Preferential Querying for the Semantic Web

Veronika Vaneková*
Institute of Computer Science
Faculty of Natural Sciences
Pavol Jozef Šafárik University in Košice
Jesenná 5, 04001 Košice, Slovakia
veronika.vanekova@upjs.sk

Abstract

We present a model of user preference based on fuzzy sets and aggregation. The model is implemented in a web-based system Kore enabling preferential search for many users with different preferences within one arbitrary domain. The implementation includes a tool for top-k search and a tool for learning user preference from rated objects. The system was tested by real users and we analyze their satisfaction with the results according to correlation coefficients. We propose a fuzzified description logic s-$\mathcal{EL}(\mathcal{D})$ as a formal background for the preference model. The description logic s-$\mathcal{EL}(\mathcal{D})$ contains crisp roles and fuzzy concepts that represent user preference. Fuzzy sets are a part of a concrete domain $\mathcal{D}$. Fuzzy membership values generate an order of individuals starting from the most preferred to the least preferred. Therefore we also define another description logic o-$\mathcal{EL}(\mathcal{D})$ where we dispose of the specific fuzzy values and we keep only the order of individuals. Then we study the relationship between s-$\mathcal{EL}(\mathcal{D})$ and o-$\mathcal{EL}(\mathcal{D})$ concepts. We also introduce reasoning algorithms for the two defined description logics.

Categories and Subject Descriptors

D.3.2 [**Language Classifications**]: [Expert system tools and techniques]; F.4.1 [**Mathematical Logic**]: [Representation languages]; F.4.1 [**Mathematical Logic**]: [Uncertainty, "fuzzy," and probabilistic reasoning, Deduction]; D.2.2 [**Design Tools and Techniques**]: [User interfaces]

Keywords

user preference, description logic, preferential querying

*Recommended by thesis supervisor:
Prof. Peter Vojtáš
Defended at Faculty of Natural Sciences, Pavol Jozef Šafárik University in Košice on September 20, 2010.

Vaneková, V. Preferential Querying for the Semantic Web. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 2, No. 2 (2010) 137-148

1. Introduction

One of the most challenging problems of current web is the enormous amount of available data. In addition to a standard task of finding some necessary piece of information, web users often need to perform more complex tasks like buying some item online, finding a hotel for holiday, booking a flight, applying for a job. Such tasks mean searching for the most suitable object from some domain (like notebooks, hotels, flights, job offers). User preference in such cases tends to be more complex than a sequence of keywords. It is closely related with various attributes like price, location, technical parameters, etc.

Of course, it is possible to find a hotel, notebook or other item of interest with Google and many users would do so, but it also requires some manual comparison of found results. This way we obtain too many results and it is very hard to filter out the irrelevant ones with keywords. For example, it is difficult to restrict the search only to notebooks with a dual-core processor and a price below 400 EUR (a dual-core processor can be also indicated with "Core 2 Duo", "Pentium D" or similar). We also lack the ability to order the search results according to various attributes like price.

Some e-shops and portals allow users to search for items according to the attributes and to specify their preferred values. The specified values are used by the system to generate a conjunctive query and evaluate it in a database of all items. However, if we specify one of the requirements as "a price below 400 EUR", the system would exclude also an item that costs 401 EUR, even if it has excellent values of all other attributes. Therefore using conjunctive queries sometimes leads to losing potentially relevant results. If a user specifies multiple requirements, it is not necessarily a conjunction. Some requirements can be conflicting (e.g. if the user wants both low price and high quality) and the user may sometimes dispense one requirement for the sake of other requirement. For example, the user may be able to pay a slightly higher price if it means a significantly higher quality. Instead of returning a set of items that fulfill the conjunction of the user's requirements, it is more appropriate to return a *sequence* of items starting from the most preferred to the least preferred.

This is the problem addressed by preferential search. According to one of the possible definitions used in economics, "preference refers to the set of assumptions relating to a real or imagined choice between alternatives and the possibility of rank ordering of these alternatives, based on the degree of happiness, satisfaction, gratifica-

tion, enjoyment, or utility they provide"[1]. This definition points out several interesting aspects of user preference – firstly, preference creates an *ordering* of objects (ordered from the most preferred to the least preferred) and secondly, this ordering is based on different *degrees of preference*. We will address both aforementioned aspects of preference – the representation of user preference, its acquisition and querying are the main topics of our research.

Our own model of user preference is described in Section 2. It is based on preferences to single attributes, specified as fuzzy sets of preferred attribute values. Instead of a conjunctive query, we process multiple requirements as a fuzzy aggregation. The preferential search is performed with top-k algorithm [6].

The model of user preference is then defined within an underlying formal language called *description logic*, which offers the advantage of automated reasoning about the knowledge. Section 3, contains our own fuzzified description logic together with the description of reasoning tasks, algorithms and theoretical results. Section 5 investigates an interesting problem of viewing preference within a description logic as an order of objects. We also study how the proposed description logics are related to one another in Section 6.

Section 7 focuses on an experimental implementation of a preferential search system. We describe the user interface, representation and access to data and also the software tools integrated in the system. Then we present the results of tests and experiments performed to evaluate the system.

2. Fuzzy Model of User Preference

Let us consider a user searching for a notebook. She can choose the best notebook according to many attributes: price, disk size, processor speed, RAM, screen diagonal, graphic card type, number of USB ports, weight, battery life, eventually according to specific accessories like infraport, bluetooth and various memory card slots. The user usually considers only a subset of all possible attributes, so we will also use only a few attributes for the purpose of this example. Table 1 shows a sample dataset that will occur in several examples. The data is collected from different web pages offering notebooks.

Table 1: A sample set of four notebooks with attribute values.

id	brand	price (EUR)	speed (GHz)	screen
nb1	Acer	365	2.1	16”
nb2	Asus	500	2.2	17”
nb3	Toshiba	647	1.2	12,1”
nb4	Lenovo	986	1.66	16”

Table 2: A sample dataset of four notebooks with preference values of user `U1`.

id	$\texttt{Cheap}_{\texttt{U1}}$	$\texttt{Fast}_{\texttt{U1}}$	$\texttt{Widescreen}_{\texttt{U1}}$	preferred
nb1	1	0.66	0.5	0.8
nb2	0.66	0.72	1	0.73
nb3	0.18	0.12	0	0.13
nb4	0	0.4	0.5	0.22

[1] `http://en.wikipedia.org/wiki/Preference`

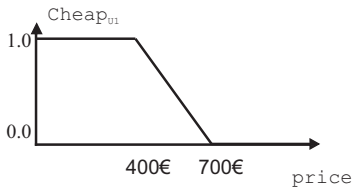

Figure 1: Attribute preference $\texttt{Cheap}_{\texttt{U1}}$ with left-trapezoidal function.

If the user searches for cheap, fast, widescreen notebook, her possible attribute preferences are shown on the figures 1, 2 and 3. For example, Figure 1 shows that the user `U1` is fully satisfied with notebooks that cost up to 400 EUR. However, she does not prefer notebooks over 700 EUR at all. Notebooks with price between 400 and 700 EUR are preferred partially. Notebook `nb1` from the table above is cheap to a degree 1, while `nb2` only to 0.66.

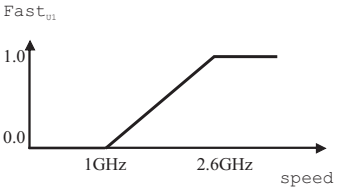

Figure 2: Attribute preference $\texttt{Fast}_{\texttt{U1}}$ with right-trapezoidal function.

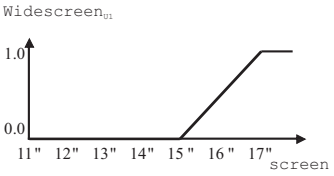

Figure 3: Attribute preference $\texttt{Widescreen}_{\texttt{U1}}$ with right-trapezoidal function.

Every membership functions shows where an “ideal” value (or an interval of ideal values) of the attribute lies. $\texttt{Cheap}_{\texttt{U1}}$ from Figure 1 shows that an ideal price for user `U1` would be from 0 to 400. However, the ideal value depends on the user. Other user’s ideal prices could be e.g. from 0 to 600. This is the reason why we add the user identifier like `U1`, `U2` into the name of the fuzzy set.

An attribute preference also depends on the attribute domain. The attributes mentioned above (`price`, `speed` and `screen`) have numeric values, so they have a natural ordering. Such attributes are called *ordinal*. If two attribute values are near on the x-axis, their degrees of preference are usually also similar. Thus user preference can be often represented with basic fuzzy sets of triangular and trapezoidal shape (see Figure 4).

The first type (left-trapezoidal function) is typical for attributes like `price`. Products with lower prices are preferred more. Right-trapezoidal function is the most common. It occurs for attributes like speed, power and capacity, where we can say “the more, the better”. If the user prefers one value (or an interval of values) somewhere around the middle of the attribute domain, we get the

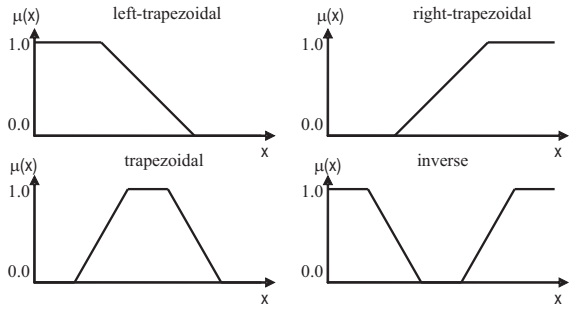

Figure 4: Basic types of membership functions.

trapezoidal type. This may be the case of the attribute `screen`. The user may prefer 14" screen, while 12" may already be too small and 17" may be considered too large and unwieldy. The last type (inverse) is not very likely to occur in the domain of notebooks, but it is possible e.g. in the domain of hotels, for the attribute `distance` (from the city center). This type would mean that the user wants to live either very close to the center or somewhere in a quiet suburb.

Some attribute domains can not be naturally ordered. Such attributes are called *nominal* or *crisp* attributes. The attribute `brand` is an example from the domain of notebooks. There is only lexicographic order of the words, but it has nothing to do with preference. Values that are near in the lexicographic order do not have to be preferred in similar degrees. However, most users have some preferred brands. This preference can be represented as a classical (crisp) set of values, but we use fuzzy sets that allow some values to be preferred more than the others. An example can be $\mu(\texttt{Toshiba}) = 0.9$, $\mu(\texttt{Lenovo}) = 0.8$ and $\mu(\texttt{Acer}) = 0.7$. Other values will be preferred to the degree 0. Compared to nominal attributes, we are not able to determine the type of the fuzzy set here. Thus nominal attributes can be a little more complicated for preferential search and indexing.

If the user has a preference to e.g. three attributes, we end up with three different preference values (one for each attribute). The overall preference value is obtained with her global preference (aggregation). It is formally defined as a function $@^\bullet : [0,1]^n \to [0,1]$, which is monotone in all arguments and $@^\bullet(1,\ldots,1) = 1$ must hold. One possible aggregation function is a weighted average. The weights indicate how much the corresponding attribute is important for the user. If the user cares about price and speed of the notebook more than about screen, her aggregation function can be:

$$\frac{3 \cdot \texttt{Cheap}_{\texttt{U1}}(\texttt{x}) + 2 \cdot \texttt{Fast}_{\texttt{U1}}(\texttt{x}) + \texttt{Widescreen}_{\texttt{U1}}(\texttt{x})}{6}$$

The preference degrees for attributes `price`, `speed` and `screen` and overall preference using the aggregation function above, are shown in Table 2. Notebook `nb1` is the most preferred because the weight related with `Cheap` is quite high. If we used other weights, e.g. 1 for $\texttt{Cheap}_{\texttt{U1}}$, 1 for $\texttt{Fast}_{\texttt{U1}}$ and 2 for $\texttt{Widescreen}_{\texttt{U1}}$, then `nb1` would be preferred to 0.67 and `nb2` to 0.85. It is easy to see that the weights (and more generally, the choice of aggregation function) have a significant influence on the result.

Other possible aggregation functions are OWA operators [5], and also fuzzy t-norms and t-conorms. Aggregation can be viewed as a generalization of both fuzzy conjunction and fuzzy disjunction. It is also possible to represent global preference with a set of rules learned from a rated sample of objects [8].

- $\texttt{GoodNotebook}_{\texttt{U1}}(\texttt{x}) > 0.8$ IF $\texttt{Cheap}_{\texttt{U1}}(\texttt{x}) > 0.8$ AND $\texttt{Fast}_{\texttt{U1}}(\texttt{x}) > 0.5$
- $\texttt{GoodNotebook}_{\texttt{U1}}(\texttt{x}) > 0.7$ IF $\texttt{Cheap}_{\texttt{U1}}(\texttt{x}) > 0.6$ AND $\texttt{Widescreen}_{\texttt{U1}}(\texttt{x}) > 0.4$ AND $\texttt{Fast}_{\texttt{U1}}(\texttt{x}) > 0.5$

Every rule consists of a head (e.g. $\texttt{GoodNotebook}_{\texttt{U1}}(\texttt{x}) > 0.8$) and a body. The body is a conjunction of clauses like $\texttt{Cheap}_{\texttt{U1}}(\texttt{x}) > 0.8$. If the conjunction is satisfied for some object `x`, then it will be preferred at least in the degree specified in the head of the rule. However, the object can be preferred to a higher degree as well. If the object satisfies more than one rule, its overall preference will be maximum from the values specified in rule heads. For example, consider the notebook `nb1` from Table 2 – it has local preferences $\texttt{Cheap}_{\texttt{U1}}(\texttt{nb1}) = 1$, $\texttt{Fast}_{\texttt{U1}}(\texttt{nb1}) = 0.66$, $\texttt{Widescreen}_{\texttt{U1}}(\texttt{nb1}) = 0.5$. It is easy to see that both rule bodies above are satisfied. Thus both inequalities $\texttt{GoodNotebook}_{\texttt{U1}}(\texttt{nb1}) > 0.8$ and $\texttt{GoodNotebook}_{\texttt{U1}}(\texttt{nb1}) > 0.7$ must hold. The overall preference will be 0.8, so both inequalities are satisfied.

3. Scoring Description Logic $s\text{-}\mathcal{EL}(\mathcal{D})$

Description logics (DLs) denote a group of formal languages for knowledge representation. Their main advantage is the ability of reasoning, i.e. inference of the implicit knowledge. The knowledge is represented with *concepts* and *roles*. Concepts can be viewed as unary predicates and roles as binary predicates, expressing the relationships within the domain. There are several description logics differing with their complexity and expressive power. The difference is in the syntactic constructors allowed to create new, complex concepts. The more constructors we allow, the more complex will the resulting DL be. This also leads to a higher complexity of reasoning tasks.

The language of our description logic $s\text{-}\mathcal{EL}(\mathcal{D})$ consists of a set of *atomic concept* names N_C, *atomic role* names N_R, individual names N_I and constructors N_K. DL knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. The TBox contains all necessary definitions of complex concepts. Every definition (TBox axiom) has the form $C \equiv D$ where C is a name of the new concept and D is an expression made of simpler concepts and constructors. This definition is treated as a logical equivalence. ABox can use complex concept and role names defined in the TBox to create *assertions* about individuals. We distinguish role assertions $R(a,b)$ and concept assertions $\langle C(a) \geq t \rangle$, where t is a truth value. Thus the TBox contains general knowledge and the ABox contains concrete knowledge.

Complex concepts in the TBox are created from atomic concepts according to special syntax rules using constructors (see Table 3). DL $s\text{-}\mathcal{EL}(\mathcal{D})$ allows only two basic constructors, concept conjunction $C \sqcap D$ and full existential restriction $\exists R.C$ to create complex concepts. In Table

Table 3: Syntax and semantics of s-$\mathcal{EL}(\mathcal{D})$

Constructor	Syntax	Semantics
atomic concept	A	$A^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0,1]$
role	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
top concept	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \times 1$
existential restriction	$\exists R.C$	$(\exists R.C)^{\mathcal{I}}\ (a) = \sup\{C^{\mathcal{I}}(b) : (a,b) \in R^{\mathcal{I}}\}$
existential restriction	$\exists u.P$	$(\exists u.P)^{\mathcal{I}}(a) = \sup_{b \in \Delta^{\mathcal{D}}} \{P(b) :\ (a,b) \in u^{\mathcal{I}}\}$
concept conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}}\ (a) = \min\{C^{\mathcal{I}}(a), D^{\mathcal{I}}(a)\}$
top-k constructor	top-k(C)	top-k$(C)^{\mathcal{I}}(a) = \begin{cases} C^{\mathcal{I}}(a); & \left\lvert b \in \Delta^{\mathcal{I}} : C^{\mathcal{I}}(a) < C^{\mathcal{I}}(b)\right\rvert < k \\ 0; & \text{otherwise} \end{cases}$
aggregation	$@(C_1, \ldots, C_m)$	$@(C_1, \ldots, C_m)^{\mathcal{I}}(a) = @^{\bullet}(C_1^{\mathcal{I}}(a), \ldots, C_m^{\mathcal{I}}(a))$

3, A is an atomic concept name, C, D are complex concept names, R is a role, u is a concrete role and a, b are individuals.

The interpretation $\mathcal{I}$ consist of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\bullet^{\mathcal{I}}$. Table 3 shows that the concepts are interpreted as fuzzy sets of individuals. Roles, however, are interpreted as classical, crisp binary relations. An interpretation $\mathcal{I}$ is a model of TBox definition $C \equiv D$ iff $\forall x \in \Delta^{\mathcal{I}} :\ C^{\mathcal{I}}(x) = D^{\mathcal{I}}(x)$, ABox concept assertion $\langle C(a) \geq t \rangle$ iff $C^{\mathcal{I}}(a) \geq t$ and role assertions $R(a,b)$ iff $(a,b) \in R^{\mathcal{I}}$.

Apart from standard constructors $\top$, $C \sqcap D$ and $\exists R.C$, description logic s-$\mathcal{EL}(\mathcal{D})$ contains two new constructors, namely aggregation @ and top-k. The latter is a special modifier that leaves the first k membership values of the concept C unchanged, but it sets all other values to 0. Note that $\left\lvert b \in \Delta^{\mathcal{I}} : C^{\mathcal{I}}(a) < C^{\mathcal{I}}(b)\right\rvert$ is a set of elements preferred more than a. If this set contains at least k elements, then a is not among k most preferred individuals according to the concept C.

If $C^{\mathcal{I}}$ defines a strict ordering of the first k elements, i.e. $C^{\mathcal{I}}(a_1) > \cdots > C^{\mathcal{I}}(a_k)$, then top-k$(C)$ is unique. However, if we wanted to choose top-3(C) for $C^{\mathcal{I}}(x) = 0.9$; $C^{\mathcal{I}}(y) = 0.8$; $C^{\mathcal{I}}(z) = 0.7$; $C^{\mathcal{I}}(p) = 0.7$; $C^{\mathcal{I}}(q) = 0.6$; we could take the "strict" order either as (x, y, z, p, q) or (x, y, p, z, q). So the top-three objects can be (x, y, z) as well as (x, y, p). Note that top-k(C) from Table 3 includes an element a if there are less than k elements that have strictly greater membership value, which means including all the ties.

Aggregation @ is a generalization of fuzzy conjunctions and disjunctions, so it can be used as an alternative to concept conjunction $C \sqcap D$. Aggregation functions are m-ary fuzzy functions $@_U^{\bullet} : [0,1]^m \to [0,1]$, monotone in all arguments and such that $@_U^{\bullet}(1, \ldots, 1) = 1$ and $@_U(0, \ldots, 0) = 0$.

We use a fuzzy concrete domain $\mathcal{D}$ introduced by U. Straccia [10], defined as $\mathcal{D} = (\Delta^{\mathcal{D}}, Pred^{\mathcal{D}})$. In our case, the domain $\Delta^{\mathcal{D}} = \mathbb{R}$ is a set of real numbers. If we need other than numerical attribute values, we can easily extend the concrete domain to contain other values as well (the combination of multiple concrete domains is defined in [2]). The set of fuzzy predicates $Pred^{\mathcal{D}} = \{lt_{a,b}(x), rt_{a,b}(x), trz_{a,b,c,d}(x), inv_{a,b,c,d}(x)\}$ contains monotone and trapezoidal fuzzy sets (unary fuzzy predicates). The interpretation of fuzzy predicates is fixed, handled by the concrete domain.

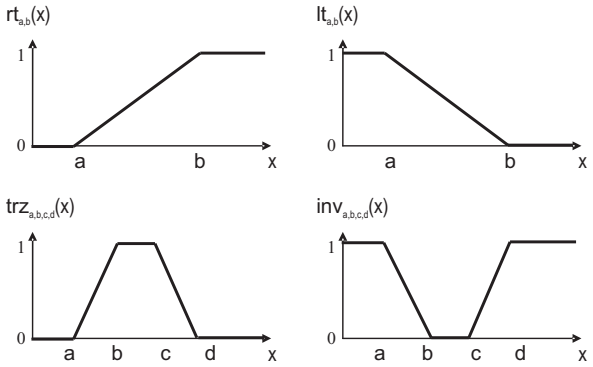

Figure 5: Basic fuzzy sets as elements of the concrete domain $\mathcal{D}$.

Figure 5 shows $lt_{a,b}(x)$ (left trapezoidal membership function), $rt_{a,b}(x)$ (right trapezoidal), $trz_{a,b,c,d}(x)$ (trapezoidal) and $inv_{a,b,c,d}(x)$ (inverse trapezoidal) with one variable x and parameters a, b, c, d. Note that the parameters influence the shape of the membership function to match user's attribute preference exactly. The membership function `Cheap` from Figure 1 would be defined as $lt_{400,700}$ because the function is left trapezoidal (i.e. non-increasing), all prices below 400 have a membership value 1 and all prices above 700 have a membership value 0. The parameters can be chosen arbitrarily, but they must be fixed for a concrete predicate. Note that the parameters are subjective for an individual user `U`, so that different users would have different parameters. In case of complex preferential concepts, we add a user identifier like `U1` to the subscript, e.g. $\texttt{Cheap}_{\texttt{U1}}$.

If we want to associate a fuzzy predicate with some role, i.e. to say that $lt_{400,700}$ contains only the values of `price`, we can use a special case of existential restriction $\exists u.P$ (e.g. $\exists \texttt{price}.lt_{400,700}$). It is similar to $\exists R.C$, but $P \in Pred^{\mathcal{D}}$ is a concrete predicate and u is a concrete role.

4. Subsumption Problem and Instance Problem in s-$\mathcal{EL}(\mathcal{D})$

We have already mentioned that the main advantage of DLs is automated reasoning. One of the most commom reasoning problems is called *subsumption problem* – a concept C is subsumed by D with respect to a TBox $\mathcal{T}$, $C \sqsubseteq_{\mathcal{T}} D$, if for every model $\mathcal{I}$ of $\mathcal{T}$ and every individual $a \in \Delta^{\mathcal{I}}$ holds $C^{\mathcal{I}}(a) \leq D^{\mathcal{I}}(a)$. The subsumption problem in s-$\mathcal{EL}(\mathcal{D})$ can be decided with a structural algorithm, similar to the algorithm proposed in [9], by finding homo-

morphism between description trees for concepts D and C. In the following text, we present our own structural algorithms, extended from [9] to handle fuzzy predicates. Note that the algorithms use a subset of $s\text{-}\mathcal{EL}(\mathcal{D})$ without aggregations and top-k constructor. Let us begin with some definitions from graph theory.

Definition 1. *Let $G = (V, E)$ and $G' = (V', E')$ be graphs. A* homomorphism *from G to G' is a mapping $\phi : V \to V'$ such that $\forall(u, v) \in E : \ (\phi(u), \phi(v)) \in E'$.*

We also need the ability to attach a label to any vertex or edge. Thus we will use labeled graphs:

Definition 2. *A* labeled graph *is defined as a triple $G = (V, E, l)$, where (V, E) is a directed graph and l is a labeling function $l : V \cup E \to L$ for some set of labels L. For every $x \in V \cup E$, the element $l(x) \in L$ is called the* label *of x. A* labeled tree *is a connected labeled graph without cycles.*

It is possible to represent an $\mathcal{EL}$ concept as a labeled tree [9]. If the concept is atomic, it will be represented with a single vertex, labeled with the concept name. Complex concepts are treated differently – at first we need to rewrite the concept C into an equivalent *normal form* $C \equiv D_1 \sqcap \cdots \sqcap D_n \sqcap \exists R_1.C_1 \sqcap \cdots \sqcap \exists R_m.C_m$ where D_i denotes an atomic concept and C_i can be either a complex concept in the normal form or a conjunction of concrete domain predicates (if R_i is a concrete role). Roles $R_1, \ldots, R_m$ must be distinct. This transformation can be done using commutativity of conjunctions and the fact that $\exists R.(C \sqcap D) \equiv (\exists R.C) \sqcap (\exists R.D)$. Subsequently, we can create the tree: concepts $D_1, \ldots, D_n$ will form a label of a vertex v_0 and we create a new edge labeled with R_i for every $\exists R_i.C_i$. The edge is oriented from the v_0 to a new vertex obtained recursively from C_i. If C_i is a conjunction of concrete domain predicates, the label will be a set of predicate names.

Definition 3. *A $s\text{-}\mathcal{EL}(\mathcal{D})$ description tree is defined as a labeled tree $G = (V, E, l)$, where the labeling function maps every vertex $v \in V$ either to a subset of atomic concept names or a subset of concrete predicates $l(v) \subset N_C \cup Pred^{\mathcal{D}}$. Every edge $e \in E$ is mapped to a role name $v(e) \in N_R$.*

Concrete domain predicates (like $lt_{a,b}$, $rt_{a,b}$, $trz_{a,b,c,d}$, $inv_{a,b,c,d}$) are allowed to occur only as a part of $\exists R.P$ in complex concept descriptions, where R is a concrete role and P is a concrete predicate. Thus the normal from can also contain a conjunction of concrete predicates. However, concrete predicates cannot occur in the same conjunction with concept names. We can solve these two cases separately.

Crisp subsumption between two fuzzy concepts C and D can be checked by creating a $s\text{-}\mathcal{EL}(\mathcal{D})$ description tree for each concept (let us denote them $G(C)$ and $G(D)$) and searching for a homomorphism from $G(D)$ to $G(C)$. Because $s\text{-}\mathcal{EL}(\mathcal{D})$ trees are labeled, the homomorphism must match edges with identical labels. If a vertex v is labeled with concept names, it must be mapped to a vertex v' labeled with a superset of the concept names in $l(v)$ (so that the first concept is more general). On the other hand, if v, v' are labeled with predicate names, we must check if the conjunction of predicates from $l(v)$ is more general than the conjunction of $l(v')$. If such homomorphism is found, the concept D is more general than C, thus the subsumption $C \sqsubseteq_{\mathcal{T}} D$ holds.

Definition 4. *Let $G(D) = (V_D, E_D, l_D)$ and $G(C) = (V_C, E_C, l_C)$ be $s\text{-}\mathcal{EL}(\mathcal{D})$ description trees for concepts D and C, respectively. A mapping $\phi : V_D \to V_C$ is a subsumption-homomorphism from $G(D)$ to $G(C)$ iff all of the following conditions are satisfied:*

1. *if v_0 is a root of $G(D)$ and w_0 is a root of $G(C)$, then $\phi(v_0) = w_0$,*
2. *$l_D(v) \subseteq l_C(\phi(v))$ for every vertex $v \in V_D$ which is labeled with concept names ($l_D(v) \subseteq N_C$),*
3. *$(\bigwedge^{\bullet} l_C(\phi(v)))(x) \leq (\bigwedge^{\bullet} l_D(v))(x)$ for every $x \in \mathcal{D}$ and for every vertex $v \in V_D$ which is labeled with concrete predicate names ($l_D(v) \subseteq Pred^{\mathcal{D}}$),*
4. *$(\phi(v), \phi(w)) \in E_C$ for every edge $(v, w) \in E_D$ and the labels are the same $l_D(v, w) = l_C(\phi(v), \phi(w))$.*

Baader, Küsters and Molitor [3] showed that the homomorphism can be found in polynomial time for crisp $\mathcal{EL}$. The only substantial difference in our subsumption algorithm is the presence of concrete predicates. Note that the definition uses a "big operator" notation of Gödel fuzzy conjunction $\bigwedge^{\bullet}$. The expression $\bigwedge^{\bullet} l_D(v)$ is a fuzzy conjunction of concrete predicates that occur in the label of v and $\bigwedge^{\bullet} l_C(\phi(v))$ is a fuzzy conjunction of predicates in the label of $\phi(v)$. Condition 3 in fact requires a crisp subsumption between $\bigwedge^{\bullet} l_C(\phi(v))$ and $\bigwedge^{\bullet} l_D(v)$. Now we need a method to decide subsumption between two conjunctions of concrete predicates in polynomial time.

Let $\bigwedge^{\bullet} l_C(\phi(v))$ be $P_1 \wedge \cdots \wedge P_m$ and let $\bigwedge^{\bullet} l_D(v)$ be $P_{m+1} \wedge \cdots \wedge P_{m+n}$, where $P_1, \ldots, P_m, P_{m+1}, \ldots, P_{m+n} \in Pred^{\mathcal{D}}$ are arbitrary concrete domain predicates in the form $lt_{a,b}, rt_{a,b}, trz_{a,b,c,d}$ or $inv_{a,b,c,d}$. Every P_i has parameters a, b or a, b, c, d that specify points from the concrete domain. The membership function is linear between these endpoints (see Figure 5). Tables 4 and 5 show how we determine the membership value on different subintervals.

Table 4: Membership values for left-trapezoidal and right-trapezoidal type.

predicate	$(-\infty, a]$	$[a, b]$	$[b, \infty)$
$lt_{a,b}$	1	$\frac{b-x}{b-a}$	0
$rt_{a,b}$	0	$\frac{x-a}{b-a}$	1

Table 5: Membership values for trapezoidal and inverse trapezoidal type.

predicate	$(-\infty, a]$	$[a, b]$	$[b, c]$	$[c, d]$	$[d, \infty)$
$trz_{a,b,c,d}$	0	$\frac{x-a}{b-a}$	1	$\frac{d-x}{d-c}$	0
$inv_{a,b,c,d}$	1	$\frac{b-x}{b-a}$	0	$\frac{x-c}{d-c}$	1

If we use Gödel t-norm, fuzzy conjunction is a minimum of fuzzy values of all the conjuncts. The resulting fuzzy

predicate is also defined by a function, which is piecewise linear on subintervals of the domain. The endpoints of the subintervals can be chosen from the parameters a, b or a, b, c, d of the original predicates, or from the intersections of the increasing or decreasing parts of the membership functions. Figure 6 shows the conjunction (thick black line) of a trapezoidal and an inverse trapezoidal predicate (gray lines). Thin dashed lines mark the parameters a, b, c, d and the dotted lines mark the intersections. It is easy to see that if we partition the domain into all possible subintervals (using all parameters and intersections), we only need to choose a function formula according to Table 4 or Table 5 for each predicate in the subinterval, substitute arbitrary value from the open subinterval for x and compare the results. Thus we obtain the function formulas for the conjunction.

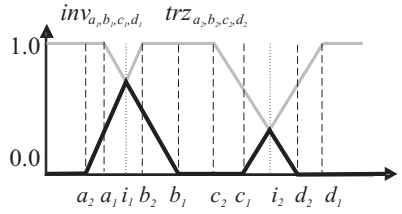

Figure 6: A conjunction of a trapezoidal and an inverse trapezoidal predicate.

If we want to find a conjunction of the fuzzy predicates from Figure 6, we need to check the intervals $(-\infty, a_2]$, $[a_2, a_1]$, $[a_1, i_1]$, $[i_1, b_2]$, $[b_2, b_1]$, $[b_1, c_2]$, $[c_2, c_1]$, $[c_1, i_2]$, $[i_2, d_2]$, $[d_2, d_1]$, $[d_1, \infty)$. Let us take interval $[a_1, i_1]$ as an example. Since it is a subinterval of $[a_1, b_1]$, the first predicate inv_{a_1,b_1,c_1,d_1} is defined by the formula $\frac{b_1-x}{b_1-a_1}$ (see Table 5). It is also a subinterval of $[a_2, b_2]$, so trz_{a_2,b_2,c_2,d_2} is defined as $\frac{x-a_2}{b_2-a_2}$. After substituting any value from $[a_1, i_1]$ for x, we find out that $\frac{b_1-x}{b_1-a_1} \geq \frac{x-a_2}{b_2-a_2}$ (note that all the parameters would be numbers in a real example). Thus the conjunction is defined by a formula $\frac{x-a_2}{b_2-a_2}$ on the interval $[a_1, i_1]$. Some of the subintervals will have the same formula in the result, so they can be unified (like $[a_2, a_1]$ and $[a_1, i_1]$ on Figure 6).

The last part of the algorithm is to check crisp subsumption between two conjunctions of fuzzy predicates. We take all subinterval endpoints from both conjunctions. Then we check the condition $(P_1 \wedge \cdots \wedge P_m)(x) \leq (P_{m+1} \wedge \cdots \wedge P_{m+n})(x)$ for every x in the subintervals. If we find any intersection inside a subinterval, we can stop, because the condition could not be satisfied. Otherwise we substitute a point from the subinterval for x and check if the first membership function has lower values than the second function. If the inequality is checked successfully for all subintervals, then the crisp subsumption relationship holds.

Every predicate has at most 4 parameters (a, b, c, d), so we gain $4m$ interval endpoints for m predicates in a conjunction. Moreover, we have to add the number of intersections, which is bounded by $2(m^2 - m)$ (there are $\frac{1}{2}(m^2 - m)$ possible pairs of predicates, every predicate membership function has at most two strictly monotonic parts, so we have to check four intersections for every pair of predicates). Thus we have at most $2m^2 + 2m + 1$ intervals. Then we check m function formulas to find the minimum (the substitution and comparison needs only constant time for each function). Thus the algorithm for finding the conjunction of m fuzzy predicates has a complexity $O(m^3)$.

The *s-*$\mathcal{EL}(\mathcal{D})$ description trees also help to answer the question to which degree an individual a is an instance of a concept C. In addition to description trees, we also need a graph structure to represent assertions from the ABox. Such structure will be called *s-*$\mathcal{EL}(\mathcal{D})$ description graph.

Definition 5. *A s-*$\mathcal{EL}(\mathcal{D})$ *description graph is defined as a labeled graph* $G = (V, E, l)$*, where the labeling function maps every vertex* $v \in V$ *either to a fuzzy subset of atomic concept names or to a concrete value* $l : V \rightarrow N_C \times [0, 1] \cup \mathcal{D}$*. Every edge* $e \in E$ *is mapped to a role name* $v(e) \in N_R$*.*

A *s-*$\mathcal{EL}(\mathcal{D})$ description graph can be created from an ABox the following way:

1. We replace complex concept names with their definitions, so that the ABox contains only assertions with atomic concepts. This means that we replace $\langle (C \sqcap D)(a) \geq t \rangle$ with $\langle C(a) \geq t \rangle$ and $\langle D(a) \geq t \rangle$. The assertion $\langle (\exists R.C)(a) \geq t \rangle$ is replaced with $R(a, d)$ and $\langle C(d) \geq t \rangle$ where d is a new individual name. The ABox obtained this way is equivalent to the original ABox (i.e. it is satisfied by the same models).
2. We define a set $nodes(a) = \{(C, t) |\ \langle C(a) \geq t \rangle \in \mathcal{A}\}$ for every individual a occurring in the ABox.
3. If $(C, t_1) \in nodes(a)$ and $(C, t_2) \in nodes(a)$ at the same time and $t_1 \geq t_2$, we remove the element (C, t_2).
4. We create a new vertex labeled $nodes(a)$ for every individual a from the ABox.
5. We also create a new vertex for every concrete value occurring in the ABox. The label is identical with the concrete value.
6. For every abstract role assertion $R(a, b)$ from the ABox we create a new edge labeled with R from $nodes(a)$ to $nodes(b)$.
7. For every concrete role assertion $u(a, b)$ we create a new edge labeled with R from $nodes(a)$ to b.

To check the instance problem, it is sufficient to create a *s-*$\mathcal{EL}(\mathcal{D})$ description tree of a concept C, a *s-*$\mathcal{EL}(\mathcal{D})$ description graph for ABox $\mathcal{A}$ and try to find a suitable homomorphism ϕ and an evaluation e_ϕ. If such ϕ and e_ϕ exist, then the individual is an instance of the concept C to a degree $e_\phi(v_0)$, where v_0 is the root of the description tree. Otherwise the degree is 0.

Definition 6. *Let* $H = (V_H, E_H, l_H)$ *be a s-*$\mathcal{EL}(\mathcal{D})$ *description graph for an ABox* $\mathcal{A}$*, let* $G = (V_G, E_G, l_G)$ *be a s-*$\mathcal{EL}(\mathcal{D})$ *description tree for a concept* C *and let* a *be arbitrary, but fixed individual. Then an instance-homomorphism* $\phi : V_G \rightarrow V_H$ *is a mapping such that:*

1. *for every vertex $v \in V_G$ and every concept $C \in l_G(v)$ there exists $(C, n) \in l_H(\phi(v))$,*
2. *for every edge $(v, w) \in E_G$ holds $(\phi(v), \phi(w)) \in E_H$ and moreover the labels remain the same $l_G(v, w) = l_H(\phi(v), \phi(w))$*
3. *if v_0 is a root of G, then $\phi(v_0) = a$.*

DEFINITION 7. *Let $G = (V_G, E_G, l_G)$ be a s-$\mathcal{EL}(\mathcal{D})$ description tree for a concept C, let $H = (V_H, E_H, l_H)$ be a s-$\mathcal{EL}(\mathcal{D})$ description graph for an ABox $\mathcal{A}$ and let ϕ be an instance-homomorphism from V_G to V_H. Let $\wedge_G$ be an infix notation of Gödel conjunction. An* evaluation *is a function $e_\phi : V_G \to [0, 1]$ defined by induction from the leaves of the description tree:*

1. *for every leaf $v \in V_G$ which is labeled with concept names $l_G(v) = \{C_1, \ldots, C_n\}$ the evaluation is defined $e_\phi(v) = C_1(\phi(v)) \wedge_G \cdots \wedge_G C_n(\phi(v))$, where $C_i(\phi(v)) = t$ if $(C_i, t) \in nodes(v)$, 0 otherwise.*
2. *for every leaf $v \in V_G$ which is labeled with concrete predicate names $l_G(v) = \{P_1, \ldots, P_n\}$ the evaluation is $e_\phi(v) = P_1(\phi(v)) \wedge_G \cdots \wedge_G P_n(\phi(v))$, where $P_i(\phi(v))$ is determined by the concrete domain*
3. *for any other vertex $w \in V_G$, $l_G(w) = \{C_1, \ldots, C_n\}$ there are successors $u_1, \ldots, u_m$. The evaluation is defined $e_\phi(w) = C_1(\phi(w)) \wedge_G \cdots \wedge_G C_n(\phi(w)) \wedge_G e_\phi(u_1) \wedge_G \cdots \wedge_G e_\phi(u_m)$, where $C_i(\phi(w)) = t$ if $(C_i, t) \in nodes(w)$, 0 otherwise.*

The paper [9] proves that the existence of the homomorphism can be checked in a polynomial time. We extended the algorithm with removing complex concepts from the ABox, computing $nodes(a)$ and the evaluation, which can be done in polynomial time. Therefore our algorithm also finishes in polynomial time.

5. Description Logic o-$\mathcal{EL}(\mathcal{D})$ with Preorders

Compared to other fuzzified DLs, s-$\mathcal{EL}(\mathcal{D})$ defined in the previous section uses fuzzy membership values to represent the notion of preference, not vagueness. Every preference concept in s-$\mathcal{EL}(\mathcal{D})$ orders all the individuals from the domain according to their preference values. Conjunctions or aggregations are necessary to obtain the overall order. This principle is similar to the rules of decathlon: athletes compete in ten disciplines, each discipline is awarded according to scoring tables. All scores are summed up to determine the final order. This is the case when all precise scores are important to determine the final score.

There are other cases when the score itself is not important, only the order. Recall another example from sport, namely from Formula 1 car racing: the first ten drivers gain points according to the point table (25, 18, 15, 12, 10, 8, 6, 4, 2, 1) regardless of their exact time, speed or headstart. The final order is also determined by summing up all points. A similar system is used in Tour de France, where riders can earn points at the end of each stage. The stages are divided into several types (flat, medium mountain, high mountain) and each type has its own point table. It is possible to gain extra points for winning time trials and sprints. These examples show that sometimes we can neglect the membership values and to consider only the order of individuals.

Since preference is defined as an ordering of objects, we will interpret the preference concepts as ordering of the domain. Such description logic will be denoted o-$\mathcal{EL}(\mathcal{D})$ where o stands for "order" (preliminary versions of this DL were published in [11, 13, 12]). This approach is original and does not appear elsewhere, in contrast to fuzzified DLs which are very popular in DL community.

Every concept in o-$\mathcal{EL}(\mathcal{D})$ is interpreted as a non-strict preorder of the domain. A preorder is a reflexive and transitive relation. We do not require the antisymmetry condition (as is required for partial orders) because there can be two individuals that are not identical despite being equally preferred. We call such individuals *indiscernible* according to the preference concept C. A preorder is *total* if for every pair of distinct individuals a, b holds $a \leq b$ or $b \leq a$.

To emphasize that o-$\mathcal{EL}(\mathcal{D})$ concepts are preorders, we will write them as $\leq_C$ and we will sometimes use the infix notation $a \leq_C^{\mathcal{J}} b$ instead of $\leq_C^{\mathcal{J}} (a, b)$ for atomic concepts. Interpretations in o-$\mathcal{EL}(\mathcal{D})$ will be denoted $\mathcal{J} = (\Delta^{\mathcal{J}}, \bullet^{\mathcal{J}})$ to be clearly distinguished from the interpretations in s-$\mathcal{EL}(\mathcal{D})$. The interpretation domain $\Delta^{\mathcal{J}}$ is identical, the difference is only in the interpretation function $\bullet^{\mathcal{J}}$. If $a \leq_C^{\mathcal{J}} b$, we say that a belongs to a concept $\leq_C$ less or equal than b. (If $\leq_C$ represents user preference, we say that b is preferred to a (or equally) according to $\leq_C$. Complex concepts in o-$\mathcal{EL}(\mathcal{D})$ are constructed according to Table 6 (compare with Table 3). Note that roles are crisp and their interpretation is the same as for s-$\mathcal{EL}(\mathcal{D})$. Also note that lowercase letters in Table 6 denote individuals, except for u which denotes a concrete role. $\leq_C, \leq_D$ stand for concepts, R stands for a role and P for a concrete predicate.

We use the same concrete domain as in case of s-$\mathcal{EL}(\mathcal{D})$. Typical TBox definitions are $\leq_C \equiv \leq_D$. ABox concept assertions have the form $a_1 \leq_C a_2$ or $\leq_C(a_1, a_2)$ and role assertions $R(a, c)$.

An interpretation $\mathcal{J}$ is a model of the TBox definition $\leq_C \equiv \leq_D$ if the two preorders are equal $\leq_C^{\mathcal{J}} = \leq_D^{\mathcal{J}}$. A model of role assertions $R(a, b)$ is defined exactly as in s-$\mathcal{EL}(\mathcal{D})$, while $\mathcal{J}$ is a model of a concept assertion $a_1 \leq_C a_2$ if $(a_1, a_2) \in \leq_C^{\mathcal{J}}$.

If an individual a is preferred to b in two order concepts $\leq_C$ and $\leq_D$, the same relationship will hold in the concept conjunction $\leq_C \sqcap \leq_D$. The main disadvantage is that the concept conjunction often produces a partial preorder, even if $\leq_C$ and $\leq_D$ are total preorders. According to the *order-extension principle*, it is possible to extend $(\leq_C \sqcap \leq_D)^{\mathcal{J}}$ to a total preorder, but this extension does not have to be unique. The extension is trivial for finite domains and it is also possible for infinite domains using the axiom of choice. However, sometimes it is more convenient to use aggregation $@_U$ instead of concept conjunction, especially when we consider a conjunction of more than two concepts.

The semantics of $\exists R.\leq_C$ is very unusual at the first sight, so we will explain it with help of s-$\mathcal{EL}(\mathcal{D})$. Let s-$\mathcal{EL}(\mathcal{D})$ knowledge base contain the following assertions:

Table 6: Concept constructors in o-$\mathcal{EL}(\mathcal{D})$

Syntax	Semantics
$\leq_A$	$\leq_A^{\mathcal{J}} \subseteq \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}}$
R	$R^{\mathcal{J}} \subseteq \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}}$
$\leq_\top$	$\Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}}$
$\exists R.\leq_C$	$\{(a_1,a_2) : \forall c_1\ (a_1,c_1) \in R^{\mathcal{J}}\ \exists c_2\ (a_2,c_2) \in R^{\mathcal{J}} : c_1 \leq_C^{\mathcal{J}} c_2\}$
$\exists u.P$	$\{(a_1,a_2) : \forall c_1\ (a_1,c_1) \in u^{\mathcal{J}}\ \exists c_2\ (a_2,c_2) \in u^{\mathcal{J}} : P(c_1) \leq P(c_2)\}$
$\leq_C \sqcap \leq_D$	$\{(a_1,a_2) : a_1 \leq_C^{\mathcal{J}} a_2 \wedge a_1 \leq_D^{\mathcal{J}} a_2\}$
$@_U(\leq_{C_1},\ldots,\leq_{C_m})$	$@_U^{\bullet}(\leq_{C_1},\ldots,\leq_{C_m})$
top-k($\leq_C$)	defined below

$\langle \mathtt{C}(\mathtt{c_1}) \geq 0.8 \rangle$
$\langle \mathtt{C}(\mathtt{c_2}) \geq 0.3 \rangle$
$\mathtt{R}(\mathtt{a_1}, \mathtt{c_1})$
$\mathtt{R}(\mathtt{a_1}, \mathtt{c_2})$
$\langle \mathtt{C}(\mathtt{c_3}) \geq 0.5 \rangle$
$\langle \mathtt{C}(\mathtt{c_4}) \geq 0.6 \rangle$
$\langle \mathtt{C}(\mathtt{c_5}) \geq 0.6 \rangle$
$\mathtt{R}(\mathtt{a_2}, \mathtt{c_3})$
$\mathtt{R}(\mathtt{a_2}, \mathtt{c_4})$
$\mathtt{R}(\mathtt{a_2}, \mathtt{c_5})$

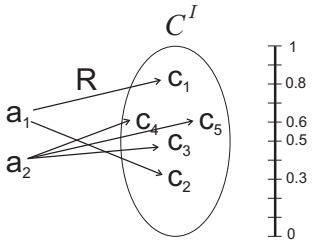

Figure 7: A graphical representation of s-$\mathcal{EL}(\mathcal{D})$ knowledge base.

The same knowledge base is shown on Figure 7. Note that membership degrees of different individuals in C are indicated by their vertical position – the individual c_1 with the highest membership degree 0.8 is on the top. Role assertions are indicated with the arcs, as is usual in ontology visualization.

If we want to find a membership degree of the individual a_1 in $\exists R.C$, it is sufficient to find a supremum from $C^{\mathcal{I}}(c_1)$, $C^{\mathcal{I}}(c_2)$ and analogously a supremum from $C^{\mathcal{I}}(c_3)$, $C^{\mathcal{I}}(c_4)$, $C^{\mathcal{I}}(c_5)$ for the individual a_2. We find out that a_1 must belong to $\exists R.C$ to a degree at least 0.8 in every model, while a_2 has a degree at least 0.6. Note that the supremum is usual in the interpretation of existential quantifier in all papers concerning fuzzy DLs, regardless of whether they use crisp or fuzzy roles. It also affects the properties of fuzzy $\exists R.C$: if a_1 has a higher degree of membership than a_2, we know that for every individual c_i connected with a_2 via role R, there exists a better individual c_j connected with a_1 (better with respect to the preference concept C). This feature of fuzzy existential quantifier is used in our interpretation of $\exists R.C$ in order description logic o-$\mathcal{EL}(\mathcal{D})$.

Figure 8 shows the analogous knowledge base for o-$\mathcal{EL}(\mathcal{D})$. Instead of fuzzy membership degrees, we have only the

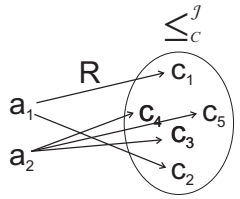

Figure 8: A graphical representation of analogous o-$\mathcal{EL}(\mathcal{D})$ knowledge base.

preorder relation to work with. The interpretation of $\exists R.\leq_C$ will be a preorder as well, so the definition must specify which pairs of individuals will belong to it. The pairs (a_1,a_2) have to fulfill the condition $\forall c_1\ (a_1,c_1) \in R^{\mathcal{J}}\ \exists c_2\ (a_2,c_2) \in R^{\mathcal{J}} : c_1 \leq_C^{\mathcal{J}} c_2$, which is just a formalization of the condition stated above: for every individual c_i connected with a_2 via role R, there exists a better individual c_j connected with a_1.

Also note that if the concept $\leq_C$ was a total preorder, then $\exists R.\leq_C$ will be also total. If the knowledge base contains two role assertions $R(a_1,b_1)$ and $R(a_2,b_2)$, then their order in the concept $\exists R.\leq_C$ is the same as in $\leq_C$ (which is total, so it contains either the pair (a_1,a_2), or (a_2,a_1), or both). If the knowledge base contains only $R(a_1,c_1)$ and no such assertion for a_2 then for every individual connected with a_2 via role R (which is none in our case), there exists a better individual c_1 connected with a_1, so the pair (a_2,a_1) clearly belongs to $\exists R.\leq_C$ according to the definition. If there are no role assertions for a_1 and a_2, then both (a_1,a_2) and (a_2,a_1) fulfill the condition from the definition and so both pairs belong to $\exists R.\leq_C$. Thus all individuals that have role assertions are ordered according to $\leq_C$ and all other individuals share the lowest level of $\exists R.\leq_C$. A similar principle was used in the definition s-$\mathcal{EL}(\mathcal{D})$ – if some individual a did not occur in role assertions with R, then the supremum $\sup\{C^{\mathcal{I}}(b) : (a,b) \in R^{\mathcal{I}}\}$ had to be chosen from an empty set and $\sup\emptyset$ was defined to be 0. The individual a would have the lowest possible membership value in $\exists R.C$.

For every $@_U$ with arity m and for every m-tuple of order concepts $\leq_{C_j} \subseteq \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}}$ an aggregation $@_U^{\bullet}(\leq_{C_1},\ldots,\leq_{C_m})^{\mathcal{J}} \subseteq \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}}$ is a preorder such that the following holds: if $a \leq_{C_j}^{\mathcal{J}} b$ for every $j = 1,\ldots,m$, then $(a,b) \in @_U^{\bullet}(\leq_{C_1},\ldots,\leq_{C_m})^{\mathcal{J}}$.

There are many possibilities to define aggregation $@_U^{\bullet}$. We will show one possibility inspired by Formula 1 car race point tables mentioned before. First of all, it is necessary to define the *level* of an individual a in the interpretation of a concept C. It is the biggest possible length

of a sequence such that the first element is a and every following element is strictly greater than its predecessor.

$$level(a, \leq_C^{\mathcal{J}}) = \max_{l \in \mathbf{N}} \{l : \exists b_1, \ldots, b_l \in \Delta^{\mathcal{J}} \ \forall i \in \{1, \ldots, l-1\} : \ b_i \leq_C^{\mathcal{J}} b_{i+1} \ \wedge \ b_{i+1} \not\leq_C^{\mathcal{J}} b_i \ \wedge \ b_1 = a\}$$

Next we define a *point table* for a concept $\leq_C$ and user $\mathtt{U}$. A point table is a non-increasing function over natural numbers $point^U_{\leq_C} : \mathbb{N} \to \mathbb{N}$ with values like 10, 8, 6, 5, 4, 3, 2, 1, 0, 0, ... ending with zeros. The differences between adjacent values are non-increasing. A pair (a, b) belongs to aggregation $@^{\bullet}_U(\leq_{C_1}, \ldots, \leq_{C_m})^{\mathcal{J}}$ if

$$\sum_{j=1}^{m} point^U_{\leq_{C_j}}(level(a, \leq_C^{\mathcal{J}})) \leq \sum_{j=1}^{m} point^U_{\leq_{C_j}}(level(b, \leq_C^{\mathcal{J}})).$$

This means finding the level of individual a in every preference concept $\leq_{C_j}^{\mathcal{J}}$, then determining the corresponding points for these levels and sum up all the points. If the individual b has better levels in the preference concepts $\leq_{C_j}^{\mathcal{J}}$ than individual a, it will also have more points in $point^U_{\leq_{C_j}}$.

It is also straightforward to define top-k queries. Let $C_a = \{c \in \Delta^{\mathcal{J}} : a \leq_C^{\mathcal{J}} c \ \wedge \ c \not\leq_C^{\mathcal{J}} a\}$ be a set of individuals strictly greater than a in ordering concept $\leq_C$. Then $(a, b) \in \text{top-k}(\leq_C)^{\mathcal{J}}$, iff:

1. $a \leq_C^{\mathcal{J}} b \ \wedge |C_a| < k$ or
2. $|C_a| \geq k$

If $\leq_C$ was a total preorder, then top-k($\leq_C$) will be also total. Top-k constructor preserves the original order of the first k individuals, including the ties. Note that the first k individuals often occupy less than k levels because some of them are ties. Concerning the ties on the last included level (not necessarily the k-th level), we can either choose only some of them to fill up the needed amount of elements, or we can return them all. Here we choose the latter possibility, even if we end up with more than k elements in the result, because it makes our definition deterministic. If some element a has more than k strictly greater elements in $\leq_C^{\mathcal{J}}$, so that it is beyond the last included level (see condition 2), it is made lower or equal to all other elements, which moves it to the last level in top-k$(\leq_C)^{\mathcal{J}}$.

5.1 Order-Oriented Reasoning

Interpreting concepts as preorder relations has one main drawback – it complicates the reasoning. The relationship between various concept (or role) constructors and the complexity of reasoning is well explored in description logics. Reasoning with complex roles is known to be more time-consuming than reasoning in a DL with concept constructors only [1]. Because concepts in o-$\mathcal{EL}(\mathcal{D})$ are also binary relations, we can expect the reasoning to have higher complexity than the polynomial complexity in s-$\mathcal{EL}(\mathcal{D})$.

Some reasoning algorithms for o-$\mathcal{EL}$ can be designed by modification of structural or tableaux approaches from classical DLs. There is one more complication – the use of aggregation functions may easily cause undecidability of reasoning. Paper [4] proves decidability only for $\mathcal{EL}(\mathcal{D})$ with atomic negation and a concrete domain $\mathcal{D}$ with functions min, max, sum. Although aggregation functions are more suitable to represent user preferences, we replace them with concept conjunctions for the purpose of reasoning. We also leave out top-k constructor which will be used only in top-k queries [6]. In this section we focus on *instance order problem* – to find out if $a \leq_C^{\mathcal{J}} b$ holds for every model $\mathcal{J}$ of the knowledge base.

At first we construct the expansion and the transitive closure of an ABox $\mathcal{A}$:

1. add $a \leq_C b$ (the assertion that we want to check)
2. replace all complex concept names in $\mathcal{A}$ with their definitions
3. if $a \leq_C b \in \mathcal{A}$ and $b \leq_C c \in \mathcal{A}$ and $a \leq_C c \notin \mathcal{A}$, then add $a \leq_C c$

Afterwards we decompose $\mathcal{A}$ by applying rules. Every rule replaces some ABox assertion with new assertions or constraints. New assertions are allowed to contain variables instead of individual names. Constraints have the form $P(x, y)$, where x, y can be values from the concrete domain or variables and P is a concrete predicate. The rule $R\exists$ can be used for both abstract and concrete roles.

Table 7: Replacement rules for o-$\mathcal{EL}(\mathcal{D})$ ABoxes.

Rule	Original assertions	Replace with
$R_\sqcap$	$(\leq_C \sqcap \leq_D)^{\mathcal{J}}(a_1, a_2)$	$a_1 \leq_C^{\mathcal{J}} a_2$ **and** $a_1 \leq_D^{\mathcal{J}} a_2$
$R_\exists$	$(\exists R. \leq_C)^{\mathcal{J}}(a_1, a_2)$	$R^{\mathcal{J}}(a_2, x)$ **and** $c_1 \leq_C^{\mathcal{J}} x$ for all c_1 such that $(a_1, c_1) \in R^{\mathcal{J}}$ where x is a new variable

After no more rules can be applied on ABox $\mathcal{A}$, we end up with constraints like $P(x_1, x_2)$ and assertions of the form $x_3 \leq_C x_4$ and $R(x_5, x_6)$. As the next step we try to substitute individuals and concrete values for the variables so that all constraints and assertions are fulfilled. For each variable x and role R we create a set of candidate values $CV(x, R) = \{y \in \Delta^{\mathcal{J}} \cup \Delta^{\mathcal{D}} : \ R(a, y) \in \mathcal{A} \wedge y \neq x\}$. We have to choose one value from each candidate set such that no predicate assertion $P(x, y)$ is violated. If there is such a substitution, then $a \leq_C^{\mathcal{J}} b$ for every model $\mathcal{J}$. If not, $a \leq_C^{\mathcal{J}} b$ may hold only for some models. A substitution can be found with bMIP (bounded mixed integer programming) method described in [10].

6. The Relationship between Scoring and Ordering Approach

Definitions for s-$\mathcal{EL}(\mathcal{D})$ and o-$\mathcal{EL}(\mathcal{D})$ are much similar, but the two logics are not equivalent. At the first sight, it is obvious that o-$\mathcal{EL}(\mathcal{D})$ drops exact membership degrees, thus it loses the ability to express some features of s-$\mathcal{EL}(\mathcal{D})$. If we have a "constant" fuzzy concept $C^{\mathcal{I}}(a) = w \in TV_n$ for every $a \in \Delta^{\mathcal{I}}$, the corresponding order concept in o-$\mathcal{EL}(\mathcal{D})$ will be $\leq_C^{\mathcal{J}} = \Delta^{\mathcal{J}} \times \Delta^{\mathcal{J}}$, regardless of the value w. Similarly, if $D^{\mathcal{I}}(a) \leq D^{\mathcal{I}}(b)$, the corresponding order concept contains the pair $(a, b) \in \leq_D^{\mathcal{J}}$, but we lose information about the difference $D^{\mathcal{I}}(b) - D^{\mathcal{I}}(a)$.

If we compare a scoring concept C with an ordering concept $\leq_C$, we are concerned about the order of individu-

als. It is straightforward to define corresponding order-preserving concept $\leq_A$ for every primitive concept A and for any interpretation $A^{\mathcal{I}}$. We define $a \leq_A^{\mathcal{J}} b$ iff $A^{\mathcal{I}}(a) \leq A^{\mathcal{I}}(b)$. Concept constructors should also preserve order of individuals. We start from a scoring concept A, transform it to a corresponding ordering concept $\leq_A$, use corresponding constructors on both concepts and finally compare order of individuals in the results.

DEFINITION 8. *Let C be a s-$\mathcal{EL}(\mathcal{D})$ concept, $\leq_C$ a o-$\mathcal{EL}(\mathcal{D})$ concept. We say that $\leq_C$ is* generated by C *if the following holds in all interpretations $\mathcal{I}, \mathcal{J}$: $\forall a, b \in \Delta^{\mathcal{I}}$: $a \leq_C^{\mathcal{J}} b$ iff $C^{\mathcal{I}}(a) \leq C^{\mathcal{I}}(b)$.*

Note that the concrete domain $\mathcal{D}$ is already defined in such a way that $\exists u.P$ is order-preserving. In addition, the following properties of the concept constructors can be proved:

THEOREM 1. *Let $\leq_C$ be generated by C. Then $\exists R.\leq_C$ is generated by $\exists R.C$.*

THEOREM 2. *Let $\leq_C$ be generated by C. Then top-$k(\leq_C)$ is generated by top-$k(C)$.*

The constructor $\leq_C \sqcap \leq_D$ produces partial preorders. Because of the minimum function in $(C \sqcap D)^{\mathcal{I}}$, we cannot model this constructor exactly in o-$\mathcal{EL}(\mathcal{D})$. There is no way of comparing elements without fuzzy degrees in two different preorders.

THEOREM 3. *Let $\leq_C$ be generated by C and $\leq_D$ be generated by D. Then $\leq_C \sqcap \leq_D$ can be extended to a total preorder which is generated $C \sqcap D$.*

Note that aggregations are defined differently for o-$\mathcal{EL}(\mathcal{D})$ and s-$\mathcal{EL}(\mathcal{D})$, so we cannot claim such a strong relationship between them. The main problem is that we neglect the exact membership values and we only keep the order of individuals as we move from s-$\mathcal{EL}(\mathcal{D})$ to o-$\mathcal{EL}(\mathcal{D})$. Our aggregations are defined in a rather universal way that allows us to adjust them according to our needs (we can set the weights in s-$\mathcal{EL}(\mathcal{D})$ and the point tables in o-$\mathcal{EL}(\mathcal{D})$). Thus if we want scoring and order aggregations to correspond, we have to set the weights and point tables properly.

THEOREM 4. *Let $\leq_{C_1}, \ldots, \leq_{C_m}$ be o-$\mathcal{EL}(\mathcal{D})$ concepts generated by $C_1, \ldots, C_m$. For every s-$\mathcal{EL}(\mathcal{D})$ aggregation $@_U$ with weights $w_1, \ldots, w_m$ there exists an o-$\mathcal{EL}(\mathcal{D})$ aggregation $@'_U$ such that $@'_U(\leq_{C_1}, \ldots, \leq_{C_m})$ is generated by $@_U(C_1, \ldots, C_m)$.*

Note that if the domain $\Delta^{\mathcal{J}}$ is changed (some elements are added or removed), this relationship does not have to hold anymore. This lemma shows that the point tables use similar principles as in scoring DLs.

7. Implementation

Our fuzzy preference model was implemented within a system for preferential search [7]. The system is named Kore[2] and it consists of three main software tools implemented in Java – acquiring and managing user preferences (UPreA), preferential top-k search (top-k, see [6]) and learning user preference from rated objects (IGAP [8]). Figure 9 shows the functionality of the system. As the first step, the user can select relevant attributes and specify her attribute preferences. These preferences are used in top-k search (step 1) and a sorted list of results R_0 is returned. The user can rate the results in a 5-degree scale (step 2), thus creating a rated list U_0. It is used as an input for preference learning tool IGAP and the search is performed with new preferences. This process can be repeated a few times until the user is content with the results. Every set of rated objects can help to refine the user preferences.

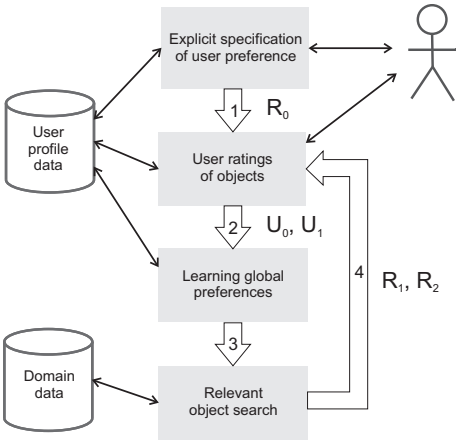

Figure 9: Overview of the preferential search system Kore.

7.1 UPreA: The Acquisition of User Preference

We need the user to specify which attribute values are preferred and to what degree, and moreover the user has to do this for every attribute that she wants to be considered. Thus our interface has to be intuitive and simple enough not to discourage and dismay users. We tested several versions of GUI components until we decided to use sliders an palettes (Figure 10).

We divided the process of specifying user preferences into two subsequent forms. The first form shows only one slider for every attribute and these sliders serve to specify the weights. The second form contains either a slider or a palette for every attribute selected by the user. Those attributes that were marked as "not important" on the first screen are left out.

The sliders on the second form represent ordinal attributes and they show all possible (ordered) attribute values. In case of the attribute `screen`, the user chose the value 17" which means that the corresponding fuzzy set has a non-decreasing membership function and the value 17" is the most preferred. Thus we obtain only a simplified model of user preference – the selected value is preferred to 1.0,

[2]Kore is available online at `http://x64.ics.upjs.sk:8080/kore/`

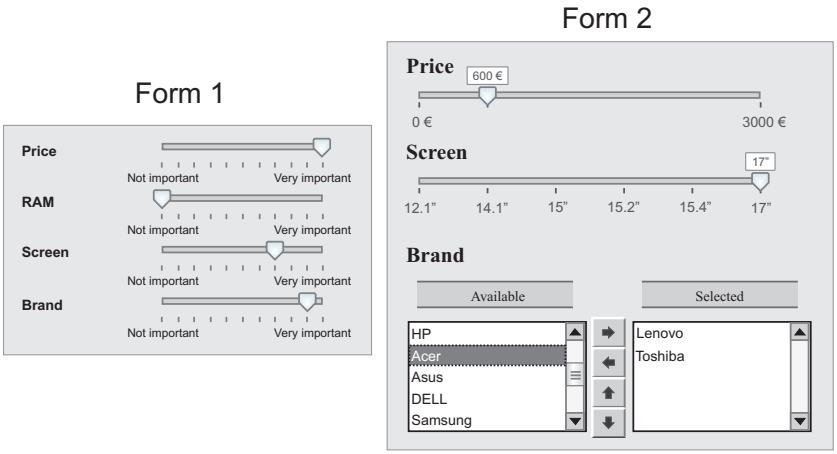

Figure 10: The third version of user interface with sliders and palettes.

while other values have evenly decreasing membership degrees. This approach means hiding the complexity of the model from users. It is similar to any standard search system, where the user is allowed to select only one preferred value for each attribute. Despite this simplification, our system is still able to order objects by their preference degrees and also retrieves objects with values close to the selected value.

Nominal attributes are handled with a component called palette. The list on the left side contains available attribute values, while the list on the right contains values selected by the user. It is possible to move values from left to right and vice versa using the buttons in the middle. Selected values can be ordered using the buttons with arrows pointing up and down. Then the system automatically creates a fuzzy set where the first selected value is preferred to a degree 1.0 and the preference degrees of the following values evenly decrease. Again, we hide the inner complexity of our model from the user and we just require the user to select her preferred values.

The GUI is implemented within Wicket framework. We added a new specialized slider component for wicket, using a freeware Javascript code by E. Khmelev[3]. We enhanced the slider component with new functions like automatic setting of the "step" value (i.e. how will the selected value change if we move the indicator by 1 pixel) and a "snap to ticks" feature for attributes with a finite number of possible values. Thus the slider in the implementation behaves differently for continuous attributes like `price` (the slider can be moved arbitrarily, the selected value is shown on a flag above the indicator) and for discrete attributes like `screen` (possible values are shown by the ticks below the slider, the indicator will snap to the closest tick after moving).

7.2 Evaluation

The implementation was tested by real users who looked for the ideal objects in the domain of job offers and rated the results in five different grades to refine their search. The experiments involved 136 different users who rated more than 500 sets of results. Some users rated the results only once, but others went through several cycles (see Figure 9 for the illustration of cycles). We analyzed the first five cycles for each user, which cover the major part of our experimental results.

Now we are interested in the correlation of R_i and U_i, i.e. the correlation of the order defined by the search tool and by the user. If the correlation is high, then our preference model resembles real user preference. We used five different measures to determine the similarity of two ordered lists. The first is called τ-correlation, $\tau(\leq_1, \leq_2) = \frac{2D}{\frac{1}{2}\cdot n\cdot(n-1)} - 1$, where D is a number of discordant pairs in both orders (i.e. such pairs (a, b) that $a \leq_1 b$ and $b \leq_2 a$) and C is the number of concordant pairs. The second is a modification of τ-correlation called *weighted order similarity* – we choose a decreasing vector of weights such that differences between adjacent values are also decreasing, e.g. $w = (20, 15, 11, 8, 6, 5, 4, 3, 2, 1)$ for $n = 10$. Weighted order coefficient is then defined as $\mathcal{W} = 1 - \sum_{j=1}^{n} \left| w_{p_j} - w_{q_j} \right|$ where p_j is a sequence number of R_j^i and q_j is a sequence number of U_j^i. Other three measures are Gödel, Łukasiewicz and product fuzzy equality $\equiv_G^\bullet (C_1, C_2) = \inf_{x\in\Delta}\{(C_1 \to^\bullet C_2) \vee^\bullet (C_2 \to^\bullet C_1)\}$, which means that C_1, C_2 are equal if $C_1(x)$ implies $C_2(x)$ and vice versa, for all x from the domain.

Finally, we compare results of these fuzzy measures on Fig. 11. τ-correlation and weighted similarity seem to be the most suitable measures. Fuzzy equalities impose too strict conditions on fuzzy values. We do not expect result lists and user ratings to be totally equal because we learn new preferences from their difference. Correlation increases most significantly between the first and the second cycle where we start to use fuzzy rules. So experiments proved that fuzzy rules are suitable representation of user preference.

Another interesting point is that τ-correlation and weighted similarity have the same trends up to fifth cycle, while all fuzzy equalities have the same trends from second to fifth cycle. Łukasiewicz equality has the greatest results and Gödel equality has the smallest results because the corresponding fuzzy implications also have this feature.

8. Conclusions

Our model of user preference is designed to support preferential top-k queries. It is divided into attribute prefer-

[3] `http://blog.egorkhmelev.com/2009/11/jquery-slider-safari-style/`

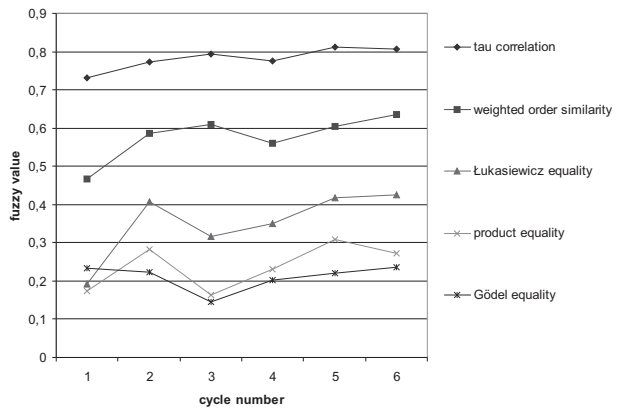

Figure 11: Comparison of different fuzzy measures.

ences and an aggregation function to obtain overall preference values. The model has been implemented in a web-based system Kore[4] and integrated with a tool for preferential search (top-k) and inductive learning of user preference (IGAP). The system was tested by real users and the data logged from the experiment indicates that there is a positive correlation between the order of results returned by our system and the order generated by user ratings. It also shows that inductive learning from ratings improves the preference model.

We created a new scoring description logic $s\text{-}\mathcal{EL}(\mathcal{D})$ as a language to describe the preference model. We extended the structural algorithms from [9] to handle a fuzzy concrete domain $\mathcal{D}$. If we use a simplified version of $s\text{-}\mathcal{EL}(\mathcal{D})$ without aggregation and top-k constructor, the extended reasoning algorithm retains polynomial complexity. We used the same approach in case of instance problem and we present a structural algorithm with polynomial complexity.

DL $s\text{-}\mathcal{EL}(\mathcal{D})$ differs from other fuzzified description logics because it has crisp roles. Fuzzy concepts do not represent uncertain information, but rather vague user preference. Every such preferential concept generates an order of individuals from the most preferred to the least preferred. We propose another DL $o\text{-}\mathcal{EL}(\mathcal{D})$ which discards the fuzzy membership degrees and keeps only the information about order of individuals. Therefore concepts in $o\text{-}\mathcal{EL}(\mathcal{D})$ are interpreted as preorders of the domain. This approach is new and it significantly changes the perspective for concept constructors and reasoning problems. We introduced an order-oriented modification of instance problem, called instance order problem – to find out if individual a is preferred more than b according to a concept $\leq_C$ in every possible interpretation. We also present a tableaux-like algorithm to solve instance order problem.

We further study the relationship between $s\text{-}\mathcal{EL}(\mathcal{D})$ concepts and $o\text{-}\mathcal{EL}(\mathcal{D})$ concepts. If two such concepts $\leq_C, C$ define the same order of individuals, we say that an order concept $\leq_C$ is generated by a scoring concept C. Then we prove that constructors $\exists$ and top-k preserve this property. The situation is more complicated for concept conjunctions (because order-oriented conjunction produces partial preorders) and aggregations (because they depend on how we set weights and point tables).

[4] `http://x64.ics.upjs.sk:8080/kore/`

References

[1] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider. *Description Logic Handbook*. Cambridge University Press, 2002.

[2] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Technical Report RR-91-10*. Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), 1991.

[3] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proceedings of IJCAI'99*, pages 96–101. Morgan Kaufmann, 1999.

[4] F. Baader and U. Sattler. Description logics with aggregates and concrete domains. *Information Systems*, 28, 2003.

[5] R. Fullér. Owa operators in decision making. In *Exploring the Limits of Support Systems*, pages 85–104. Turku Centre for Computer Science, TUCS General Publications, 1996.

[6] P. Gurský. Towards better semantics in the multifeature querying. In *Proceedings of Dateso*, 2006.

[7] P. Gurský, T. Horváth, J. Jirásek, S. Krajči, R. Novotný, J. Pribolová, V. Vaneková, and P. Vojtáš. User preference web search – experiments with a system connecting web and user. *Computing and Informatics*, 28:515–553, 2009.

[8] T. Horváth and P. Vojtáš. Ordinal classification with monotonicity constraints. *ICDM 2006. LNCS*, 4065:217–225, 2006.

[9] R. Küsters and R. Molitor. Approximating most specific concepts in description logics with existential restrictions. In *Proceedings of the Joint German/Austrian Conference on Artificial Intelligence*, pages 33–47, 2001.

[10] U. Straccia. Fuzzy $\mathcal{ALC}$ with fuzzy concrete domains. In *Proceedings of the International Workshop on Description Logics (DL-05)*, 2005.

[11] V. Vaneková and P. Vojtáš. A description logic with concept instance ordering and top-k restriction. *Frontiers in Artificial Intelligence and Applications*, 190:139–153, 2009.

[12] V. Vaneková and P. Vojtáš. Comparison of scoring and order approach in description logic $\mathcal{EL}(\mathcal{D})$. *Lecture Notes in Computer Science*, 5901/2010:709–720, 2010.

[13] V. Vaneková and P. Vojtáš. Order-oriented reasoning in description logics. *Advances in Soft Computing*, 67/2010:219–228, 2010.

Selected Papers by the Author

V. Vaneková, P. Vojtáš. Fuzziness as a Model of User Preference in Semantic Web Search. Proceedings of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference. J. P. Carvalho, D. Dubois, U. Kaymak and J. M. C. Sousa, editors. ISBN 978-989-95079-6-8.

P. Gurský, T. Horváth, J. Jirásek, R. Novotný, J. Pribolová, V. Vaneková, P. Vojtáš. Knowledge Processing for Web Search – An Integrated Model and Experiments Scalable Computing: Practice and Experience, ISSN 1895-1767, 9:51—59, 2008.

P. Gurský, V. Vaneková, J. Pribolová. Fuzzy User Preference Model for Top-k Search. Proceedings of IEEE World Congress on Computational Intelligence (WCCI), Hong Kong, 2008.

V. Vaneková, P. Vojtáš. Fuzziness as a Model of User Preference in Semantic Web Search. Proceedings of the Joint 2009 International Fuzzy Systems Association World Congress and 2009 European Society of Fuzzy Logic and Technology Conference. J. P. Carvalho, D. Dubois, U. Kaymak and J. M. C. Sousa, editors. ISBN 978-989-95079-6-8.