } (f_i, v_i) \in K,$$

$$\text{maxunsat}^K(m) = -\infty \text{if } m \models f_i \text{ for all } (f_i, v_i) \in K,$$

$$\text{max}\{i \mid (f, i) \in K, m \not\models f\}.$$ 

Now we can define the corresponding orderings on models:

- m1 ≥κ m2 iff maxsatκ(m1) ≥ maxsatκ(m2).
- m1 ≥κ m2 iff maxunsatκ(m1) ≤ maxunsatκ(m2).
- m1 ≥κ m2 iff Kκ(m1) = Kκ(m2) for all n, or there is an n such that Kκ(m1) ⊆ Kκ(m2), and for all j > n: Kκ(j) = Kκ(j).
- m1 ≥κ m2 iff Kκ(m1) = Kκ(m2) for all n, or there is an n such that Kκ(m1) > Kκ(m2), and for all j > n: Kκ(j) = Kκ(j).

The strategies can be described informally as follows:

- ⊤ prefers m1 over m2 whenever the most important goal satisfied by m1 is more important than the most important goal satisfied by m2. It was used in [4] in the context of bipolar representations. With this strategy the intuitive reading of (f, r) is: if f is true, then the total satisfaction is at least r.
- κ prefers m1 over m2 whenever the most important goal not satisfied by m1 is less important than the most important goal not satisfied by m2, in other words, if the rank r such that all goals of rank r and higher are satisfied is lower in m1 than the corresponding rank in m2. This is the κ-ranking used in System Z.
• to check whether \( \subseteq \) prefers \( m_1 \) over \( m_2 \) we start from the most important goals and go down stepwise to less important ones. If, at the first rank reached this way for which the formulas satisfied by
the two models differ, we have that \( m_1 \) satisfies a superset of the formulas satisfied by \( m_2 \), then \( m_1 \) is preferred. This is the order used in [8].

• \( \# \) is similar to \( \subseteq \), but rather than checking the sets of formulas satisfied for each rank, their cardinality is considered. This is the proposal of Benferhat and colleagues in [3].

Among the preorders on models generated by these strategies only \( \succsim_\kappa \) is partial. The others are total, that is, the ordering on models is again a ranking. To illustrate the strategies let us consider the RKB:

\[
K = \{(a, 2), (b, 2), (c, 2), (d, 1), (e, 1)\}
\]

We represent models by a sequence of atoms true in the model. For example, \( acd \) represents the model in which \( a, c \) and \( d \) are true, \( b \) and \( e \) are false. Also, whenever \( K \) is clear from context we omit the upper index \( K \) from the relation symbols. We have \( ad \succsim_\kappa de \) since \( ad \), contrary to \( de \), satisfies a goal of rank 2. On the other hand, \( ad \nsuccsim_\kappa de \) since both models falsify a goal of rank 1. Furthermore, \( abc \succsim_\kappa bd \) since \( abc \) satisfies all goals of rank 2, that is, the maximal rank of a violated goal is 1. On the other hand \( abc \nsgtr \kappa bd \) since both satisfy a goal of rank 2. \( abd \) is incomparable to \( cd \) according to \( \subseteq \), however \( abd \succsim_\kappa cd \) since the former satisfies two goals of rank 2.

The different strategies are not independent of each other. We have the following results:

**Proposition 2** Let \( m_1 \) and \( m_2 \) be models, \( K \) a ranked knowledge base. The following relationships hold:

\[
\begin{align*}
m_1 \succsim_\kappa m_2 & \text{ implies } m_1 \succsim_\kappa m_2, \\
m_1 \nsgtr_\kappa m_2 & \text{ implies } m_1 \nsgtr_\kappa m_2, \\
m_1 \nsgtr_\kappa m_2 & \text{ implies } m_1 \nsgtr_\kappa m_2, \\
m_1 \nsgtr_\kappa m_2 & \text{ implies } m_1 \nsgtr_\kappa m_2.
\end{align*}
\]

The first 4 relationships can be illustrated using the following figure:

\[
\begin{array}{c}
\top \quad > \quad \subseteq \quad \nsgtr \quad \# \\
K
\end{array}
\]

**Fig.1: Relationship among basic orderings**

More relationships can be established if we allow \( K \) to be modified.

**Proposition 3** Let \( K \) be a ranked knowledge base, \( m_1 \) and \( m_2 \) models. Let

\[
K_\land = \{(C_i, i) \mid C_i \text{ conjunction of all } f \text{ with } (f, j) \in K, j \geq i\},
\]

\[
K_\lor = \{(C_i, i) \mid C_i \text{ disjunction of all } f \text{ with } (f, j) \in K, j \geq i\}.
\]

Then \( m_1 \succsim_\kappa m_2 \iff m_1 \succsim_\kappa m_2 \) and \( m_1 \nsgtr_\kappa m_2 \iff m_1 \nsgtr_\kappa m_2 \).

Moreover, since \( \subseteq \) and \( \# \) are equivalent if for each rank there is only a single formula possessing this rank, the proposition also holds if we use \( \# \) instead of \( \subseteq \).

Ranked rejections [4] can be modeled using the \( \kappa \)-strategy:

**Lemma 4** Let \( \mathcal{R} = \{(R_1, v_1), \ldots, (R_n, v_n)\} \) be a set of rejections. \( m_1 \) is more acceptable than \( m_2 \) (see [4], Sect.4) \( \text{iff } m_1 \succsim_\kappa m_2 \) where \( \mathcal{R}' = \{\neg R_1, v_1), \ldots, (\neg R_n, v_n)\} \).

**4 The preference language**

So far we discussed basic preference descriptions only. A user may have different ways of modeling her preferences for different aspects of a problem. Therefore, we also want to allow more complex descriptions representing combinations of the corresponding preorders.

We now give the full definition of our logical preference description language. For reasons which will become clear later, we use standard propositional connectives together with a new connective \( > \) expressing preference among expressions.

**Definition 5** The logical preference description language \( \text{LPD} \) is inductively defined as follows:

1. Each basic preference description is in \( \text{LPD} \).
2. If \( d_1 \) and \( d_2 \) are in \( \text{LPD} \), then the expressions \( (d_1 \land d_2) \), \( (d_1 \lor d_2) \), \( (d_1 > d_2) \) and \( \neg d_1 \) are in \( \text{LPD} \).

The formal definition of the meaning of a (non-basic) \( \text{LPD} \) expression, that is the definition of its associated preorder on models, is as follows:

**Definition 6** Let \( R_1 \) and \( R_2 \) be the preorders on models represented by \( d_1 \) and \( d_2 \), respectively. Let \( \text{tr}(R) \) denote the transitive closure of a relation \( R \), \( R^{-1} \) the inverse relation. \( \text{Ord}(\text{lpd}) \), the preorder represented by the complex \( \text{LPD} \) expression \( \text{lpd} \), is defined as follows:

\[
\begin{align*}
\text{Ord}(d_1 \land d_2) &= R_1 \cap R_2, \\
\text{Ord}(d_1 \lor d_2) &= \text{tr}(R_1 \cup R_2), \\
\text{Ord}(\neg d_1) &= R_1^{-1}, \\
\text{Ord}(d_1 > d_2) &= (R_1 \cap R_2) \cup (R_2 \setminus R_1^{-1}).
\end{align*}
\]

d_1 \land d_2 \) corresponds to the well-known Pareto ordering: a model \( m_1 \) is at least as good as \( m_2 \) if it is at least as good as \( m_2 \) with respect to both \( d_1 \) and \( d_2 \). \( m_1 \) is strictly better if it is better according to one of the suborders, and at least as good as \( m_2 \) with respect to the other. \( d_1 \lor d_2 \) considers \( m_1 \) at least as good as \( m_2 \) if at least one of the composed orderings does (or one of the agents if \( d_1 \) and \( d_2 \) represent preferences of 2 agents). The definition needs the transitive closure since the union of two orderings is not necessarily transitive. The \( > \) operator just reverses the original ordering. Double application of \( > \) obviously gives back the original ordering. Note, however, that other properties of negation do not hold for \( > \), in particular the de Morgan laws do not hold. For instance, \( (\neg d_1 \land d_2) \) differs from \( (\neg d_1 \land \neg d_2) \).

\( d_1 > d_2 \) is the lexicographic ordering of \( R_1 \) and \( R_2 \) which gives more priority to \( R_1 \) and uses \( R_2 \) only to distinguish between models which are equally good w.r.t. \( R_1 \). \( m_1 \) is strictly better than \( m_2 \) if it is strictly better w.r.t. \( R_1 \), or as good as \( m_2 \) w.r.t. \( R_1 \) and strictly better w.r.t. \( R_2 \). \( R_1 \setminus R_1^{-1} \) is the strict partial order associated with \( R_1 \).

The binary operators \( \lor, \land \) and \( > \) are associative. We omit brackets if this does not cause confusion, assuming binding strength decreases in the order \( \land, \lor, > \).

The language \( \text{LPD} \) gives us flexible means of representing preferences on models. We next discuss some properties of the language.

Under certain circumstances expressions can be simplified. We say a preference expression \( d_1 \) implies an expression \( d_2 \) iff \( \text{Ord}(d_1) \subseteq \text{Ord}(d_2) \).
Ord(d_2). We say two preference expressions are equivalent if they induce the same preorder on models, that is, iff $Ord(d_1) = Ord(d_2)$. Let $s \in Strat$ be any of our basic strategies, then:

$$\{(f_1, r_1), \ldots, (f_n, r_n)\}^s > \{(s_1, r'_1), \ldots, (s_m, r'_m)\}^s$$

is equivalent to

$$\{(f_1, c + r_1), \ldots, (f_n, c + r_n), (s_1, r'_1), \ldots, (s_m, r'_m)\}^s \quad \text{where} \quad c = \max\{r'_i\} + 1.$$

Note that this result depends on the fact that the two basic preference expressions use the same strategy. A similar result for different strategies does not hold. Also, for $\land$ such simplifications are not possible, even if the strategies of the subexpressions coincide. The only weak result we get is:

**Proposition 7** Let $K_1$ and $K_2$ be RKBs.

$$(K_1^\land \land K_2^\land) \implies (K_1 \land K_2)^\land.$$  

The other direction does not hold (to see this, consider the case where we split an RKB such that formulas with high rank are in $K_1$, formulas with low rank in $K_2$. For the cardinality based strategy, using the union of 2 RKBs, that is $(K_1 \cup K_2)^\land$, clearly is different from $(K_1^\land \land K_2^\land)$. In the general case complex expressions are not reducible to single ones which use the same formulas, even if the ranks are allowed to change.

### 5 Example: Selecting a Movie

In this section we want to illustrate the use of our language with a commonsense example. Assume you are planning to go to the cinema with your girl friend. Both of you prefer comedies over action movies over tragedies. Your girl friend loves to see Hugh Grant and Brad Pitt, followed by Leonardo di Caprio. Your favourite actors are Julia Roberts and Nicole Kidman, followed by Gwyneth Paltrow and Halle Berry. You both feel that the type of movie is as important as the actors. Moreover, since it is your girl friend’s birthday, her actors’ preferences are more important today than yours.

We can represent this information using the following RKBs:

- $K_1 = \{(\text{Hugh}, 2), (\text{Brad}, 2), (\text{Leo}, 1)\}$
- $K_2 = \{(\text{Julia}, 2), (\text{Nicole}, 2), (\text{Gwyneth}, 1), (\text{Halle}, 1)\}$

We assume the background knowledge contains information that the mentioned types of movies are mutually exclusive, models thus will make at most one of the types true. Since seeing more of the favourite actors is more fun we use the cardinality based strategy. Our preferences can thus be represented as the LPD expression:

$$(K_1^\land > K_2^\land) \land K_4^\land$$

Assume we have the following information about the movies shown tonight:

- $M_1 : \text{comedy, Hugh, Brad}$
- $M_2 : \text{comedy, Hugh, Leo, Julia}$
- $M_3 : \text{comedy, Brad, Leo, Julia, Halle}$
- $M_4 : \text{action, Brad, Hugh, Nicole}$
- $M_5 : \text{action, Brad, Leo, Julia, Halle}$
- $M_6 : \text{tragedy, Brad, Leo, Julia, Nicole}$

We assume that the list of actors mentioned for each movie is complete, that is, if one of the names appearing in the RKBs is not listed, then this actor is not in the corresponding movie.

We represent the information listed above in the background knowledge in the form of logical implications. For instance, for $M_1$ we get:

$M_1 \rightarrow \text{comedy} \land Hugh \land Brad \land \neg Leo$

$\land \neg Julia \land \neg Nicole \land \neg Gwyneth \land \neg Halle$

We also represent that exactly one of the 6 movies needs to be chosen, that is exactly one of $\{M_1, \ldots, M_6\}$ must be true in each model. All models thus contain one selected movie together with its type and its actors.

According to our preference expression, $M_1$ is preferred over $M_2$ and over $M_3$ because two of your girl friend’s most favourite actors play in $M_1$. $M_4$ is preferred over $M_2$ since it is as good with respect to your girl friend’s preferences (trading Hugh for Brad), but better according to your preferences since it additionally gives you Halle.

$M_4$ and $M_6$ are incomparable: $M_1$ is the better type of movie, but $M_6$ is better with respect to its actors. $M_5$ is worse than both $M_4$ (worse actors according to your girl friend) and $M_3$ (worse type), and thus also worse than $M_1$. $M_6$ is less preferred than both $M_4$ and $M_1$: it has less preferred actors and a worse type. $M_6$ is incomparable to $M_5$.

The only non-dominated movies are thus $M_1$ and $M_4$. The preference structure among models (represented through the selected movies) is illustrated in the following figure (arrows point to strictly preferred models):

![Fig.2: Strict preferences among movies](image-url)

### 6 Discussion

In this paper we developed a flexible preference representation language. The basic building blocks of the language are ranked knowledge bases together with a model selection strategy. Ranked knowledge bases allow us to represent prioritized goals conveniently. We investigated four different strategies known from the literature, all of them qualitative in the sense that the induced total preorder on formulas is what counts, rather than the actual numbers.

Our language also allows for combinations of preference expressions. Conjunction naturally leads to Pareto orderings based on the underlying subexpressions. The connective $\land$ allows us to define lexicographic orderings. The language also has disjunction and a form of negation which simply reverses the original order.

The work presented in this paper shares some motivation with [9]. Also in that paper a language, called PDL, for expressing complex preferences is presented. However, there are several major differences which are due to the fact that PDL is tailored towards answer set optimization:

1. **PDL** is rule based rather than goal based. The basic building blocks are rules with prioritized heads rather than ranked knowledge bases.
2. Since **PDL** is used to assess the quality of answer sets (i.e., sets of literals) rather than models, it becomes important to distinguish between an atom not being in an answer set and its negation being in an answer set. In other words, the distinction between classical negation and default negation (negation as failure) is relevant. Since we are interested in preferences among models here, this distinction does not play a role in **LPD**.
3. PDL distinguishes between penalty producing and other strategies. Both numerical and qualitative combination strategies are thus used. On the other hand, combinations corresponding to our disjunction and negation operators are lacking.

Although we restricted our discussion to purely qualitative approaches, there is no principle obstacle against integrating numerical approaches as well, at least at the level of basic preference expressions. For instance, we could use ranks as penalties or rewards and define the preorder on models on the basis of the actual rank values. The reader should be aware, though, that this only works on the basic level. The connectives we defined operate on the preorders and do not take numerical information into account. Any numerical information would thus be lost in our language at the level of complex preference expressions.

An interesting related paper is [16] which introduces a preference language for planning. The language is based on a temporal logic and is able to express preferences among trajectories. As in LPD, preferences can be combined via binary operators - somewhat different from ours. The major difference certainly is that our approach aims at being application-independent, whereas [16] focuses on planning.

Also related is [1]. The authors investigate combinations of priority orderings based on a generalized lexicographic combination method. This method is more general than usual lexicographic orderings - including the ones expressible through our $>$ operator - since it does not require the combined orderings to be linearly ordered. It is based on so-called priority graphs where the suborderings to be combined are allowed to appear more than once. The authors also show that all orderings satisfying certain properties derived from Arrow’s conditions [2] can be obtained through their method. This is an interesting result. On the other hand, we found it somewhat difficult to express examples like our movie example using the method. We believe our language is closer to the way people actually describe their preferences.

In [7] CP-networks are introduced, together with corresponding algorithms. These networks are a graphic representation, somewhat reminiscent of Bayes nets, for conditional preferences among feature values under the ceteris paribus principle. Our approach differs from CP-networks in several respects: (1) Preferences in CP-networks are always total orders of the possible values of a single variable. We are able to represent arbitrary prioritized goals. (2) The ceteris paribus interpretation of preferences is very different from our goal-based interpretation. The former views the available preferences as (hard) constraints on a global preference order. Each preference relates only models which differ in the value of a single variable. A set of ranked goals, on the other hand, is more like a set of different criteria in multi-criteria optimization. In particular, goals can be conflicting. Conflicting goals may neutralize each other, but do not lead to inconsistency.

Although our work was mainly motivated by several approaches developed in the area of nonmonotonic reasoning, many related ideas can be found in constraint satisfaction, in particular valued (sometimes also called weighted) constraint satisfaction [11, 10, 15, 6]. A valued constraint, rather than specifying hard conditions a solution has to satisfy, yields a ranking of solutions. A global ranking of solutions then is obtained from the rankings provided by the single constraints through some combination rule. This is exactly what happens in our approach on the level of basic preference expressions. Also in constraint satisfaction we find numerical as well as qualitative approaches. In MAX-CSP [11], for instance, constraints assign penalties to solutions, and solutions with the lowest penalty sum are preferred. In fuzzy CSP [10] each solution is characterized by the worst violation of any constraint. Preferred solutions are those where the worst violation is minimal. This corresponds to the $\kappa$ strategy. We are not aware of any approach in constraint satisfaction trying to combine different strategies. For this reason we believe the language developed here will be of interest also for the constraint community.

In future work we plan to investigate the use of partially ordered rather than ranked knowledge bases on the level of basic preference expressions. We also plan to investigate computational issues related the approach. In particular, it would be interesting to see whether a generate and improve method like the one developed for answer set optimization in [9] can be used here as well.

Acknowledgements

The paper greatly benefitted from discussions with and suggestions by Jérôme Lang. The author acknowledges support from DFG (Computationale Dialektik: BR 1817/1-5).

REFERENCES