4
Related Work on Preference Representation

Given that we presented in the previous chapter a preference metamodel based on a study of how humans express preferences, we now introduce existing research work in the context of preferences that proposes representation models of preferences. These models may be part of approaches that include algorithms to reason about preferences, but our focus in this chapter is to describe how these approaches *represent* preferences, and to compare which kind of preferences can be *explicitly* expressed by the different proposed models, showing their limitations with respect to our metamodel.

Preference representation models are split into five groups. First, we present approaches that use constraints to represent preferences in Section 4.1. The second group consists of approaches that represent preferences using graphs, which are detailed in Section 4.2. Most of the approaches of these two groups are investigated within the artificial intelligence research area, but there are preference representation models proposed in the area of databases and semantic web, which are described in Sections 4.3 and 4.4, respectively. Finally, we present a recent approach that provides a non-parametric way of representing preferences in Section 4.5. We compare the presented research work in Section 4.6, and conclude in Section 4.7.

4.1
Constraint-based Approaches

4.1.1
Soft-constraints

Soft constraints model quantitative preferences by generalising the traditional formalism of hard constraints. Bistarelli et al. (Bistarelli et al. 1997) define a constraint solving framework where all such extensions, as well as classical Constraint Satisfaction Problems (CSPs), can be cast. The main idea is based on the observation that a semiring (i.e. a domain plus two operations satisfying certain properties) is all that is needed to describe many constraint satisfaction schemes. The domain of the semiring provides the levels of consistency (which can be interpreted, for example, as cost, degrees of preference, probabilities), and the two operations define a way to combine constraints together. So, in a soft constraint, each assignment to the variables of a constraint is annotated with a level of its desirability, and the desirability of a complete assignment is computed by a combination operator applied to the local preference values.

Soft constraint-based approaches rely on Soft Constraint Satisfaction Problems (SCSPs), which are an extension of CSPs. The definition of a SCSP is based on the definitions of constraint system and constraint, and both involve the choice of a c-semiring — in this term, "c" stands for "constraint," meaning that this kind of semiring is a natural structure to be used when handling constraints. Basically, a SCSP defines the c-semiring being used, a set of attributes that describe an application area, and their associated domains. In addition, there is a set of constraints, each associated with a value that is interpreted according to the chosen c-semiring. Formally, a SCSP is defined as follows (Bistarelli et al. 1997).

1. **Constraint system**. A constraint system is a tuple $CS = \langle S, D, V \rangle$, where S is a c-semiring, D is a finite set, and V is an ordered set of attributes.

2. **Constraint**. Given a constraint system $CS = \langle S, D, V \rangle$, a constraint over CS is a pair $\langle def, con \rangle$, where

   - $con \subseteq V$, it is called the *type* of the constraint;
   - $def : D^k \to A$ (where k is the cardinality of con and A is the set of possible values representing the penalty, cost, preference, weight, etc.) is called the *value* of the constraint.

3. **Constraint problem**. Given a constraint system $CS = \langle S, D, V \rangle$, a constraint problem P over CS is a pair $P = \langle C, con \rangle$, where C is a set of constraints over CS and $con \subseteq V$. We also assume that $\langle def_1, con' \rangle \in C$ and $\langle def_2, con' \rangle \in C$ implies $def_1 = def_2$.

Bistarelli et al. use this framework to deal with *bipolar* preferences (Bistarelli et al. 2010), i.e. problems with both positive and negative preferences. They argue that soft-constraints *only can model negative preferences*, since in this framework preference combination returns lower preferences. So, they adopt the soft constraint formalism based on semirings to model negative preferences and also define a new algebraic structure to model positive preferences. Then, to model bipolar problems, these two structures are linked by a combination operator between positive and negative preferences to model preference compensation. Another extension to soft-constraints consists of *interval preferences* (Gelain et al. 2010), which instead of representing the *value* of the constraint as a specific number, it uses an interval, which may be easier to be specify.

4.1.2 Preference-based Problem Solving for Constraint Programming

An approach based on multi-objective optimisation was proposed by Junker (Junker 2008), considering a finite set of attributes X where each attribute $x \in X$ has a domain $D(x)$. The problem space of X is restricted by defining constraints on attributes in X. A constraint c has a scope $X_c \subseteq X$ and a "relation," which is expressed by a set R_c of assignments to the scope X_c.

Users must provide as input preferences on certain properties of the option. These criteria are mathematical functions from the problem space to an outcome domain. Formally, a criterion z with domain Ω is an expression $f(x_1, ..., x_n)$ where $x_1, ..., x_n$ are attributes from X and f is a function with signature $D(x_1) \times ... \times D(x_n) \rightarrow \Omega$. Function f can be formulated with the operators of the constraint language (e.g. sum, min, max, conditional expression) or by a table.

The user can compare the different outcomes in a domain Ω and formulate preferences between them. Preferences are modelled in form of a preorder $\gtrsim$ on Ω, which consists of a strict part $>$ and an indifference relation $\sim$. The user may also formulate *wishes* about the properties that an option should have. Such a wish is a soft constraint that should be satisfied if possible. A wish for constraint c can thus be modelled by a preference $\langle z_c, > \rangle$, which is abbreviated by $wish(c)$.

Users can inspect the conflicts between criteria and the way they have been resolved. If they are not satisfied with such a conflict resolution, they can change it by reordering the criteria. This importance is ordered in terms of a strict partial order $I \subseteq Z \times Z$ on the criteria set $Z := z_1, ..., z_n$.

4.2 Graphically-structured Approaches

4.2.1 CP-nets: Conditional Ceteris Paribus Preference Statements

Boutilier et al. (Boutilier et al. 2004) have proposed a graphical representation, namely CP-nets,[1] which can be used for specifying preference relations in a relatively compact and structured manner using conditional ceteris paribus (all else being equal) preference statements. The inference techniques for CP-nets focus on two questions: how to perform preferential comparison between outcomes, and how to find the optimal outcome given a partial assignment to the problem attributes.

In CP-nets, a set of variables $V = X_1, ..., X_n$ is assumed, over which the decision maker has preferences. Each variable X_i is associated with a domain

[1]This structures are conditional preference networks or CP-networks (CP-nets, for short).

$Dom(X_i) = x_1^i, ..., x_{n_i}^i$ of values it can take. An assignment x of values to a set $X \subseteq V$ of variables (also called an instantiation of X) is a function that maps each variable in X onto an element of its domain; if $X = V$, x is a complete assignment, otherwise x is called a partial assignment. The set of all assignments to $X \subseteq V$ is denoted by $Asst(X)$. Based on this assumption, two definitions are made.

- A set of variables X is preferentially independent of its complement $Y = V - X$, for all $x_1, x_2 \in Asst(X)$ and $y1, y2 \in Asst(Y)$, we have $x_1 y_1 \succcurlyeq x_2 y_1$ if and only if $x_1 y_2 \succcurlyeq x_2 y_2$.
- X is conditionally preferentially independent of Y given an assignment z to Z if and only if, for all $x_1, x_2 \in Asst(X)$ and $y_1, y_2 \in Asst(Y)$, we have $x_1 y_1 z \succcurlyeq x_2 y_1 z$ if and only if $x_1 y_2 z \succcurlyeq x_2 y_2 z$.

A CP-net over variables $V = X_1, ..., X_2$ is a directed graph G over $X_1, ..., X_2$ whose nodes are annotated with conditional preference tables $CPT(X_i)$ for each $X_i \in V$. Each conditional preference table $CPT(X_i)$ associates a total order $\succ_u^i$ with each instantiation u of X_i's parents $Pa(X_i) = U$.

In summary, the kinds of statements captured by CP-Nets are those that establish order among attribute values, conditioned to values set to other attributes (parent attributes).

4.2.2
TCP-nets: Modelling of Preference and Importance

Brafman et al. (Brafman et al. 2006) provide an extension of the CP-nets formalism in order to handle another class of qualitative statements — statements of **relative importance** of attributes. These statements have the form: "*It is more important to me that the value of X be better than that the value of Y be better.*" A more refined notion of importance, which is also addressed, is that of **conditional relative importance**, having this form: "*A better assignment for X is more important than a better assignment for Y given that $Z = z_0$.*"

The resulting extended formalism, TCP-nets (for tradeoffs-enhanced CP-nets), maintains the ideas of CP-nets, as it remains focused on using only simple and natural preference statements. In addition, it also uses the ceteris paribus semantics, and utilises a graphical representation of this information to reason about its consistency and to perform, possibly constrained, optimisation using it. The extra expressiveness it provides allows better modelling of trade-offs, with attribute importance relationships. When the TCP-net structure is "acyclic," the set of preference statements represented by the TCP-net is guaranteed to be consistent. TCP-nets are annotated graphs with three types of edges.

- A first type of (directed) edges, which comes from the original CP-nets model and captures direct preferential dependencies.
- A second (directed) edge type captures relative importance relations. The existence of such an edge from attribute X to attribute Y implies that X is more important than Y.
- A third (undirected) edge type captures conditional importance relations: such an edge between nodes X and Y exists if there exists a non-empty attribute subset $Z \subseteq V - \{X, Y\}$ for which $RI(X, Y \mid Z)$ (relative importance of X and Y conditioned on Z) holds.

In addition to the conditional preference table (CPT) of CP-Nets, in TCP-nets, each undirected edge is annotated with a conditional importance table (CIT). The CIT associated with such an edge (X, Y) describes the relative importance of X and Y given the value of the corresponding importance-conditioning variables Z.

4.3 Database Approaches

4.3.1 Scoring Function

Agrawal and Wimmers (Agrawal and Wimmers 2000) propose a framework for *expressing* and *combining* user preferences. In this framework, preferences for an entity are expressed by a numeric score between 0 and 1, vetoing it, or explicitly stating indifference (by default, indifference is assumed). The framework consists of three elements, as shown below.

- A set of **(base) types**, which typically include `ints`, `strings`, `floats`, `booleans`, etc.
- A data type called **score** that represents a user preference. Formally, this is $[0, 1] \cup \natural, \perp$. A score of 1 indicates the highest level of user preference, while a score of 0 indicates its lowest level. The "$\natural$" score, represents a veto, and the "$\perp$" score represents that no user preference has been indicated.
- A set of **record types**, which are pairs $name_1 : type_2, ..., name_n : type_n$ in which all n names (a name is simply a non-empty string) are different (although the types are allowed to be the same). In this case, $name_i$ is the name of a field in the record and $type_i$ is the type of that field. An option is a record, where each field takes on a value in the type of that field. In addition, a type is called wild if it contains "*," used to indicate a wild card that "matches" any value.

Based on these three elements, a preference function is defined as a function that maps options of a given record type to a score. Since sometimes a preference function is applied to an option with more fields than are present in the domain of the preference function, a projection operator is introduced to eliminate the extra fields.

4.3.2 Preference Formulae in Relational Queries

Chomicki (Chomicki 2003) proposes a framework for specifying preferences using logical formulae and their embedding into relational algebra. Preferences are defined using binary preference relations between tuples, which are specified using first-order formulae. The focus is mostly on *intrinsic* preference formulae, which can refer only to built-in predicates. Two infinite domains are considered in the approach: D (uninterpreted constants) and Q (rational numbers).

> Given a relation schema $R(A_1...A_k)$ such that U_i, $1 \leq i \leq k$, is the domain (either D or Q) of the attribute A_i, a relation $\succ$ is a preference relation over R if it is a subset of $(U_1 \times ... \times U_k) \times (U_1 \times ... \times U_k)$.

In addition, this preference relation has the following properties: irreflexivity, asymmetry, transitivity, negative transitivity and connectivity.

> A preference formula (pf) $C(t1, t2)$ is a first-order formula defining a preference relation $\succ$ in the standard sense, namely $t1 \succ_c t2$ iff $C(t1, t2)$. An intrinsic preference formula (ipf) is a preference formula that uses only built-in predicates.

Because two specific domains are considered, D and Q, there are two kinds of attributes, D-attributes and Q-attributes, and two kinds of atomic formulae: (i) equality constraints; and (ii) rational-order constraints. Without loss of generality, the author assumes that ipfs are in DNF (Disjunctive Normal Form) and quantifier-free. Moreover, atomic formulae are closed under negation (also satisfied by the above theories). *Indifferent* is also defined. Every preference relation $\succ_c$ generates an indifference relation $\sim_c$: two options $t1$ and $t2$ are indifferent ($t1 \sim_c t2$) if neither is preferred to the other one, that is, $t1 \succ_c t2$ and $t2 \succ_c t1$.

Finally, there is preference composition, which can be unidimensional or multidimensional. In unidimensional composition, a number of preference relations over a single database schema are composed, producing another preference relation over the same schema. In multidimensional composition, there is a number of preference relations defined over several database relation schemas, and a preference relation over the Cartesian product of those relations is defined. Unidimensional composition can be: (i) *boolean composition*: union, intersection and difference of preference relations; (ii) *prioritised composition*: preference over preferences, represented by the $\rhd$ operator; and (iii) *transitive closure*. Multidimensional composition, in turn, can be pareto composition and lexicographic composition.

4.3.3 Foundations of Preferences in Database Systems

A preference model tailored for database systems, proposed by Kießling (Kießling 2002), unifies and extends existing approaches for non-numerical and numerical ranking. He defined a declarative semantics of preference queries under the Best-Matches-Only (BMO) query model. His preference model includes a set of preference constructors, which allows the expression of different preferences, whose definition is given below.

Preference $P = (A, <P)$. Given a set A of attribute names, a preference P is a strict partial order $P = (A, <P)$, where $<P \subseteq dom(A) \times dom(A)$. $<P$ is irreflexive and transitive (which imply asymmetry). Important is this intended interpretation: "$x <P\ y$" is interpreted as "I like y better than x."

In order to build *base* preferences, there are two classes of constructors (detailed in Table 4.1): (i) *non-numerical base preferences*, which include positive and negative preferences; and (ii) *numerical base preferences*, which include preferences that specify preferred numerical values of an attribute using, for example, intervals. A preference term (i.e. a preference that is valid according to the provided constructors) can also be composed of other preferences. Given preference terms $P1$ and $P2$, P is a preference term if and only if P is one of the following.

1. Any **base** preference: $P := basepref_i$.
2. Any **subset** preference: $P := P1^{\subseteq}$
3. Any **dual** preference: $P := P1^{\partial}$
4. Any **complex** preference P gained by applying one of the following preference constructors:
   - Accumulating preference constructors:
     - Pareto accumulation: $P := P1 \otimes P2$
     - Prioritised accumulation: $P := P1\ \&\ P2$
     - Numerical accumulation: $P := rank_F(P1, P2)$ (applied only to SCORE preferences)
   - Aggregating preference constructors:

Non-numerical base preferences	
POS(A, POS-set)	A desired value should be in a finite set of favourites $POS - set \subseteq dom(A)$. If this infeasible, any other value from $dom(A)$ is acceptable.
NEG(A, NEG-set)	A desired value should not be any from a finite set NEG-set of dislikes. If this is infeasible, any disliked value is acceptable.
POS/NEG(A, POS-set; NEG-set)	A desired value should be one from a finite set of favourites. Otherwise it should not be any from a finite set of disjoint dislikes. If this is not feasible either, any disliked value is acceptable.
POS/POS(A, POS1-set; POS2-set)	A desired value should be amongst a finite set POS1-set. Otherwise it should be from a disjoint finite set of alternatives POS2-set. If this is not feasible either, any other value is acceptable.
EXP(A, E-graph)	Let $E - graph = (val_1, val_2), ...$ represent a finite acyclic 'better-than' graph, V be the set of all val_i occurring in $E - graph$. A strict partial order $E = (V, <E)$ is induced as follows: (i) $(val_i, val_j) \in E - graph$ implies $val_i <E\ val_j$; and (ii) $val_i <E\ val_j \wedge val_j <E\ val_k$ imply $val_i <E\ val_k$. P is an EXPLICIT preference, if: $x <P\ y$ iff $x <E\ y \vee (x \notin range(<E) \wedge y \in range(<E))$.
Numerical base preferences	
AROUND(A, z)	The desired value should be z. If this is infeasible, values with shortest distance apart from z are acceptable.
BETWEEN(A, [low, up])	A desired value should be between the bounds of an interval. If this is infeasible, values with shortest distance apart from the interval boundaries will be acceptable.
LOWEST(A), HIGHEST(A)	A desired value should be as low (high) as possible.
SCORE(A, f)	Assume a scoring function $f : dom(A) \rightarrow \mathbb{R}$. Let $<$ be the familiar 'less-than' order on $\mathbb{R}$. P is called SCORE preference, if for $x, y \in dom(A)$: $x <P\ y$ iff $f(x) < f(y)$. No intuitive interpretation is given.

Table 4.1: Base Preference Constructors.

– Intersection aggregation: $P := P1 \blacklozenge P2$
– Disjoint union aggregation: $P := P1 + P2$
– Linear sum aggregation: $P := P1 \oplus P2$

4.3.4 Personalisation of Queries based on User Preferences

Koutrika and Ioannidis (Koutrika and Ioannidis 2006) have addressed query personalisation, by proposing (i) a model for representing and storing preferences in user profiles, (ii) a query personalisation framework that specifies which kind of personalised answer is generated given a query and a user profile; and (iii) query personalisation algorithms. Preferences may be expressed for values of attributes, and for relationships between entities, which indicate to what degree, if any, entities related depend on each other. The following preferences are part of the preference model.

(i) **Atomic Selection Preferences**. For any atomic selection condition q on attribute $R.A$ (of a relational table R_A, with D_A as its domain specific values), a user's preference for values satisfying (or not) q is expressed by the degree of interest in q, denoted by $doi(q)$, which is defined as follows:

$$doi(q) = \langle d_T(u), d_F(u) \rangle$$

where $\forall\, u \in D_A$, satisfying q, $d_T(u), d_F(u) \in [-1, 1]$ and $d_T(u) * d_F(u) \leq 0$. Based on this definition, three aspects of preferences can be modelled, as shown below.

(a) **Valence**. Preferences may be *positive* (expressing liking), *negative* (expressing dislike) or *indifferent* (expressing don't care). This is expressed by giving a positive or negative value to $d_T(u)$, $d_F(u)$.

(b) **Concern**. A user's concern is captured by the pair $\langle d_T(u), d_F(u) \rangle$, and $d_T(u)$ captures a user's concern for the *presence* of values u of $R.A$ (or any other path of the schema leading to $R.A$) that make q evaluate to true, while $d_F(u)$ captures a user's concern for the *absence* of the same values, i.e. for q evaluating to false.

(c) **Elasticity**. Preferences may be *exact* or *elastic* depending on whether the domain D_A is categorical or numeric. Constants are attributed to $d_T(u), d_F(u)$ to represent a constant preference for a value, and functions are adopted to represent preference that varies according to the value.

(ii) **Join Preferences.** Join preference indicates a preference for joining two entities, it expresses the dependence of the left part of the join on the right part (using database query vocabulary). A user preference for a join condition q is expressed by the degree of interest in q, $doi(q)$, defined as: $doi(q) = \langle d \rangle$, where $d \in [0, 1]$.

(iii) **Implicit Preferences.** User's preferences over the contents of a database can be expressed on top of a personalisation graph (Figure 4.1). This is a directed graph $G(V, E)$ and it is an extension of the database schema graph. Nodes in V are (a) *relation nodes*, one for each relation in the schema, (b) *attribute nodes*, one for each attribute of each relation in the schema, and (c) *value nodes*, one for each value that is of any interest to a particular user. Likewise, edges in E are (a) *selection edges*, from an attribute node to a value node; such an edge represents the potential selection condition connecting the attribute and the value, and (b) *join edges*, from an attribute node to another

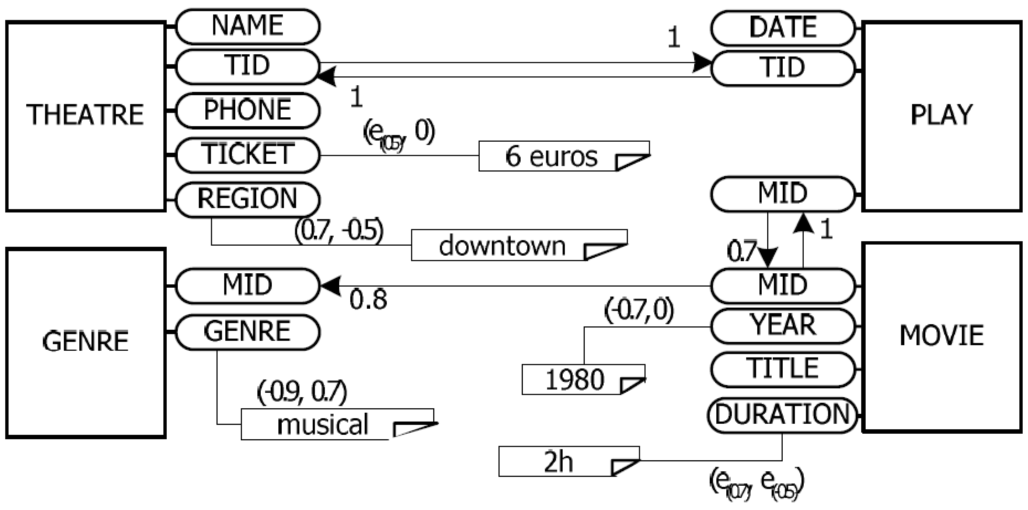


Figure 4.1: Personalisation Graph (Koutrika and Ioannidis 2006).

attribute node; such an edge represents the potential join condition between these attributes.

Implicit preferences are those derived from atomic preferences. An implicit join preference is mapped onto a path in the personalisation graph between two attribute nodes. An implicit selection preference is mapped to a path in the personalisation graph from an attribute node to a value node.

(iv) **Combination Preferences.** *Satisfaction* of an atomic or implicit selection preference $\langle q, d_T, d_F \rangle$ is equivalent to satisfaction of q if $d_T \geq 0$ or failure of q if $d_F \leq 0$. *Failure* of a preference is the exact opposite. Thus, the doi in the satisfaction of a preference is $d^+(u) = max(d_T(u), d_F(u))$. The degree of interest in the failure is $d^-(u) = min(d_T(u), d_F(u))$. The overall degree of interest in a combination of preferences is calculated using a *ranking function*. Three cases are distinguished: (a) all preferences are satisfied (positive combination), (b) none of the preferences is satisfied (negative combination), and (c) some preferences are satisfied and others not (mixed combination).

(v) **Preference Order.** The notion of degree of criticality is introduced for ordering preferences and selecting the top K of preferences. Intuitively, the most important or critical preference is that with the highest d^+, and the lowest d^-.

Even though there are five different kinds of preferences, only the first two (atomic selection and join preferences) are expressed by users; the remaining three are preferences derived from preferences represented as atomic selection and join preferences.

Preference	Definition
AroundPreference	It represents the preference for a value that is around a reference value.
IntervalPreference	It is similar to the AroundPreference, but it is associated with a range or interval of values.
ExtremalPreference	It represents the preference for maximising or minimising the value of a property.
NegativePreference	It indicates a value that is less preferred than others.
PositivePreference	It indicates a value that is more preferred than others.

Table 4.2: Types of AttributeValuePreference.

4.4 Semantic Web Approaches

4.4.1 OWLPref: a Declarative Preference Representation

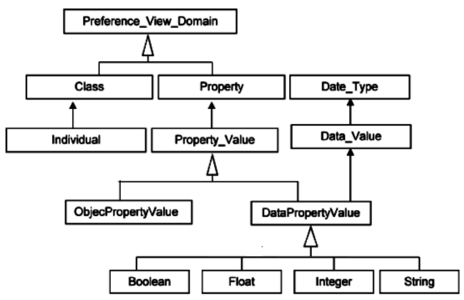
Ayres and Furtado (Ayres and Furtado 2007) proposed a declarative, domain-independent way of representing preferences in OWL, namely OWLPref. It is an ontology that defines different kinds of preferences, which are split into two groups: SimplePreference and CompositePreference. The latter makes compositions of the former.

There are three different types of simple preferences: (i) ClassPreference: it represents a preference than indicates that a certain class is preferred to another; (ii) AttributePreference: it represents a preference than indicates that a certain attribute is preferred to another; and (iii) AttributeValuePreference: it represents preferences for attribute values. The latter has five different subtypes, detailed in Table 4.2.

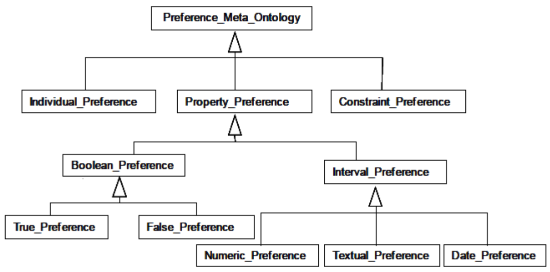
Composite preferences have three different types as well, and they indicate a relationship between two simple preferences. The first case, ParetoPreference, indicates when these two preferences are equally preferred. In order to indicate when two simple preferences are mutually exclusive, a DisjunctionPreference should be used. Last, an OrderPreference indicates which of two preferences is preferred to another.

Moreover, OWLPref allows the representation of ConditionalPreference. This concept models preferences that vary according to the context. It is composed of two properties: onCondition and hasPreference. The first allows representing the conditions to which the preference is applicable, while the second is a reference to the preference that is conditioned to the stated condition.

Finally, even if it is not explicitly defined, the authors mention that preferences can be organised into hierarchies, so that one can define priority among them.



4.2(a): PVD.



4.2(b): PMO.

Figure 4.2: Metamodels (Tapucu et al. 2008).

4.4.2 Metamodelling Approach to Preference Management

A metamodeling approach to preference management was presented by Tapucu et al. (Tapucu et al. 2008). They propose a preference metamodel, which consists of concepts and semantic relations to represent interests of users. Their motivation is that users may have the same type of preferences in different domains, and therefore metamodeling can be used to define similar preferences for interoperability in different domains. The metamodel structure consists of: (i) **FOAF ontology**, which has the personal data about the user; (ii) **Domain Ontology**, which defines the data about the domain knowledge; (iii) **Preference View Domain Ontology (PVD)** (Figure 4.2(a)), which decomposes the domain ontology and stores ontology resources in a hierarchy based on Ontology Definition Metamodel; and (iv) **Preference Meta Ontology (PMO)** (Figure 4.2(b)), which defines different preference types.

The PMO includes different types of preferences, which are listed below.

– **Boolean preferences** are modelled by means of OWL Boolean Data Type Property Construct. *Example*: Hotel Hilton Paris has sauna.

– **Constraint preferences** are modelled by means of the OWL Object Property construct. Constraint preferences show the related domain class. *Example*: Tennis Court has opening hours from 10:00 am to 22:00 pm.

– **Individual preferences** take value from domain class. *Example*: User prefers hotel Hilton Paris.

– **Interval preferences** have numerical (**Numeric preference**, e.g. User prefers hotel prices between 150-250 Euros), textual (**Textual preference**, e.g. User prefers the beer named “Aloha”) and date (**Date preference**, e.g. User prefers to swim on the weekend) values either discrete or continuous.

4.4.3 Situated Preferences for Personalised Database Applications

Holland and Kießling (Holland and Kießling 2004) present a framework for modelling situations and situated preferences. They claim that, since preferences do not always hold in general, personalised applications have to consider the current situation of users. In this context, they propose a general metamodel for situations, which can be applied as foundation for situation models of applications. The metamodel consists of five entities, listed next.

- **Situation** is the most general entity type of situation models. It can contain any attributes describing the situational context of people, agents, applications, etc.
- **Timestamp** denotes the date and time of situations. Attributes can be SQL data types like date, time, time zone, etc.
- **Location** can describe the current position. Attributes are, for example, city, zip-code, or global positioning system (GPS) coordinates.
- **Influences** describe other aspects affecting a situation.
- **Personal Influences** denote human factors of a situation like physical state or current emotion.
- **Surrounding Influences** describe outer influences like weather condition or other people currently accompanying the user.

The metamodel is generic, so it defines only the main entities and their relationships. In addition, it focuses only on modelling situations and the approach is kept **independent from the underlying preference model**. Preferences can be associated with situations, and according these associations, they can be classified in three categories: (i) **Long-term preference**: preferences that hold generally; (ii) **Singular preference**: preferences that hold in exactly one situation; and (iii) **Non-singular preference**: preferences that hold in more than one situation.

4.5 Non-parametric Representation of User Preferences

Domshlak and Joachims (Domshlak and Joachims 2007) present an approach to ordinal utility revelation from a set of qualitative preference statements. According to the authors, their approach is able to handle heterogeneous preference statements both efficiently and effectively, consisting of a computationally tractable, non-parametric transformation into a space where ordinal utility functions decompose linearly.

The main type of preference statements addressed by the approach is **dyadic** statements (*I prefer A to B*). However, it has a particularity in comparison to existing approaches: statements refer to multiple features of alternatives seen in a unified non-parametric way, e.g. "*green sport car*," which is not defined by two parameters (green and sport). Additionally, the approach also covers **monadic** statements, by qualifying alternatives as good and bad. These qualifiers are defined in terms of the "better than" expressed in dyadic statements. Finally, statements as "**A is preferred to B more than C is preferred to D**" are also covered. These two types of preferences are interpreted based on dyadic statements. For monadic statements, one can establish what is indifferent using dyadic statements, and therefore good (bad) is more (less) than indifferent. And for the third type of statement, the difference between A and B must be greater than the difference between C and D. Statements are represented as a qualitative *preference expression*, as shown next.

$$S = \{s_1, ..., s_m\} = \{\langle \varphi_1 \otimes_1 \psi_1 \rangle, ..., \langle \varphi_m \otimes_m \psi_m \rangle\}$$

consisting of a set of preference statements $s_i = \varphi_i \otimes_i \psi_i$, where φ_i, ψ_i are logical formulae over X (set of attributes), $\otimes_i \in \{\succ, \succeq, \sim\}$, and $\succ, \succeq, \sim$ have the standard semantics of strong preference, weak preference, and preferential equivalence, respectively. The authors assume that attributes X are boolean (denoting $Dom(X_i) = \{x_i, \overline{x_i}\}$), and φ_i, ψ_i are propositional logic formulae.

4.6 Comparison of Preference Representation Models

Based on the introduced preference representation approaches, we summarise and compare them in Tables 4.3 and 4.4, showing which kind of preferences and targets each of the approaches addresses. It can be seen that most of the approaches address different kinds of preferences, but they are restricted in allowing stating these preferences only over attributes values. *Exceptions* are OWLPref (Ayres and Furtado 2007) and Tapucu et al.'s approach (Tapucu et al. 2008). In addition, even though Domshlak and Joachims's work (Domshlak and Joachims 2007) also refers to attribute values, it considers it in a *non-parametric way*. Three database approaches have some preferences in their preference model (Kießling 2002, Chomicki 2003, Koutrika and Ioannidis 2006) that are not shown in these tables, because these preference constructions do not correspond to how people express preferences, but to algebraic constructions necessary for the respective approaches.

It can be seen that the presented tables are very sparse, indicating that many existing preference models are very restricted. The approach that is able to represent

Table 4.3: Comparison of Preference Representation Approaches (1).

	Soft-Constraints	**(Junker 2008)**	**CP-nets**	**TCP-nets**	**Scoring function**	**(Chomicki 2003)**
Research Area	Constraint Programming	Constraint Programming	AI	AI	Databases	Databases
Preference Targets						
Concept						
Attribute						
Enumeration Value						
Instance						
Restrictive Preferences						
Optimisation (attribute)		Yes.				
Optimisation (value)						
Constraint / Attribute value	Yes.	Yes.	Yes.	Yes.	Based on record types.	Preference formula.
• Interval					BETWEEN	
• Around						
Preference Statements						
Rating		Wish for a constraint.			Score	
• Bipolar	Yes.					
Qualifying						
Order		Yes.	Yes.	Yes.		Yes.
Don't care					Wild type (*)	
Indifferent						Yes.
Relative Preference[a]						
Unknown					No preference (⊥).	
Preference Interaction						
Attribute priority				Yes.		
Attribute indifference						
Preference priority		Yes.			Veto (♮).	Prioritised preference.
Preference indifference		Yes.				
Mutually Exclusive Preference						
Conditionality						
Condition			Yes.	Yes.		
Context						

[a]A is preferred to B more than C is preferred to D.

Table 4.4: Comparison of Preference Representation Approaches (2).

	(Kießling 2002)	**(Koutrika and Ioannidis 2006)**	**OWLPref**	**(Tapucu et al. 2008)**	**Situated Preferences**	**Non-parametric preferences**
Research Area	Databases	Databases	Semantic Web	Semantic Web	Databases	AI
Preference Targets						
Concept			ClassPreference	Class		
Attribute			AttributePreference	Property		
Enumeration Value						
Instance				Individual preferences		
Restrictive Preferences						
Optimisation (attribute)	LOWEST, HIGHEST		ExtremalPreference			
Optimisation (value)						
Constraint / Attribute value	Yes.	Atomic selection preferences.		Constraint preference		Yes.
• Interval			IntervalPreference	Interval preferences (numerical, textual, date)		
• Around	AROUND		AroundPreference			
Preference Statements						
Rating	EXP and SCORE	Atomic selection preferences and Join preferences.				Monadic statements
• Bipolar	POS, NEG, POS/NEG, POS/POS		PositivePreference and NegativePreference	Boolean preferences		
Qualifying						
Order			OrderPreference			Dyadic statements
Don't care						
Indifferent			ParetoPreference			Dyadic statements
Relative Preference[a]						Yes.
Unknown						
Preference Interaction						
Attribute priority						
Attribute indifference						
Preference priority	Prioritised preference and Numerical preference		OrderPreference			
Preference indifference	Pareto preference		ParetoPreference			
Mutually Exclusive Preference			DisjuntionPreference			
Conditionality						
Condition			ConditionalPreference	Yes.		
Context					Abstract metamodel for situations.	

[a] A is preferred to B more than C is preferred to D.

most of the preference types is OWLPref, but it still has limitations and allows only representing interval and around preferences, not having the flexibility to represent constraints. Considering preference types, note that there are two of them that are not represented in any of the approaches: (i) qualifying preferences; (ii) attribute indifference. Qualifying preferences, with the use of expressive speech acts, were widely used in our study of how humans express preferences, and this is an important limitation of existing approaches.

The comparison tables present three preference types that are not part of our proposed metamodel: (i) bipolar; (ii) relative preferences; and (iii) unknown. The first is a categorisation of constraints, but users, when expressing preferences, use expressive speech acts or rates to make this distinction. Moreover, these expressive speech acts and rates indicate how positive or negative preferences are, thus being more expressive than bipolar preferences. The second type of preferences not explicitly represented by our metamodel is a limitation in comparison with existing approaches, but they were not observed in our study. This type of preference indicates a form of resolving trade-offs between options, and our study indicates that people tend to not state this kind of preference (as none of the participants provided it), unless they are specifically asked to do so. Finally, unknown preferences are equivalent to the absence of preferences, i.e. if no preference is given with respect to a target, nothing can be inferred.

4.7 Final Remarks

In this chapter, we reviewed existing preference representation models, discussing the types of preferences they can explicitly represent, and the targets of those preferences. We showed that most of these models are very limited, and even in those that allow the representation of many preference types, there are important preferences that cannot be explicitly represented, such as qualifying preferences. Although there are few preference types that are part of existing models and cannot be represented in our metamodel, they either are not typically adopted by people or are expressed in a different way, according to our study of how humans express preferences. As our goal is to provide means for expressing preferences in a way close to natural language, our metamodel does not consider these preference types.

Representing preferences adequately is an important issue in the context of preference research, as it allows capturing and storing user preferences, and indicates the kinds of preferences that should be taken into account in preference handling approaches. It is extremely important to provide means for reasoning about those preferences, to support user decision making or even automate this process, and this is the issue that will be addressed in the next part of this thesis.