An Intuitive and Formal Description of Preferences for Semantic Web Service Discovery and Ranking

José María García, David Ruiz and Antonio Ruiz-Cortés
{josemgarcia,druiz,aruiz}@us.es

Applied Software Engineering Research Group
University of Seville, Spain
December 2012

Technical Report ISA-12-TR-07

This report was prepared by the

Applied Software Engineering Research Group (ISA)
Department of computer languages and systems
Av/ Reina Mercedes S/N, 41012 Seville, Spain
http://www.isa.us.es/

Support: This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project SETI (TIN2009-07366), by the Andalusian Government under projects ISABEL (TIC-2533) and THEOS (TIC-5906), by the EU FP7 IST project 27867 SOA4All, and by the EC FP7 Network of Excellence 215483 S-CUBE.

List of changes

Version	Date	Description
1.0	December 2012	First release

Abstract

Preference modeling constitutes an essential component for the execution of Semantic Web Service (SWS) discovery and, especially, ranking processes, providing facilities to define user requests and preferences. In this technical report we describe our proposed preference model, introducing in Section 1 the existing challenges on this topic that motivates our research work. Section 2 presents an abstract upper ontology to define both services and user requests, that serves as a common model to make our proposal independent from concrete SWS frameworks. Then, Section 3 further describes, both intuitively and formally, our preference model and its facilities to define preferences within a user request. Finally, we sum up the main characteristics of our solution to model preferences for discovery and ranking in Section 4, discussing its fulfillment degree with respect to our identified challenges on preference modeling.

Contents

List of Figures

1 Introduction

SWS definition frameworks provide comprehensive tools to describe services and their interactions. Although they offer facilities to also state user requests, preferences cannot be described at the same detail level, *i.e.* users cannot define complex desires for a concrete service request. For instance, Web Service Modeling Ontology (WSMO) user requests, denoted by goals [12], only support the description of requirements about a request in the form of capabilities and interfaces. In turn, preferences to rank services fulfilling those requirements cannot be directly expressed by using a standard WSMO goal definition, which only provides means to define non-functional properties / values pairs. In other words, preferences are not considered first-class citizens in WSMO, in comparison to service capabilities, whose definitions are more expressive. Other frameworks, such as OWL Ontology of Services (OWL-S) [10] or Semantic Annotations for WSDL and XML Schema (SAWSDL) [3], do not even define a specific model to describe user requests at all.

Discovery and ranking proposals try to fill this gap, extending SWS frameworks to support preferences definition [14, 15], or just providing separate user preferences descriptions [11, 9], using different formalisms. Consequently, these formalisms actually determine the level of expressiveness of each proposal, while resulting in a high dependence between user preferences definition and its corresponding discovery and ranking implementations.

In order to overcome these identified challenges in current proposals, we present in this technical report the Semantic Ontology of User Preferences (SOUP), which is a highly expressive, intuitive model of user preferences. This proposal adapts a well-known model designed for database systems [8] that allows to define preferences constructively and user-friendly. Starting from an abstract model that defines both service, user requests and preferences description at the same semantic level, next sections describe our model in detail, also introducing elements that conform the foundations of other proposals on improving discovery [6] and ranking integration [4]. Additionally, in [5] we presented an early version of this model and a thorough validation that consists on the complete definition of a discovery scenario from the SWS Challenge[1]. Particularly, we validated our model using the Logistics Management scenario, that contains several service descriptions and user requests contextualized in a transportation and logistics domain. In the following we use concepts from these domains to illustrate the different facilities provided by our model to define preferences.

2 An Abstract Upper Ontology of Services

As discussed before, service descriptions, user requests and preferences should be semantically described at the same detail level. Therefore, there is a need for the definition of an ontological model that leverages preference descriptions as first-class citizens in the discovery and ranking scenario. Moreover, this model has to provide intuitive and user-friendly facilities to easily define both requirements and preferences, so that service descriptions can be matched with user requests. Furthermore, these facilities have to conform a sufficiently expressive model so that a user can fully describe any preference, without being limited by a concrete formalism or representation.

[1] `http://sws-challenge.org`

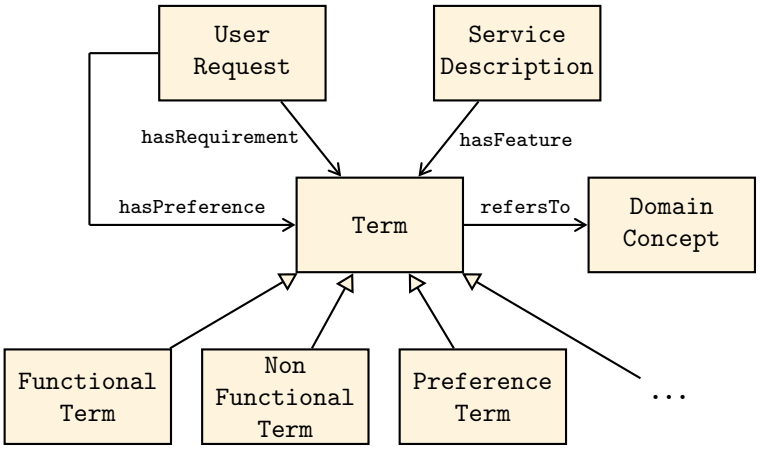

Figure 1: Upper ontology of services

In order to specify a preference model, firstly we need to establish a clear separation between requirements that have to be met, and preferences that have to be taken into account once requirements have been fulfilled. Typically, requirements are hard constraints that are used to filter service repositories in the discovery process, while preferences are used to rank previously discovered services so that the most appropriate service can be selected after the ranking process. Therefore, preferences define a strict partial order in our model, providing a framework to compare and rank a set of services.

Figure 1 shows the upper ontology of SOUP, which is represented using a UML-based notation for Ontology Web Language (OWL) ontologies [1] that we also use throughout the rest of this report. `UserRequest` and `ServiceDescription` are the root concepts in our proposal. On the one hand, a `ServiceDescription` describes features provided by the service itself, using the `hasFeature` object property to link corresponding terms about functionality, non-functional property (NFP), input and output parameters, among others. Listing 1 shows an excerpt of an abstract service description from the logistics scenario using our upper ontology, where some of the functional and NFP terms of service `:ws1` are defined. Its graphical representation is also depicted in Figure 2, where namespaces are omitted for the sake of clarity.

Listing 1: Example of an abstract service description

```
:ws1 a soup:ServiceDescription;
  soup:hasFeature :transOrder, :basePrice. # among others...

:transOrder a soup:FunctionalTerm;
  soup:refersTo logistics:TransportOrder.
:basePrice a soup:NonFunctionalTerm;
  soup:refersTo logistics:BasePrice.
```

On the other hand, a `UserRequest` is the materialization of user desires with respect to a particular service request. These desires are described using requirements and preference terms, which are linked with the particular `UserRequest` instance using respectively `hasRequirement` and `hasPreference` object properties. Terms related with requirements state hard constraints that have to be fulfilled in order to consider a certain service as a matching candidate with respect to the user request. For instance, users searching for services usually interpret functionality, service classification terms, input and output parameters, among others, as requirements on their desired service. In turn, preferences can be considered as soft constraints whose degree

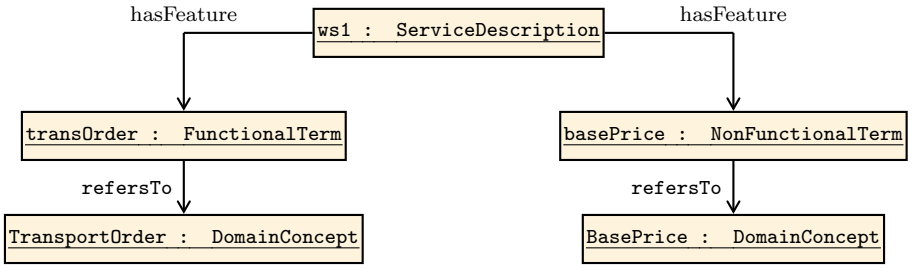

Figure 2: Graphical representation of the abstract service

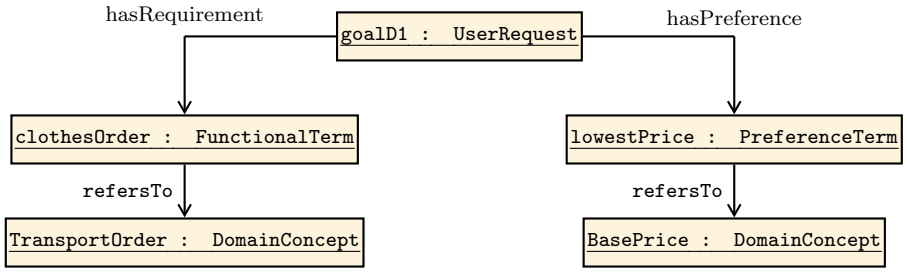

Figure 3: Graphical representation of the abstract user request

of fulfillment determine to what extent a candidate service is preferred against other candidate services that also fulfill the user requirements. In other words, ranking mechanisms evaluate preference terms in order to obtain the best candidate service with respect to a user request.

An example of a user request `:goalD1` defined within the SWS Challenge logistics scenario is showcased in Listing 2, along with its graphical representation in Figure 3. This request comprises a complex functional requirement term, which may contain pickup and delivery time among other information regarding the transportation of clothes, and a preference term referring to the base price.

Listing 2: Example of an abstract user request

```
:goalD1 a soup:UserRequest;
  soup:hasRequirement :clothesOrder;
  soup:hasPreference :lowestPrice.

:clothesOrder a soup:FunctionalTerm;
  soup:refersTo logistics:TransportOrder.
:lowestPrice a soup:PreferenceTerm;
  soup:refersTo logistics:BasePrice.
```

Both requirements and preferences are related with one or more `DomainConcept` classes, which are referred inside each term, and explicitly stated using the `refersTo` object property. Domain concepts usually represent service properties related to the domain-specific ontology used for service description, such as functional classification, input and output parameters types, process description, behavioral parameters, and non-functional properties, with the latter being specially important for preference terms definition. The above examples contains some logistics concepts such as a transport order and the base price for shipping.

Both functional and non-functional requirements specification has been widely discussed in the literature [13], and SWS frameworks provide sufficiently expressive facilities to define them, so in the following we will focus on preference modeling. Moreover, the validation scenario

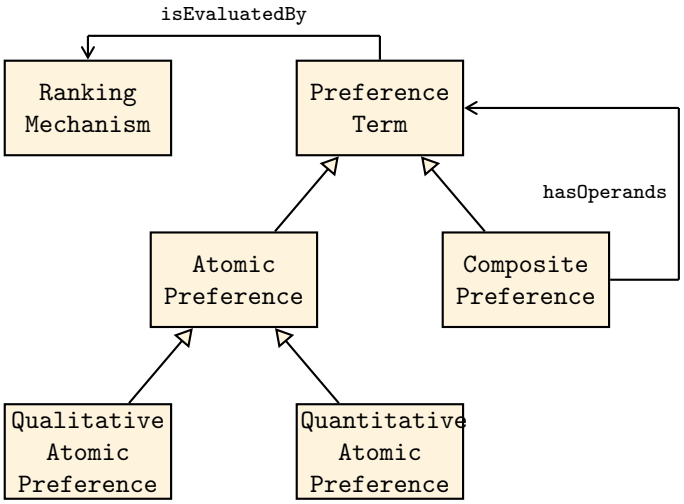

Figure 4: Middle ontology of preferences

described in [5] consists on a series of user requests whose requirement terms can be simply considered as property/property value pairs, so it is not necessary to define a complex hierarchy of functional terms in order to validate the upper ontology. Nevertheless, concrete applications may extend our upper ontology adding specialized terms in order to achieve a better integration with their discovery and ranking mechanisms, as in the case of our filtered discovery solution presented in [6].

3 SOUP: Defining an Ontology of User Preferences

Concerning preference terms, Figure 4 presents the middle ontology of SOUP, where we differentiate atomic preferences from composite ones. Thus, a `PreferenceTerm` can be an `Atomic Preference`, or a composition of two preference terms by applying a `Composite Preference`. On the one hand, atomic preferences are those which refers to a single domain concept, and can describe either a qualitative or a quantitative preference that users may have with respect to the referred service concept. On the other hand, composite preferences relate different preferences between them, so that a complex preference can be described using the `hasOperands` to associate a composite preference with its components.

As a preference is always related to some domain concepts, it can be intuitively expressed as "I prefer y rather than x", where x and y are instances of those concepts. This relationship between concept instances can be mathematically interpreted as a strict partial order. Therefore, we define a preference in general as:

Definition 1 (Preference) *Let $\mathcal{C}$ be a non-empty set of domain concepts, and* $\mathrm{dom}(\mathcal{C})$ *the set of all possible instances of those concepts. We define a preference as $\mathcal{P} = (\mathcal{C}, <^{\mathcal{P}})$, where $<^{\mathcal{P}} \subseteq \mathrm{dom}(\mathcal{C}) \times \mathrm{dom}(\mathcal{C})$ is a strict partial order (irreflexive, transitive and asymmetric), and if $x, y \in \mathrm{dom}(\mathcal{C})$, then $x <^{\mathcal{P}} y$ is interpreted as "I prefer* y *rather than* x*".*

If we consider a finite set of concept instance pairs $(x, y) \in <^{\mathcal{P}}$, $\mathcal{P}$ can be represented as a directed acyclic graph, also known as *Hasse diagrams* [2], where each node corresponds to

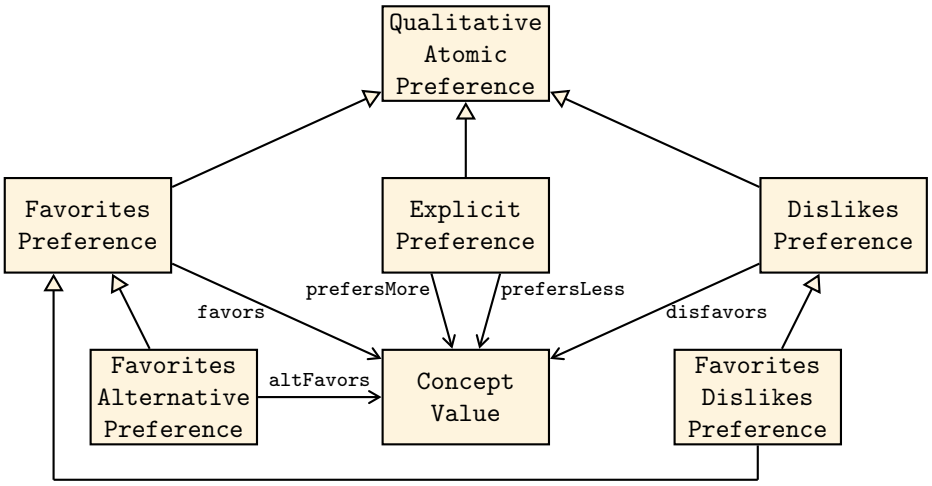

Figure 5: Qualitative atomic preference terms hierarchy

a concept instance, and edges represent the preference relationship $<^{\mathcal{P}}$. Furthermore, each preference term instance defines its order depending on the concrete concepts referred ($\mathcal{C}$) and some operand values that determine the evaluation of the $<^{\mathcal{P}}$ relation.

At this level we also add information about which particular ranking implementation is able to analyze and evaluate a certain preference term, associating this term with an instance of a `Ranking Mechanism` via the `isEvaluatedBy` object property. Our integrated ranking solution discussed in [4] makes use of this information in order to dynamically instantiate relevant ranking mechanisms when evaluating a particular preference term. Consequently, in order to abstract our preference model definition from the ranking implementation, we intentionally omit this property in the following.

Each preference construct derived from the hierarchy shown in Figure 4 is defined both formally and intuitively in the following, including a motivating example described in natural language from the SWS Challenge scenario used to validate our proposed model in [5], where some of these constructs are applied to describe that scenario goals. A formal discussion of the algebra of the described preference terms can be found at [8], where the foundations of our model are thoroughly discussed.

3.1 Qualitative atomic preferences

The first group of preferences that we present in the following corresponds to the qualitative and atomic constructs, which means that every preference $\mathcal{P} = (\mathcal{C}, <^{\mathcal{P}})$ that belongs to this kind refers to a single domain concept that represents a non-numerical service property, *i.e.* $|\mathcal{C}| = 1$. Figure 5 shows the available hierarchy of preference terms belonging to this group. Note that different terms use specific object properties to associate their operands to values from the referred domain concept, depending on the semantics of each preference construct. In each example, *italics* text correspond to service property values or instances used as operands, while `typewriter` text are used to denote domain concept classes that represents those properties.

Definition 2 (`FavoritesPreference`) *Let* $FAV \subseteq \mathrm{dom}(\mathcal{C})$ *be a non-empty, finite set of preferred values for property* $\mathcal{C}$*, and* $x, y \in \mathrm{dom}(\mathcal{C})$ *property values from two services.* $\mathcal{P}_{FAV} = (\mathcal{C}, <^{\mathcal{P}_{FAV}})$ *is a* ***`FavoritesPreference`*** *iff*

$$x <^{\mathcal{P}_{FAV}} y \iff x \notin FAV \wedge y \in FAV$$

A *favorites preference* defines a finite set of property values that constitute the desired values of the referred service property. Thus, services whose value for that property is a member of the *favorite set* are preferred to services that provide any other values from the property domain. An instance of this preference constructor has many operands as the cardinality of the favorite values set, associated using the `favors` object property.

Example: I prefer services that provide *carriageForward* as a possible `PaymentMethod`.

Definition 3 (`DislikesPreference`) *Let* $DIS \subseteq \mathrm{dom}(\mathcal{C})$ *be a non-empty, finite set of disliked values for property* $\mathcal{C}$*, and* $x, y \in \mathrm{dom}(\mathcal{C})$ *property values from two services.* $\mathcal{P}_{DIS} = (\mathcal{C}, <^{\mathcal{P}_{DIS}})$ *is a* ***`DislikesPreference`*** *iff*

$$x <^{\mathcal{P}_{DIS}} y \iff y \notin DIS \wedge x \in DIS$$

As opposite to `FavoritesPreference`, a *dislikes preference* defines a set of property values that the service should not provide for the referred property in order to be preferred to another service whose property values coincide with any of the values in the associated *dislikes set*. In this case, operands are linked to the term via the `disfavors` object property.

Example: I prefer SWSs that do not offer *refundForDamage* as an available `Insurance` option.

Definition 4 (`FavoritesAlternativePreference`) *Let* $FAV \subseteq \mathrm{dom}(\mathcal{C})$ *and* $ALT \subseteq \mathrm{dom}(\mathcal{C})$ *be two non-empty, finite sets of preferred values for property* $\mathcal{C}$*, and* $x, y \in \mathrm{dom}(\mathcal{C})$ *property values from two services.* $\mathcal{P}_{FAV,ALT} = (\mathcal{C}, <^{\mathcal{P}_{FAV,ALT}})$ *is a* ***`FavoritesAlternativePreference`*** *iff*

$$\begin{aligned} x <^{\mathcal{P}_{FAV,ALT}} y \iff & (x \in ALT \wedge y \in FAV) \vee \\ & (x \notin FAV \wedge x \notin ALT \wedge y \in ALT) \vee \\ & (x \notin FAV \wedge x \notin ALT \wedge y \in FAV) \end{aligned}$$

A *favorites or alternative preference* is an extension of `FavoritesPreference`, where there are two favorite sets. The second set is called *alternative set*, and links their values with the `altFavors` object property. In this case, services whose property values are in the favorite set are the most preferred. Otherwise their values should be on the alternative set. If this is not the case either, then the corresponding services will be undesirable, because their property values are not member of any of the two sets. Note that `favors` property is inherited because of the subclass relationship between `FavoritesPreference` and `FavoritesAlternativePreference`.

Example: I prefer SWSs whose `PaymentMethod` is *carriagePaid*, but if that is infeasible, then it should be *carriageForward*.

Definition 5 (`FavoritesDislikesPreference`) *Let $FAV \subseteq \mathrm{dom}(\mathcal{C})$ and $DIS \subseteq \mathrm{dom}(\mathcal{C})$ be two non-empty, finite sets of preferred and disliked values for property $\mathcal{C}$, and $x, y \in \mathrm{dom}(\mathcal{C})$ property values from two services. $\mathcal{P}_{FAV,DIS} = (\mathcal{C}, <^{\mathcal{P}_{FAV,DIS}})$ is a `FavoritesDislikesPreference` iff*

$$x <^{\mathcal{P}_{FAV,DIS}} y \iff (x \in DIS \wedge y \notin FAV) \vee (x \notin DIS \wedge x \notin FAV \wedge y \in FAV)$$

It is also possible to combine a `FavoritesPreference` with a `DislikesPreference` in the following form: a given service property should have a value on the defined favorite set. Otherwise, values should not belong to the dislikes set. If none of these two conditions hold, then the service will be less preferred than others fulfilling the first or the second condition. Again, subclass relationships bring both `favors` and `disfavors` object properties to this preference term.

Example: I prefer SWSs that provide *refundForLoss* as an option for `Insurance`, but if that is infeasible, then it should not be *refundForDamage*.

Definition 6 (`ExplicitPreference`) *Let $G = \{(v_1, v_2), \dots\}$ be a non-empty, finite directed acyclic graph that represents "better-than" relationships between its nodes $v_i \in \mathrm{dom}(\mathcal{C})$ corresponding to values of property $\mathcal{C}$, and V be the set of nodes belonging to G. Then, a strict partial order $E = (V, <^E)$ is induced as follows:*

$$a)\ (v_i, v_j) \in G \implies v_i <^E v_j$$
$$b)\ v_i <^E v_j \wedge v_j <^E v_k \implies v_i <^E v_k$$

Therefore, given $x, y \in \mathrm{dom}(\mathcal{C})$ property values from two services, $\mathcal{P}_E = (\mathcal{C}, <^{\mathcal{P}_E})$ is an `ExplicitPreference` iff:

$$x <^{\mathcal{P}_E} y \iff x <^E y \vee (x \notin V \wedge y \in V)$$

An *explicit preference* can be used to explicitly represent the strict partial order between a pair of property values. Thus, a directed acyclic graph comprising better-than relationships can be defined using several *explicit preferences*. In this case, `prefersMore` denote the value that is considered better than the `prefersLess` value.

Example: SWSs that provide *carriageForward* as a possible value for the `PaymentMethod` are more preferred than those that provide the *carriagePaid* value.

3.2 Quantitative atomic preferences

When the referred domain concept of an atomic preference is a numerical property, the quantitative constructs shown in Figure 6 may be used to express user preferences on that single property. Therefore, $\mathrm{dom}(\mathcal{C})$ values are numbers that support the total order operator $<$ and the subtraction $-$.

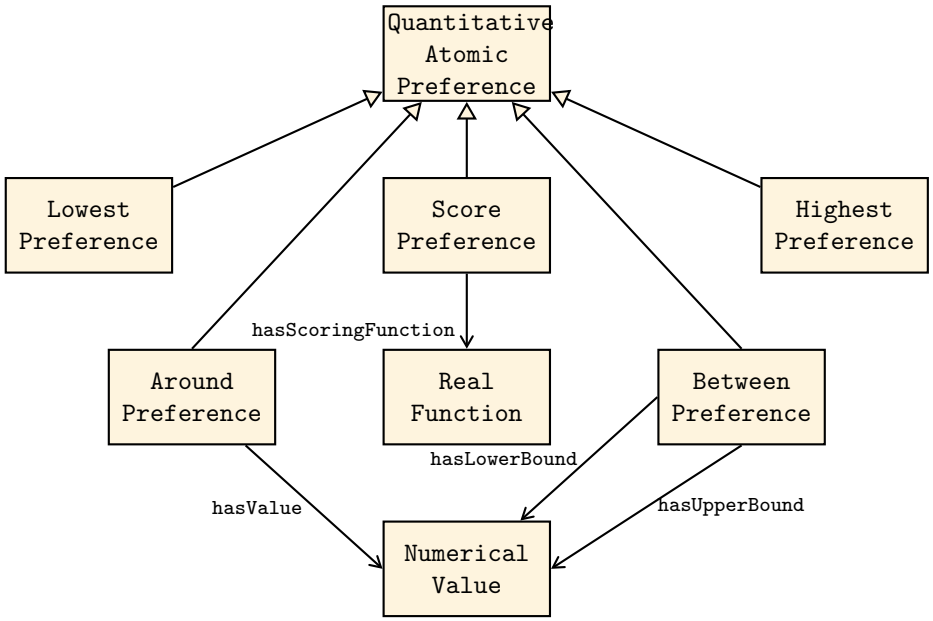

Figure 6: Quantitative atomic preference terms hierarchy

Definition 7 (`LowestPreference`) *Let $x, y \in \mathrm{dom}(\mathcal{C})$ be values for property $\mathcal{C}$ from two services. $\mathcal{P}_L = (\mathcal{C}, <^{\mathcal{P}_L})$ is a* ***`LowestPreference`*** *iff:*

$$x <^{\mathcal{P}_L} y \iff x > y$$

A *lowest preference* does not have any operand, but prefer services whose property values are as low as possible for the referred service property.

Example: I prefer SWSs that provide a `BasePrice` as low as possible.

Definition 8 (`HighestPreference`) *Let $x, y \in \mathrm{dom}(\mathcal{C})$ be values for property $\mathcal{C}$ from two services. $\mathcal{P}_H = (\mathcal{C}, <^{\mathcal{P}_H})$ is a* ***`HighestPreference`*** *iff:*

$$x <^{\mathcal{P}_H} y \iff x < y$$

In opposition to the last constructor, a *highest preference* is used when property values should be as high as possible.

Example: I prefer SWSs that provide a `PaymentDeadline` as long as possible.

Definition 9 (`AroundPreference`) *Let $z \in \mathrm{dom}(\mathcal{C})$ be the most preferred value of $\mathcal{C}$. For all values $v \in \mathrm{dom}(\mathcal{C})$ we define:*

$$dist(v, z) = |v - z|$$

Then, given $x, y \in \mathrm{dom}(\mathcal{C})$ property values from two services, $\mathcal{P}_z = (\mathcal{C}, <^{\mathcal{P}_z})$ is an ***`AroundPreference`*** *iff:*

$$x <^{\mathcal{P}_z} y \iff dist(x, z) > dist(y, z)$$

An *around preference* determines which property value is better by determining the distance of each values to a concrete value provided as an operand of this preference term using the `hasValue` object property. Thus, services which provide exactly that value are preferred to the rest of them. If this is infeasible, services with closer values to the operand are preferred.

Example: I prefer SWSs that provide a `BasePrice` closer to *180 Euros*.

Definition 10 (`BetweenPreference`) *Let* $[low, up] \in \mathrm{dom}(\mathcal{C}) \times \mathrm{dom}(\mathcal{C})$ *be the preferred values interval of* $\mathcal{C}$*. For all values* $v \in \mathrm{dom}(\mathcal{C})$ *we define:*

$$dist(v, [low, up]) = \begin{cases} 0 & \text{if } v \in [low, up] \\ low - v & \text{if } v < low \\ v - up & \text{if } v > up \end{cases}$$

In this case, given $x, y \in \mathrm{dom}(\mathcal{C})$ *property values from two services,* $\mathcal{P}_{[low,up]} = (\mathcal{C}, <^{\mathcal{P}_{[low,up]}})$ *is a* ***`BetweenPreference`*** *iff:*

$$x <^{\mathcal{P}_{[low,up]}} y \iff dist(x, [low, up]) > dist(y, [low, up])$$

In this case, a service should have values for the referred property between a range that are defined as operands in the preference (using `hasLowerBound` and `hasUpperBound` to actually define range bounds). If this is not the case, *between preferences* prefer services closer to the interval boundaries, computing the distance as in around preferences.

Example: I prefer SWSs that provide a `PaymentDeadline` within the interval of $[45, 60]$ days.

Definition 11 (`ScorePreference`) *Let* $f\colon \mathrm{dom}(\mathcal{C}) \to \mathbb{R}$ *be a scoring function and* $<$ *the usual less-than order in* $\mathbb{R}$*.* $\mathcal{P}_f = (\mathcal{C}, <^{\mathcal{P}_f})$ *is a* ***`ScorePreference`*** *iff for* $x, y \in \mathrm{dom}(\mathcal{C})$*:*

$$x <^{\mathcal{P}_f} y \iff f(x) < f(y)$$

A *score preference* basically defines a scoring function (*i.e.* a utility function like in [7], linked via `hasScoringFunction`) that takes a property value as its argument and returns a real value that can be interpreted in the following form: the higher the value returned by the function is, the more preferred the property value entered as the argument. Note that this kind of preference is not as intuitive as the rest, but it is still useful when a user wants to express complex grades of preference, using for instance a piecewise function depending on the property values.

Example: I prefer SWSs with the highest score with respect to `Price PerKg`, where the scoring function is defined as:

$$f(pricePerKg) = \frac{-1}{50} pricePerKg + 1$$

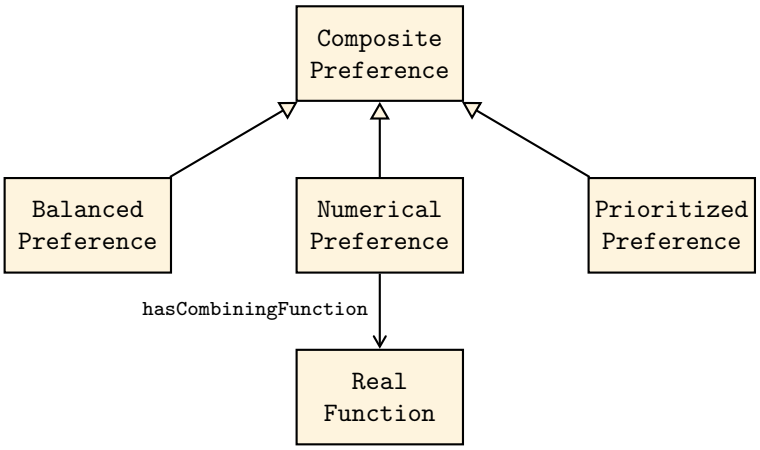

Figure 7: Composite preference terms hierarchy

3.3 Composite Preferences

The last group of preference constructs are used to compose two different preference terms by stating the preference relationship between each component term, which can be also a composite preference. Composite preferences `refersTo` property associate the preference with the union of the properties referred by component preferences.

These complex constructors are defined in the following for two preferences, though they can be trivially generalized to a greater number of preferences. Consequently, our model does not initially restrict the number of preference terms that can be composed using composite preferences.

Definition 12 (`BalancedPreference`) *Let $\mathcal{P}_1 = (\mathcal{C}_1, <^{\mathcal{P}_1})$ and $\mathcal{P}_2 = (\mathcal{C}_2, <^{\mathcal{P}_2})$ be two different preferences defined after $\mathcal{C}_1$ and $\mathcal{C}_2$ properties, and $x = (x_1, x_2), y = (y_1, y_2) \in \mathrm{dom}(\mathcal{C}_1) \times \mathrm{dom}(\mathcal{C}_2)$ be two value tuples for each property. $\mathcal{P} = (\mathcal{C}_1 \cup \mathcal{C}_2, <^{\mathcal{P}_1 \otimes \mathcal{P}_2})$ is a `BalancedPreference` iff:*

$$x <^{\mathcal{P}_1 \otimes \mathcal{P}_2} y \iff (x_1 <^{\mathcal{P}_1} y_1 \wedge (x_2 <^{\mathcal{P}_2} y_2 \vee x_2 = y_2)) \vee (x_2 <^{\mathcal{P}_2} y_2 \wedge (x_1 <^{\mathcal{P}_1} y_1 \vee x_1 = y_1))$$

A *balanced preference* $\mathcal{P}$ combines two preference terms $\mathcal{P}_1$ and $\mathcal{P}_2$ using the *Pareto-optimality principle*, which considers that $\mathcal{P}_1$ and $\mathcal{P}_2$ are equally important preferences. Thus, a service SWS_1 is better than another service SWS_2 with respect to $\mathcal{P}$, if SWS_1 is better than SWS_2 with respect to $\mathcal{P}_1$ and SWS_1 is not worse than SWS_2 with respect to $\mathcal{P}_2$, and *vice versa*. Intuitively, this preference balance the fulfillment of each preference component, so that the composite preference is the average degree of preference taking both components into account.

Example: I prefer SWSs that best fit (with an average satisfaction) the following three (atomic) preferences: the lowest `BasePrice`, the `PaymentDeadline` within the interval of $[45, 60]$ days, and provided `Insurance` options of *refundForLoss* or *refundForDamage*.

Definition 13 (`PrioritizedPreference`) *Let $\mathcal{P}_1 = (\mathcal{C}_1, <^{\mathcal{P}_1})$ and $\mathcal{P}_2 = (\mathcal{C}_2, <^{\mathcal{P}_2})$ be two different preferences defined after $\mathcal{C}_1$ and $\mathcal{C}_2$ properties, and $x = (x_1, x_2), y = (y_1, y_2) \in \mathrm{dom}(\mathcal{C}_1) \times$*

$\mathrm{dom}(\mathcal{C}_2)$ *be two value tuples for each property.* $\mathcal{P} = (\mathcal{C}_1 \cup \mathcal{C}_2, <^{\mathcal{P}_1 \& \mathcal{P}_2})$ *is a* ***PrioritizedPreference*** *iff:*

$$x <^{\mathcal{P}_1 \& \mathcal{P}_2} y \iff x_1 <^{\mathcal{P}_1} y_1 \vee (x_1 = y_1 \wedge x_2 <^{\mathcal{P}_2} y_2)$$

In the case of a *prioritized preference* $\mathcal{P}$ that compose two preference terms $\mathcal{P}_1$ and $\mathcal{P}_2$, $\mathcal{P}_1$ is considered more important than $\mathcal{P}_2$. Thus, $\mathcal{P}_2$ is evaluated only if $\mathcal{P}_1$ does not mind (*i.e.* service property values compared using $\mathcal{P}_1$ do not return enough information to rank those services). In this case, operands have to be evaluated in a specific order, so the `hasOperands` property should be properly specialized to account for operands ordering. For instance, the range of the property could be defined as an Resource Description Framework (RDF) list.

Example: I prefer SWSs that provide *carriageForward* as a possible `PaymentMethod`. In the case of equal satisfaction degree on that preference, I prefer SWSs whose `BasePrice` are closer to *180 Euros*.

Definition 14 (`NumericalPreference`) *Let* f *and* g *be two scoring functions that define score preferences* $\mathcal{P}_f = (\mathcal{C}_1, <^{\mathcal{P}_f})$ *and* $\mathcal{P}_g = (\mathcal{C}_2, <^{\mathcal{P}_g})$*, respectively, and* $F \colon \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ *be a combining function. For* $x = (x_1, x_2), y = (y_1, y_2) \in \mathrm{dom}(\mathcal{C}_1) \times \mathrm{dom}(\mathcal{C}_2)$*,* $\mathcal{P} = (\mathcal{C}_1 \cup \mathcal{C}_2, <^{\mathrm{rank}_F(\mathcal{P}_f, \mathcal{P}_g)})$ *is a* ***NumericalPreference*** *iff:*

$$x <^{\mathrm{rank}_F(\mathcal{P}_f, \mathcal{P}_g)} y \iff F(f(x_1), g(x_2)) < F(f(y_1), g(y_2))$$

Finally, a *numerical preference* is the combination of a number of score preferences using a function that takes the values returned by the score preferences as its arguments and returns another real number that gives information about the global preference, considering all the properties referred by concrete score preferences. Notice that component preferences must be score preferences in order to properly compose them using a combining function, which is associated with this term using the `hasCombiningFunction` object property.

Example: Provided that $f(basePrice)$ and $g(pricePerKg)$ are already defined and they range within the interval $[0, 1]$, I prefer SWSs that have a higher combined score, where the combining function is defined as:

$$F(basePrice, pricePerKg) = 0.8 * f(basePrice) + 0.4 * g(pricePerKg)$$

4 Conclusions

In this report, a highly expressive preference model for SWS discovery and ranking named SOUP is described. This model, specified as an ontology, represents a novel approach that leverages preference descriptions so that they become a first-class citizen in SWS frameworks. Additionally, SOUP has been validated using a complex discovery scenario from the SWS Challenge in order to prove the applicability of our solution to an actual discovery and ranking scenario [5]. The main benefits of our proposed model can be summarized as follows:

- **Expressiveness**. The model is sufficiently expressive to describe complex user desires about requested services, providing a comprehensive hierarchy of preference terms.

- **Intuitive semantics**. Based on a strict partial order interpretation of preferences, the model is user-friendly and machine-readable, so preferences may be automatically processed and inferred.
- **Qualitative and Quantitative**. Available constructs allow to express both qualitative and quantitative preferences, and even combine them in a general preference term.
- **Independence**. Our proposal is not coupled with a concrete SWS solution, neither with a discovery nor ranking mechanism, so it is not limited by the formalisms used to implement these mechanisms.
- **Extensibility**. Because the model is presented as an ontology, it can be further extended with new preference constructs with ease.
- **Applicability**. Our model can be implemented within any SWS framework, extending current proposals to leverage preference descriptions.

References

[1] Saartje Brockmans, Raphael Volz, Andreas Eberhart, and Peter Löffler. Visual modeling of owl dl ontologies using uml. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 198–213. Springer, 2004.

[2] Brian A. Davey and Hilary A. Priestley. *Introduction to Lattices and Order (2. ed.)*. Cambridge University Press, 2002.

[3] Joel Farrell and Holger Lausen. Semantic annotations for WSDL and XML Schema. Technical report, World Wide Web Consortium, August 2007.

[4] José María García, Martin Junghans, David Ruiz, Sudhir Agarwal, and Antonio Ruiz-Cortés. Integrating semantic web services ranking mechanisms using a common preference model. *Knowledge-Based Systems*, 2012. Under Review.

[5] José María García, David Ruiz, and Antonio Ruiz-Cortés. A model of user preferences for semantic services discovery and ranking. In Lora Aroyo, Grigoris Antoniou, Eero Hyvönen, Annette ten Teije, Heiner Stuckenschmidt, Liliana Cabral, and Tania Tudorache, editors, *ESWC (2)*, volume 6089 of *Lecture Notes in Computer Science*, pages 1–14. Springer, 2010.

[6] José María García, David Ruiz, and Antonio Ruiz-Cortés. Improving semantic web services discovery using sparql-based repository filtering. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:12–24, 2012.

[7] José María García, Ioan Toma, David Ruiz, and Antonio Ruiz-Cortés. A service ranker based on logic rules evaluation and constraint programming. In Flavio de Paoli, Ioan Toma, Andrea Maurino, Marcel Tilly, and Glen Dobson, editors, *NFPSLA-SOC'08*, volume 411 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[8] Werner Kießling. Foundations of preferences in database systems. In *VLDB*, pages 311–322. Morgan Kaufmann, 2002.

[9] Steffen Lamparter, Anupriya Ankolekar, Rudi Studer, and Stephan Grimm. Preference-based selection of highly configurable web services. In Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, and Prashant J. Shenoy, editors, *WWW*, pages 1013–1022. ACM, 2007.

[10] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara. OWL-S: Semantic markup for web services. Technical report, DAML, 2006.

[11] E. Michael Maximilien and Munindar P. Singh. A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, 8(5):84–93, 2004.

[12] Dumitru Roman, Uwe Keller, Holger Lausen, Jos de Bruijn, Rubén Lara, Michael Stollberg, Axel Polleres, Cristina Feier, Christoph Bussler, and Dieter Fensel. Web service modeling ontology. *Applied Ontology*, 1(1):77–106, 2005.

[13] Antonio Ruiz-Cortés, Octavio Martín-Díaz, Amador Durán, and Miguel Toro. Improving the automatic procurement of web services using constraint programming. *Int. J. Cooperative Inf. Syst.*, 14(4):439–468, 2005.

[14] Xia Wang, Tomas Vitvar, Mick Kerrigan, and Ioan Toma. A qos-aware selection model for semantic web services. In Asit Dan and Winfried Lamersdorf, editors, *ICSOC*, volume 4294 of *Lecture Notes in Computer Science*, pages 390–401. Springer, 2006.

[15] Chen Zhou, Liang-Tien Chia, and Bu-Sung Lee. Daml-qos ontology for web services. In *ICWS*, pages 472–479. IEEE, IEEE Computer Society, 2004.