Fundamenta Informaticae XX (2012) 1–42
DOI 10.3233/FI-2012-0000
IOS Press

Interactive Exploration of Multidimensional and Hierarchical Information Spaces with Real-time Preference Elicitation

Yannis Tzitzikas and Panagiotis Papadakos
Institute of Computer Science, FORTH-ICS, GREECE, and
Computer Science Department, University of Crete, GREECE
Email: $\{$*tzitzik*$|$*papadako*$\}$*@ics.forth.gr*

Abstract. Current proposals for preference-based information access seem to ignore that users should be acquainted with the information space and the available choices for describing effectively their preferences. Furthermore users rarely formulate complex (preference or plain) queries. The interaction paradigm of *Faceted Dynamic Taxonomies (FDT)* allows users to explore an information space and to restrict their focus without having to formulate queries. Instead the users can restrict their focus (object set, or set of choices in general) gradually through a simple set of actions, each corresponding to a more refined query (formulated on-the-fly) which can be enacted by a simple click. In this paper we extend this interaction paradigm with actions that allow users to dynamically express their *preferences*. The proposed model supports progressive preference elicitation, inherited preferences and scope-based resolution of conflicts over single or multi-valued attributes with hierarchically organized values. Finally we elaborate on the algorithmic perspective and the applicability of the model over large information bases.

1. Introduction

The powerful and expressive query languages that are usually offered for structured information (e.g. for databases or for the Semantic Web) are not directly utilized by end users, because query formulation is a laborious and difficult task for them. Efforts for exploiting such languages in user friendly general-purpose models of exploration/navigation have started to come up [34, 31, 18, 10]. For instance, the interaction paradigm of *Faceted Dynamic Taxonomies (FDT)* [47] allows users to explore the information space and to restrict their focus (object set) without having to formulate queries, but through a small set of actions (zoom in/out/side) each corresponding to a query (formulated on-the-fly) which can be enacted by a simple click. In this paper we investigate how we can extend these actions in order to further ease the interaction and to speed up the restriction of the focus to those parts of the information space that the user is interested in. Specifically, we extend the interaction with user specified *preferences*.

Works on preference management over structured data (e.g. [25, 13, 4, 26]) also suffer from the same problem: the user either has to formulate complex preference queries, or the application developer has to develop specialized interfaces which internally construct such queries. Moreover, and more importantly, for formulating an effective preference specification the user should be already acquainted with the information space and the available choices. The above observations justify the need for flexible and universal (i.e. general purpose) access methods that offer exploration services and *real-time preference elicitation*. The requirements for such explorative environments include: a) *generality*, they should be capable of capturing a wide range of information spaces and user information needs, b) *expressiveness*, it should be possible for the user to interactively specify complex preference structures and c) *usability*, the users should be able to use and understand the interaction immediately, and the resulting interaction should be effective and desired by the users.

In this paper we propose an interactive preference elicitation approach, specifically we extend FDT with preference elicitation and support. In other words, FDT allows constructing queries by simple navigation/exploration actions, and our work extends the set of actions for offering preference-compliant exploration. More precisely, the key points and contributions of our work are:

- Introducing a *language for preference elicitation* during FDT exploration. Most works on preference management focus only on the order of objects, while we focus also on the order of facet/zoom-points, i.e. on the order of "queries" the user can select for changing his/her focus.

- Defining the semantics of an *incremental* preference elicitation mode which allows the user to define the desired preference structure *gradually* and *flexibly*.

- Supporting attributes with *hierarchically organized values*, which can be single or *multi-valued*, and introducing novel methods for managing them, i.e. *inherited actions* and *scope-based conflict resolution rules*.

- Elaborating on the algorithmic perspective of the proposed approach, and proposing methods that allow applying the approach over *large information spaces*.

To the best of our knowledge this is the first work that supports the above. We shall use a running example to motivate the proposed approach. Consider an international dealer of used cars and suppose that the available cars are stored in a relational table of the form: `Car(id, manufacturer, category, price, color, power, year, mileage, fuel, location, comment, accessories)`. In addition there are four *taxonomies* that have been designed in order to provide an hierarchical organization for the values of the attributes `manufacturer`, `fuel`, `location` and `category`. The leaves of these taxonomies are the domains of the corresponding attributes which are recorded in the tuples of the relational table[1]. Specifically, assume the following four taxonomies:

[1]Our model also allows tuples that contain values which are not necessarily leaves of the corresponding taxonomy.

```
MANUFACTURER            |   FUEL
  European              |       Diesel
    German              |       Ecological
        Audi            |           Hybrid
        BMW             |-----------------------------
        Porsche         |   CATEGORY
    French              |       Cabrio
        Citroen         |       Jeep
        Peugeot         |       Polymorphic
    Italian             |       Sedan
        Alfa Romeo      |-----------------------------
        Ferrari         |   LOCATION
        Fiat            |       Europe
        Lamborghini     |           Greece
  Asian                 |               Crete
    Japanese            |                   Chania
        Toyota          |                   Heraklion
        Lexus           |                   Rethymnon
    Korean              |               Cefalonia
        Kia             |           Italy
  American              |               Pisa
    U.S.A.              |           UK
        Chrysler        |               London
        Dodge           |
```

Now suppose a user who (a) likes European cars, (b) does not like Italian cars, (c) likes Ferrari, and (d) prefers low prices. According to the framework that we propose, the user can express the above preferences straightforwardly, i.e. without having to refer to particular European countries or Italian manufacturers (for expressing (a) and (b)) thanks to the *hierarchically organized values*, and *preference inheritance*). Furthermore, he does not have to express all the above in one shot. He can provide them gradually and in any order, say (b)-(a)-(d)-(c), and there is no need to define priorities for resolving the conflicts (e.g. the fact that he likes Ferrari but he does not like Italian cars). The priority will be deduced automatically by a *scope-based conflict resolution rule*. For instance, the scope of (b), i.e. Italian cars, is contained in the scope of (a), which is the set of European cars, so (b) prevails on Italian cars. Analogously (c) prevails on Ferrari cars (despite the fact that Ferrari is Italian).

Moreover the user can express more expressive statements like (e) I prefer Asian to European cars, and (f) I prefer Italian to Korean cars, and from these statements we can deduce that the user prefers Fiat to Kia, and prefers Toyota to Peugeot. The above are examples of just some of the functionalities offered by the proposed approach.

The rest of this paper is organized as follows. Section 2 provides the required background information on FDT and preference management. Section 3 defines the syntax and semantics of a preference specification language for multidimensional hierarchical information spaces. Section 4 elaborates on the algorithmic perspective of the proposed approach and introduces a number of optimizations which are crucial for the applicability of the framework. Section 5 discusses user effort, specifically it analyzes FDT convergence and provides a comparison between plain FDT and preference-based FDT systems regarding user effort. Section 6 discusses related work and finally, Section 7 concludes the paper and identifies issues that are worth further work and research. Proofs and supplementary material is given in the Appendix.

2. Background

For reasons of self-containedness Section 2.1 reviews FDT and Section 2.2 preferences.

2.1. Information Exploration through FDT

Most DB (Database) and IR (Information Retrieval) applications, as well as most WSEs (Web Search Engines), are very effective for *focalized search*, i.e. they make the assumption that users can accurately describe their information need using a query which is usually a small sequence of words. However, as several user studies have shown, a high percentage of search tasks are *exploratory* [1], the user does not know accurately his information need (e.g. in WSE users provide in average 2.4 words [20]), and focalized search very commonly leads to inadequate interactions and poor results. The available UIs in most cases do not aid the user in query formulation, and do not provide any exploration services. The returned answers are simple ranked lists of results, with no organization. For this reason users typically reformulate their initial query, inspect the top elements of the returned answer, and so on.

Modern environments should guide users in exploring the information space and in expressing their information needs in a *progressive* manner. Systems supporting FDT offer a simple, efficient and effective way for explorative tasks [47]. *Dynamic taxonomies* (faceted or not) is an interaction framework based on a multidimensional classification of (may heterogeneous) data objects allowing users to browse and explore the information space in a guided, yet unconstrained way through a simple visual interface. Features of this framework include: (a) display of current results in multiple categorization schemes (called facets - or just attributes), (b) display of categories (i.e. attribute values) leading to non-empty results only, (c) display of the count information of the indexed objects of each category (i.e. the number of results the user will get by selecting that category), and (d) the user can refine his focus gradually, i.e. it is a session-based interaction paradigm in contrast to the query-and-response dialog of current WSE which is stateless. Moreover, and as shown in [36] this interaction paradigm can act complementarily to the query-and-response dialog of the current WSE.

An example of the idea of dynamic taxonomies assuming only one facet, is shown in Figure 1. Figure 1(a) shows a taxonomy comprising 7 terms (A-G) and 8 indexed objects (1-8). Figure 1(b) shows the dynamic taxonomy if we restrict our focus to the objects $\{4,5,6\}$. Notice that it comprises only 5 terms, those that lead to objects in $\{4,5,6\}$. Figure 1(c) shows the browsing structure that could be provided at the GUI layer (e.g. at the left side bar), and Figure 1(d) sketches user interaction, based on the restriction shown in Figure 1(b). Notice the count number next to each term.

Examples of applications offering faceted exploration include: e-commerce (e.g. ebay), library and bibliographic portals (e.g. DBLP), museum portals (e.g. [19] and Europeana[2]), mobile phone browsers (e.g. [22]), specialized search engines and portals (e.g. [32]), Semantic Web (e.g. [18, 31]), general purpose WSE (e.g. `Mitos` [36]), collaborative enviroments (e.g. mSpace[49]), and booking applications (e.g. booking.com).

User Interaction. The user explores or navigates the information space by setting and changing his *focus*. The notion of focus can be *intensional* or *extensional*. Specifically, any conjunction of terms (or

[2]http://www.europeana.eu

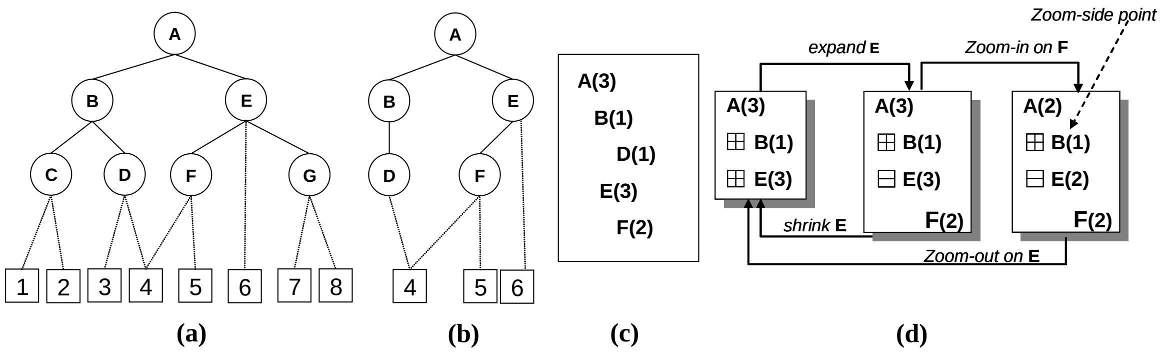


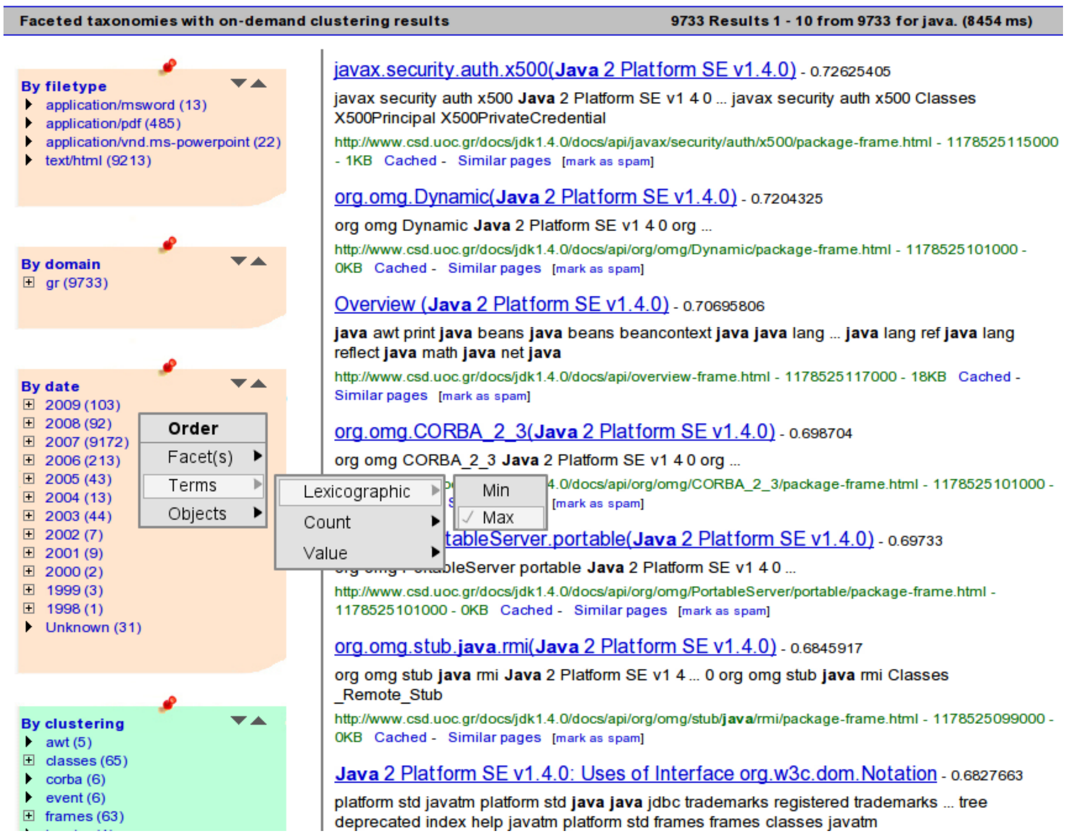
Figure 1. Dynamic Taxonomies

any boolean expression of terms in general) is a possible *focus*. For example, the initial focus can be the empty, or the top term of a facet. However, the user can also start from an arbitrary set of objects, and this is the common case in the context of a WSE. In that case we can say that the focus is defined *extensionally*. Specifically, if A is the result of a free text query q (or if A is a set of tuples returned by an SQL query q), then the interaction is based on the *restriction* of the faceted taxonomy on A (Figure 1(b) shows the restriction of a taxonomy on the objects $\{4,5,6\}$). At any point during the interaction, we compute and provide to the user the *immediate zoom-in/out/side points* along with count information (as shown in Figure 1(d)). When the user selects one of these points then the selected term is added to the focus (corresponding to a more refined query), and so on.

As an application example, Fig. 2 shows a screenshot of a WSE that supports FDT exploration specifically it shows the screen after the user submitted the query `java`. In that screenshot, 4 different facets are shown, each corresponding to one metadata attribute (at the left bar). The values (zoom-points) of two of these facets (*By date* and *By clustering*) are hierarchically organized, while the values of the rest two facets (*By filetype* and *By language*) are flat (no hierarchical organization). The results of the current focus appear at the right frame. For more on this application the reader can refer to [36], while the real-time snippet-based results clustering algorithm employed is described in [27]. The figure also sketches some GUI enhancements for supporting the preferences actions which we will introduce at Section 3. Specifically, by right clicking the facet *By date*, and selecting *Terms* from the *Order* menu, the user is able to sort the terms of this facet lexicographically, based on their count information, or based on their value, in an ascending or descending order respectively.

Notions and Notations. Table 1 defines formally and introduces notations for *terms*, *terminologies*, *taxonomies*, *faceted taxonomies*, *interpretations*, *descriptions* and *materialized faceted taxonomies*, that will be used in the sequel. The upper part of the table describes taxonomies. The middle part of the table describes *materialized faceted taxonomies*, which is actually the kind of information sources that we consider. We should note at this point that in many cases the contents and structuring of the browsable part of an information source is defined by a query, implying that its structure may be different from that of the original source. This means that one could define a materialized faceted taxonomy on top of a relational database, or IR system, or a semantic network-based (e.g. RDF/S) knowledge base.

In brief, *Obj* is a set of objects (e.g. the set of all documents indexed by a WSE), each described with respect to one or more aspects (facets), where the description of an object with respect to one



9733 Results 1 - 10 from 9733 for java. (8454 ms)

javax.security.auth.x500(**Java** 2 Platform SE v1.4.0) - 0.72625405
javax security auth x500 **Java** 2 Platform SE v1 4 0 ... javax security auth x500 Classes X500Principal X500PrivateCredential
http://www.csd.uoc.gr/docs/jdk1.4.0/docs/api/javax/security/auth/x500/package-frame.html - 1178525115000 - 1KB Cached - Similar pages [mark as spam]

org.omg.Dynamic(**Java** 2 Platform SE v1.4.0) - 0.7204325
org omg Dynamic **Java** 2 Platform SE v1 4 0 org ...
http://www.csd.uoc.gr/docs/jdk1.4.0/docs/api/org/omg/Dynamic/package-frame.html - 1178525101000 - 0KB Cached - Similar pages [mark as spam]

Overview (**Java** 2 Platform SE v1.4.0) - 0.70695806
java awt print **java** beans **java** beans beancontext **java java** lang ... **java** lang ref **java** lang reflect **java** math **java** net **java**
http://www.csd.uoc.gr/docs/jdk1.4.0/docs/api/overview-frame.html - 1178525117000 - 18KB Cached - Similar pages [mark as spam]

org.omg.CORBA_2_3(**Java** 2 Platform SE v1.4.0) - 0.698704
org omg CORBA_2_3 **Java** 2 Platform SE v1 4 0 org ...
4.0/docs/api/org/omg/CORBA_2_3/package-frame.html - 1178525101000 -
[mark as spam]

tableServer.portable(**Java** 2 Platform SE v1.4.0) - 0.69733
bleServer portable **Java** 2 Platform SE v1 4 0 ...
http://www.csd.uoc.gr/docs/jdk1.4.0/docs/api/org/omg/PortableServer/portable/package-frame.html - 1178525101000 - 0KB Cached - Similar pages [mark as spam]

org.omg.stub.**java**.rmi(**Java** 2 Platform SE v1.4.0) - 0.6845917
org omg stub **java** rmi **Java** 2 Platform SE v1 4 ... 0 org omg stub **java** rmi Classes _Remote_Stub
http://www.csd.uoc.gr/docs/jdk1.4.0/docs/api/org/omg/stub/**java**/rmi/package-frame.html - 1178525099000 - 0KB Cached - Similar pages [mark as spam]

Java 2 Platform SE v1.4.0: Uses of Interface org.w3c.dom.Notation - 0.6827663
platform std javatm platform std **java java** jdbc trademarks registered trademarks ... tree deprecated index help javatm platform std frames frames classes javatm

Figure 2. FTD-based GUI of the `Mitos` WSE

facet consists of assigning to the object one or more terms from the taxonomy that corresponds to that facet. I is the interpretation function, while $\bar{I}$ takes into account the semantics. For example, and assuming the example of Figure 1(a), we have $I(C) = \{1, 2\}$, $I(B) = \emptyset$, while $\bar{I}(C) = \{1, 2\}$ and $\bar{I}(B) = \{1, 2, 3, 4\}$.

The lower part of the table describes the FDT-interaction. and is given in detail for reasons of self-containedness. For example, Figure 1(b) depicts the *restriction* over the set $A = \{4, 5, 6\}$, and the *reduced terminology* T_A is the set of shown terms.

Regarding scalability we should mention that FDT can provide real-time exploration services for millions of objects using techniques like those proposed in [56, 45, 7]. Thorough experimental results over `FleXplorer` (that we have developed) are given in [54]. As expected, the computation of zoom-in points with count information is more expensive than without: in 1 sec we can compute the zoom-in points of around 240.000 results (i.e. $|A| = 240.000$) with count information, while without count information we can compute the zoom-in points of around 540.000 results.

TAXONOMY		
Name	**Notation**	**Definition**
terminology	T	a set of *terms* (can capture categorical/numeric values)
subsumption	$\leq$	a partial order (reflexive, transitive and antisymmetric)
taxonomy	$(T, \leq)$	T is a terminology, $\leq$ a subsumption relation over T
broaders of t	$B^+(t)$	$\{ t' \mid t < t' \}$
narrowers of t	$N^+(t)$	$\{ t' \mid t' < t \}$
direct broaders of t	$B(t)$	$minimal_<(B^+(t))$
direct narr. of t	$N(t)$	$maximal_<(N^+(t))$
Top element	$\top_i$	$\top_i = maximal_\leq(T_i)$
MATERIALIZED FACETED TAXONOMIES		
faceted taxonomy	$\mathcal{F} = \{F_1, ..., F_k\}$	$F_i = (T_i, \leq_i)$, for $i = 1, ..., k$ and all T_i are disjoint
object domain	Obj	any denumerable set of objects
interpretation of T	I	any function $I : T \rightarrow 2^{Obj}$
materialized faceted taxonomy	$(\mathcal{F}, I)$	$\mathcal{F}$ is a faceted taxonomy $\{F_1, ..., F_k\}$ and I is an interpretation of $T = \bigcup_{i=1,k} T_i$
ordering over interpretations	$I \sqsubseteq I'$	$I(t) \subseteq I'(t)$ for each $t \in T$
model of $(T, \leq)$ induced by I	$\bar{I}$	$\bar{I}(t) = \cup\{I(t') \mid t' \leq t\}$
Descr. of *o* wrt I	$D_I(o)$	$D_I(o) = \{ t \in T \mid o \in I(t)\}$
Descr. of *o* wrt $\bar{I}$	$D_{\bar{I}}(o) \equiv \bar{D}_I(o)$	$\{ t \in T \mid o \in \bar{I}(t)\} = \cup_{t \in D_I(o)}(\{t\} \cup B^+(t))$
FDT-INTERACTION: BASIC NOTIONS AND NOTATIONS		
focus	ctx	any subset of T such that $ctx = minimal(ctx)$
projection on F_i	ctx_i	$ctx \cap T_i$
Kinds of zoom points w.r.t. a facet i while being at ctx		
zoom points	$AZ_i(ctx)$	$\{ t \in T_i \mid \bar{I}(ctx) \cap \bar{I}(t) \neq \emptyset\}$
zoom-in points	$Z_i^+(ctx)$	$AZ_i(ctx) \cap N^+(ctx_i)$
immediate zoom-in points	$Z_i(ctx)$	$maximal(Z_i^+(ctx)) = AZ_i(ctx) \cap N(ctx_i)$
zoom-side points	$ZR_i^+(ctx)$	$AZ_i(ctx) \setminus \{ctx_i \cup N^+(ctx_i) \cup B^+(ctx_i)\}$
immed. zoom-side points	$ZR_i(ctx)$	$maximal(ZR^+(ctx))$
Restriction over an object set $A \subseteq Obj$		
reduced interpretation	I_A	$I_A(t) = I(t) \cap A$
reduced terminology	T_A	$\{ t \in T \mid \bar{I}_A(t) \neq \emptyset\} =$ $\{ t \in T \mid \bar{I}(t) \cap A \neq \emptyset\} = \cup_{o \in A} B^+(D_I(o))$

Table 1. Basic notions and notations

2.2. Preference Management

Commonly, preferences are not hard constraints, but wishes, simple or complicated ones (covering one or more aspects), which might or might not be satisfied. Such wishes might be independent, or might affect each other, even in conflicting ways.

A survey of major questions and approaches for preference handling in applications such as recommender systems, personal assistant agents and personalized user interfaces is given at [39], while [40] proposes guidelines and reports examples for product search and recommender systems. Below we show and briefly discuss some distinctions of preference management approaches from various perspectives.

```
Personalization of            |    Nature of Preference
    Content                   |       Qualitative
    Visualization             |       Quantitative
    Available Services        |
    Interaction               |
------------------------------------------------------------
Certainty of Preference       |   Source of Preferences
    Crisp                     |       Explicit User Input
    Fuzzy                     |       Implicit User Input
                              |       Collaborative
------------------------------------------------------------
Information Space             |
    Relational DB             |
    DB with hierarchical values |
    Unstructured data (text)    |
```

Subject of Personalization. In general, a user can express preferences regarding the informational contents of an application, the visualization of the contents, the services that the user has access to at any time, or the interaction between the user and the application.
Nature of Preferences. Preferences can be defined either using a *qualitative* approach [25, 13, 17] or a *quantitative* approach [3, 5, 30]. According to the former, preferences are described directly, using a preference relation $\succ_{Pref}$ (i.e. $x \succ_{Pref} y$), while according to the latter, preferences are described indirectly by defining scoring functions (i.e. $Score(x) > Score(y)$). The qualitative approach is more powerful and expressive than the quantitative approach, since not every preference can be modeled using scoring functions, according to [13, 15]. There are also approaches that support a mixture of qualitative and quantitative preferences [43]. This can be done by putting together a CP-net[3] and a set of constraints.
Certainty of Preferences. The above approaches can be further specialized depending on whether the expressed preferences are crisp or fuzzy (e.g. [6] reviews the literature on fuzzy preferences).
Sources of Preference. Preferences can be specified *explicitly* by the users (either through a query language that supports preferences, or through the mediation of an application that produces such queries), or *implicitly*, by tracking silently user actions and monitoring user behaviour. The latter category includes works like [16, 24]. The *Collaborative* category includes works like *collaborative filtering systems* [48, 41], based on the assumption that similar people like similar things. Machine learning has also been applied for learning preference value functions, e.g. [14] presents a method for learning preferences over sets of items, by taking as input a collection of positive examples.
Subject Information Space. Finally, another criterion is the structure of the underlying information space (texts, relational spaces, multi-dimensional spaces with hierarchically organized attribute domains, etc).

Our focus. With respect to the characteristics described earlier, our work focuses on multi-dimensional spaces with hierarchically organized attribute domains, and explicitly-specified and crisp qualitative user preferences. We also focus on the preference elicitation process. According to Wikipedia, *preference elicitation* refers to the problem of developing a decision support system capable of generating recom-

[3]A CP-net is a directed graph G over attributes V, whose nodes are annotated with conditional preference tables for each attribute [9].

mendations to a user, thus assisting him in decision making. It is important for such a system to model user's preferences accurately, find hidden preferences and avoid redundancy. A survey of preference elicitation methods is given in [11]. Most of these methods focus on the quantitative approach, i.e. on the elicitation of multi-criteria value functions. In this paper we use the term *real-time preference elicitation* because according to our approach: (a) the system requires from the user to express his preferences *only* for those facets/values that are involved in the available (and restricted) set of choices (i.e. not for the entire value space), and (b) we exploit the hierarchical organization of terms for reducing the number of preferences that have to be explicitly specified.

3. Extending FDT with Preference Actions (Syntax, Semantics and Algorithms)

Here we extend the interaction of FDT with user actions for preference specification/elicitation. Let us first motivate the benefits of FDT for *decision making* over our running example.

Example 3.1. Assume that somebody, call him James, wants to change his car. He is interested in a family car, although he preferred sport cars when he was younger. His wife prefers Jeeps but he is reluctant due to the extra parking space required and because the garage of his home is somehow small. He believes that Japanese and German cars are more reliable than French or Korean cars. He likes the fact that Hybrid cars consume less, are more ecological and that the annual taxes are lower for such cars. James lives at the city of Heraklion, so cars owned by persons that do not live in the island of Crete (where Heraklion resides) are less preferred for him (due to the traveling time and cost) unless the case is exceptional. In addition he cannot afford an expensive car. Ideally he would like a Porsche with four doors (e.g. Porsche Panamera) and enough space for luggage, hybrid with consumption less than 6lt/100Km, bigger than Panamera (to satisfy his wife) but smaller than Cayenne, with less than 10 thousands kilometers, in sale by his favorite neighbor and at a very good price (e.g. less than 30K Euros), but this is a utopian desire. James aims at buying one car, but it is probable that he would buy a "Porsche Carrera 4s" if available at a very good price, and another decent but inexpensive family car to satisfy the rest requirements.

Although lengthy, the above description is by no means complete. There are a lot of other aspects that would determine James' final decision (accessories, grip, airbugs, Euro NCAP stars, color, etc). What we want to stress with this example is that the specification of preferences is a laborious, cumbersome and time consuming task, and that the resulting descriptions are in most cases incomplete. Pragmatically, decision making is based on complex trade-offs that involve several (certain or uncertain) attributes as well as user's attitude towards risk [23]. Moreover preferences are not stable over time.

We believe that it is beneficial to provide users with an interaction method in which the preference specification cost is paid *gradually* and *depends on the available choices*. For example, why spending time for expressing complex tradeoffs between Porsche models with 4 doors versus those with 2 doors if no Porsche car is in sale. Therefore an effective interaction that shows users the available choices is important for reducing the preference specification cost and for speeding up decision making.

In brief, the preference specification actions that we propose affect the presentation order of:

- **facets**, i.e. the order by which facets (i.e. criteria) appear,

- **terms**, i.e. the order of the zoom-in/side points (i.e. criteria values) appear, and
- **objects** (of the focus), i.e. the order by which the objects (i.e. choices) appear.

3.1. Syntax

Here we introduce a language consisting of statements that we call *preference actions*. Each action has a scopeType (either `facets order`, `terms order`, or `object order`) determining which kind of elements it affects (facets, terms, or objects). Each action is "anchored" to one element which can be a facet or a term, and this allows enacting the preference actions through the GUI straightforwardly[4]. Each action is associated to a rank description ($rankSpec$) which can be *lexicographic* (for ordering strings), *count* (for ordering elements based on the number of objects that are classified to them), and *value* (for ordering numerically-valued facets). The language also defines actions for defining *best/worst* (i.e. preferred/non-preferred) elements, and *relative* preferences.

Syntactically, preference actions are defined through the following grammar (in BNF):

$$
\begin{array}{rcll}
\langle stmt \rangle & ::= & \langle scopeType \rangle \langle spec \rangle & \\
\langle scopeType \rangle & ::= & \texttt{facets order :} \mid \texttt{terms order :} \mid \texttt{object order :} & \\
\langle spec \rangle & ::= & \langle anchor \rangle \langle rankSpec \rangle & \\
\langle anchor \rangle & ::= & \texttt{facet } \langle F_i \rangle & \\
& \mid & \texttt{term } \langle t_j \rangle & \\
& \mid & \epsilon & // \text{ the empty string} \\
\langle rankSpec \rangle & ::= & \{\texttt{l}exicographic \mid count \mid value\} \ \{\texttt{min}|\texttt{max}\} & \\
& \mid & \texttt{best} \mid \texttt{worst} & \\
& \mid & \texttt{use scoreFunction } \langle score() \rangle \ \{\texttt{min}|\texttt{max}\} &
\end{array}
$$

In the above grammar F_i and t_j denote names that match a facet or term respectively, while $score$ is the name of a real-valued function provided by the user or the application programmer. Some examples follow:

```
(b1) facets order:  count max
(b2) facets order:  facet Manufacturer best
(b3) terms order:  facet Year value max
(b4) object order:  term Location.Cefalonia best
(b5) object order:  facet Relevance value max
(b6) object order:  use scoreFunction Relevance*dist(price,20K) max
```

Before explaining formally their semantics, let us first describe them informally. The 1st action specifies the order of facets to be in decreasing order with respect to their count information (i.e. max counts are preferred). The 2nd places the facet *Manufacturer* at the top of the facets list. The 3rd specifies that the order of the terms of the facet *Year* to be in decreasing order. The 4th places all objects classified (directly or indirectly) under the term *Cefalonia* at the top of the object ordering.

Now suppose a user who starts the car seeking process by formulating a free text query which the WSE evaluates over the attribute `comment` of the database. In this case the user would like to see the

[4]If the user right-clicks on an element e, a pop-up window will show and allow the user to select the desired preference action. The selected action will be anchored to e.

objects in decreasing order with respect to their relevance. The 5th action captures this requirement where the facet called *Relevance* corresponds to the score returned by the WSE. Finally, the 6th action orders the objects based on a function over the relevance score and distance from a given price.

We can extend the syntax so that to support *interval-anchored* actions, *distance* functions, and *named actions* (that eases the formulation of more complex preferences):

$$\begin{array}{rcl} \langle anchor \rangle & | & \texttt{term interval } [\,\langle t_i \rangle \langle t_j \rangle] \\ \langle rankSpec \rangle & | & \texttt{use distFunction } \langle dist() \rangle \ \{\texttt{min}|\texttt{max}\} \\ \langle namedStmt \rangle & ::= & \texttt{NamedAction } \langle String \rangle : \langle stmt \rangle \end{array}$$

Furthermore, we can extend the syntax to support *relative preferences* over facets and terms.

$$\begin{array}{rcl} \langle stmt \rangle & | & \texttt{facets order}: \ \texttt{prefer} \ \langle F_i \rangle \ \texttt{to} \ \langle F_j \rangle \\ \langle stmt \rangle & | & \texttt{terms order}: \ \texttt{prefer} \ \langle t_i \rangle \ \texttt{to} \ \langle t_j \rangle \\ \langle stmt \rangle & | & \texttt{object order}: \ \texttt{prefer} \ \langle t_i \rangle \ \texttt{to} \ \langle t_j \rangle \end{array}$$

Regarding object ranking, we can extend the syntax to allow *skyline* [38] operators[5]:

$$\langle stmt \rangle \quad | \quad \texttt{object order}: \ \texttt{skylineOf} \ \langle rankSpecList \rangle$$

Although we could easily extend the syntax so that to allow complex expressions over plain or named actions that synthesize two or more actions with complex preference constructors (Pareto, prioritized synthesis, etc), such actions are more difficult for the user to specify interactively, and thus not our primary concern. Instead, in Section 3.4 (and in the application example described at appendix B) we sketch a session-based automatic prioritization, which syntactically would correspond to actions of the form:

$$\langle stmt \rangle \quad | \quad \texttt{object order}: \ \texttt{Priorities}: \langle F_{p1} \rangle \ \ldots \ \langle F_{pm} \rangle$$

where F_{p*} are facet names.

3.2. The Domain of Semantics

The actions specified by the syntax allow structuring (ordering) the materialized faceted taxonomy according to the preferences. Independent of how many actions have been issued and what their semantics are, the defined preference at each point in time, comprises $k+2$ preference relations. Specifically:

- One over the facets: $(\{F_1, \ldots, F_k\}, \succ_F)$,
- k preferences relations, one for each facet F_i (of the form $(T_i, \succ_i)$), and
- one preference relation for the objects $(A, \succ_{Obj})$.

Let $\mathcal{B}$ be the set of user actions the user has issued. We can partition this set to $k+2$ subsets (where k is the number of facets) as follows: $\mathcal{B}_F$ holds the user actions for facets, $\mathcal{B}_{T_i}$ holds the user actions for the terms of each facet F_i and $\mathcal{B}_{Obj}$ holds the user actions regarding the objects' preferences. So

[5] In brief, the skylines are the maximal (w.r.t. preference) elements, i.e. those which are not dominated by others. This set is also called efficient set, or Pareto optimal set.

$\mathcal{B} = \mathcal{B}_F \cup (\bigcup_i \mathcal{B}_{T_i}) \cup \mathcal{B}_{Obj}$. As each of these sets can contain more than one action, we have to specify how the corresponding preference relation is defined, e.g. from the actions in $\mathcal{B}_{T_i}$ to define the preference relation $(T_i, \succ_i)$.

Let us now introduce some required notions about preference relations. Consider a set $E = \{Porsche, Ferrari, Fiat\}$ and a preference relation $R_\succ$ over E consisting of one relationship, specifically $R_\succ = \{Porsche \succ Ferrari\}$. We shall use $dom(R_\succ)$ to denote the elements of E that participate in $R_\succ$, here $dom(R_\succ) = \{Porsche, Ferrari\}$, and call *inactive* the elements of E which are not members of $dom(R_\succ)$, in our case $Fiat$. Given a preference relation $\succ$, with $\prec$ we will denote its *dual* order. Commonly, preference relations are illustrated using Hasse diagrams. In our case $(E, R_\succ)$ can be illustrated as:

```
Porsche
   |
   |          Fiat
Ferrari
```

Def. 1. (Valid Preference)
We consider a preference relation R over a set of elements E to be valid*, if it is acyclic.*

Def. 2. *We say that a linear or bucket order*[6] *L over E* respects *a preference relation R, if $R \subseteq L$.*

3.3. Syntax to Semantics

To describe formally the semantics of the syntax we have to define what the various keywords of the syntax, like `count`, `best`, `worst`, mean precisely.

At first note that the semantics of `lexicographic`, `count`, `value`, and `use ScoreFunction` are straightforward, and each defines a linear or bucket order. Note however that `count` is not applicable to objects. For a term t, $t.count$ is the number of objects in A, indexed by term t, or a narrower term of t. For a facet F_i, $F_i.count$ is the number of the elements in A which are indexed by terms of F_i. For example, consider the example of Figure 1(a) where we have only one facet, say F_1. At that point we have $F_1.count = 8$ while for the term B we have $B.count = 4$. In the restriction on the set $A = \{4, 5, 6\}$ that is shown in Figure 1(b), we have $F_1.count = 3$ and $B.count = 1$. Formally, and using the notations of Table 1, we have $t.count = |\bar{I}(t) \cap A|$ and $F_i.count = |J(F_i) \cap A|$ where $J(F_i) = \{o \in Obj \mid D(o) \cap T_i \neq \emptyset\}$.

3.3.1. Flat Single-Valued Attributes

We will now define the semantics of actions that express *qualitative preferences*, i.e. actions of the form `best`(e_i), `worst`(e_i), and `prefer` e_i `to` e_j, starting from the case of single-valued and flat attributes.

Let B (resp. W) be the elements of E on which a `best` (resp. `worst`) action has been defined. Let $R_\succ$ be the relative preferences (of the form $e_i \succ e_j$) over E provided by the user. We shall now introduce an algorithm, Alg. `Apply`, which takes as input these sets and derives one *linear* or *bucket* order. The algorithm also takes a parameter $Policy$ which determines the ordering of the *inactive elements* (will be explained later on).

[6]A bucket order is linear order of blocks where each block is a set, and all blocks are pairwise disjoint.

Algorithm 1 Apply(E, $B, W, R_{\succ}, Policy$)
Input: the set of elements E, the set of best elements B, the set of worst elements W, a set of relative relationships $R_{\succ}$, and $Policy$ for *inactive* elements
Output: a bucket order over E that respects R

```
1: $R_{bw} \leftarrow \{(b, w) \mid b \in B, w \in W\}$ // each best is preferred than each worst
2: $R \leftarrow R_{bw} \cup R_{\succ}$ //add relative prefs
3: $L \leftarrow$ SourceRemoval($R$) //produce blocks with boundaries
4: $I \leftarrow E \setminus (B \cup W \cup dom(R_{\succ}))$ // $I$ contains the inactive elements
5: $L' \leftarrow$ addInactiveElements($L, I, Policy$)
6: return $L'$
```

Algorithm 2 SourceRemoval(R)
Input: a binary relation R over E
Output: a bucket order over E that respects R

```
1: $L \leftarrow \langle\rangle$
2: repeat
3:     $S \leftarrow maximal_{\succ}(R)$
4:     $R \leftarrow R \setminus \{(x \succ y) \in R \mid x \in S\}$ // Remove maximal
5:     $L \leftarrow L.append(S)$ // Append a bucket to $L$
6: until $S \neq \varnothing$
7: return $L$
```

At first the algorithm constructs a graph by connecting each best to each worst element (so best/worst are interpreted as "each best is preferred to each worst"). Then it adds to the graph the relationships in $R_{\succ}$. We should note here that the parameters B and W actually define a set of relationships (R_{bw} at line 3 of the algorithm), so they could have been expressed directly through the $R_{\succ}$ parameter, however we keep them separate as they constitute an easily enacted (for the user) shorthand. Although a linear or bucket order could be produced by traversing the graph in a breadth first search (BFS) manner (where the first block will contain the more preferred elements, the second the next more preferred, etc), if the transitive reduction is a DAG (Directed Acyclic Graph, i.e. not a tree), then BFS could yield wrong results. Using *topological sorting*[7] instead of BFS, e.g. Alg. `SourceRemoval` as shown above, we can always get a linear order that *respects* R. In particular, Alg. `SourceRemoval` is based on the *source removal algorithm* described in [21], satisfying the condition that all removed maximal nodes are inserted in the same bucket. It begins by finding all the maximal elements of R, moves them into a bucket, and continues with the maximal elements of their children, and so on.

To give an example, let $B = \{Ferrari\}$, $W = \{Fiat, Lancia\}$ and $R_{\succ} = \{Porsche \succ Ferrari, Porsche \succ Fiat\}$. At the left of the figure that follows we can see the diagram of R, and at the right the result of topological sorting (as derived by step 5 of `Apply`), i.e. $L = \langle Porsche, Ferrari, \{Fiat, Lancia\}\rangle$, meaning that the bucket order consists of three blocks (the first two are singletons).

[7]Topological sorting yields a linear ordering of the nodes of a DAG such that each node comes before all nodes to which it has outbound edges.

```
   Porsche                                Porsche          high
     /  \            Topological             |             |
    / Ferrari           Sort              Ferrari          | preference
   /  /    \            ==>                  |             |
  Fiat    Lancia                        {Fiat, Lancia       low
```

Regarding *inactive elements* (elements not participating in any action), they can be considered as maximal or minimal elements according to the application needs (controlled by parameter $Policy$ of Alg. `Apply`). For example consider a facet F_i with values from a set T_i, and a number of actions that define the sets $B_i, W_i, R_{\succ_i}$. By using $E = T_i$ and calling Alg. `Apply`, in line 7, we compute $I = E \setminus (B_i \cup W_i \cup dom(R_{\succ_i}))$ (where $dom(R_{\succ_i})$ is the elements of E that participate in $R_{\succ_i}$). Now by using the command `addInactiveElements` (line 8 of Alg. `Apply`) , and passing as parameters the bucket order L', the inactive elements I and the policy based on the application needs, which can be maximal (resp. minimal), we put the inactive elements at the beginning (resp. end) of L' as a new block.

3.3.2. Set-Valued Attributes

Multi-valued attributes appear in several cases (social tags, clustering, etc). In our running example suppose that the attribute `accessories` of the table `Car` is multi-valued, taking values like ABS, ESP (Electronic Stability Program), AT (Auto-Transmission), DVD, etc.

Def. 3. (Induced Preference over Sets: *MoreWins-Rule*)
If s, s' are two subsets of E, with $wins(s, s')$ we will denote the number of "times" s beats s' according to $\succ$. Formally:

$$wins(s, s') = |\{(e, e') \mid e \in s, e' \in s', e \succ e'\}|$$

Any subset S of the powerset of E (i.e. $S \subseteq P(E)$), can be ordered according to a preference relation that we will be denoted by $\succ_{\{\}}$, defined by the following rule:

$$s \succ_{\{\}} s' \text{ iff } wins(s, s') > wins(s', s) \quad \Box$$

As an example consider a set $T = \{ABS, ESP, AT, DVD\}$ and two statements which define ABS as *best* and ESP as *worst*. Now consider the following family of sets: $S = \{\{ABS\}, \{ESP\}, \{ABS, ESP\}, \{AT, ABS\}, \{AT, ESP\}, \{DVD, ESP\}\}$. The $win(s, s')/win(s', s)$ values of all pairs of sets from the above family are shown in the next table (the last column shows the number of clear winnings - not ties).

$w(s, s')/w(s', s)$	{ABS}	{ESP}	{ABS,ESP}	{AT, ABS}	{AT, ESP}	{DVD, ESP}	all
{ABS}	0/0	1/0	1/0	1/0	2/0	2/0	5/0
{ESP}	0/1	0/0	0/1	0/2	0/1	0/1	0/5
{ABS,ESP}	0/1	1/0	1/1	1/2	2/1	2/1	3/2
{AT,ABS}	0/1	2/0	2/1	1/1	3/0	3/0	4/1
{AT,ESP}	0/2	1/0	1/2	0/3	1/1	1/1	1/3
{DVD,ESP}	0/2	1/0	1/2	0/3	1/1	1/1	1/3

By using Def. 3 (i.e. $\succ_{\{\}}$) and then applying topological sorting we get the following bucket order $\langle \{ABS\}, \{AT, ABS\}, \{ABS, ESP\}, \{\{AT, ESP\}, \{DVD, ESP\}\}, \{ESP\}\rangle$.

Now suppose that both ABS and ESP are defined as best elements, and that both AT and DVD are defined as worst. In that case it holds:

$$\begin{aligned} wins(\{ABS\}, \{ABS, ESP\}) &= wins(\{ABS, ESP\}, \{ABS\}) = 0 \\ wins(\{AT\}, \{AT, DVD\}) &= wins(\{AT, DVD\}, \{AT\}) = 0 \end{aligned}$$

This means that with $wins$ we get 0 whenever sets with best only elements are compared, and sets with worst only elements are compared. If we would like to break such ties we could adopt a *MoreGoodLessBad*-rule (the more best elements the better and the less worst elements the better). To define it formally, we first have to introduce some notations. Given an element e we use $sup(e)$ to denote the number of elements that e dominates, minus 1. Formally, $sup(e) = |\{e' \in E \mid e \succ e'\}| - 1$. Notice that each worst element takes a negative value. Given a set of values e we define the support of s, denoted by $Support(s)$, by summing up the support of its terms, i.e. $Support(s) = \sum_{e \in s} sup(e)$. Note that since a worst value takes -1 we can discriminate an s having one worst term from one s' having 10 worst terms ($Support(s) = -1$, while $Support(s') = -10$). We can now proceed and define:

Def. 4. (Breaking ties: *MoreGoodLessBad-rule*)
If $wins(s, s') = wins(s', s) = 0$ *and* $Support(s) > Support(s')$ *then* $s \succ_{\{\}} s'$. □

In our example it holds: $Support(\{ABS, ESP\}) = 2 > Support(\{ABS\}) = 1 > Support(\{AT\}) = -1 > Support(\{AT, DVD\}) = -2$, and the induced ordering, i.e. $\langle$ $\{ABS, ESP\}$, $\{ABS\}$, $\{AT\}$, $\{AT, DVD\}$ $\rangle$, satisfies the *MoreGoodLessBad*-rule.

To conclude, in case we have preferences over atomic values but the information space has set-valued attributes, then it is enough to use Alg. `Apply` with a small modification. Initially, we follow the first two steps of Alg. `Apply`, in order to compute the relation $\succ$ of the atomic values We should stress at this point that in order to compute correctly $wins$ we have to take into account the *transitive closure* of the preference relation, e.g. if $a \succ b$ and $b \succ c$ and we want to compute $wins(\{a, e\}, \{c, e\})$ we should consider that $a \succ c$ during this computation. In other words, we should anticipate the topological sorting of `Apply` over individual values before computing $wins$ over sets. Then we compute the $wins$ (and the $Support$ to break ties), to define $\succ_{\{\}}$. Afterwards we continue with the next steps of Alg. `Apply`, i.e. topological sorting and so on, eventually yielding the final bucket order of the sets.

The steps are given in more detail at Alg. 3.

3.3.3. Best/Worst Preferences over Hierarchically Organized Values

So far we have considered single-valued and set-valued attributes over flat (non hierarchically organized) value domains. Let us now consider hierarchically organized values. As an example if the user is interested in "Italian" cars and marks them as "best" then it is reasonable to apply "best" also to its narrower terms, i.e. to Ferrari, Fiat, etc. It is not hard to see that the approach described in the previous subsection is not adequate for terms which are not leaves. For example suppose the following set of actions (using an informal syntax): $\mathcal{B} = \{Best(European), Worst(Italian), Best(Ferrari)\}$, which define the sets $B = \{European, Ferrari\}$, $W = \{Italian\}$. If we apply Alg. `Apply` without taking into account

Algorithm 3 `ApplyOverFamiliesOfSets`$(E, B, W, R_{\succ}, Policy)$
Input: the set of elements E **(here each element of E is a set)**, the set of best elements B, the set of worst elements W, a set of relative relationships $R_{\succ}$, and $Policy$ for *inactive* elements
Output: a bucket order over E

```
1: $R_{bw} \leftarrow \{(b, w) \mid b \in B, w \in W\}$
2: $R \leftarrow R_{bw} \cup R_{\succ}$
3: $R \leftarrow Closure_{transitivity}(R)$ // Addition of the transitively induced links
4: for each $e, e' \in E$, s.t. $e \neq e'$ do
5:     if $wins(e, e') > wins(e', e)$ then
6:         set $e \succ_{\{\}} e'$
7:     else if $wins(e, e') < wins(e', e)$ then
8:         set $e' \succ_{\{\}} e$
9:     else if $wins(e, e') = wins(e', e)$ then
10:        resolve the tie by computing the $support(e)$ and $support(e')$
11: $L \leftarrow$ SourceRemoval$(\succ_{\{\}})$
12: $I \leftarrow E \setminus dom(\succ_{\{\}})$ // $I$ is the set of inactive elements
13: $L' \leftarrow$ addInactiveElements$(L, I, Policy)$
14: return $L'$
```

the taxonomy we would get the preference relation:

```
Ferrari   European
    \       /
     Italian
```

which does not make much sense, nor helps us to derive the intended ordering of cars.

It follows that without proper exploitation of the subsumption relation, the user would have to issue a high number of actions, all anchored to leaf terms. To tackle this problem, below we introduce a form of *preference inheritance* where preferences are "inherited" to the narrower terms. Let b be an action in $\mathcal{B}$. We shall use $scope(b)$ to denote the *scope* of the action b, which is the set of elements (either facets, terms, or objects) that are affected by this action. To capture inheritance we will redefine the scope of actions which are anchored to terms of a taxonomy.

Def. 5. (Scope and Inheritance)
Let b be an action $b = \langle e, rs\rangle$ where e is its anchor and rs the other part of the action. The scope of b is defined as $scope(b) = \cup_{e' \in N^*(e)} scope(\langle e', rs\rangle)$ where $N^*(e)$ stands for e and the narrower elements of e, formally $N^*(e) = \{e\} \cup N^+(e) = \{e' \mid e' \leq e\}$. $\Box$

In other words, the scope of b is the union of the scopes of the actions obtained by replacing the anchor e with a narrower term of e. Table 2 defines exactly the scope for each action.

The scopes of the actions of our example according to Def. 5, are shown in the first two columns of Table 3.

Note that the set of action $\mathcal{B} = \{b1, b2, b3\}$ defines a *valid* preference, i.e. no cycles are formed (recall Def. 1). However, if we "unfold" each $b \in \mathcal{B}$, based on its scope, then we will get a $\mathcal{B}'$ that does not define a valid preference, e.g. Ferrari will be both best and worst and this forms a cycle. To tackle

Table 2. Scopes (direct and under inheritance)

scopeType	anchor	(D)irect scope	(I)nherited scope
	facet F_i	T_i	T_i
terms order	term t_j	$\{t_j\}$	$N^*(t_j)$
objects order	term t_j	$I(t_j)$	$\bar{I}(t_j)$

Table 3. Scopes: Example

action	scope	active scope
b1: Best(Europe)	{European, German, Audi, BMW, Porsche, French, Citroen, Peugeot, Italian, Alfa Romeo, Ferrari, Fiat, Lamborghini }	$\mathit{scope}(b1) \setminus \mathit{scope}(b2)$
b2: Worst(Italian)	{Italian, AlfaRomeo, Ferrari, Fiat, Lamborgini}	$\mathit{scope}(b2) \setminus \mathit{scope}(b3)$
b3: Best(Ferrari)	{Ferrari}	scope(b3)

this problem, and to provide an intuitive interpretation of user's actions, we will introduce what we call *active scope*, after first introducing some required definitions.

Def. 6. *We say that an action b is* equally or more refined *than an action b', denoted by $b \sqsubseteq b'$, if $\mathit{scope}(b) \subseteq \mathit{scope}(b')$.* □

In this way a preorder (reflexive and transitive) relation over $\mathcal{B}$, denoted by $(\mathcal{B}, \sqsubseteq)$ is defined. In our case the Hasse diagram of $\sqsubseteq$ is:

```
b1: Best(European)
       |
b2: Worst(Italian)
       |
b3: Best(Ferrari)
```

The objective is to use $(\mathcal{B}, \sqsubseteq)$ for resolving the conflicts incurred due to inheritance. Specifically, we introduce the following rule:

Def. 7. (Scope-based Dominance Rule)
If $A \subseteq \mathit{scope}(b) \subseteq \mathit{scope}(b')$ then b' is dominated by b on A*, and thus action b' should not determine the ordering of A.* □

We can now define the *active scope* of each action, by excluding from its scope the scopes of its direct children with respect to $\sqsubseteq$. Specifically, we can define active scope as:

Def. 8. (Active Scope)
If $C(b)$ denotes the direct children of b with respect to $\sqsubseteq$, then the active scope of b*, denoted by $aScope(b)$, is defined as: $aScope(b) = \mathit{scope}(b) \setminus (\bigcup_{b' \in C(b)} \mathit{scope}(b'))$* □

In our example, the active scopes are shown in the last column of Table 3. From these we obtain $B = ascope(b1) \cup ascope(b3)$, and $W = ascope(b2)$, which define a valid preference. Specifically, Alg. `Apply` will return (assuming inactive elements go at the end) the following bucket order:
$\langle$ $\{European, Ferrari, German, Audi, BMW, Porsche, French, Citroen, Peugeot\}$,
$\{AlfaRomeo, Fiat, Lamborghini\}$,
$\{Asian, Japanese, Toyota, Korean, Kia, American, U.S.A., Chrysler, Dodge\}$ $\rangle$

Let see another example with *object*-scoped actions. Consider the same set of actions $\mathcal{B}$ but suppose that they are *object*-scoped instead of term-scoped, and assume that the table `Cars` contains the following tuples:

Id	Manuf	...
P	Porsche	
L	Lancia	
A	AlfaRomeo	
F	Ferrari	
T	Toyota	

The (plain and active) scopes in this case are:

action	scope	active scope
b1: Best(Europe)	{P, L, A, F}	{P}
b2: Worst(Italian)	{L, A, F}	{L, A}
b3: Best(Ferrari)	{F}	{F}

The sets B and W of the active scopes are: $B = \{P, F\}$ and $W = \{L, A\}$. With these parameters Alg. `Apply` will yield the ordering: $\langle\{P, F\}, \{L, A\}, \{T\}\rangle$.

Prop. 1. *If $B \cap W = \emptyset$ and $(T, \leq)$ is a* tree*, then in the expanded (through active scopes) actions, a term cannot be both Best and Worst.* $\diamond$.

This means that the inheritance of preferences over tree-structured facets cannot create any ambiguity. However if $(T, \leq)$ is a DAG (Directed Acyclic Graph), then Prop. 1 does not always hold, e.g. consider a term having two direct fathers one defined as best, the other as worst. Such actions do not define a valid preference and below we show how we can detect such cases. Let
$effAnchors(t) = minimal\{\ t' \mid t \leq t'$ and t' is anchor of one preference action$\}$.

Prop. 2. *If $B \cap W = \emptyset$, then there is not any ambiguity about a term t iff the actions in $effAnchors(t)$ are all either $Best$ or $Worst$.* $\diamond$

Note that Prop. 1 is a special case of Prop. 2 (since in trees for each term t it holds $|effActions(t)| \leq 1$). Algorithmically we can check whether the actions defined over a DAG-structured facet create an ambiguity by checking the condition of Prop. 2 only for those terms which have more than one direct fathers.

Now the algorithm that supports inherited preferences and scope-based resolution of conflicts is Alg. `PrefOrder`. It starts by computing the scopes of each action $b \in \mathcal{B}$ (line 2) using Def. 5 in order to compute the preorder relation $(\mathcal{B}, \sqsubseteq)$ (line 3). Afterwards, it computes the active scopes using the Def. 8 (line 5), and expands the original set of actions $\mathcal{B}$ to a new set of actions $\mathcal{B}'$, by including the new actions computed by the active scopes (line 6). Then, it parses the new actions set $\mathcal{B}'$ in order to get the B, W and $R_{\succ}$ (line 8). Finally, it calls Alg. `Apply` (line 9).

Algorithm 4 `PrefOrder`(E, $\mathcal{B}$, $Policy$)
Input: the set of elements E, the set of actions $\mathcal{B}$, and $Policy$ for *inactive* elements
Output: a bucket order over E

```
1: // Part (i): Computation of $(\mathcal{B}, \sqsubseteq)$
2: Compute the scopes of the actions in $\mathcal{B}$
3: Form $(\mathcal{B}, \sqsubseteq)$
4: // Part (ii): Efficient Computation of Act. Scopes
5: Use $(\mathcal{B}, \sqsubseteq)$ to compute the active scopes of the actions in $\mathcal{B}$
6: Use the active scopes to expand the set $\mathcal{B}$ to a set $\mathcal{B}'$
7: //Part (iii): Derivation of the final bucket order
8: $(B, W, R_{\succ}) \leftarrow$ Parse($\mathcal{B}'$)
9: return Apply($E, B, W, R_{\succ}, Policy$) // call to Alg. 1
```

Prop. 3. *Alg.* `PrefOrder` *respects the scope-based dominance rule (Def. 7).*

3.3.4. Relative Preferences over Hierarchically Organized Values

To complete the expressive power of the proposed actions, here we study the case of *relative* (qualitative) preferences over hierarchically organized values, despite the fact that their enactment from a GUI is not as simple (for the end users) as the rest actions. Specifically, our objective is to support sets of preferences of the form:
b1: $Asian \succ European$
b2: $European \succ Kia$
b3: $BMW \succ Asian$
b4: $Kia \succ Fiat$
b5: $Toyota \succ Kia$
whose semantics take into account inheritance, and conflicts are resolved in an intuitive manner. To this end we will define the *scope* and the *expansion* of such preferences.

Def. 9. (Scope of Relative Preferences)
The scope of a preference relationship $e_i \succ e_j$, denoted by $scope(e_i \succ e_j)$, is defined as: $scope(e_i \succ e_j) = (N^*(e_i) \times N^*(e_j)) \cup (N^*(e_j) \times N^*(e_i))$. □

Def. 10. (Expansion of Relative Preferences)
The expansion of a preference relationship $e_i \succ e_j$, denoted by $expansion(e_i \succ e_j)$, is defined as: $expansion(e_i \succ e_j) = \{e'_i \succ e'_j \mid e'_i \in N^*(e_i), e'_j \in N^*(e_j)\}$ □

This means that $expansion(e_i \succ e_j)$ actually "unfolds" the preference relationship $e_i \succ e_j$ on the basis of the subsumption relationships, while $scope(e_i \succ e_j)$ does not contain any preference relationship (it is used for resolving conflicts as we shall see below). The scope-based ordering of such actions is defined as before (Def. 6), i.e. $b \sqsubseteq b'$ iff $scope(b) \subseteq scope(b')$. We can now define the *active scope* of a preference $e_i \succ e_j$ by excluding from its expansion all relationships $e'_i \succ e'_j$ such that (e'_i, e'_j) belongs to the scope of a child (w.r.t. $\sqsubseteq$) action.

Def. 11. (Active Scope of Relative Preferences)
The active scope of a preference relationship b, in the context of a set of preference actions $\mathcal{B}$ is defined as: $aScope(b) = \{e_i \succ e_j \in expansion(b) \mid \nexists b' \in \mathcal{B} \; s.t. \; b' \sqsubseteq b, \; and \; (e_i, e_j) \in scope(b')\}$*, and this is equivalent to:* $aScope(b) = expansion(b) \setminus (\bigcup_{b' \sqsubseteq b} \{e_i \succ e_j \mid (e_i, e_j) \in scope(b')\})$ $\square$

For example, if the taxonomy of manufacturers has the form shown in the left diagram, then the scope-based ordering of preferences b1-b5 is that shown in the right diagram:

```
          MANUFACTURER                  |          b1
         /            \                 |         /  \
   European            Asian            |        b2  b3     b5
     /    \          /   |   \          |        |
   BMW    Fiat     Kia Toyota Lexus     |        b4
```

Now the expansion and the active scopes of the actions b1-b5 are shown in Table 4.

Table 4. Scopes: Example

preference	expansion	active scope
b1: $Asian \succ European$	$Asian \succ European$, $Asian \succ BMW$, $Asian \succ Fiat$, $Kia \succ European$, $Kia \succ BMW$, $Kia \succ Fiat$, $Toyota \succ European$, $Toyota \succ BMW$, $Toyota \succ Fiat$, $Lexus \succ European$, $Lexus \succ BMW$, $Lexus \succ Fiat$	$Asian \succ European$, $Asian \succ Fiat$, $Toyota \succ European$, $Toyota \succ Fiat$, $Lexus \succ European$, $Lexus \succ Fiat$
b2: $European \succ Kia$	$European \succ Kia$, $BMW \succ Kia$, $Fiat \succ Kia$	$European \succ Kia$, $BMW \succ Kia$
b3: $BMW \succ Asian$	$BMW \succ Asian$, $BMW \succ Kia$, $BMW \succ Toyota$, $BMW \succ Lexus$	$BMW \succ Asian$, $BMW \succ Kia$, $BMW \succ Toyota$, $BMW \succ Lexus$
b4: $Kia \succ Fiat$	$Kia \succ Fiat$	$Kia \succ Fiat$
b5: $Toyota \succ Kia$	$Toyota \succ Kia$	$Toyota \succ Kia$

As in the case of Best/Worst preferences (and Prop. 1 and 2), here we have to examine whether the expansion of relative preferences creates ambiguities (conflicts), apart from those which are resolved by the scope-based rule, and how we can identify such cases.

Let $\mathcal{B}$ be a set of relative preference actions, which define a valid preference relation $R_{\succ}$ over a $(T, \leq)$. We will examine whether a preference relationship between two terms e and e' of T (either $e \succ e'$ or $e' \succ e$), can be in the *active scope* of more than one action in $\mathcal{B}$. If this holds then this means both $e \succ e'$ and $e' \succ e$ could belong to the expanded (through the active scopes) preference relation, and thus that preference relation would be invalid.

Let us make the hypothesis that a relationship $e \succ e'$ belongs to the active scope of two actions b_i and b_j such that $b_i \neq b_j$. Suppose that $b_i : t_i \succ t_{i'}$ and $b_j : t_j \succ t_{j'}$. Certainly $e \succ e'$ should belong to the expansions of both b_i and b_j. Membership to expansion of b_i means: $e \leq t_i$ and $e' \leq t_{i'}$. Membership to expansion of b_j means: $e \leq t_j$ and $e' \leq t_{j'}$. We can identify the following cases:

(i) if $t_i \leq t_j$ and $t_{i'} \leq t_{j'}$ then it holds $b_i \sqsubseteq b_j$ and thus $e \succ e'$ can belong to the active scope of b_i only (and not of b_j).

(ii) if $t_j \leq t_i$ and $t_{j'} \leq t_{i'}$ then it holds $b_j \sqsubseteq b_i$ and thus $e \succ e'$ can belong to the active scope of b_j only (and not of b_i).

(iii) if $t_i \leq t_j$ and $t_{j'} \leq t_{i'}$ then neither $b_i \sqsubseteq b_j$ nor $b_j \sqsubseteq b_i$ holds. This means that in such cases it could belong to the active scopes of both. An example is shown at Figure 3 (left).

(iv) If $t_i || t_j$ and/or $t_{j'} || t_{i'}$, again we have $b_i \not\sqsubseteq b_j$ and $b_j \not\sqsubseteq b_i$, meaning that $e \succ e'$ would belong to the active scopes of both. Note that the case $t_i || t_j$ and $t_{j'} || t_{i'}$ can occur in DAGs, and an example is shown at Figure 3 (right). For the case of trees we cannot have $t_i || t_j$, since we know that $e \leq t_i$ and $e \leq t_j$ (it is not possible to hold all these three relationships). For the same reason in trees it cannot hold $t_{j'} || t_{i'}$.

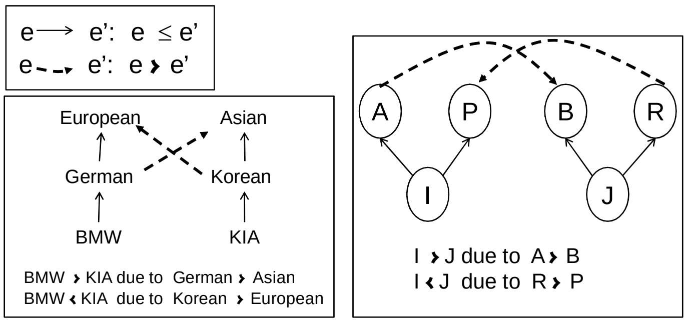


Figure 3. Examples of conflicts

The cases (iii) and (iv) are indicative situations when conflict can occur. Note that case (iii) can occur both in trees and DAGs, while case (iv) only in DAGs.

It follows from the above that we need methods for detecting the cases where inheritance causes invalidities. One method to do so, is to compute the expansion and then check for cycles. This means that a classical cycle detection algorithm (e.g. topological sort) is enough for detecting such cases.

We could also avoid the expansion step in some cases. Below we elaborate on a method that could be applied for the case of *tree-structured* taxonomies. To begin with, let R_e denote the expanded (through the notion of active scopes) preference relation of $R_{\succ}$ (obviously, $R_{\succ} \subseteq R_e$).

Prop. 4. (Relative Inherited Preferences and Conflicts)
For tree-*structured taxonomies, the* expansion through active scopes *of a valid preference relation* $R_{\succ}$

(yielding a preference relation R_e) can create a conflict iff *(if and only if) there are two actions in $R_\succ$ (not necessarily different) of the form $a \prec b$ and $c \prec d$ such that either:*
(i) $a \leq d$ and $c \leq b$ hold, or
(ii) $b \leq c$ and $d \leq a$, hold.
If these actions are the same, meaning that $a = c$ and $d = b$, the formulation of the proposition becomes: R_e has a conflict iff there is an action $a \prec b$ and either $a \leq b$ or $b \leq a$. $\diamond$

Based on the above proposition, below we describe an algorithmic method for identifying such problems without having to expand $R_\succ$, i.e. without having to compute R_e. For each pair of statements (i.e. for each pair of relationships in $R_\succ$) we check whether the condition of Proposition 4 holds. This means that we need to check the proposition $|R_\succ|(|R_\succ| - 1)/2$ times. To check the proposition once, we have to check whether four $\leq$ relationships hold. It the transitive closure of $\leq$ is stored then this can be checked fast (one scan, or even faster if indexes exist). If the transitively induced $\leq$ relationships are not stored, then we can check whether $t \leq t'$ by applying the reachability algorithm with cost analogous to the average depth of the taxonomy. If however the taxonomy has been *labeled* (e.g. using [2]), then we can check whether $t \leq t'$ in $\mathcal{O}(1)$.

At application level, the detected invalidities can be managed in various ways. For instance, we can inform the user and ask him/her to revise his preferences or to resolve the ambiguity. Alternatively one could consider the preference invalid and thus ignore it, or "cut" the inheritance at some points (e.g. at the points of conflicts), or employ other conflict resolution rules (e.g. the closer in $\leq$ hierarchy prevails, or the more recent action prevails, etc). All these are application-specific issues that go beyond the focus of this paper.

Returning to preference-based order, Alg. `PrefOrder` can be applied as it is (assuming the scope as defined in this section). Specifically, to produce the induced ordering we just have to pass to Alg. `Apply` (in the parameter $R_\succ$) all actives scopes of the actions in $\mathcal{B}$. Figure 4 shows the transitive reduction (i.e. the Hasse Diagram) of the relation $R_\succ$ for the preferences over the *Manufacturer* attribute. The derived bucket order by Alg. `PrefOrder` in our example is: $\langle\ \{BMW\}, \{Asian, Toyota, Lexus\}, \{European\}, \{Kia\}, \{Fiat\}\ \rangle$, while its restriction on the leaves of the taxonomy is: $\langle\ \{BMW\}, \{Toyota, Lexus\}, \{Kia\}, \{Fiat\}\ \rangle$ which captures the intuition.

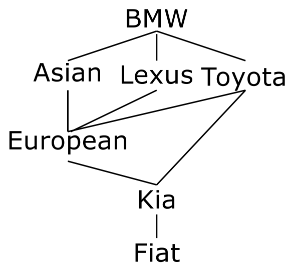


Figure 4. The relation R

3.3.5. Preferences over Hierarchical Set-Valued Attributes

In case we have set-valued attributes over hierarchically organized value domains, we can again exploit inheritance to order the sets. In particular, consider the *scope* and *active scope* as defined earlier, i.e. in a way that captures relative preferences. We can apply Alg. `PrefOrder` up to line 8 (i.e. just before calling the algorithm Apply), and then apply the algorithm described in section 3.3.2 (based on the relation $\succ_{\{\}}$), to derive the final bucket order.

The steps are sketched in more detail at Alg. 5.

Algorithm 5 PrefOrderSetValued(E, $\mathcal{B}$, $Policy$)
Input: the set of elements E **(here E is a family of sets)**, the set of actions $\mathcal{B}$, and $Policy$ for *inactive* elements
Output: a bucket order over E

```
1: // As in Alg. 4:
2: Compute the scopes of the actions in $\mathcal{B}$ and form $(\mathcal{B}, \sqsubseteq)$
3: Use $(\mathcal{B}, \sqsubseteq)$ to compute the active scopes of the actions in $\mathcal{B}$
4: Use the active scopes to expand the set $\mathcal{B}$ to a set $\mathcal{B}'$
5: $(B, W, R_{\succ}) \leftarrow$ Parse($\mathcal{B}'$)
6: // As in Alg. 3:
7: $R_{bw} \leftarrow \{(b, w) \mid b \in B, w \in W\}$
8: $R \leftarrow R_{bw} \cup R_{\succ}$
9: $R \leftarrow Closure_{transitivity}(R)$ // Addition of the transitively induced links
10: Compute $\succ_{\{\}}$ based on $wins$ and $support$ as in Alg. 3
11: $L \leftarrow$ SourceRemoval($\succ_{\{\}}$)
12: $I \leftarrow E \setminus dom(\succ_{\{\}})$ // $I$ is the set of inactive elements
13: $L' \leftarrow$ addInactiveElements($L, I, Policy$)
14: return $L'$
```

3.4. Multi-Facet Preferences

Here we describe the case where we have actions that concern more than one facets. The user can define separately a preference for each facet (using one or more actions) and then compose them using *priority* or *skyline* operators. We will focus on the prioritized composition of object-scoped actions.

Prioritized composition [25] of two preference relations $P1$ and $P2$, denoted by $P1 \rhd P2$, meaning that $P1$ has more priority than $P2$, is defined as:

$$x \succ_{P1 \rhd P2} y \text{ iff } x_1 \succ_{P1} y_1 \vee (x_1 = y_1 \wedge x_2 \succ_{P2} y_2)$$

Let $\mathcal{B}_i$ and $\mathcal{B}_j$ be two sets of object-scoped actions. $\mathcal{B}_i$ (resp. $\mathcal{B}_j$) consists of actions which have anchors from F_i (resp. F_j). Suppose the user has defined $\mathcal{B}_i \rhd \mathcal{B}_j$, and let A be the current object set (the focus). The ordering of A with respect to $\mathcal{B}_i \rhd \mathcal{B}_j$, is derived by ordering each block defined by the preference $\mathcal{B}_i$, using the preferences in $\mathcal{B}_j$. The exact steps are given in Alg. MFOrder. At Step 1 we derive the blocks defined by the preference $\mathcal{B}_i$. At Step 2 we order the elements of each block derived from the first step, using the actions in $\mathcal{B}_j$. Finally, at Step 3 we just put these blocks in the order specified by Step 1.

One can easily see that the produced bucket order interprets prioritized composition ($\rhd$) as follows:

$$x \succ_{P1 \rhd P2} y \text{ iff } x_1 \succ_{P1} y_1 \vee (x_1 \equiv_{P1} y_1 \wedge x_2 \succ_{P2} y_2)$$

Algorithm 6 MFOrder(A, $\mathcal{B}_i$, $\mathcal{B}_j$)
Input: the objects of current focus A, the set of actions $\mathcal{B}_i$ for facet F_i, and the set of actions $\mathcal{B}_j$ for facet F_j
Output: a bucket order of A corresponding to $\mathcal{B}_i \rhd \mathcal{B}_j$

```
1: We call the Alg. PrefOrder($A, \mathcal{B}_i$) and let $L = \langle A_1, \ldots A_M \rangle$ be the produced bucket order where $M$ is the number of blocks returned.
2: For each block $A_m$ of $L$ ($1 \leq m \leq M$) where $|A_m| > 1$, we call PrefOrder($A_m, \mathcal{B}_j$), returning a bucket order $L_m = \langle A_{m1}, \ldots, A_{mz} \rangle$.
3: We replace each block $A_m$ of $L$ with its bucket order $L_m$ and this yields the final bucket order $L = \langle L_1, \ldots, L_M \rangle$.
```

where $x_1 \equiv_{P1} y_1$ means that x_1 and y_1 are in the same block in the bucket order produced by $P1$. This means that the relative ordering of the blocks defined by $P1$ is preserved, and this policy is aligned with what the user expects to see.

The above algorithm can be straightforwardly generalized to more than two facets. For example, assume the user has defined
$\mathcal{B}_{Loc} \rhd \mathcal{B}_{Manuf} \rhd \mathcal{B}_{price}$, where
$\mathcal{B}_{Loc} = \{Best(Crete), Worst(Chania)\}$,
$\mathcal{B}_{Manuf} = \{Best(European), Worst(Italian), Best(Ferrari)\}$ (i.e. as in the Section 3.3.3), and
$\mathcal{B}_{Price} = \{price\ min\}$,
and suppose that the current focus A consists of the following tuples:

Id	Location	Manuf.	Price	...
L	Heraklion	Lancia	10	
B	Chania	BMW	20	
A1	Athens	Audi	20	
A2	Athens	Audi	21	
F1	Heraklion	Ferrari	100	
F2	Rethymno	Ferrari	80	

The constituent and final bucket orders are shown below (for the composed preferences we use nesting to make clear how each block was derived):

$$\begin{aligned}
L_{\mathcal{B}_{Loc}} &= \langle \{L, F1, F2\}, \{B\}, \{A1, A2\} \rangle \\
L_{\mathcal{B}_{Manuf}} &= \langle \{B, A1, A2, F1, F2\}, \{L\} \rangle \\
L_{\mathcal{B}_{Price}} &= \langle \{L\}, \{B, A1\}, \{A2\}, \{F2\}, \{F1\} \rangle \\
L_{\mathcal{B}_{Loc} \rhd \mathcal{B}_{Manuf}} &= \langle \langle \{F1, F2\}, \{L\} \rangle \{B\}, \{A1, A2\} \rangle \\
L_{\mathcal{B}_{Loc} \rhd \mathcal{B}_{Manuf} \rhd \mathcal{B}_{Price}} &= \langle \langle \langle \{F2\}, \{F1\} \rangle \{L\} \rangle \{B\}, \langle \{A1\}, \{A2\} \rangle \rangle \\
&= \langle F2, F1, L, B, A1, A2 \rangle
\end{aligned}$$

4. Complexity and Optimizations

At first (section 4.1) we discuss the computational complexity of the algorithms presented in the previous sections. Then at section 4.2 we introduce more efficient algorithms for object-ordering (including algorithms for set-valued facets which can be used also for evaluating the top-K elements of the object order). Finally, at section 4.3 we discuss some optimizations for multi-facet preferences. We have to clarify that these are a sort of "logical optimizations" which can be used as a basis for selecting the particular data structures and devising more efficient application-specific implementations.

4.1. Computational Complexity

Alg. 1 (`Apply`)

In the worst case all elements of E are involved and the most expensive task is that of topological sorting. The topological sorting is in $\mathcal{O}(|E| + |R|)$, thus w.r.t. E we can say that it is in $\mathcal{O}(|E|^2)$. If the actions are object-scoped, i.e. E corresponds to Obj, then the complexity of `Apply` is in $\mathcal{O}(|Obj|^2)$.

Alg. 3 (`ApplyOverFamiliesOfSets`)

Suppose E is a set of terms over $|T_i|$. The computation of the closure at line (3) is in $\mathcal{O}(|T_i|^3)$. Then we have to make $|E|^2$ computation of *wins* (between all pairs of element of E). Since to compute $wins(s, s')$ we need $\mathcal{O}(|s| * |s|)$ steps, the cost for computing all $wins$ is in $\mathcal{O}(|E|^2 setSizeAvg^2)$ where $setSizeAvg$ is the average size of the sets in E. Note that for some of the pairs we may have to compute the *support* of the involved sets. Since for computing the *support* of one atomic element the cost is $|T_i|$, the computation of $Support(s)$ is in $\mathcal{O}(|s||E|)$. Altogether, the computation of $wins$ and $support$ is in $\mathcal{O}(|T_i|^3 + |E|^2 setSizeAvg^2)$.

As regards the size of E, for a facet with $|T_i|$ values we can have at most $2^{|T_i|}$ sets, therefore $|E| \leq 2^{|T_i|}$. However $|E|$ cannot be bigger than $|A|$, therefore we can write $|E| \leq \min(2^{|T_i|}, |A|)$.

Alg. 4 (`PrefOrder`)

Let us now elaborate on the computational complexity of `PrefOrder` and suppose that all actions in $\mathcal{B}$ are object-scoped, i.e. E corresponds to Obj. Line 2 requires computing the scopes of all actions in $\mathcal{B}$. The computation of the scope of an action depends on $|Obj|$, and the size of the scope can be $|Obj|$ in size, i.e it is in $\mathcal{O}(|\mathcal{B}| * |Obj|)$. Line 3 requires $|\mathcal{B}|^2$ comparisons of sets, where each set can be $|Obj|$ in size, i.e. it is in $\mathcal{O}(|\mathcal{B}|^2|Obj|)$. Line 5 requires computing the active scopes and this depends on $|\mathcal{B}|$ and $|Obj|$, i.e it is in $\mathcal{O}(|\mathcal{B}||Obj|)$. Line 6 requires firstly to compute the parameters B, W and then to run Alg. `Apply`. The cost of the latter is in $\mathcal{O}(|Obj|^2)$ as discussed earlier. It follows that the overall cost of Alg. `PrefOrder` is in $\mathcal{O}(|Obj|(|Obj| + |\mathcal{B}|^2))$.

Alg. 6 (`MFOrder`)

Consider the algorithm `MFOrder` of Section 3.4. Assume that we have k facets, i.e. we have to order the elements according to a prioritized composition of actions over each facet $(\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_k)$.

Suppose that we have only two facets. In that case we have to apply Alg. `PrefOrder` with cost $\mathcal{O}(|Obj|(|Obj| + |\mathcal{B}_i|^2))$, and then for each block of the produced backet order to call Alg. `PrefOrder` with actions $|\mathcal{B}_j|$. The cost of each such call is in $\mathcal{O}(|Obj|(|Obj| + |\mathcal{B}_i|^2))$.

Overall we can say that the cost is $\mathcal{O}(|Obj|(|Obj| + |\mathcal{B}|^2))$. Now the cost of the algorithm for k facets is in $\mathcal{O}(k|Obj|(|Obj| + |\mathcal{B}|^2))$.

4.2. Optimizations for Deriving the Preference-based Order

Facet and Zoom Point Ordering.
Since the set of facets $\mathcal{F} = \{F_1, \ldots, F_k\}$ is usually small, the computation of $\succ_F$ is not expected to be expensive and we can use the proposed algorithms straightforwardly. The same is true for ordering the zoom points of each facet (as $|T_i|$ is usually small). Also note that we do not have to order the entire T_i but only the "active" terms (i.e. the zoom points $Z_i(ctx)$ and $ZR_i(ctx)$ as defined in Table 1) which are subsets of T_i.

Object Ordering.
Let us now focus on object ordering. If $|Obj|$ (and thus all $|A|$'s) is small, then we could again apply the proposed algorithm straightforwardly. If $|Obj|$ is high then $|A|$ can be high too.

At such cases we propose exploiting the benefits of adopting the FDT approach, i.e. the *fast convergence* to *small* results sets with a *few* clicks. Converge is discussed in detail (and it is quantified) at Section 5. This means that an acceptable and feasible policy is to order according to preference the set A only if $|A|$ is below a given threshold (say a few hundreds). For these reasons, below we present an algorithm, Alg. $\texttt{PrefOrder}_{\texttt{Opt}}$, which is an optimized version of Alg. `PrefOrder`, and whose complexity *does not depend on* $|Obj|$, but only on $|A|$ and $|\mathcal{B}|$, so it can be applied to large information bases. We could call this algorithm *focus-based.*

An alternative algorithm which can be beneficial in cases A is large is given in section 4.2.2.

4.2.1. A Focus-based Algorithm

Alg. $\texttt{PrefOrder}_{\texttt{Opt}}$ takes as input the set A to be ranked which we can assume that is not big (due to the fast convergence of FDT). First we present some auxiliary functions and the main idea (ignoring the case of relative preferences and set-valued attributes), and then the full algorithm.

We can start by the observation that if we have a function that checks whether $b1 \sqsubseteq b2$ holds, where $b1$ and $b2$ are actions, then we can form the relation $\sqsubseteq$. An algorithm that implements such a function, denoted by `CheckSubScopeOf`($b1$, $b2$), is given below. The key point is that we can decide whether $b1 \sqsubseteq b2$ holds, *without having to compute the scopes of these actions*. Instead we can base our approach on the definition of action scopes (Table 2). Specifically, if the anchor of $b1$ is not empty, while the one of $b2$ is empty then it returns True. The rest cases follow the general rule: *terms are more refined to facets.* In case of two term-anchored actions whose terms are $\leq$-related, then the actions are $\sqsubseteq$-related too (see line 6). If furthermore *labeling* is used (e.g. [2]) which is good choice in such applications, then the cost of this function is always in $\mathcal{O}(1)$.

```
1: function CheckSubScopeOf($b1$, $b2$): Boolean
2:     if $(b1.anchor \neq \epsilon) \wedge (b2.anchor = \epsilon)$ then
3:         return True
```

```
4:   if $(b1.anchor = \langle t_i \rangle) \wedge (b2.anchor = \langle F_j \rangle)$ then
5:       return True
6:   if $(b1.anchor = \langle t_i \rangle) \wedge (b2.anchor = \langle t_j \rangle) \wedge (t_i \leq t_j)$ then
7:       return True
8:   return False
```

For defining the intended algorithm we also need a boolean function `IsInScopeOf`(o, b) that returns True if o belongs to the scope of b. This function can be implemented as follows.

```
1: function IsInScope($o$, $b$): Boolean
2:     if b.scopeType="object order:" then
3:         if b.anchor="facet $F_i$" then
4:             if $D(o) \cap T_i \neq \emptyset$ then return True
5:             else return False
6:         else if b.anchor="term $t_j$" then
7:             if $t_j \in \bar{D}(o)$ then return True
8:             else return False
9:     return False
```

The main cost of `IsInScope`(o, b) is the cost required to check whether a term is narrower than another (line 7 requires checking if t_j is broader than a term assigned to o, i.e. if $t_j \geq t'_j$ where $t'_j \in D(o)$), so its cost is $\mathcal{O}(|R_{\leq}|)$ where $|R_{\leq}|$ denotes the number of relationships of a taxonomic relation. If labeling is used (e.g. [2]) then this cost is $\mathcal{O}(1)$.

We can now present the optimized version of Alg. `PrefOrder`, which is Alg. `PrefOrder`$_{\texttt{Opt}}$ shown below. It takes as input two parameters, an object set A, and a set of actions $\mathcal{B}$ (the latter is one of the $k + 2$ sets of actions).

Part (1) includes the optimized version of the lines (2-3) of `PrefOrder`, and Part (2) includes the optimized version of lines (5-6) of `PrefOrder`. We can see that the algorithm never computes the scope of any action and this is the key point for applying it in large information bases (in the sense that its computational complexity does not depend on $|Obj|$). Instead, it checks whether elements of E (recall that E has been reduced through clicks) belong to the scopes of actions.

Regarding its complexity, suppose that the taxonomy of each facet is labeled. The cost of the *first* part of the algorithm is in $\mathcal{O}(|\mathcal{B}|^2)$. Note that as long the user is not submitting a new action, $(\mathcal{B}, \sqsubseteq)$ can be preserved and reused when the user is changing his focus (so $\mathcal{O}(|\mathcal{B}|^2)$ is payed once). The *second* part of the algorithm has $|\mathcal{B}|$ iterations. Assuming labeling, the cost of each iteration is $|A|(1 + deg)$ where deg is the average number of direct children of an action w.r.t $\sqsubseteq$. It follows that the cost of the second part is $|\mathcal{B}|(|A|(1 + deg)) = |\mathcal{B}||A| + |\mathcal{B}|deg = |A|(|\mathcal{B}| + |\sqsubseteq|)$.

The *last* part of the algorithm is the cost of Alg. `Apply`, which in our context is expressed as $\mathcal{O}(|A|^2)$.

Changes for Capturing also Relative Preferences

The optimized algorithm Alg. `PrefOrder`$_{\texttt{Opt}}$ can be easily adapted so that to handle also relative preferences over hierarchically organized values (as defined in Section 3.3.4). Specifically we just have to make the changes shown at Fig. 5.

Regarding complexity, if labeling is available, then `CheckSubScopeOf` remains in $\mathcal{O}(1)$. The func-

Algorithm 7 `PrefOrder`$_{\texttt{Opt}}(E, \mathcal{B}, Policy)$
Input: the set of elements E, the set of actions $\mathcal{B}$, and $Policy$ for *inactive* elements
Output: a bucket order over E

```
1:  /** Part (1): Computation of $(\mathcal{B}, \sqsubseteq)$ */
2:  $Visited \leftarrow \emptyset$
3:  $R_{\sqsubseteq} \leftarrow \emptyset$    // $R_{\sqsubseteq}$ corresponds to $\sqsubseteq$
4:  for each $b \in \mathcal{B}$ do
5:      for each $b' \in \mathcal{B} \setminus Visited$ do
6:          if CheckSubScopeOf$(b, b')$ then
7:              $R_{\sqsubseteq} \leftarrow R_{\sqsubseteq} \cup \{(b \sqsubseteq b')\}$
8:          else if CheckSubScopeOf$(b', b)$ then
9:              $R_{\sqsubseteq} \leftarrow R_{\sqsubseteq} \cup \{(b' \sqsubseteq b)\}$
10:         $Visited \leftarrow Visited \cup \{b\}$
11:     endfor
12: endfor
13:
14: /** Part (2): Efficient Computation of Act. Scopes */
15: for each $b \in \mathcal{B}$ do
16:     $C(b) \leftarrow$ direct children of $b$ wrt $R_{\sqsubseteq}$
17:     ActiveScope[$b$] $\leftarrow \{e \in E \mid$ IsInScope$(e, b) \wedge$
18:        $(\forall c \in C(b)$ it holds IsInScope$(e, c) =$ False$)\}$
19: endfor
20:
21: Use the active scopes to expand the set $\mathcal{B}$ to a set $\mathcal{B}'$
22: /** Part (3): Derivation of the final bucket order */
23: $(B, W, R_{\succ}) \leftarrow$ Parse$(\mathcal{B}')$
24: return Apply$(E, B, W, R_{\succ}, Policy)$ // call to Alg. 1
```

`CheckSubScopeOf` : line 6	Let $b1.anchor = (e_i, e_j)$ and $b2.anchor = (e'_i, e'_j)$. We have to write: $((e_i \leq e'_i) \wedge (e_j \leq e'_j)) \vee$ $((e_j \leq e'_i) \wedge (e_i \leq e'_j))$
Alg. `PrefOrder`$_{\texttt{Opt}}$: lines (17-18)	ActiveScope[b] $\leftarrow$ $\{(e, e') \in E^2 \mid$ `IsInScope`$(e, e', b) \wedge$ $(\forall c \in C(b)$ it holds `IsInScope`$(e, e', c) =$ False$)\}$
`IsInScope`(o, o', b)	Let $b.anchor = (t_i, t_j)$. We have to write: $((t_i \in \bar{D}(o)) \wedge (t_j \in \bar{D}(o'))) \vee$ $((t_j \in \bar{D}(o)) \wedge (t_i \in \bar{D}(o')))$

Figure 5. Changes for capturing relative preferences over hierarchically organized values

tion $\texttt{IsInScope}(o, o', b)$ requires 4 checks of the form $t_j \in \bar{D}(o')$. If C_{avg} denotes the average number of terms that are directly assigned to an object $o \in Obj$, then these checks cost $\mathcal{O}(C_M)$ time. In the revised lines 17-18 of Alg. $\texttt{PrefOrder}_{\texttt{Opt}}$ the cost of each iteration is higher (in place of $|A|$ we now have $|A|^2$). Therefore the cost of the second part of the algorithm is now in $\mathcal{O}(|A|^2|\mathcal{B}|deg)$.

Synopsis. Table 5 summarizes the complexities for the 3 different parts of the algorithm, for the non-optimized and optimized version of the algorithm. The key point is that the complexity of the optimized algorithm it is independent of $|Obj|$.

Table 5. Complexity for non-optimized and optimized Alg. `PrefOrder` and $\texttt{PrefOrder}_{\texttt{Opt}}$

Part	Non-Optimized (Alg. `PrefOrder`)	Optimized (Alg. $\texttt{PrefOrder}_{\texttt{Opt}}$)	Optimized (Alg. $\texttt{PrefOrder}_{\texttt{Opt}}$) for relative prefs
Part 1	$\mathcal{O}(\|Obj\|(\|Obj\| + \|\mathcal{B}\|^2))$	$\mathcal{O}(\|\mathcal{B}\|^2)$	$\mathcal{O}(\|\mathcal{B}\|^2)$
Part 2	$\mathcal{O}(\|Obj\|\|\mathcal{B}\| + exp)$	$\mathcal{O}(\|A\|(\|\mathcal{B}\| + \| \sqsubseteq \|) = \mathcal{O}(\|A\|\|\mathcal{B}\|^2)$	$\mathcal{O}(\|A\|^2\|\mathcal{B}\|^2)$
Part 3	$\mathcal{O}(\|Obj\|^2)$	$\mathcal{O}(\|A\|^2)$	$\mathcal{O}(\|A\|^2)$
Overall	$\mathcal{O}(\|Obj\|(\|Obj\| + \|\mathcal{B}\|^2))$	$\mathcal{O}(\|A\|(\|A\| + \|\mathcal{B}\|^2))$	$\mathcal{O}(\|A\|^2\|\mathcal{B}\|^2)$

4.2.2. Optimizations for Capturing Set-Valued Attributes and Top-K Requirements

Here we provide an optimized algorithm for ordering a set of objects A for the case where (i) we have relative preferences over a facet whose values are hierarchically organized, and (ii) the object descriptions according to that facet are set-valued.

The reason for describing this case separately is because `IsInScope` was defined without considering set-valued attributes (however note that a plain vanilla algorithm was given in Sect. 3.3.5).

Let i be the facet whose terms are *hierarchically* organized and suppose that the object descriptions are *set-valued* at that facet. We start by assuming that the relation $\succ_{\{\}}$ over sets of terms of facet i has been computed, and obviously this includes inheritance resolution (computation of active scopes), and computation of the $wins$ (as we have described). Now the idea of the algorithm is the following:
(a) for the objects in A collect their descriptions w.r.t. F_i (let Z be this set),
(b) compute the restriction of $\succ_{\{\}}$ on Z,
(c) apply topological sorting on Z based on $\succ_{\{\}}$, and
(d) from the blocks of Z derive the blocks of the objects.

The exact steps of the algorithm are given in Alg. 8.

In line (1), Z is the family of sets of terms of F_i that occur in A. As stated earlier, we assume that the relation $\succ_{\{\}}$ over all values that occur in i has been defined. Now line (2) sets R to be the restriction of $\succ_{\{\}}$ on Z. Subsequently we apply topological sorting and we obtain L. Next, we start consuming L starting from the first block. Note that a block can contain one or more term sets. For each such set s we scan A and let *ob* be the objects that have this value. The elements in *ob* are appended to OL which is the order of objects. This continues until having consumed all blocks of L.

To compute $\succ_{\{\}}$ we can follow lines (1)-(10) of Alg. 3 and according to section 4.1, the complexity

Algorithm 8 PrefOrderSetValuedOpt(A, $\succ_{\{\}}$)
Input: A, an order $\succ_{\{\}}$ over a set-valued attribute with values in F_i.
Output: Ordering of A w.r.t. $\succ_{\{\}}$

```
1:  $Z = \{D_i(o) \mid o \in A\}$
2:  $R = \succ_{\{\}|Z}$ // restriction of $\succ_{\{\}}$ on $Z$
3:  $L \leftarrow$ SourceRemoval($R$)
4:  $OL \leftarrow \emptyset$
5:  for each block $b$ of $L$ do
6:      for each term set $s$ in $b$ do
7:          $ob = I(s) \cap A$ // $ob$ is the corresponding object block
8:          append $ob$ to $OL$
9:      append to $OL$ a block separator
10: return $OL$
```

for this is in $\mathcal{O}(|T_i|^3 + |E|^2 setSizeAvg^2)$. As regards the size of E, for a facet with $|T_i|$ values we can have at most $2^{|T_i|}$ sets, therefore $|E| \leq 2^{|T_i|}$. However $|E|$ cannot be bigger than $|A|$, therefore we can write $|E| \leq \min(2^{|T_i|}, |A|)$. One policy is to compute $\succ_{\{\}|Z}$ when needed. Another is to compute $\succ_{\{\}}$ over all distinct sets that occur for that facet (and to update it each time the user issues a preference action that concerns that order), to avoid recomputing it while the user restricts the set A. At run-time we just have to take its restriction of Z. We favor this policy in the given algorithm.

Generalization and Top-K Algorithm

Note that Alg. 8 essentially corresponds to the following approach: *first order the terms and from their ordering derive the object ordering*. Now suppose that we are *not* in the context of a set-valued attribute. If instead of passing the parameter $\succ_{\{\}}$, we pass an ordering over the values of T_i, then a rising question is whether we could use this algorithm, instead of Alg. 7, to produce the object order, and in what cases that algorithm would be beneficial.

Let approach this question from the computational complexity perspective. Suppose the case of relative preferences over a T_i. Instead of $\succ_{\{\}}$, we have to pass as parameter the ordering over the values of T_i. To compute this ordering we can use Alg. 7 where instead of having to order the objects in A we order the terms of T_i. In this case, and according to Table 5, the cost of this step is in $\mathcal{O}(|T_i|^2|\mathcal{B}|^2)$. Note that it is not necessary to compute Z or the restriction of the preference relation on Z (lines 1-2 of Alg. 8), in the sense that the final answer will be correct in any case due to the intersections with A at line 7. The computation of Z can be beneficial if Z is much smaller than T_i (in that case for less s's we will have to compute $I(s)$). Also note that the way A is defined can be exploited for further optimizations. For instance, if A has been defined intentionally (by one query), then we may already know the set Z without having to scan the set A.

Line (3) requires topological sorting whose cost is in $\mathcal{O}(|T_i|^2)$. The subsequent loop will have at most $|T_i|$ iterations (in particular $|Z|$), and the cost of each iteration is that of the operation $I(s) \cap A$. The operation $I(s) \cap A$ can be implemented in various ways, based on the sizes of the operants (and the data structures that are in place). E.g. if A is small it is better to scan A to select those objects whose D_i description equals s, and in this case the cost of an operation $I(s) \cap A$ is in $\mathcal{O}(|A|)$. On the other hand if

A is big, and $I(s)$ is small, then it is better to compute and scan $I(s)$ and then delete form this set those elements which are not in A. In this case the cost of an operation $I(s) \cap A$, if we assume direct access to the elements $I(s)$ and ability to perform binary search for lookups at A, is $\mathcal{O}(|I(s)| \log |A|)$. It follows that the cost of the loop is in $\mathcal{O}(|T_i| * \min(|A|, |I(s)| * \log |A|))$.

Overall, the cost of Alg. 8 for single-valued facets, including the cost for computing the preference relation to be passed to this algorithm, is in $\mathcal{O}(|T_i|^2|\mathcal{B}|^2 + |T_i| * \min(|A|, |I(s)| * \log |A|))$.

Recall that the cost of Alg. 7 (according to Table 5) is in $\mathcal{O}(|A|^2|\mathcal{B}|^2)$.

One benefit of Alg. 8 is that it can be more efficient than Alg. 7 if A is large and T_i is small. This is evident also from their complexities; Alg. 8 will have the cost $\mathcal{O}(|T_i|^2|\mathcal{B}|^2 + |T_i||I(s)| \log |A|)$, while Alg. 7 will have the cost $\mathcal{O}(|A|^2|\mathcal{B}|^2)$. Note that cases where $|A|$ can be very large may occur at application level. For instance, consider the case of a user who has expressed a number of object-scoped actions, and instead of restricting A, he would like to directly get the most preferred objects. The user wants to bypass the information thinning process probably because he believes that his preference actions are enough for bringing the most desired object to the top positions of the returned answer. Is it not hard to see that in this scenario, both plain (Alg. 4) and Alg. 7, are prohitively expensive. Alg. 8 will be more efficient, but it will still order the entire Obj. Although, according to our opinion the assumption that the user has expressed a detailed and complete description of his preferences is not very realistic (recall the discussion at the beginning of section 3), if we want to support such scenarios then a possible direction is to devise an appropriate *top-k algorithm*. Top-k algorithms for preference-aware queries have been proposed (e.g. [17, 51, 50]), however they are appropriate for plain relational sources, meaning that hierarchically organized values or set-valued attributes are not supported. However notice that Alg. 8 can be slightly changed to become a top-K algorithm. Specifically we consume blocks of L until OL has reached K objects. With this we complete the discussion of the main cases where the adoption of Alg. 8 is beneficial.

On the other hand Alg. 7, can be faster than Alg. 8 if the T_i is large in comparison to A (e.g. suppose T_i is a thesaurus and A is a set of few tens of objects). Also note that another merit of Alg. 7 is that it can be straightforwardly extended to accommodate *object-anchored* preference actions, or other *multi-facet preferences* (beyond those that we considered in this paper), due to its "scope-based" approach.

4.3. Optimizations for Multi-Facet Preferences

According to Section 4.1, the cost of `MFOrder` (presented at Section 3.4) for k facets is in $\mathcal{O}(k|Obj|(|Obj|+|\mathcal{B}|^2))$. Let us now suppose that in algorithm `MFOrder` we use `PrefOrder`$_{\texttt{Opt}}$ instead of `PrefOrder`. The cost of `MFOrder` in that case is in $\mathcal{O}(k|A|(|A| + |\mathcal{B}|^2))$. Analogously, one could adopt Alg. 8 and calculate accordingly the complexity of `MFOrder`.

Now we will introduce an alternative, approach, for supporting what we call *efficient priority-driftage*. We refer to the scenario where the user changes priorities with one click, and we want the new ordering of objects to appear instantly. To begin with, as long as the user does not submit an action, each $(\mathcal{B}_i, \sqsubseteq)$ can be kept stored and reused. Now suppose the user is inspecting an answer set A and he changes facets (i.e. he clicks on one facet) just for changing the priorities. Specifically, suppose the user has already specified $\mathcal{B}_i \rhd \mathcal{B}_j$, meaning that both $L_{\mathcal{B}_i}$ and $L_{\mathcal{B}_i \rhd \mathcal{B}_j}$ have already been computed (according to `MFOrder`). Suppose that the user now clicks on F_j just for changing the prioritized multi-facet preference to $\mathcal{B}_j \rhd \mathcal{B}_i$. According to the approach presented at Section 3.4, the application of `MFOrder`, will first compute $L_{\mathcal{B}_j}$ and finally it will produce $L_{\mathcal{B}_j \rhd \mathcal{B}_i}$. The key idea of the alternative algorithm is that we can avoid calling Alg. `PrefOrder` for each block of $L_{\mathcal{B}_j}$ at Step 2 of Alg. `MFOrder`. Specifically we will show that from $L_{\mathcal{B}_i}$ and $L_{\mathcal{B}_j}$ we can compute $L_{\mathcal{B}_j \triangleright \mathcal{B}_i}$. It can be easily proved that the first blocks of $L_{\mathcal{B}_j \triangleright \mathcal{B}_i}$ is the restriction of $L_{\mathcal{B}_i}$ on the objects of the first block, say A_{j1}, of $L_{\mathcal{B}_j}$. This means that the first blocks of $L_{\mathcal{B}_j \triangleright \mathcal{B}_i}$ can be obtained by scanning once $L_{\mathcal{B}_i}$ and deleting (skipping) each object encountered that does not belong to A_{j1}.

In the example of Section 3.4, the first two blocks of $L_{\mathcal{B}_{Loc} \triangleright \mathcal{B}_{Manuf}}$ (i.e. the blocks $\{F1, F2\}, \{L\}$), can be obtained by replacing the first block of $L_{\mathcal{B}_{Loc}}$ (i.e. $\{L, F1, F2\}$) by what is left after scanning $L_{\mathcal{B}_{Manuf}}$ and ignoring the elements that do not belong to $\{L, F1, F2\}$. Since $L_{\mathcal{B}_{Manuf}} = \langle\{B, A1, A2, \mathbf{F1}, \mathbf{F2}\}, \{\mathbf{L}\}\rangle$ we will get $\langle\{F1, F2\}, \{L\}\rangle$.

The cost of this approach, and assuming two facets, is in $\mathcal{O}(|A|^2)$, since we have to scan $|A|$ elements and for each one of them to perform a lookup to a set that consists of at most $|A|$ elements. If we have k facets then the cost is in $\mathcal{O}((k-1) * |A|^2)$. Notice that its cost is *independent of the number of user preference actions* $\mathcal{B}_i$ for each facet assuming that the user does not submit new actions. However this approach requires keeping in memory $L_{\mathcal{B}_1}, \cdots, L_{\mathcal{B}_k}$. Each has at most $|A|$ objects (according to suggested scenario), therefore the main memory cost is $k * |A|$ where k is the number of facets.

To summarize, an alternative to Alg. `MFOrder` policy is to compute and have stored the bucket order $L_{\mathcal{B}_i}$ for each $1 \leq i \leq k$. Then any prioritized composition of these sets of actions can be obtained by the method just described. The cost of priority driftage in this case does not depend on the number of preference actions but requires hosting $k|A|$ objects in main memory.

Top-K Prioritized Composition. Now suppose that the user wants (or the user screen has place for) only the *top-P* hits where P is a positive integer. We can exploit this constraint to speedup the process. In particular, from the bucket order of the first in priority $\mathcal{B}_i$, we can get the minimum number of blocks whose cardinality if summed is greater or equal to P (if this is possible, i.e. if $P \leq |A|$). For instance, if $P = 4$ in our example then we will get only the first 2 blocks of $L_{\mathcal{B}_{Loc}}$.

5. Interaction and User Effort

5.1. Convergence of FDT Exploration

The algorithm presented at Section 4.2 (Alg. `PrefOrder`$_{\texttt{Opt}}$) is based on the assumption that the focus A can be reduced very fast in a FDT-based interaction. In this section we report an analysis for justifying this claim.

Consider one taxonomy having the form of a complete and balanced tree of depth d and degree b. Let n be the number of objects in the information base (which are indexed by that tree). In that case $b * d$ is the number of *choices* a user has to see in order to reach (select) a particular leaf (i.e. the number of terms whose label the user has to read if he starts from the root of the tree), and d is the number of *decisions* (i.e. clicks) he has to make. If we want each object to have a distinct description (assuming that each object is classified to one leaf of the taxonomy), then this means that:

$$b * d = b * log_b n = \sqrt[d]{n} * d$$

The real-valued degree b that minimizes the product $b * d$ is the Euler's number e, so let us assume that $b = 3$ is the more beneficial degree. If each leaf should index 10 objects, then for $n = 10^{11}$ objects we need 10^{10} descriptions. Assuming $b = 3$ we get $b * d = 3 * log_3 10^{10} \cong 3 * 19 = 57$ choices, and $d \cong 19$ clicks.

Now suppose that we have k facets. Finding the desired description requires selecting one leaf from each T_i. As there are k facets, and we must select one leaf from each one of them, the overall displayed choices are obtained by multiplying by k. Since we have k facets, we can obtain the n distinct descriptions if each facet has $\sqrt[k]{n}$ leaves (since their cartesian product yields n distinct descriptions). In this case, the depth d of a facet equals to $log_b \sqrt[k]{n}$, and the degree b is $\sqrt[d]{\sqrt[k]{n}} = \sqrt[d*k]{n}$. It follows that the overall displayed choices are:

$$\begin{aligned} b * d * k &= b * log_b(\sqrt[k]{n}) * k = b * log_b(n^{\frac{1}{k}}) * k = \\ &= b * log_b(n^{\frac{k}{k}}) = b * log_b n \quad \text{(independent of } d\text{)} \end{aligned}$$

$$b * d * k = \sqrt[d*k]{n} * d \qquad \text{(independent of } b\text{)}$$

and the number of clicks required is $k * d$. Some combinations of values for these parameters follow:

$n/10$	k	b	d	Num. of Choices $b * d * k$	Num. of Clicks $k * d$
531.441 ($\sim 10^6$)	3	3	4	36	12
3.486.784.401 ($\sim 10^{11}$)	5	3	4	60	20
$\sim 10^{15}$	10	3	3	90	30

From the last row we can see, that in order to select the desired 10 objects from a *peta*-sized ($\sim 10^{15}$) information base, the user has to make 30 clicks.

In the previous analysis we have considered plain faceted taxonomies, not dynamic ones. According to FDT the only displayed terms during the interaction are those terms whose addition to the current selection, yields a conjunction having a non empty extension. The number of clicks will not be reduced, however the number of choices (the number of terms the user has to read) will be less, because each displayed internal term will not have all of its b children visible.

5.2. Plain FDT versus FDT with Preferences w.r.t. User Effort

Term-scoped preferences make these choices *less laborious* since the more desired options are shown first. Specifically if we assume that a preference relation for each facet has been defined, and we assume that the most preferred choice from each facet is prompted first (and it is unique), then the cost of the required decisions is not $b * d * k$ but $1 * d * k$ since the user just clicks on the first choice without having to look at the rest choices.

Returning to the context of our running example, if we assume that each of the 7 billions persons living on this planet sells one car, then for $n = 10^9$ objects we need 10^8 descriptions if we want to reach a block comprising 10 cars. Assuming $k = 10$ and degree $b = 3$ we get that $b * d * k = b * log_3 10^8 \cong 3 * 15 = 45$ choices have to be displayed (using plain faceted taxonomies), and certainly less than 45 using dynamic taxonomies. If we assume that preferences have been defined for each of the $k = 10$ facets, then the choices are reduced to 15, which is equal to the number of required clicks.

From this small example we can realize the potential of FDT on rapidly reducing very big information spaces.

We should also mention that the analysis in [46] shows that 3 zoom operations on leaf terms are sufficient to reduce an information base of 10^7 objects described by a taxonomy with 10^3 terms to an average of 10 objects.

6. Discussion and Related Work

Preferences and Decision Making in General

With respect to the characteristics described at section 2.2 our work focuses on multi-dimensional spaces with hierarchically organized attribute domains, and explicitly-specified and crisp qualitative user preferences. One main thesis of this paper is that effective preference specification presupposes knowledge of the information space and of the available choices. FDT-based interaction can aid users in getting acquainted with the information space and the available choices[8]. The computation and display of zoom points reduces the need for specifying complex preference profiles and users can explore the available choices (or the most preferred) *non linearly*. For instance, by clicking on the zoom points the user can inspect the available choices based on the desired values. Without effective exploration services the user is obliged to explore linearly blocks of objects and the derivation of small blocks (equivalently many blocks) requires rich preference specifications which are cumbersome to acquire. Let's consider a set of attributes and suppose the user selects one zoom point of the first attribute. The FDT approach will show those values of the rest attribute that are "active". Such browsing can aid users in identifying for each attribute those values for which it is worth specifying a complex value tradeoff (e.g. using a quantitative approach).

On a multidimensional space where user preferences for each dimension have been specified, the *efficient set* (else called skyline, or Pareto optimal set) is indeed very useful if the user is interested in one hit (e.g. one car to buy, one hotel to book). In the FDT approach and with the actions that we propose (specifically with term-scoped actions), the most preferred values from each dimension are shown as zoom points and at decreasing order of preference. We know that all objects that have these values (i.e. those objects that have at least one of the most preferred values of an attribute), are certainly part of the skyline. So the preference-extended FDT interaction inherently provides partially skyline support. However, to compute the entire skyline we need to apply one skyline algorithm (e.g. [38]), so skyline computation can be considered as a helpful complementary service (that's why we included it in the extended syntax). The computed skyline can then be explored using the FDT method.

FDT and Preferences

Regarding FDT and personalization, there is little research done. Most FDT systems, output facets in lexicographical order, or by taking into consideration the number of indexed documents. A *collaborative approach*, with explicit user ratings to design a personalized FDT system is proposed in [28], where several algorithms are proposed and evaluated. Moreover, the authors provide an evaluation methodology for personalized faceted search research, in order to complement user studies by being cheap, repeatable,

[8]Therefore FDT can aid preference elicitation even if instead of the preference actions proposed in this paper, the other well known approaches (e.g. Preference SQL [26]) are employed for expressing user preferences and/or deriving the corresponding object ranking.

and controllable. In contrast to ours, that work does not allow the user to express any attribute-based preference. Another collaborative approach for personalization of faceted search, by content filtering based on (manually or automatically) created ontologies is proposed in [53, 52]. In that approach, relevance to users is measured by calculating the distance between values in the hierarchical ontology. Again, it is a collaborative approach and the personalization supported is simplistic and concerns only the visualization layer. Another approach for data warehouses, is described in [44] where at each step of navigation, the system asks the user one or more questions in order to fetch the most promising set of facets, based on a simple approximation algorithm. Again, that work does not allow users to express preferences, and the framework does not support hierarchically organized values.

IR and Preferences
Preference management and personalization in IR, has been approached from various perspectives. For instance, query reformulation through relevance feedback [42, 12, 8], or retrieval models that rank objects based on user-defined reference points [29] fall into this category. Such approaches affect only object ranking, do not exploit metadata (i.e. facets), and with respect to our proposal they can be considered as complementary techniques that are based solely on the textual content of the objects. Recall that our model can incorporate IR-like rankings (recall the *Relevance* facet).

To conclude and to the best of our knowledge this is the first work that proposes an *incremental* preference elicitation mode over attributes with *hierarchically organized values* and possibly *set-valued*, and employs a *scope-based conflict resolution rule*.

7. Concluding Remarks

In this paper we motivated the need for real-time preference elicitation and we introduced a language (including its syntax, semantics and GUI-level exploitation methods) for enriching the interaction paradigm of Faceted Dynamic Taxonomies with *preference elicitation* and *preference-based interaction*. Key aspects of the proposed approach include, the support of *hierarchically organized values*, the support of *set-valued* attributes, and the *incremental preference specification* mode with the *scope-based method for resolving conflicts*. In addition, the *rapid reduction* of the information space that is possible with FDT, makes preference-based ordering feasible on large information bases, since the introduced algorithms for producing the preference order *are independent of the size of the information base*; they depend on the size of the focus, and the number of the preference actions enacted by the user. Furthermore, we provided a top-k variation of the algorithm suitable for the case where the size of the focus is big. To the best of our knowledge no other approach supports the above.

There are several issues that are worth further research. Apart from application-specific optimizations (based on kind of information source at hand), as faceted taxonomies are appropriate for combining multiple sources of evidence, a promising topic is to investigate how the proposed approach can be synthesized with automatic matching and mediator approaches (like those presented in [55]) to build *dynamic preference-based integration systems*. Finally, another direction is to consider information spaces with more complex structure, like RDF/S sources or even Fuzzy RDF/S, i.e. to extend the browsing approach presented at [33] with preferences.

Acknowledgements

This research has been co-financed by the European Union (European Social Fund - ESF) and Greek national funds through the Operation Program ”Education and LifeLong Learning” of the National Strategic Reference Framework (NSRF) - Research Funding Program: Herakleitus II. Investing in knowledge society through the European Social Fund.

References

[1] Special issue on Supporting Exploratory Search, *Communications of the ACM*, **49**(4), April 2006.

[2] Agrawal, R., Borgida, A., Jagadish, H.: Efficient management of transitive relationships in large data and knowledge bases, *ACM SIGMOD Record*, **18**(2), 1989, 253–262.

[3] Agrawal, R., Wimmers, E. L.: A framework for expressing and combining preferences, *SIGMOD ’00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, ACM, New York, NY, USA, 2000, ISBN 1-58113-217-4.

[4] Andreka, H., Ryan, M., Schobbens, P.-Y.: “Operators and Laws for Combining Preference Relations”, *Journal of Logic and Computation*, **12**(1), 2002, 13–53.

[5] Balke, W.-T., Güntzer, U.: Multi-objective query processing for database systems, *VLDB ’04: Proceedings of the Thirtieth international conference on Very large data bases*, VLDB Endowment, 2004, ISBN 0-12-088469-0.

[6] Barrett, R., Salles, M.: *Social Choice With Fuzzy Preferences*, Economics working paper archive (university of rennes 1 & university of caen), Center for Research in Economics and Management (CREM), University of Rennes 1, University of Caen and CNRS, 2006.

[7] Ben-Yitzhak, O., Golbandi, N., Har’El, N., Lempel, R., Neumann, A., Ofek-Koifman, S., Sheinwald, D., Shekita, E., Sznajder, B., Yogev, S.: Beyond basic faceted search, *Proceedings of the international conference on Web search and web data mining*, 2008, 33–44.

[8] Bot, R. S., Wu, Y. B.: ”Improving Document Representations Using Relevance Feedback: The RFA Algorithm”, *Procs of the 13th ACM Intern. Conf. on Information and Knowledge Management*, Washington, USA, 2004.

[9] Boutilier, C., Brafman, R. I., Domshlak, C., Hoos, H. H., Poole, D.: CP-nets: A Tool for Representing and Reasoning with Conditional Ceteris Paribus Preference Statements, *Journal Of Artificial Intelligence Research*, **21**, 2004, 135–191.

[10] Chakrabarti, K., Chaudhuri, S., Hwang, S.: “Automatic Categorization of Query Results”, *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, 2004, 755–766.

[11] Chen, L., Pu, P.: *Survey of Preference Elicitation Methods*, Technical report, Swiss Federal Institute of Technology in Lausanne (EPFL), 2004.

[12] Choi, J., Kim, M., Raghavan, V. V.: ”Adaptive Feedback Methods in an Extended Boolean Model”, *ACM SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*, New Orleans, LA, September 2001.

[13] Chomicki, J.: Preference formulas in relational queries, *ACM Trans. Database Syst.*, **28**(4), 2003, 427–466, ISSN 0362-5915.

[14] desJardins, M., Eaton, E., Wagstaff, K. L.: Learning user preferences for sets of objects, *ICML '06: Proceedings of the 23rd international conference on Machine learning*, ACM, New York, NY, USA, 2006, ISBN 1-59593-383-2.

[15] Fishburn, P.: *Utility Theory for Decision Making*, Wiley, New York, 1970.

[16] Gadanho, S. C., Lhuillier, N.: Addressing uncertainty in implicit preferences, *RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems*, ACM, New York, NY, USA, 2007, ISBN 978-1-59593-730–8.

[17] Georgiadis, P., Kapantaidakis, I., Christophides, V., Nguer, E. M., Spyratos, N.: Efficient Rewriting Algorithms for Preference Queries, *ICDE*, 2008.

[18] Hildebrand, M., van Ossenbruggen, J., Hardman, L.: "/facet: A Browser for Heterogeneous Semantic Web Repositories", *Procs of ISWC '06*, Athens, GA, USA, Nov. 2006.

[19] Hyvönen, E., Mäkelä, E., Salminen, M., Valo, A., Viljanen, K., Saarela, S., Junnila, M., Kettula, S.: "MuseumFinland – Finnish Museums on the Semantic Web", *Journal of Web Semantics*, **3**(2), 2005, 25.

[20] Inan, H.: *Search Analytics: A Guide to Analyzing and Optimizing Website Search Engines*, Book Surge Publishing, 2006.

[21] Kahn, A. B.: Topological sorting of large networks, *Commun. ACM*, **5**(11), 1962, 558–562, ISSN 0001-0782.

[22] Karlson, A. K., Robertson, G. G., Robbins, D. C., Czerwinski, M. P., Smith, G. R.: "FaThumb: a Facet-Based Interface for Mobile Search.", *Procs of the Conference on Human Factors in computing systems, CHI'06*, New York, NY, USA, Apr. 2006.

[23] Keeney, R. L., Raiffa, H.: *"Decisions with Multiple Objectives: Preferences and Value Tradeoffs"*, John Wiley & Sons, 1976.

[24] Kelly, D., Teevan, J.: Implicit feedback for inferring user preference: a bibliography, *SIGIR Forum*, **37**(2), 2003, 18–28, ISSN 0163-5840.

[25] Kießling, W.: Foundations of preferences in database systems, *VLDB'02: Procs of the 28th intern. conf. on Very Large Data Bases*, VLDB Endowment, 2002.

[26] Kießling, W., Kostler, G.: "Preference SQL - Design, Implementation, Experiences", *Procs of the 28th Intern. Conf. on Very Large Data Bases (VLDB)*, Hong Kong, China, 2005.

[27] Kopidaki, S., Papadakos, P., Tzitzikas, Y.: STC+ and HSTC: Two Novel Online Results Clustering Methods for Web Searching, *WISE'09: Procs of the 10th Intern. Conf. on Web Information Systems Engineering*, October 2009.

[28] Koren, J., Zhang, Y., Liu, X.: Personalized interactive faceted search, *WWW'08: Procs of the 17th intern. conf. on World Wide Web*, ACM, New York, NY, USA, 2008, ISBN 9781605580852.

[29] Korfhage, R. R.: *"Information Storage and Retrieval"*, John Wiley & Sons, 1997.

[30] Koutrika, G., Ioannidis, Y.: Personalized Queries under a Generalized Preference Model, *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, IEEE Computer Society, Washington, DC, USA, 2005, ISBN 0-7695-2285-8.

[31] Mäkelä, E., Hyvönen, E., Saarela, S.: Ontogator - A Semantic View-Based Search Engine Service for Web Applications, *International Semantic Web Conference*, Athens, GA, USA, Nov. 2006.

[32] Mäkelä, E., Viljanen, K., Lindgren, P., Laukkanen, M., Hyvönen, E.: Semantic yellow page service discovery: The veturi portal, *In poster paper at ISWC '05*, Nov. 2005.

[33] Manolis, N., Tzitzikas, Y.: “Interactive Exploration of Fuzzy RDF Knowledge Bases”, *Procs of the 8th Extended Semantic Web Conference (ESWC’11)*, Heraklion, Greece, 2011.

[34] Oren, E., Delbru, R., Decker, S.: ”Extending Faceted Navigation for RDF Data”, *Procs of ISWC ’06*, Athens, GA, USA, Nov. 2006.

[35] Papadakos, P., Armenatzoglou, N., Kopidaki, S., Tzitzikas, Y.: ”On Exploiting Static and Dynamically Mined Metadata for Exploratory Web Searching”, *Knowl. Inf. Syst.*, **30**(3), 2012, 493–525.

[36] Papadakos, P., Kopidaki, S., Armenatzoglou, N., Tzitzikas, Y.: Exploratory web searching with dynamic taxonomies and results clustering, *ECDL’09: Procs of the 13th European Conf. on Digital Libraries*, Corfu, Greece, September 2009.

[37] Papadakos, P., Tzitzikas, Y., Zafeiri, D.: ”An Interactive Exploratory System with Real-Time Preference Elicitation”, *Proceedings of the 13th International Conference on Web Information Systems Engineering (WISE’12)*, November 2012.

[38] Papadias, D., Ta, Y., Fu, G., Seeger, B.: Progressive skyline computation in database systems, *ACM Trans. Database Syst.*, **30**(1), 2005, 41–82, ISSN 0362-5915.

[39] Peintner, B., Viappiani, P., Yorke-Smith, N.: Preferences in Interactive Systems: Technical Challenges and Case Studies, *AI Magazine*, **29**(4), Winter 2008, 13–24.

[40] Pu, P., Chwn, L.: User-Involved Preference Elicitation for Product Search and Recommender Systems, *AI Magazine*, **29**(4), Winter 2008, 93–103.

[41] Rashid, A. M., Albert, I., Cosley, D., Lam, S. K., Mcnee, S. M., Konstan, J. A., Riedl, J.: Getting to know you: learning new user preferences in recommender systems, *IUI ’02: Proceedings of the 7th international conference on Intelligent user interfaces*, ACM Press, New York, NY, USA, 2002, ISBN 1581134592.

[42] Rochio, J.: “Relevance Feedback in Information Retrieval”, in: *The SMART Retrieval System* (G. Salton, Ed.), Prentice Hall, Englewood Cliffs, NJ, 1971, 313–323.

[43] Rossi, F., Venable, K. B., Walsh, T.: Preferences in constraint satisfaction and optimization, *AI Magazine*, **28**(4), 2008.

[44] Roy, S. B., Wang, H., Das, G., Nambiar, U., Mohania, M.: Minimum-effort driven dynamic faceted search in structured databases, *CIKM’08: Procs of the 17th ACM conf. on Information and knowledge management*, New York, NY, USA, 2008, ISBN 978-1-59593-991-3.

[45] Sacco, G.: Some Research Results in Dynamic Taxonomy and Faceted Search Systems, *SIGIR’2006 Workshop on Faceted Search*, 2006.

[46] Sacco, G. M.: Analysis and Validation of Information Access Through Mono, Multidimensional and Dynamic Taxonomies, *FQAS*, 2006.

[47] Sacco, G. M., (Editors), Y. T.: *Dynamic Taxonomies and Faceted Search: Theory, Practice and Experience*, Springer, 2009, ISBN 978-3-642-02358-3, ISBN = 978-3-642-02358-3.

[48] Schafer, J. B., Konstan, J. A., Riedl, J.: E-Commerce Recommendation Applications, *Data Min. Knowl. Discov.*, **5**(1-2), 2001, 115–153, ISSN 1384-5810.

[49] Schraefel, M., Karam, M., Zhao, S.: “mSpace: Interaction Design for User-Determined, Adaptable Domain Exploration in Hypermedia”, *Procs of Workshop on Adaptive Hypermedia and Adaptive Web Based Systems*, Nottingham, UK, Aug. 2003.

[50] Spyratos, N., Sugibuchi, T., Yang, J.: “Personalizing Queries over Large Data Tables”, *Procs of the 15th East-European Conference on Advances in Databases and Information System (ADBIS 2011)*, Vienna, Austria, September 2011.

[51] Stefanidis, K., Drosou, M., Pitoura, E.: PerK: personalized keyword search in relational databases through preferences, *EDBT*, 2010.

[52] Tvarožek, M., Barla, M., Frivolt, G., Tomša, M., Bieliková, M.: Improving Semantic Search Via Integrated Personalized Faceted and Visual Graph Navigation., *SOFSEM*, 4910, Springer, 2008, ISBN 978-3-540-77565-2.

[53] Tvarožek, M., Bieliková, M.: Personalized Faceted Browsing for Digital Libraries, *ECDL*, 2007.

[54] Tzitzikas, Y., Armenatzoglou, N., Papadakos, P.: FleXplorer: A Framework for Providing Faceted and Dynamic Taxonomy-based Information Exploration, *Procs of FIND'2008 (at DEXA '08)*, Torino, Italy, Sept. 3, 2008, ISSN 1529-4188, Doi=10.1109/DEXA.2007.4312888.

[55] Tzitzikas, Y., Spyratos, N., Constantopoulos, P.: "Mediators over Taxonomy-based Information Sources", *VLDB Journal*, **14**(1), 2005, 112–136.

[56] Yee, K., Swearingen, K., Li, K., Hearst, M.: "Faceted Metadata for Image Search and Browsing", *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2003, 401–408.

A. Proofs

Prop. 1
If $B \cap W = \emptyset$ and $(T, \leq)$ is a *tree*, then in the expanded (through active scopes) actions, a term cannot be both Best and Worst.
Proof:

> Since $(T, \leq)$ is a tree, for each term t there is only one and unique path starting from t and ending to the root of the tree. The term t will be in active scope of the closest action, i.e. in the active scope of an action anchored on t, or on its father, or on the father of its father, and so on. Therefore it can be in the active scope of an action anchored to its closest (in the path) term. Since $B \cap W = \emptyset$ that anchor can be either in B or in W (not both), therefore t cannot be both Best and Worst.

◇
Prop. 2
If $B \cap W = \emptyset$, then there is not any ambiguity about a term t iff the actions in $effAnchors(t)$ are all either $Best$ or $Worst$.
Proof:

> It is a straightforward consequence of the definitions that a term t will be in active scopes of the actions anchored in the terms that belong to the set $effAnchors(t)$ = $minimal$ $\{ t' \mid t \leq t'$ and t' is anchor of one preference action$\}$. If all such actions are Best (resp. Worst) statements, then t will be Best (resp. Worst) in the expanded statements. If however some of these actions are Best and some are Worst, then (since t will be in the active scopes of all of them) t will be both Best and Worst, and thus we expansion will create ambiguities (and hence an invalid preference).

◇
Prop. 3
Alg. `PrefOrder` respects the scope-based dominance rule (Def. 7).
Proof:

> Suppose the opposite, i.e. suppose that $\exists\, b, b'$ and $A \subseteq Obj$ s.t. $A \subseteq scope(b) \subseteq scope(b')$ and that `PrefOrder` orders the elements of A on the basis of action b'. This cannot be true, since according to Def. 8, the active scope of b' will not contain A. Notice that although in the definition of *active*

scopes (Def. 8) only the direct children are used, the *scope* (as defined in Def. 5) is based on $N^*(e)$ so it takes into account all children wrt $\leq$. For this reason it is enough at Def. 8 (and actually more efficient at implementation level) to consider only the direct children.

$\diamond$

Prop. 4
For *tree*-structured taxonomies, the *expansion through active scopes* of a valid preference relation $R_\succ$ (yielding a preference relation R_e) can create a conflict *iff* (if and only if) there are two actions in $R_\succ$ (not necessarily different) of the form $a \prec b$ and $c \prec d$ such that either:
(i) $a \leq d$ and $c \leq b$ hold, or
(ii) $b \leq c$ and $d \leq a$, hold.
If these actions are the same, meaning that $a = c$ and $d = b$, the formulation of the proposition becomes:
R_e has a conflict iff there is an action $a \prec b$ and either $a \leq b$ or $b \leq a$.
Proof:

(Direction: If the conditions of the proposition hold then R_e has a conflict)
As we can see from Figure 6(i), if the conditions of the proposition hold, then R_e contains a conflict (either between a and c, or between d and b).
Regarding the special case (where the two actions are the same), note that if $b \leq a$ then we get the cycle $b \prec b$ (see Figure 6(ii-left)). If $a \leq b$ then we get the cycle $a \prec a$ (see Figure 6(ii-middle)). Note than non trivial cycles (i.e. not self-cycles) can also occur, e.g. if $c \leq b \leq a$, with the expansion we will get $c \prec b$ and $b \prec c$ (see Figure 6(ii-right)).

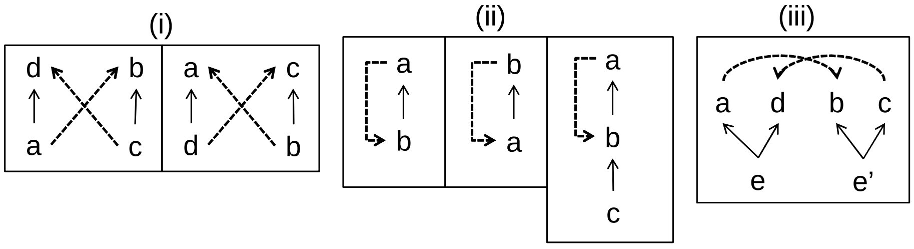


Figure 6. Examples

(Direction: if R_e has a conflict then the conditions of the proposition holds)
Trivial Cycle
Suppose that R_e has a trivial cycle of the form $a \succ a$. Since this relationship cannot belong to $R_\succ$ (which is acyclic by assumption), it should be result of an inherited action, therefore a should have a superclass, say sp, for which there is an action $sp \succ sb$, and this action for being inheritable to a, it should also be $a \leq sb$. Therefore it should hold $a \leq sb$ and $a \leq sp$. However, since $\leq$ is a tree, sb and sp cannot be incomparable (i.e. it cannot be $sb||sp$), therefore it should either be $a \leq sb \leq sp$ or $a \leq sp \leq sb$. We reached to the conclusion that there exists an action $sp \succ sb$ and either $sb \leq sp$ or $sp \leq sb$. This is exactly what the proposition states.

Cycle of the form $e \prec e' \prec e$
A relationship $e \prec e'$ can belong to R_e either because it belongs to $R_\succ$, or due to an action $a \prec b$ to whose active scope the relationship $e \prec e'$ belongs. In the latter case it should be $e \leq a$ and $e' \leq b$.

Analogously, a relationship $e' \prec e$ can belong to R_e either because it belongs to $R_\succ$, or due to an action $c \prec d$ to whose active scope the relationship $e \prec e'$ belongs. In the latter case it should be $e' \leq c$ and $e \leq d$.

The situation is illustrated at Figure 6(iii).

However, since $\leq$ is a tree it cannot be $a||d$ nor $b||c$. Therefore we can have one of the following four cases (also illustrated at Figure 7.

(i) $a \leq d$ and $b \leq c$

(ii) $a \leq d$ and $c \leq b$

(iii) $d \leq a$ and $b \leq c$

(iv) $d \leq a$ and $c \leq b$

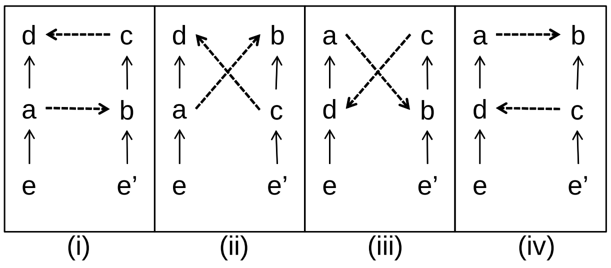
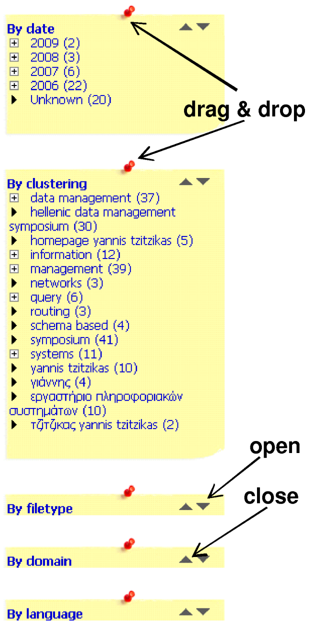


Figure 7. Examples

We cannot be in case (i) because in that case $e \succ e'$ would not be in the active scope of $c \prec d$ (that would contradict one of our hypothesis).
Similarly, we cannot be in case (iv) because in that case $e \prec e'$ would not be in the active scope of $a \prec b$.
So only (ii) and (iii) can hold. Notice that we reached to the exact conditions that the proposition states.

$\diamond$



By date
2009 (2)
2008 (3)
2007 (6)
2006 (22)
Unknown (20)

By clustering
data management (37)
hellenic data management symposium (30)
homepage yannis tzitzikas (5)
information (12)
management (39)
networks (3)
query (6)
routing (3)
schema based (4)
symposium (41)
systems (11)
yannis tzitzikas (10)
γιάννης (4)
εργαστήριο πληροφοριακών συστημάτων (10)
τζίτζικας yannis tzitzikas (2)

By filetype

By domain

By language

Figure 8. GUI for facets' order and visibility

B. A Possible Application in Web Searching

Here we discuss the application of our approach in Web searching. Commonly, the various static metadata that are available to a search engine (e.g. domain, language, date, filetype, etc.) are exploited only through the advanced (form-based) search facilities that some WSEs offer (and users rarely use). An approach that exploits such metadata by adopting the interaction paradigm of FDT exploration is described in [36] (for a more complete presentation that also includes the results of a user study please refer to [35]). In addition, the FDT interaction scheme has already been implemented over the `Mitos` WSE [9].

Capturing Default Behaviour.
Regarding preferences, the default operation mode of `Mitos` (and of most FDT search engines) is captured by the following actions that are captured by the proposed syntax:

```
facets order:  lexicographic min
terms order:  count max
objects order:  Relevance value max
```

Enaction of Preference Actions
Recall that Figure 2 sketched how some of the preference actions can be enacted. In the current version of the preference-aware GUI of `Mitos`, the user can specify the desired order of facets also by drag and drop and he can expand or collapse them as shown in Figure 8. Also note that although the syntax of Section 3.1 allows actions that do not make sense, e.g. "facets order: term Location.Rhodes best", the GUI enacted actions can be designed to allow forming only meaningful actions (in general, such tests are responsibility of a *validation* module).

Prioritized Composition
At section 3.4 we described prioritized composition. At application level, the priorities can be defined manually by the user through the GUI. For instance a small bar at the top that has k slots of the form 1[ ], 2[ ], ..., k[ ], each corresponding to a drop down list (with the names of the facets), can be used for this purpose. To further ease interaction, and since FDT exploration is *session*-based, a complementary approach is to provide a method that does not require any input from the user. One method to do so, is to exploit the sequence of clicks to determine the prioritization. Specifically, the first in priority can be considered to be the facet that was clicked last, the second in priority the facet that was clicked before, and so on. Such *automatic session-based* can be generalized to include the results of complex log analysis.

Finally, another demo application of our framework over hierarchical information spaces expressed using RDF/S, is described in [37].

[9]Under development by the Department of Computer Science of the University of Crete and FORTH-ICS (http://groogle.csd.uoc.gr:8080/mitos/).