

Preference Queries with SV-Semantics

Werner Kießling

Department of Applied Computer Science, University of Augsburg

Universitätsstraße 14

D- 86159 Augsburg, Germany

+49 821 598 2134

kiessling@informatik.uni-augsburg.de

ABSTRACT

Personalization of database queries requires a semantically rich, easy to handle and flexible preference model. Building on preferences as strict partial orders we provide a variety of intuitive base preference constructors for numerical and categorical data, including so-called d-parameters. As a novel semantic concept for complex preferences we introduce the notion of ‘substitutable values’ (SV-semantics), characterizing equally good values amongst indifferent values. Pareto and prioritized preference construction preserves strict partial orders, which instantly solves crucial well-known problems for preference queries. We can point out a new semantic-guided way to cope with the infamous flooding effect of query engines. Contrary to a wide-spread belief we can give evidence that the result sizes of Pareto or skyline queries not necessarily explode for multiple attributes. Moreover, we can show that known laws from preference relational algebra remain valid under SV-semantics. Since most of these laws rely on transitivity, preservation of strict partial order is essential to algebraically optimize complex preference queries. Similarly, well-known efficient evaluation algorithms for the preference selection operator rely on transitivity. In a nutshell, preference constructors with SV-semantics enable an intuitive and powerful personalization of database queries and at the same time are the key to efficient preference query evaluation.

1. INTRODUCTION

Personalization of database queries is an increasingly important issue. For instance let’s consider e-procurement, which is one of the fastest growing application areas for e-commerce. There are multiple reasons, why today the sales process in e-procurement is still a business with lots of costly human interaction. The misery starts already with the product search. Often B2B customers are forced to manually scroll through huge electronic product catalogs. Frequently commercial search engines simply interpret the customer’s search conditions as hard constraints, yielding the embarrassing ‘empty result’ effect. A failing solution attempt is to interpret the

search constraints as ‘or’-conditions, causing the ‘flooding’ effect. Another time-consuming and error-prone approach is parametric search. Offering a plain full-text search is no remedy either, because B2B product search is largely an attribute based search, if e-catalog standards (being mostly XML-based) are in place. Thus state-of-the-art approaches to find products are not enough for the B2B customer ([25]). In fact, a good product search demands a personalized search engine that can handle attribute-rich e-catalog data, that can be personalized to the customer’s wishes, roles and situations, and that fully automatically delivers best alternatives when there is no perfect match.

Let’s consider the following motivating example from an e-procurement setting.

Example 1 Personalized query composition

Let’s assume that an embodied virtual agent called Homer is a notebook reseller. A business woman named Marge contacts him telling her purchase interests:

- /1/ “I am interested in notebooks. The CPU speed *must* be at least 2 GHz.
- /2/ The order quantity *should* be *around* 40, and *equally important*, the main memory capacity *should* be the *highest possible*.”

Homer as a clever salesman maintains a preference repository and thus knows that Marge has long-term preferences, too:

- /3/ “Her *favorite* manufacturers are Toshiba and Hewlett Packard, which is *equally important* to her explicit customer preferences.”

Naturally, Homer has his own *vendor* preferences:

- /4/ “I want to *maximize* my profit margin. But since I am a fair dealer for Marge, all her customer preferences are *more important* to me.” ☀

This example emphasizes the need for various components interoperating during the personalization process: *Personalized query composition* has to inductively assemble the query from various sources, including explicit customer preferences entered e.g. through the search mask, long-term customer preferences matching the current situation, and current vendor preferences. Long-term preferences should be detectable by automatic *preference mining* algorithms and be managed intelligently in a *preference repository*.

In our example we carefully differentiated between hard constraints (“must”) and preferences (“should”). Extending the exact-match query paradigm of database query languages by preferences to specify soft constraints is considered a necessity for successful personalization by many researchers. Research on preferences in

databases reaches back quite some time (see e.g. [23], [17], [21]). A powerful framework founded on *preferences as strict partial orders* has been proposed recently by [13]. This constructor-driven approach was implemented by *Preference SQL* ([20]), its first commercial release being available in 1999, and by *Preference XPath* for XML databases ([19]). Skyline queries ([6]) and numerical ranking ([1]) are special cases of [13]. To date an extensive amount of theoretical results on preferences exists (see e.g. [2], [7]).

Since even simple scenarios like above example cannot be modeled by numerical ranking or by skyline queries, more powerful preference frameworks including Pareto preferences and prioritized preferences as introduced in [13] are required. However, skyline queries (hence also more general Pareto queries) over multiple attributes have the wide-spread bad reputation to yield very large query results. On the other hand, if Pareto preferences are defined by use of indifference relations as suggested e.g. in [7], then strict partial orders are not preserved in general. This in turn has a three-fold negative impact as discussed later on: First, the declarative model and fixpoint semantics of preference query languages is invalidated, secondly the optimization of preference queries is severely impeded in two ways: Losing transitivity would invalidate many law from preference relational algebra, hence resulting in poorer query execution plans. The subsequent cost-based optimization phase for choosing best algorithms to evaluate preference selection operators would suffer as well, since efficient algorithms again rely on transitivity of the preference relation.

The rest of this paper is organized as follows: We revisit needed concepts in section 2. In section 3 we show how to customize base preference constructors by so-called d-parameters. Section 4 is dedicated to enhancing preferences with more intuitive semantics (called SV-semantics) regarding indifferent values. In section 5 we analyze the impact of this extension on preference query optimization. Section 6 investigates the issue of preference query cardinalities and exhibits striking impacts on the annoying flooding effect. A discussion of related work in section 7 and a summary and outlook in section 8 conclude this work. Due to space limitations only some selected proofs are given in the Appendix; the remaining ones can be found in [14].

2. BASIC CONCEPTS REVISITED

As a matter of fact most preferences appearing in practical database applications can be modeled by *strict partial orders*. To be self-contained let's revisit needed concepts from [13].

2.1 Preferences

Definition 1 Basic definitions for preferences

Let $A = \{A_1, A_2, \dots, A_k\}$ be a set of attributes A_j with domains $\text{dom}(A_j)$, $1 \leq j \leq k$. The domain of A is defined as $\text{dom}(A) := \times_{A_j \in A} \text{dom}(A_j)$.

- A **preference** P on a set of attributes A is defined as $P = (A, \prec_P)$, where $\prec_P \subseteq \text{dom}(A) \times \text{dom}(A)$ is a *strict partial order* (i.e. irreflexive and transitive). $x \prec_P y$ is interpreted as “I like y better than x ”.
- The **indifference** relation $\parallel_P \subseteq \text{dom}(A) \times \text{dom}(A)$ is defined as: $x \parallel_P y$ iff $\neg(x \prec_P y) \wedge \neg(y \prec_P x)$
- A preference P is a **chain** (synonym **total order**) iff for all $x, y \in \text{dom}(A)$, $x \neq y$: $x \prec_P y \vee y \prec_P x$
- A preference P is an **anti-chain** iff $\prec_P = \emptyset$. The anti-chain on an attribute A is denoted as A^{\leftrightarrow} .

- A preference P is a **weak order**, if negative transitivity holds, i.e. for all $x, y, z \in \text{dom}(A)$:
 $\neg(x \prec_P y) \wedge \neg(y \prec_P z)$ implies $\neg(x \prec_P z)$
- The **maximal** values of $P = (A, \prec_P)$ are defined as: $\text{max}(P) := \{v \in \text{dom}(A) \mid \neg \exists w \in \text{dom}(A): v \prec_P w\}$

Note that in general \parallel_P is reflexive and symmetric, but *not* transitive. If P is a weak order, then \parallel_P is transitive.

To specify a preference $P = (A, \prec_P)$ we allow a great flexibility. Any first order predicate on $\text{dom}(A)$ can be given for \prec_P , possibly using built-in predicates including equality of values ($=$, \neq) and numeric constraints ($<$, \leq , $>$, \geq). But \prec_P may also be written in some programming language. For the ease of use we promote a **constructor-based** approach, distinguishing between **base** preference constructors and **complex** preference constructors. Throughout this paper we will use the following notation:

- Creating a base preference constructor:
`base bname(A, paramlist) {definition of \prec_{P_new} };`
- Defining a base preference P :
`P := bname(actual_A, actual_params);`
- Creating a complex preference constructor:
`complex Pref1 cname Pref2 {definition of \prec_{P_new} };`
- Defining a complex preference P :
`P := actual_Pref1 cname actual_Pref2;`

To illustrate this notation let's repeat some preference constructors presented in [13].

Example 2 Sample use of preference constructors

```
base SCORE(A, f) {x  $\prec_{P\_new}$  y iff f(x) < f(y)};
base HIGHEST(A) {x  $\prec_{P\_new}$  y iff x < y};
base AROUND(A, z) {x  $\prec_{P\_new}$  y iff
  abs(x - z) > abs(y - z)};
complex Pref1  $\otimes$  Pref2 {(x1, x2)  $\prec_{P\_new}$  (y1, y2) iff
  (x1  $\prec_{Pref1}$  y1  $\wedge$  (x2  $\prec_{Pref2}$  y2  $\vee$  x2 = y2))  $\vee$ 
  (x2  $\prec_{Pref2}$  y2  $\wedge$  (x1  $\prec_{Pref1}$  y1  $\vee$  x1 = y1))};
```

The preferences labeled /2/ in Example 1 can be stated as:

$P_1 := \text{AROUND}(\text{quantity}, 40)$;
 $P_2 := \text{HIGHEST}(\text{capacity})$; $P_3 := P_1 \otimes P_2$; ☼

Note that *paramlist* for base constructors is optional, like in HIGHEST. Defining P_1 by AROUND constructs a base preference $P_1 = (\{\text{quantity}\}, \prec_{P_1})$, instantiating the set of attributes A by $\{\text{quantity}\}$ and \prec_{P_1} by \prec_{P_new} . This single-attribute case is also abbreviated as $P_1 = (\text{quantity}, \prec_{P_1})$. Defining P_3 by \otimes constructs a complex preference $P_3 = (\{\text{quantity}, \text{capacity}\}, \prec_{P_3})$, instantiating \prec_{P_3} by \prec_{P_new} which is defined in terms of \prec_{P_1} and \prec_{P_2} .

Definition 2 Preference sub-constructor

C_2 is a *preference sub-constructor* of C_1 , if the definition of $\prec_{C_2_new}$ can be gained from $\prec_{C_1_new}$ by some specializing constraints.

For instance, HIGHEST is a sub-constructor of SCORE: specialize $f(x) := x$ (see Example 2). Since sub-constructors are due to specializing constraints, sub-constructor hierarchies are taxonomic. This observation economizes proof efforts: **Properties proved for a constructor are inherited to all its sub-constructors.**

2.2 Preference Query Languages

In personalized database applications a cooperative query model is needed that supplements the exact-match query of SQL or XPath.

Personalized constraints may be hard constraints (in which case the exact-match model is appropriate) or preferences, i.e. *soft constraints*. Whether preferences can be satisfied depends on the current database contents, capturing the status of the real world. Thus a *match-making* between wishes and reality has to be accomplished. To this purpose the **BMO** query model (“*Best Matches Only*”) has been introduced in [13].

Given a schema $R(A_1: \text{dom}(A_1), \dots, A_m: \text{dom}(A_m))$ we consider a preference $P = (A, \prec_P)$, where $A \subseteq \{A_1, \dots, A_m\}$. For an instance of R let P^R denote the subset preference obtained by restricting P from $\text{dom}(A)$ to $\pi_A(R)$, i.e. the currently available A -values in R .

Definition 3 Preference selection, BMO-size

- a) **Preference selection** $\sigma[P](R)$ is defined as:
 $\sigma[P](R) = \{t \in R \mid t[A] \in \max(P^R)\}$
- b) $t \in \sigma[P](R)$ is a **perfect match** iff $t[A] \in \max(P)$.
- c) $\text{card}(\sigma[P](R))$ is called the **BMO-size** of $\sigma[P](R)$.

$\sigma[P](R)$ retrieves all maximal values from the instance of R . Not all of them are necessarily perfect matches of P . Thus the principle of *query relaxation* is implicit in above definition. Moreover, any non-maximal values of P^R are excluded; hence can be considered as *discarded on the fly*. Thus only best matching tuples are retrieved.

In many personalized e-commerce applications $\sigma[P](R)$ serves as an intelligent pre-selection, which is the basis for further sales negotiations. Therefore it is often important that BMO-sizes come in “handy portions”.

Example 3 A preference query and its BMO result

Continuing Example 2 we evaluate $\sigma[P_3](\text{Sales})$, given this small sales relation:

$\text{Sales}(\text{quantity}, \text{capacity}, \text{notebook}) =$
 $\{(45, 768, 1), (20, 1024, 2), (30, 1024, 3), (45, 512, 4)\}$

Due to the definition of ‘ \otimes ’ in Example 2 notebook 1 is better than 4, because it has the same quantity but a better capacity; 3 is better than 2, but 1 and 3 are indifferent. Thus we get a result with BMO-size 2:

$\sigma[P_3](\text{Sales}) = \{(45, 768, 1), (30, 1024, 3)\}$ ☼

Note that $\sigma[P]$ has been implemented in Preference SQL (as PREFERRING-clause) and in Preference XPath. For brevity we will stick to the algebraic notation $\sigma[P]$.

3. d-PARAMETERS

When dealing with numerical scores, it is a common practice to group ranges of scores together; e.g., forming grades at school, or offering differential price discounts in e-procurement, or setting target deadlines for events to happen (e.g. “payment due within two weeks”), etc. Now we show how to model such real-world practice on top of a given preference constructor.

3.1 Base Preference Constructor SCORE_d

Definition 4 SCORE_d

Given a utility function $f: \text{dom}(A) \rightarrow \mathbb{R}$ and some $d \in \mathbb{R}_{0^+}$, we define for all $v \in \text{dom}(A)$:

$$f_d: \text{dom}(A) \rightarrow \{\text{if } d = 0 \text{ then } \mathbb{R} \text{ else } \mathbb{Z}\}, \text{ where}$$

$$f_d(v) := \{\text{if } d = 0 \text{ then } f(v) \text{ else } \lceil f(v) / d \rceil\}$$

$$\text{base } \text{SCORE}_d(A, f) \{x \prec_{P_{\text{new}}} y \text{ iff } f_d(x) < f_d(y)\};$$

Each preference constructed by SCORE_d is constructible by SCORE and vice versa. Thus due to [9] SCORE_d constructs a weak order. Choosing $d > 0$ effects that values with identical f_d -values become indifferent: $x \parallel_{P_{\text{new}}} y$ iff $f_d(x) = f_d(y)$. In this way a categorical view on numerical scores is achieved. As we will see later on, certain indifferent values can be interpreted as ‘substitutable’ or ‘equally good’.

Now we present several sub-constructors of SCORE_d , focusing on preferences $P = (A, \prec_P)$, where A is a *single* attribute with a *numerical* domain, i.e. $\text{dom}(A) \subseteq \mathbb{R}$.

3.2 Sub-Constructors of SCORE_d

Given $v, \text{low}, \text{up} \in \text{dom}(A)$ we define the *distance* of v from the closed interval $[\text{low}, \text{up}]$ as follows:

$$\text{dist}[\text{low}, \text{up}]: \text{dom}(A) \rightarrow \mathbb{R}_{0^+}$$

$$\text{dist}[\text{low}, \text{up}](v) := \{\text{if } v \in [\text{low}, \text{up}] \text{ then } 0 \text{ else}$$

$$\text{if } v < \text{low} \text{ then } \text{low} - v \text{ else } v - \text{up}\}$$

Given $d \in \mathbb{R}_{0^+}$ we group distances together as follows:

$$\text{dist}_d[\text{low}, \text{up}]: \text{dom}(A) \rightarrow \{\text{if } d = 0 \text{ then } \mathbb{R}_{0^+} \text{ else } \mathbb{N}0\}$$

$$\text{dist}_d[\text{low}, \text{up}](v) := \{\text{if } d = 0 \text{ then } \text{dist}[\text{low}, \text{up}](v)$$

$$\text{else } \lceil \text{dist}[\text{low}, \text{up}](v) / d \rceil\}$$

Definition 5 BETWEEN_d

$$\text{base } \text{BETWEEN}_d(A, [\text{low}, \text{up}]) \{x \prec_{P_{\text{new}}} y \text{ iff}$$

$$\text{dist}_d[\text{low}, \text{up}](y) < \text{dist}_d[\text{low}, \text{up}](x)\};$$

For $d > 0$ a BETWEEN_d preference can be envisioned as a one-dimensional *dart board*: Perfect matches hit the interval $[\text{low}, \text{up}]$ with dist_d being 0, second bests are those with dist_d being 1, and so on. Values with identical dist_d -values become indifferent.

Special cases of this construction are obtained by identifying $\text{low} = \text{up} =: z$ (setting $\text{dist}_d[z, z] =: \text{dist}_d[z]$) and by choosing z as the *finite* infimum or supremum of $\text{dom}(A)$.

Definition 6 AROUND_d, LOWEST_d, HIGHEST_d

$$\text{base } \text{AROUND}_d(A, z)$$

$$\{x \prec_{P_{\text{new}}} y \text{ iff } \text{dist}_d[z](y) < \text{dist}_d[z](x)\};$$

$$\text{base } \text{LOWEST}_d(A)$$

$$\{x \prec_{P_{\text{new}}} y \text{ iff } \text{dist}_d[\text{inf}_A](y) < \text{dist}_d[\text{inf}_A](x)\};$$

$$\text{base } \text{HIGHEST}_d(A)$$

$$\{x \prec_{P_{\text{new}}} y \text{ iff } \text{dist}_d[\text{sup}_A](y) < \text{dist}_d[\text{sup}_A](x)\};$$

Example 4 AROUND_d, LOWEST_d

Let $\text{dom}(\text{age}) = [6, 20] \subseteq \mathbb{R}$, hence $\text{inf}_A = 6$. Further let $d = 2$ and $R(\text{age}) = \{7, 8, 11, 13\}$.

- Let $P := \text{AROUND}_2(\text{age}, 10)$: Since $\text{dist}_2[10](v) = 1$ if $v \in \{8, 11\}$ and $\text{dist}_2[10](v) = 2$ if $v \in \{7, 13\}$, we get
 $\sigma[P](R) = \{8, 11\}$.
- Let $P := \text{LOWEST}_2(\text{age})$: Since $\text{dist}_2[6](7) = 1 = \text{dist}_2[6](8)$, $\text{dist}_2[6](11) = 3$ and $\text{dist}_2[6](13) = 4$, we get $\sigma[P](R) = \{7, 8\}$.
☼

It should be emphasized that above parameter d defines a symmetrical distance from the perfect hit. If, however, the application

semantics suggests an *unsymmetrical* treatment, d has to be replaced properly by two parameters d_1 and d_2 .

Now we study *categorical* data, not requiring any numerical operations for defining the preference order.

Definition 7 **LAYERED_m**

Let $L = (L_1, \dots, L_{m+1})$, $m \geq 0$, be an ordered list of sets with the following properties:

- L is a partition of $\text{dom}(A)$.
- Exactly m out of these $m+1$ sets are given as finite enumerations of values from $\text{dom}(A)$.
- The remaining set is specified as ‘*other values*’.

We define a function $\text{layer}: \text{dom}(A) \rightarrow \mathbb{N}$ as follows:

$$\text{for } i \in \{1, \dots, m+1\}, \text{ for all } v \in L_i: \text{layer}(v) := i.$$

$$\text{base LAYERED}_m(A, L) \{x <_{p_{\text{new}}} y \text{ iff } \text{layer}(y) < \text{layer}(x)\};$$

LAYERED_m is a sub-constructor of SCORE_d, specializing $d = 0$ and $f(v) = -\text{layer}(v)$. Relating LAYERED_m to constructors on categorical attributes like POS/POS and POS/NEG as defined in [13] yields:

Proposition 1 **Sub-constructors of LAYERED_m**

- a) POS/POS is a sub-constructor of LAYERED_m:
 $m = 2$, $L = (\text{POS}_1\text{-set}, \text{POS}_2\text{-set}, \text{'other values'})$
- b) POS/NEG is a sub-constructor of LAYERED_m:
 $m = 2$, $L = (\text{POS-set}, \text{'other values'}, \text{NEG-set})$
- c) POS is a sub-constructor of LAYERED_m:
 $m = 1$, $L = (\text{POS-set}, \text{'other values'})$
- d) NEG is a sub-constructor of LAYERED_m:
 $m = 1$, $L = (\text{'other values'}, \text{NEG-set})$
- e) ANTI-CHAIN is a sub-constructor of LAYERED_m:
 $m = 0$, $L = (\text{'other values'})$

The categorical view of numerical data by BETWEEN_d for $d > 0$ is reflected by this relationship to LAYERED_m: Defining $\text{layer}(v) := \text{dist}_d[\text{low}, \text{up}](v) + 1$ for all $v \in \text{dom}(A)$, then BETWEEN_d maps values onto $m = \max\{\text{dist}_d[\text{low}, \text{up}](\text{inf}_A), \text{dist}_d[\text{low}, \text{up}](\text{sup}_A)\}$ layers.

These results plus relationships, which can be proved for our numerical base constructors and the EXPLICIT constructor from [13], are visualized in Figure 1.

This constructor repertoire can be extended if required by the application semantics (see [14]). Now we turn our attention to complex constructors.

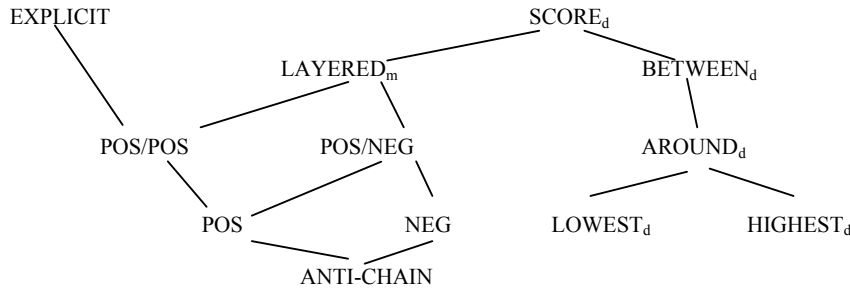


Figure 1. Base preference sub-constructor hierarchy

4. PREFERENCES WITH SV-SEMANTICS

A distinctive feature of strict partial orders is that indifferent values may exist. Often people have a strong opinion about better-than relationships for a selected choice of options, but without being complete: For some values they don’t mind or some values may be equally good for them in a given scenario, etc. Thus the freedom of not having to specify a total order is not a bug, but rather a valuable asset for real world modeling.

4.1 SV-Relations

Our interest concentrates on a *debatable* effect caused by indifferent values. To this end we will study the Pareto constructor ‘ \otimes ’ as stated in Example 2 more closely.

Example 5 **The impact of indifferent values**

Given $\text{dom}(A_i) = [-10, 10] \subseteq \mathbb{R}$ for $i \in \{1, 2, 3\}$, we consider $P_1 := \text{AROUND}_0(A_1, 0)$, $P_2 := \text{LOWEST}_0(A_2)$, $P_3 := \text{HIGHEST}_0(A_3)$, $P_4 := P_1 \otimes P_2$ and $P_5 := P_3 \otimes P_4$.

For $v = (-5, 3, 4)$ and $w = (5, 1, 8) \in \times \text{dom}(A_i)$ we get:

- $-5 \parallel_{P_1} 5, \quad 3 <_{P_2} 1, \quad 4 <_{P_3} 8$
- $(-5, 3) \parallel_{P_4} (5, 1), \quad v \parallel_{P_5} w$

But there are situations where some indifferent values should be treated as *substitutable* (synonym: *equally good*): For P_1 a mismatch of +5 or -5 from the perfect match 0 might be equally good for us and we would not mind, if +5 and -5 are substituted for each other. If so, then it is reasonable to re-assess the relationship of v with w in P_4 . The *intuitive feeling* is that we would rather expect:

$$(-5, 3) <_{P_4} (5, 1)$$

This is because w and v , though not syntactically equal are substitutable wrt. P_1 and w is better than v wrt. P_2 . This in turn would change the rating for v and w in P_5 : $v <_{P_5} w$ ☼

The challenge now becomes to find out, how the definition of ‘ \otimes ’ must be adapted. To this end we claim that the *intuitive semantics of substitutability* should be as follows:

- a) Values x and y can only be substitutable, if both are indifferent.
- b) If x is better than z and x can be substituted by y , then y should be better than z as well.
- c) Dually, if z is better than x and x can be substituted by y , then z should be better than y as well.
- d) Substitutability should be an equivalence relation.

Thus not all indifferent values need to be substitutable! E.g., point d) is no consequence from a), since \parallel_P is not transitive in general. This is the very reason, why simply replacing ‘ $x_1 = y_1$ ’ by ‘ $x_1 \parallel_{P_1} y_1$ ’ and ‘ $x_2 = y_2$ ’ by ‘ $x_2 \parallel_{P_2} y_2$ ’ in the definition of ‘ \otimes ’ is semantically not justified in general. Technically this flaw impacts that in general ‘ \otimes ’ does not preserve the strict partial order property ([7]).

Formalizing our stated intuition for substitutable values yields:

Definition 8 SV-relation

Given $P = (A, <_P)$, \cong_P is called *substitutable values relation (SV-relation)* for short) iff for all $x, y \in \text{dom}(A)$:

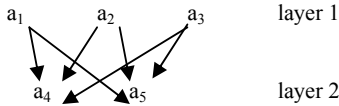
- a) $x \cong_P y$ implies $x \parallel_P y$
- b) $x \cong_P y \wedge \exists z : z <_P x$ implies $z <_P y$
- c) $x \cong_P y \wedge \exists z : x <_P z$ implies $y <_P z$
- d) \cong_P is reflexive, symmetric and transitive.

Indifferent values that are not substitutable are called *alternative* values, which cannot be substituted for each other.

The crucial question is of course how complex preference constructors can be adapted to preserve this SV-semantics. But let’s do some case studies before.

Example 6 SV-relations

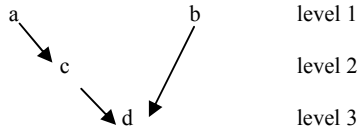
Case study 1: $P := \text{POS}(A, \{a_1, a_2, a_3\})$, where $\text{dom}(A) = \{a_1, a_2, a_3, a_4, a_5\}$, has this ‘better-than’ graph (see [13]):



Given $x, y \in \text{dom}(A)$, SV-relations are e.g.:

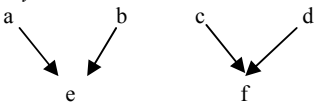
- $x \cong_P y$ iff $\text{layer}(x) = \text{layer}(y)$ // regular case
- $x \cong_P y$ iff $x = y \vee x, y \in \{a_1, a_2\} \vee x, y \in \{a_4, a_5\}$
- $x \cong_P y$ iff $x = y$ // trivial case

Case study 2: Consider an EXPLICIT preference P with this ‘better-than’ graph:



There is only the trivial SV-relation: $x \cong_P y$ iff $x = y$. Because $c <_P a$ and $\neg(c <_P b)$, a and b are *alternatives*. Also, c and b are *alternatives*, since $c <_P a$ but $\neg(a <_P b)$.

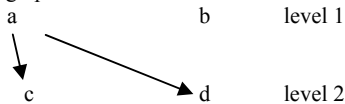
Case study 3: Consider P with this ‘better-than’ graph:



- $x \cong_P y$ iff $x = y \vee x, y \in \{a, b\} \vee x, y \in \{c, d\}$

Negative transitivity is violated, but \cong_P is non-trivial.

Case study 4: Consider an EXPLICIT preference P with this ‘better-than’ graph:



- $x \cong_P y$ iff $x = y \vee x, y \in \{c, d\}$

This is a non-trivial SV-relation. Note that negative transitivity is violated here; hence P is no weak order. ☼

Proposition 2 Properties of SV-relations

- a) ‘=’ is an SV-relation for each preference P (called **trivial SV-relation**).
- b) If P is the anti-chain A^{\leftrightarrow} , each partition of $\text{dom}(A)$ defines an SV-relation.

We just have seen cases, where P is not a SCORE_d preference, but has a non-trivial SV-relation. Also we have seen instances of alternative values, which are not substitutable (not equally good). For SCORE_d it turns out that the full indifference relation is a valid SV-relation.

Proposition 3 Regular SV-relation for SCORE_d

Given a SCORE_d preference P, let’s define for all $x, y \in \text{dom}(A)$: $x \cong_P y$ iff $x \parallel_P y$

- a) \cong_P is an SV-relation (called **regular SV-relation**).
- b) If P is not a chain, then \cong_P may be non-trivial.

This result says that *all* values with equal f_d -values can be designated as ‘substitutable’ or ‘equally good’. However, this nice behavior does not hold for other constructors like EXPLICIT, which are no weak orders.

4.2 Enriching the Preference Definition

From now on we will enrich our definition of preferences to accommodate the semantics of SV-relations.

Definition 9 Preferences with SV-semantics

Enriching Definition 1 a) we use the following notation:

- A preference P with an SV-relation \cong_P is denoted as:
 $P = (A, <_P, \cong_P)$
- Each base constructor receives one additional parameter for the SV-relation.

Let’s explore the impact of SV-relations for *inductive* preference construction. These aspects are important:

- Consider a base preference $P_i = (A_i, <_{P_i}, \cong_{P_i})$. Then \cong_{P_i} **does not** affect $<_{P_i}$ itself, but expresses that it is admissible to substitute \cong_{P_i} -values for each other.
- A complex constructor C, using P_i *recursively* in its definition for $<_{P_{\text{new}}}$, can make use of \cong_{P_i} . This **does** affect $<_{P_{\text{new}}}$! Moreover, if an SV-relation $\cong_{P_{\text{new}}}$ can be determined for P, then C can *recursively* be used itself for defining more complex preferences.

4.3 Complex Constructors with SV-Semantics

We present in detail, how Pareto and prioritized construction can be enriched by SV-semantics. A Pareto preference combines *equally important* preferences P_1 and P_2 , whereas the combination of P_1 with a *less important* P_2 is modeled by a prioritized preference.

Definition 10 Pareto and prioritized constructors

We assume $P_1 = (A_1, <_{P_1}, \cong_{P_1})$ and $P_2 = (A_2, <_{P_2}, \cong_{P_2})$.

a) **Pareto constructor ‘ \otimes ’**

- A_1 and A_2 don't overlap:
 $\text{complex } P_1 \otimes P_2 \{(x_1, x_2) <_{p_new} (y_1, y_2) \text{ iff}$
 $(x_1 <_{p_1} y_1 \wedge (x_2 <_{p_2} y_2 \vee x_2 \cong_{p_2} y_2)) \vee$
 $(x_2 <_{p_2} y_2 \wedge (x_1 <_{p_1} y_1 \vee x_1 \cong_{p_1} y_1));$
 $(x_1, x_2) \cong_{p_new} (y_1, y_2) \text{ iff } x_1 \cong_{p_1} y_1 \wedge x_2 \cong_{p_2} y_2\};$
- Otherwise: Identify overlapping attributes above.

b) **Prioritized constructor ‘ $\&$ ’**

- A_1 and A_2 don't overlap:
 $\text{complex } P_1 \& P_2 \{(x_1, x_2) <_{p_new} (y_1, y_2) \text{ iff}$
 $x_1 <_{p_1} y_1 \vee (x_1 \cong_{p_1} y_1 \wedge x_2 <_{p_2} y_2);$
 $(x_1, x_2) \cong_{p_new} (y_1, y_2) \text{ iff } x_1 \cong_{p_1} y_1 \wedge x_2 \cong_{p_2} y_2\};$
- Otherwise: Identify overlapping attributes above.

Note that \cong_{p_new} is defined recursively using \cong_{p_1} and \cong_{p_2} .

Example 7 Pareto preferences with SV-semantics

Let's revisit Example 5 introducing SV-relations, e.g.:

$$P_1 := \text{AROUND}_0(A_1, 0, \cong_{p_1}) \text{ where } \cong_{p_1} \text{ is regular}$$

$$P_2 := \text{LOWEST}_0(A_2, '=') , P_3 := \text{HIGHEST}_0(A_3, '=')$$

$$P_4 := P_1 \otimes P_2, P_5 := P_3 \otimes P_4$$

For $v = (-5, 3, 4)$ and $w = (5, 1, 8)$ we now can state:

- Since \cong_{p_1} does not change $<_{p_1}$ we get as before:
 $-5 \parallel_{p_1} 5, 3 <_{p_2} 1, 4 <_{p_3} 8$
- Since $-5 \cong_{p_1} 5$ we now get: $(-5, 3) <_{p_4} (5, 1)$
- Since $4 <_{p_3} 8, (-5, 3) <_{p_4} (5, 1)$ we now get: $v <_{p_5} w$

This is our intuitively expected result, provided that \cong_{p_1} holds in the given application situation. ☀

Example 8 Prioritization with SV-semantics

Assuming $P_1 := \text{POS}(\text{category}, \{\text{luxury}, \text{sport}\}, '=')$ and $P_2 := \text{POS}(\text{color}, \{\text{red}\}, '=')$, let's consider $P := P_1 \& P_2$.

Given $R = \{(\text{luxury}, \text{black}), (\text{sport}, \text{green}), (\text{sport}, \text{red})\}$, we get the following results:

- Since $\text{sport} = \text{green} <_{p_2} \text{red}$:
 $(\text{sport}, \text{green}) <_p (\text{sport}, \text{red})$
- Since $\text{luxury} \neq \text{sport}$: $(\text{luxury}, \text{black}) \parallel_p (\text{sport}, \text{red})$
 $(\text{luxury}, \text{black}) \parallel_p (\text{sport}, \text{green})$

Thus $\sigma[P](R) = \{(\text{sport}, \text{red}), (\text{luxury}, \text{black})\}$.

Now suppose that 'luxury' and 'sport' shall be substitutable: $P_1 := \text{POS}(\text{Category}, \{\text{luxury}, \text{sport}\}, \cong_{p_1})$ where \cong_{p_1} is regular. This changes the picture:

- As before: $(\text{sport}, \text{green}) <_p (\text{sport}, \text{red})$
- Since $\text{luxury} \cong_{p_1} \text{sport}$ we get:
 $\text{black} <_{p_2} \text{red}$ implies $(\text{luxury}, \text{black}) <_p (\text{sport}, \text{red})$
 $\text{black} \parallel_{p_2} \text{green}$ implies $(\text{luxury}, \text{black}) \parallel_p (\text{sport}, \text{green})$

Therefore $\sigma[P](R) = \{(\text{sport}, \text{red})\}$, offering less alternatives than before. ☀

The following main theorem is the result of our semantically well-founded approach.

Theorem 1 Preservation of strict partial order

Given $P_1 = (A_1, <_{p_1}, \cong_{p_1})$ and $P_2 = (A_2, <_{p_2}, \cong_{p_2})$, consider $P := P_1 \otimes P_2$ and $\tilde{P} := P_1 \& P_2$, respectively. Then $P = (A_1 \cup A_2, <_P, \cong_P)$ is a preference with SV-semantics, i.e.:

- $<_P$ is a strict partial order on $A_1 \cup A_2$.
- \cong_P is an SV-relation for $<_P$.

Please refer to Definition 1 for dealing with overlapping A_1 and A_2 ; $<_P$ and \cong_P are due to Definition 10a, b.

Consequently *inductive* preference construction, being essential for personalized query composition, preserves strict partial order, too! Moreover, obeying to SV-semantics is the most general approach to preserve strict partial order for '⊗' and '&'.

Theorem 2 Further properties of '⊗' and '&'

Any relaxation of SV-semantics for Pareto or prioritized construction does not preserve strict partial order.

In [13] we reported an interesting correlation between prioritization and grouping for trivial SV-relations. Now we extend this observation to arbitrary SV-relations.

Definition 11 Grouped preference

Given $P = (B, <_P, \cong_P)$ and the anti-chain $A^{\leftrightarrow} = (A, \emptyset, \cong_A)$, $A^{\leftrightarrow} \& P$ is called a *grouped preference*. As a synonym we also write: **P groupby A**

Due to Proposition 2b, \cong_A can be any partition of $\text{dom}(A)$. If \cong_A is trivial, then grouping is done wrt equal A-values. Otherwise a sophisticated grouping effect is achieved by building groups wrt substitutable A-values.

Grouped preferences appear in skyline queries ([6]). If DIFF attributes are given, then a skyline preference can be characterized as $P := (P_1 \otimes \dots \otimes P_n)$ groupby DIFF, where each P_i is a LOWEST_0 or HIGHEST_0 preference. Due to Figure 1 and Theorem 1 we can extend the class of skyline preferences without violating strict partial orders.

Definition 12 Skyline preference with SV-semantics

A *skyline preference* P is defined as

$$P := (P_1 \otimes \dots \otimes P_n) \text{ groupby DIFF}$$

where $P_i := \text{HIGHEST}_{\text{di}}(A_i, \cong_{P_i})$ or $P_i := \text{LOWEST}_{\text{di}}(A_i, \cong_{P_i})$, $i \in \{1, \dots, n\}$, and $\text{DIFF}^{\leftrightarrow} = (\text{DIFF}, \emptyset, \cong_{\text{DIFF}})$.

A *numerical preference* is constructed by the 'rank_F' constructor from SCORE_{di} preferences P_1, \dots, P_n , applying a combining function F (for details see [13]). In [14] we show how to adapt 'rank_F' to SV-semantics. There it is also shown, how the complex constructors *intersection* ('♦'), *disjoint union* ('+') and *linear sum* ('⊕') presented in [13] can be enriched by SV-semantics.

4.4 Expressiveness Results

The next result suggests that a rich repertoire of complex constructors with SV-semantics should be supported.

Theorem 3 Expressiveness of complex constructors

- Pareto is no sub-constructor of rank_F and vice versa. (Skyline preferences cannot be expressed by rank_F.)
- Pareto is no sub-constructor of '&' and vice versa.
- '&' is no sub-constructor of 'rank_F' and vice versa. (Grouped preferences are not expressible by rank_F.)

Thus personalization that solely relies on numerical ranking is of rather limited expressiveness.

Example 9 Deeply personalized query

Let's get back to our motivating Example 1. Personalized preference composition has to inductively construct a complex preference P from the statements labeled /2/ (customer preferences), /3/ (long-term preferences from the repository) and /4/ (vendor preferences). Using ' \otimes ' to model equal importance and '&' for ordered importance we can state: $P := (P_{\text{customer}} \otimes P_{\text{repository}}) \& P_{\text{vendor}}$

Our sales story leaves open the issues of how to categorize numerical data (i.e. choices of d-parameters) and of substitutability (i.e. choices of SV-relations). This knowledge can be gained in manifold ways, e.g. by interviewing the customer, from personalized long-term knowledge in the preference repository, from defaults, etc. For our sales story we assume this scenario:

- Marge lets Homer know that a deviation up to +3 or -3 from the stated quantity doesn't really worry her.
- From the preference repository it is known that Marge doesn't mind capacity differences up to 256 Mbytes and that Toshiba and HP are equally good manufacturers for her that can be substituted.

Then we can complete the definition of P e.g. as follows:

```
P_customer := AROUND3(quantity, 40, 'regular')  $\otimes$ 
             HIGHEST256(capacity, 'regular');
P_repository := POS(make, {'Toshiba', 'HP'}, 'regular');
P_vendor := HIGHEST0(profit_margin, 'regular');
```

Using Preference XPath syntax our entire sales scenario, including the hard customer constraint labeled /1/, can be expressed declaratively by one query statement:

```
/Notebook
[CPU_speed >= 2.0]
#[ (quantity around (3, 40, 'reg') and
   capacity highest(256, 'reg') and
   make in(('Toshiba', 'HP'), 'reg'))
  prior_to profit_margin
  highest(0, 'reg')]#
```

Note that hard constraints are scoped by "[...]", preferences by "#[...]#"; Pareto is 'and', prioritization is 'prior to', POS is 'in'. ☼

5. QUERY OPTIMIZATION ISSUES

After we have discussed our semantic intuition about substitutable values let's explore its implications for the optimization of deeply personalized preference queries.

5.1 Preference Algebra

In [13] we have identified many algebraic laws amongst preferences with trivial SV-relations. The subsequent main theorem is the key that these laws carry over to non-trivial SV-relations.

Definition 13 SV-order

Given $P = (A, <_P, \cong_P)$, let A/\cong_P denote the set of equivalence classes of $\text{dom}(A)$ over \cong_P . For all $X, Y \in A/\cong_P$ we define:

- $X <_{[P]} Y$ iff $\forall x \in X, \forall y \in Y: x <_P y$
- $X \cong_{[P]} Y$ iff $\forall x \in X, \forall y \in Y: x \cong_P y$

Then $[P] = (A/\cong_P, <_{[P]})$ is called **SV-order**.

Theorem 4 Every SV-order is a strict partial order

Moreover, $\cong_{[P]}$ is the trivial SV-relation, i.e. equality of equivalence classes.

Consequently all preference algebra laws given in [13] hold for any SV-order, characterizing preferences with non-trivial SV-relations. For more details see [14].

5.2 Preference Relational Algebra

Declarative query languages with hard constraints like SQL or XPath can be seamlessly extended by preference selection towards Preference SQL or Preference XPath, supporting the BMO query model. Implementing such preference query languages can be accomplished by loose coupling and query rewriting ([20]). For higher performance tight coupling is required, integrating the preference selection operator directly into the database kernel and extending relational algebra towards a *preference relational algebra*.

To date many transformation laws for preference relational algebra are known ([18], [7], [10]). For illustration, here are two laws, given a preference $P = (A, <_P)$:

- *Push preference over Cartesian product:*
If $A \subseteq \text{attr}(R)$, then $\sigma[P](R \times S) = \sigma[P](R) \times S$
- *Push preference over union:* If $A \subseteq \text{attr}(R)$, then $\sigma[P](R \cup S) = \sigma[P](\sigma[P](R) \cup \sigma[P](S))$

The proofs for such laws sometimes critically rely on the transitivity of the given preference order. Since according to Theorem 4 strict partial order property is preserved for arbitrary SV-relations, **all such transformation laws carry over to preferences with SV-semantics.**

6. OBSERVATIONS ON BMO-SIZES

The BMO query model avoids the notorious 'empty result' effect and reduces the 'flooding' effect. But the latter can exhibit two opposite effects from a personalization point of view: The BMO-size is too large, i.e. fewer alternatives would be preferable. Or, the BMO-size is too small, i.e. more alternatives would be preferable.

Our constructor-based approach offers two semantically guided opportunities to influence the BMO-sizes of preferences queries: choosing the d-parameters of base constructors and choosing the SV-relations. We will investigate in more detail both options subsequently.

6.1 BMO-Sizes for Base Preferences

Let's start with a basic property of SCORE_d for varying d . At first guess one might conjecture that $d_1 \leq d_2$ implies $\sigma[\text{SCORE}_{d_1}(A, f)](R) \subseteq \sigma[\text{SCORE}_{d_2}(A, f)](R)$, however:

Proposition 4 BMO-sizes of SCORE_d are non-monotonic in d .

Note that Proposition 4 applies to all sub-constructors of SCORE_d having d-parameters!

Thus the BMO-size cannot be influenced deterministically by d ; it also depends on the data distribution (a counterexample can be found in the Appendix). But statistically speaking, it is reasonable to assume the following *rule of thumb*: **Choosing a larger d tends to increase the BMO-size of $\sigma[\text{SCORE}_d](R)$.**

Table 1. Sample BMO-sizes for base preferences with varying parameter d

	d = 0	d = 5	d = 9	d = 15	d = 25	d = 30	d = 50
BETWEEN_d	1	20	20	27	39	56	97
AROUND_d	3	3	3	26	41	49	74
LOWEST_d	4	6	4	6	9	6	9

Note that the choice of d only impacts BMO-sizes, if there are *no* perfect matches. The BMO-sizes observed in the next example nicely demonstrate this non-monotonic behavior and support our rule of thumb stated as well.

Example 10 BMO-sizes for varying d-parameters

We used a data set taken from a real-life application. The COSIMA^{B2B} prototype ([15]), being a sophisticated sales agent for e-procurement portals, works with an XML-based electronic product catalog for storage and transport boxes and waste containers. There are attributes on numerical domains (like length, height, width, weight) and on categorical domains (like color, type of material) as well. This sample catalog comprises about 1000 such sales objects. Using Preference XPath we executed a series of test queries for different settings of d. Characteristic patterns of observed BMO-sizes for base preference queries on numerical domains are given in Table 1. ☀

6.2 BMO-Sizes for Complex Preferences

Choosing the right SV-relations is an important factor in influencing BMO-sizes for complex constructors, in particular for Pareto and prioritized preferences. The following classification of SV-relations is important.

Definition 14 More liberal than ($\cong_2 \succ_P \cong_1$)

Given SV-relations \cong_1 and \cong_2 for a preference P, \cong_2 is *more liberal than* \cong_1 ($\cong_2 \succ_P \cong_1$) if:
 $\cong_2 \succ_P \cong_1$ iff $(\forall x, y \in \text{dom}(A): x \cong_1 y \text{ implies } x \cong_2 y)$

If $\cong_2 \succ_P \cong_1$, then each substitutable value of \cong_1 is also substitutable in \cong_2 , but \cong_2 may have additional ones, which is regarded as a more liberal behavior.

Proposition 5 Properties of \succ_P

- a) \succ_P is a non-strict partial order on the set of all SV-relations of a preference P.
- b) If P is constructed by SCORE_d, then the regular (trivial) SV-relation is the greatest (smallest) element of \succ_P .

Example 11 Liberality of SV-relations

Given $\text{dom}(A) = \{a_1, a_2, a_3, a_4, b_1, b_2, b_3\}$, for $P := \text{POS}(A, \{a_1, a_2, a_3, a_4\}, \cong_P)$ we know that:

- $\text{layer}(x) = 1$ iff $x \in \{a_1, a_2, a_3, a_4\}$.
- $\text{layer}(x) = 2$ iff $x \in \{b_1, b_2, b_3\}$.

Then choices for \cong_P are e.g.:

- $x \cong_1 y$ iff $\text{layer}(x) = \text{layer}(y)$ // regular case
- $x \cong_2 y$ iff $x, y \in \{a_1, a_2\} \vee x, y \in \{a_3, a_4\} \vee x, y \in \{b_1, b_2, b_3\}$
- $x \cong_3 y$ iff $x, y \in \{a_1, a_3\} \vee x, y \in \{a_2, a_4\} \vee x, y \in \{b_1, b_2, b_3\}$

- $x \cong_4 y$ iff $x = y$ // trivial case

We get: $\cong_1 \succ_P \cong_2, \cong_1 \succ_P \cong_3, \cong_2 \succ_P \cong_4, \cong_3 \succ_P \cong_4$ ☀

The following main theorem supports the common experience that accepting more things as substitutable typically ends up in having fewer alternative choices left.

Theorem 5 Monotonicity of BMO-sizes for \otimes , &

Let $P_1 = (A_1, <_{P_1}, \cong_1)$, $P_1^* = (A_1, <_{P_1}, \cong_1^*)$, differing only wrt. the SV-relation, likewise $P_2 = (A_2, <_{P_2}, \cong_2)$, $P_2^* = (A_2, <_{P_2}, \cong_2^*)$.

- a) $\sigma[P_1 \otimes P_2](R) \subseteq \sigma[P_1^* \otimes P_2^*](R)$
if $\cong_1 \succ_{P_1} \cong_1^*$ and $\cong_2 \succ_{P_2} \cong_2^*$
- b) $\sigma[P_1 \& P_2](R) \subseteq \sigma[P_1^* \& P_2](R)$ if $\cong_1 \succ_{P_1} \cong_1^*$
- c) $\sigma[P_1 \& P_2](R) \subseteq \sigma[P_1 \otimes P_2](R)$

Theorem 6 Smallest / largest BMO-sizes for \otimes , &

Consider SCORE_d preferences $P_1 = (A_1, <_{P_1}, \cong_{P_1})$ and $P_2 = (A_2, <_{P_2}, \cong_{P_2})$. Varying \cong_{P_1} and \cong_{P_2} we have:

- a) *Trivial* \cong_{P_1} and \cong_{P_2} yield *largest* BMO-sizes for $\sigma[P_1 \otimes P_2](R)$ and $\sigma[P_1 \& P_2](R)$, resp.
- b) *Regular* \cong_{P_1} and \cong_{P_2} yield *smallest* BMO-sizes for $\sigma[P_1 \otimes P_2](R)$ and $\sigma[P_1 \& P_2](R)$, resp.

Example 12 BMO-sizes for varying SV-relations

We measured the BMO-sizes of the following queries for different choices of z, low, up, set₁, set₂ and set₃, evaluated against the same test collection as in Example 10:

- $Q_\alpha = \sigma[\text{LOWEST}_{d_1}(\text{height}, \cong_1) \otimes \text{HIGHEST}_{d_2}(\text{length}, \cong_2)](\text{test_coll})$
- $Q_\beta = \sigma[\text{AROUND}_{d_1}(\text{height}, z, \cong_1) \otimes \text{BETWEEN}_{d_2}(\text{length}, [\text{low}, \text{up}], \cong_2)](\text{test_coll})$
- $Q_\gamma = (\sigma[\text{POS}(\text{color}, \text{set}_1, \cong_1) \otimes \text{POS}(\text{material}, \text{set}_2, \cong_2)] \& \text{HIGHEST}_{d_3}(\text{height}, \cong_3)](\text{test_coll})$
- $Q_\delta = \sigma[\text{POS}(\text{color}, \text{set}_1, \cong_1) \otimes \text{NEG}(\text{material}, \text{set}_2, \cong_2) \otimes \text{AROUND}_{d_3}(\text{height}, \cong_3) \otimes \text{HIGHEST}_{d_4}(\text{length}, \cong_4) \otimes \text{HIGHEST}_{d_5}(\text{width}, \cong_5) \otimes \text{BETWEEN}_{d_6}(\text{weight}, [\text{low}, \text{up}], \cong_6)](\text{test_coll})$

In Table 2 we present selected test runs with characteristic effects concerning the flooding effect. The parameter rel_d indicates an equal proportion of the domain size. E.g., in Q_α d_1 and d_2 are chosen such that $(d_1 * 100) / (\text{sup}_{\text{height}} - \text{inf}_{\text{height}}) = (d_2 * 100) / (\text{sup}_{\text{length}} - \text{inf}_{\text{length}}) = \text{rel}_d$. ☀

The observed drops of BMO-sizes from trivial SV-relations to regular SV-relations are quite striking. In particular, looking at Q_δ in Table 2 the often heard claim that Pareto queries are inherently

Table 2. Sample BMO-sizes for complex preferences with varying SV-relations

Q_a	$rel_d = 0\%$	$rel_d = 5\%$	$rel_d = 10\%$	$rel_d = 15\%$	$rel_d = 20\%$	$rel_d = 30\%$
trivial SVs	8	10	10	28	51	116
regular SVs	8	2	2	2	1	1

Q_b	$rel_d = 0\%$	$rel_d = 5\%$	$rel_d = 10\%$	$rel_d = 15\%$	$rel_d = 20\%$	$rel_d = 30\%$
trivial SVs	4	4	15	33	51	101
regular SVs	4	4	4	5	12	19

Q_c	$rel_d = 0\%$	$rel_d = 5\%$	$rel_d = 10\%$	$rel_d = 15\%$	$rel_d = 20\%$	$rel_d = 30\%$
trivial SVs	48	48	62	67	72	104
regular SVs	8	8	10	14	14	24

Q_g	$rel_d = 0\%$	$rel_d = 5\%$	$rel_d = 10\%$	$rel_d = 15\%$	$rel_d = 20\%$	$rel_d = 30\%$
trivial SVs	388	476	519	502	545	519
regular SVs	84	44	35	31	23	5

prone to flooding seems to be refuted. The transition to equivalence classes in Theorem 4 algebraically explains this phenomenon. Note that a personalized query often has some hard constraints in addition to preferences (cmp. our Example 9), yielding even smaller query results.

7. DISCUSSION AND RELATED WORK

Preference queries with SV-semantics achieve a breakthrough regarding several synergetic issues:

1. Intuitive semantics of preference modeling:

Indifferent values are an essential feature of strict partial order preferences. Introducing the semantically well-founded notion of SV-relations we have partitioned indifferent values into substitutable (i.e. equally good) and alternative values. That this is the proper way to go is reflected by Theorem 1 and Theorem 2. For personalized query composition it is the key to inductively construct complex BMO preference queries closed under strict partial orders. For proper choices of SV-relations and d-parameters also the *flooding effect* becomes much less of an issue for personalized database queries.

In many applications BMO-results are *intelligent pre-selections*, which have to be refined afterwards. In our e-procurement example the virtual salesman Homer would apply his *presentation preferences* to decide which item to pick first from the BMO-set to start the sales negotiation ([15]). This decision depends on sales strategies and psychology, and may even involve non-transitive arguments like majority voting. (A survey on the decades-long discussion in decision theory on the sense or nonsense of non-transitive preferences can be found in [8].)

2. Formal semantics of preference query languages:

The theory of subsumption lattices ([21]) guarantees both the existence of a model theory and of a corresponding fixpoint theory for BMO queries, if strict partial order is maintained. Thus according to Theorem 1 general Pareto and prioritized preference queries preserve this crucial requirement.

3. Efficiency of preference query evaluation:

- Many transformation laws required for *heuristic optimization* of preference relational algebra rely on the transitivity of the underlying preference relation. According to Theorem 4 SV-semantics preserves this important requirement.

- Also *cost-based optimization* of evaluation algorithms for the preference selection operator (see e.g. the BNL algorithm in [6]) benefits from SV-semantics, since again transitivity is important. Moreover, personalization with a rich repertoire of preference constructors enables the chance to implement specialized efficient evaluation algorithms for each constructor (see e.g. [10]).
- Smaller BMO-sizes (see Theorems 5 and 6) often coincide with faster query evaluation, in particular if the heuristics of ‘*push preference*’ for preference relational algebra is applied.

The work of Chomicki ([7]) contains a fine survey on preference research. He investigates preference queries under the BMO model, calling it the ‘winnow’ operator. His definitions relax the strict partial order semantics of preferences. In particular, Pareto and prioritized preferences are defined using indifference instead of SV-relations, which fails to preserve strict partial orders ([7], theorem 4.14). As reported in [7], some transformation laws of preference relational algebra are invalidated, if transitivity is dropped. This negative impact on heuristic query optimization is avoided by SV-semantics.

The importance of personalization in database queries is also emphasized in [22], proposing a preference model relying on scores and numerical ranking. According to Theorem 3 adding more preference functionality as announced makes much sense (see e.g. [4] describing a personalized application that integrates numerical ranking with linguistic variables and categorical preferences). The integration of personalization and database queries with the use of structured user profiles proposed in [22] has been supported by Preference SQL and Preference XPath for sophisticated applications ([16], [15]). There persistent preference repositories can be queried e.g. by Preference XPath to find best-matching preferences for a given situation ([11]). Efficient preference mining algorithms ([12]) can feed their findings into the repository.

SV-semantics and d-parameters offer a novel means to combat the infamous flooding effect. So far in literature it has been criticized that Pareto preferences are impractical, because BMO-sizes get too large for increasing numbers of attributes. Empirical and analytical studies for skyline queries seemingly support that view (see e.g. [6], [5], [3]). But such investigations did not explore the full picture; instead only the worst case being $d = 0$ and trivial SV-relations has been investigated. Pareto preferences over categorical preferences haven’t been considered either. However, we showed that proper choices of d and the SV-relations offer a semantically

guided way to influence BMO-sizes, hence controlling the flooding effect. Naturally, just like in the conventional relational model there are always queries with large result cardinalities. Another approach to address the flooding problem is the Top-k query model. As proved in [7], Top-k and BMO can be combined consistently.

8. SUMMARY AND OUTLOOK

Personalization of database queries requires a semantically rich, easy to handle and flexible preference model for query composition. The constructor-driven foundation of preferences as strict partial orders serves all these requirements. In this paper we have extended this approach in two crucial ways. First we introduced d-parameters to model a categorical view on numerical data for base constructors. Then as the core contribution of this paper we enriched preferences by SV-semantics. We could prove that inductive preference construction with SV-semantics (including Pareto, prioritization and numerical ranking) preserves the strict partial order property.

Our better modeling capabilities even can improve preference query performance. We showed that known transformation laws of preference relational algebra remain valid under SV-semantics.

The BMO query model avoids already the embarrassing ‘empty result’ effect. Concerning the annoying flooding effect we presented novel insights. We could relate lower and upper bounds for BMO-sizes of Pareto and prioritized preference queries to regular and trivial SV-relations. Performing a series of test queries on real e-catalog data, we reported evidence that BMO-sizes can come up in handy portions. This observation makes us confident that also the flooding problem can be tamed in a semantically guided way.

There are more research challenges for personalized database applications. For cost-based query optimization, which was not the topic here, analytic methods to estimate BMO-sizes for the full constructor spectrum are essential. Preferences and user modeling are investigated within the Bavarian research cooperation FORSIP on “Situated, Individualized and Personalized Human-Computer Interaction” (www.forsip.de). E.g., the fully automated sales agent COSIMA^{B2B} ([15]) enables a deep personalization of the B2B sales process and automates skills that so far could be performed only by human vendors.

We have implemented all constructors stated in this paper including d-parameters, trivial and regular SV-relations in the latest release of Preference XPath. Moreover, we have extended Preference XPath with a nested constructor on tree-structured and set-valued XML-objects, dealing also with ontologies. All of this advanced functionality is used to build a deeply personalized notification system for future MPEG-7 multimedia libraries ([24]), where also end-user evaluation studies will be conducted.

9. ACKNOWLEDGMENTS

I would like to express my thanks to U. Güntzer for his valuable comments on a draft of this work. Many thanks also to S. Döring, T. Ehm and B. Hafenrichter who implemented the constructor extensions in Preference XPath.

10. REFERENCES

- [1] R. Agrawal, E. L. Wimmers: A Framework for Expressing and Combining Preferences. *Proc. ACM SIGMOD*, May 2000, Dallas, pp. 297 - 306.
- [2] H. Andreka, M. Ryan, P.-Y. Schobbens: Operators and Laws for Combining Preference Relations. *Journal of Logic and Computation*, 12, 1, 2002, pp. 13 - 53.
- [3] W.T. Balke, U. Güntzer, J.X. Zheng: Efficient Distributed Skylining for Web Information Systems. *Proc. Intern. Conference on Extending Database Technology (EDBT)*, Heraklion, March 2004.
- [4] W.-T. Balke, W. Kießling, C. Unbehend: Performance and Quality Evaluation of a Personalized Route Planning System. *18th Brazilian Symposium on Databases (SBBDB)*, pp. 328-340, Manaus, 2003.
- [5] J. Bentley, H. Kung, M. Schkolnick, C. Thomson: On the average number of maxima in a set of vectors and applications. *JACM* 25, 536-543 (1978).
- [6] S. Börzsönyi, D. Kossmann, K. Stocker: The Skyline Operator. *Proc. 17th Intern. Conference On Data Engineering (ICDE)*, Heidelberg, April 2001.
- [7] J. Chomicki: Preference Formulas in Relational Queries. *ACM Transactions on Database Systems*, Vol. 28, No. 4, Dec. 2003, pp. 1 - 39.
- [8] P.C. Fishburn: Nontransitive Preferences in Decision Theory. *Journal of Risk and Uncertainty*, Kluwer Academic Publishers, 4:113 - 134, 1991.
- [9] P. C. Fishburn: Preference Structures and their Numerical Representations. *Journal of Theoretical Computer*, 217, 1999, pp. 359 - 383.
- [10] B. Hafenrichter: Optimization of Relational Preference Database Queries (in German). Ph.D. thesis, Univ. of Augsburg, Jan. 2004.
- [11] S. Holland: Preference Mining and Preference Repositories: Design, Algorithms and Personalized Applications. Ph.D. thesis Univ. Augsburg, *ibidem-Verlag* Stuttgart, Jan. 2004, ISBN 3-89821-352-8.
- [12] S. Holland, M. Ester, W. Kießling: Preference Mining: A Novel Approach on Mining User Preferences for Personalized Applications. *7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pp. 204-216, Dubrovnik, Croatia, Sep. 2003.
- [13] W. Kießling: *Foundations of Preferences in Database Systems*. Proc. 28th Intern. Conf. on Very Large Databases (VLDB), pp. 311-322, Hong Kong, China, 2002.
- [14] W. Kießling: Preference Constructors for Deeply Personalized Database Queries. Technical Report 2004-7, Institute of Computer Science, Univ. of Augsburg, Febr. 2004. (extended version of this paper; www.informatik.uni-augsburg.de/forschung/techBerichte/reports/2004-7.pdf)
- [15] W. Kießling, S. Fischer, S. Döring: COSIMA^{B2B} - Sales Automation for E-Procurement. *IEEE Conference on E-Commerce Technology (CEC)*, San Diego, U.S.A., July 2004, pp. 59-68.
- [16] W. Kießling, S. Fischer, S. Holland, T. Ehm: Design and Implementation of COSIMA - A Smart and Speaking E-Sales Assistant. *3rd Intern. Workshop on Advanced Issues of E-Commerce and Web-Based Inform. Systems (WECWIS)*, pp. 21-30, San Jose, 2001. (demonstrated also at SIGMOD 2001, St. Barbara, U.S.A.)
- [17] W. Kießling, U. Güntzer: Database Reasoning - A Deductive Framework for Solving Large and Complex Problems by means of Subsumption. *Proc. 3rd Workshop on Information Systems and Artificial Intelligence*, LNCS 777, pp. 118-138, Hamburg, 1994.
- [18] W. Kießling, B. Hafenrichter: Optimizing Preference Queries for Personalized Web Services. *IASTED Intern. Conf. on Communication, Internet, Information Technology*. St. Thomas, USA, Nov. 2002, pp. 461 - 466.

- [19] W. Kießling, B. Hafenrichter, S. Fischer, S. Holland: Preference XPATH: A Query Language for E-Commerce. *Proc. 5th Intern. Conference Wirtschaftsinformatik*, pp. 427-440, Augsburg, Germany, 2001.
- [20] W. Kießling and G. Köstler: Preference SQL - Design, Implementation, Experiences. *Proc. 28th Intern. Conf. on Very Large Data Bases (VLDB)*, pp. 990-1001, Hong Kong, China, 2002.
- [21] G. Köstler, W. Kießling, H. Thöne, U. Güntzer: Fixpoint Iteration with Subsumption in Deductive Databases. *Journal of Intelligent Information Systems*, Vol. 4, pp. 123-148, Boston, USA, 1995.
- [22] G. Koutrika, Y. Ioannidis: Personalization of Queries in Database Systems. *Proc. 20th International Conference on Data Engineering (ICDE)*, Boston, USA, March 30 - April 2, 2004.
- [23] M. Lacroix, P. Lavency: Preferences: Putting more Knowledge into Queries. *Proc. 13th Intern. Conference on Very Large Databases (VLDB)*, 1987.
- [24] Q. Wang, W.-T. Balke, W. Kießling, A. Huhn: P-News: Deeply Personalized News Dissemination for MPEG-7 based Digital Libraries. *8th European Conference on Digital Libraries (ECDL)*, Bath, U.K., Sept. 2004, pp. 256-268.
- [25] B. Xiao, E. Aimeur, J. M. Fernandez: PCFinder: An Intelligent Product Recommendation Agent for E-Commerce. *IEEE International Conference on Electronic Commerce (CEC)*, pp. 181-190, Newport Beach, CA, USA, 2003.

11. APPENDIX

Here some selected proofs are presented. The remaining ones can be found in [14].

Proposition 3 (see Section 4.1)

Given a SCORE_d preference P, let's define for all $x, y \in \text{dom}(A)$:

- $x \cong_P y$ iff $x \parallel_P y$
- a) \cong_P is an SV-relation (called **regular SV-relation**).
- b) If P is not a chain, then \cong_P may be non-trivial.

Proof of a):

- By definition: $x \cong_P y$ implies $x \parallel_P y$
- $x \cong_P y \wedge z <_P x$
iff $x \parallel_P y \wedge z <_P x$
iff {if $d = 0$ then $f(x) = f(y)$
else $\lceil f(x) / d \rceil = \lceil f(y) / d \rceil$ } \wedge
{if $d = 0$ then $f(z) < f(x)$
else $\lceil f(z) / d \rceil < \lceil f(x) / d \rceil$ }
- implies {if $d = 0$ then $f(z) < f(y)$
else $\lceil f(z) / d \rceil < \lceil f(y) / d \rceil$ }
- iff $z <_P y$
- $x \cong_P y \wedge x <_P z$
iff $x \parallel_P y \wedge x <_P z$
iff {if $d = 0$ then $f(x) = f(y)$
else $\lceil f(x) / d \rceil = \lceil f(y) / d \rceil$ } \wedge
{if $d = 0$ then $f(x) < f(z)$
else $\lceil f(x) / d \rceil < \lceil f(z) / d \rceil$ }
- implies {if $d = 0$ then $f(y) < f(z)$
else $\lceil f(y) / d \rceil < \lceil f(z) / d \rceil$ }
- iff $y <_P z$

- \parallel_P is reflexive and symmetric in general. Since SCORE_d constructs a weak order, \parallel_P is transitive due to [9], hence \cong_P is transitive. qed

Proof of b):

If P is not a chain, then there are $v, w \in \text{dom}(A)$, $v \neq w$: $v \parallel_P w$, hence $v \cong_P w$ is feasible. qed

Theorem 1 (see Section 4.2, Pareto preference)

Given $P_1 = (A_1, <_{P_1}, \cong_{P_1})$ and $P_2 = (A_2, <_{P_2}, \cong_{P_2})$ consider $P := P_1 \otimes P_2$. Then $P = (A_1 \cup A_2, <_P, \cong_P)$ is a preference with SV-semantics, i.e.:

- a) $<_P$ is a strict partial order on $A_1 \cup A_2$.
- b) \cong_P is an SV-relation for $<_P$.

Proof of a):

Irreflexivity: $x <_P x$ iff $(\text{false} \wedge (\text{false} \vee \text{true})) \vee (\text{false} \wedge (\text{false} \vee \text{true}))$ iff false

Transitivity: For abbreviation we define:

$$\begin{aligned} F_1 &\equiv 'x_1 <_{P_1} y_1', F_2 \equiv 'x_2 <_{P_2} y_2', F_3 \equiv 'x_2 \cong_{P_2} y_2', \\ F_4 &\equiv 'x_1 \cong_{P_1} y_1', F_5 \equiv 'y_1 <_{P_1} z_1', F_6 \equiv 'y_2 <_{P_2} z_2', \\ F_7 &\equiv 'y_2 \cong_{P_2} z_2', F_8 \equiv 'y_1 \cong_{P_1} z_1', F_9 \equiv 'x_1 <_{P_1} z_1', \\ F_{10} &\equiv 'x_2 <_{P_2} z_2', F_{11} \equiv 'x_2 \cong_{P_2} z_2', F_{12} \equiv 'x_1 \cong_{P_1} z_1' \end{aligned}$$

Due to Definition 8 for SV-relations we can state: // ***

Because \cong_{P_1} and \cong_{P_2} did *not* change $<_{P_1}$ and $<_{P_2}$:

$$F_1 \wedge F_5 \text{ implies } F_9, F_2 \wedge F_6 \text{ implies } F_{10}$$

Because SV-relations are *transitive*:

$$F_4 \wedge F_8 \text{ implies } F_{12}, F_3 \wedge F_7 \text{ implies } F_{11}$$

Because of properties of SV-relations:

$$F_1 \wedge F_8 \text{ implies } F_9, F_2 \wedge F_7 \text{ implies } F_{10},$$

$$F_3 \wedge F_6 \text{ implies } F_{10}, F_4 \wedge F_5 \text{ implies } F_9$$

Then we get: $x <_P y \wedge y <_{P_2} z$

$$\begin{aligned} \text{iff } & [(F_1 \wedge (F_2 \vee F_3)) \vee (F_2 \wedge (F_1 \vee F_4))] \wedge \\ & [(F_5 \wedge (F_6 \vee F_7)) \vee (F_6 \wedge (F_5 \vee F_8))] \\ \text{iff } & [(F_1 \wedge F_2) \vee (F_1 \wedge F_3) \vee (F_2 \wedge F_4)] \wedge \\ & [(F_5 \wedge F_6) \vee (F_5 \wedge F_7) \vee (F_6 \wedge F_8)] \\ \text{iff } & (F_1 \wedge F_2 \wedge F_5 \wedge F_6) \vee (F_1 \wedge F_2 \wedge F_5 \wedge F_7) \vee \\ & (F_1 \wedge F_2 \wedge F_6 \wedge F_8) \vee (F_1 \wedge F_3 \wedge F_5 \wedge F_6) \vee \\ & (F_1 \wedge F_3 \wedge F_5 \wedge F_7) \vee (F_1 \wedge F_3 \wedge F_6 \wedge F_8) \vee \\ & (F_2 \wedge F_4 \wedge F_5 \wedge F_6) \vee (F_2 \wedge F_4 \wedge F_5 \wedge F_7) \vee \\ & (F_2 \wedge F_4 \wedge F_6 \wedge F_8) \end{aligned}$$

According to // *** we can now continue:

$$\begin{aligned} \text{implies } & (F_9 \wedge F_{10}) \vee (F_9 \wedge F_{11}) \vee (F_9 \wedge F_{10}) \vee \\ & (F_9 \wedge F_{10}) \vee (F_9 \wedge F_{11}) \vee (F_9 \wedge F_{10}) \vee \\ & (F_9 \wedge F_{10}) \vee (F_9 \wedge F_{10}) \vee (F_{10} \wedge F_{12}) \\ \text{iff } & (F_9 \wedge F_{10}) \vee (F_9 \wedge F_{11}) \vee ((F_9 \wedge F_{10}) \vee \\ & (F_{10} \wedge F_{12})) \\ \text{iff } & (F_9 \wedge (F_{10} \vee F_{11})) \vee (F_{10} \wedge (F_9 \vee F_{12})) \\ \text{iff } & x <_P z \end{aligned}$$

qed

Proof of b):

$$\begin{aligned} x \cong_P y \text{ iff } & x_1 \cong_{P_1} y_1 \wedge x_2 \cong_{P_2} y_2 \\ \text{implies } & x_1 \parallel_{P_1} y_1 \wedge x_2 \parallel_{P_2} y_2 \\ \text{iff } & \neg(x_1 <_{P_1} y_1) \wedge \neg(y_1 <_{P_1} x_1) \wedge \\ & \neg(x_2 <_{P_2} y_2) \wedge \neg(y_2 <_{P_2} x_2) \\ \text{implies } & x \parallel_P y \end{aligned}$$

[Please note that the given proof consistently renames variables x, y, z in Definition 8b).]

$$\begin{aligned}
& x <_p y \wedge y \cong_p z \\
& \text{iff } (F_1 \wedge (F_2 \vee F_3)) \vee (F_2 \wedge (F_1 \vee F_4)) \wedge (F_8 \wedge F_7) \\
& \text{iff } ((F_1 \wedge (F_2 \vee F_3)) \wedge F_8 \wedge F_7) \vee ((F_2 \wedge (F_1 \vee F_4)) \wedge \\
& \quad F_8 \wedge F_7) \\
& \text{implies } ((F_9 \wedge (F_2 \vee F_3)) \wedge F_7) \vee ((F_{10} \wedge (F_1 \vee F_4)) \\
& \quad \wedge F_8) \\
& \text{implies } ((F_9 \wedge (F_{10} \vee F_{11})) \vee ((F_{10} \wedge (F_9 \vee F_{12})) \\
& \text{iff } x <_p z
\end{aligned}$$

[Please note that the given proof consistently renames variables x, y, z in Definition 8c).]

$$\begin{aligned}
& x <_p y \wedge x \cong_p z \\
& \text{iff } (F_1 \wedge (F_2 \vee F_3)) \vee (F_2 \wedge (F_1 \vee F_4)) \wedge (F_{12} \wedge F_{11}) \\
& \text{iff } ((F_1 \wedge (F_2 \vee F_3)) \wedge F_{11} \wedge F_{12}) \vee ((F_2 \wedge (F_1 \vee F_4)) \wedge \\
& \quad F_{11} \wedge F_{12}) \\
& \text{implies } ((z_1 <_{p_1} y_1 \wedge (F_2 \vee F_3)) \wedge F_{11}) \vee \\
& \quad ((z_2 <_{p_2} y_2 \wedge (F_1 \vee F_4)) \wedge F_{12}) \\
& \text{implies } ((z_1 <_{p_1} y_1 \wedge (z_2 <_{p_2} y_2 \vee z_2 \cong_{p_2} y_2)) \vee \\
& \quad ((z_2 <_{p_2} y_2 \wedge (z_1 <_{p_1} y_1 \vee z_1 \cong_{p_1} y_1)) \\
& \text{iff } z <_p y
\end{aligned}$$

- \cong_p is reflexive, symmetric and transitive, since it is the intersection of equivalence relations \cong_{p_1} and \cong_{p_2} . qed

Theorem 4 (see Section 5.1)

Every SV-order is strict partial order. Moreover, $[P] = (A/\cong_p, <_{[P]})$ is a strict partial order, where $\cong_{[P]}$ is the trivial SV-relation, i.e. equality of equivalence classes.

Proof:

- $<_{[P]}$ is *well-defined*:
Consider $x <_p y$ for some $x \in X, y \in Y$. Then for each $x' \in X$ by 0b) $x' <_p y$ holds. Likewise, for each $y' \in Y$ by 0c) $x <_p y'$ holds. Thus the definition of $X <_{[P]} Y$ is independent from the chosen representative for X and Y .
- $<_{[P]}$ is *irreflexive*:
 $X <_{[P]} X$ iff $(\forall x \in X, \forall y \in X: x <_p y)$
implies $(\forall x \in X: x <_p x)$ iff false
- $<_{[P]}$ is *transitive*:
 $X <_{[P]} Y \wedge Y <_{[P]} Z$
iff $(\forall x \in X, \forall y \in Y: x <_p y) \wedge$
 $(\forall y \in Y, \forall z \in Z: y <_p z)$
Considering a fixed, but arbitrary $y_0 \in Y$ we can continue:
implies $(\forall x \in X: x <_p y_0 \wedge \forall z \in Z: y_0 <_p z)$
By transitivity of $<_p$ we get:
implies $(\forall x \in X, \forall z \in Z: x <_p z)$ iff $X <_{[P]} Z$
- $\cong_{[P]}$ is *trivial*:
For $X \neq Y$ by the definition we get:
 $X \cong_{[P]} Y$ iff $\forall x \in X, \forall y \in Y: x \cong_p y$
implies (by Proposition 2a)
 $\forall x \in X, \forall y \in Y: \text{false iff false}$
Thus $\cong_{[P]}$ is the equality of equivalence classes for A/\cong_p , hence it represents the trivial SV-relation on A/\cong_p . qed

Proposition 4 (see section 6.1)

BMO-sizes of SCORE_d are non-monotonic in d .

Proof: We assume $d_2 > d_1 > 0$.

$$\begin{aligned}
& x <_{Pd_1} y \text{ iff } \lceil f(x)/d_1 \rceil < \lceil f(y)/d_1 \rceil \\
& \text{implies } f(x)/d_1 < f(y)/d_1 \text{ iff } f(x)/d_2 < f(y)/d_2 \\
& \text{implies } \lceil f(x)/d_2 \rceil \leq \lceil f(y)/d_2 \rceil \\
& \text{iff } x <_{Pd_2} y \vee x \parallel_{Pd_2} y \\
& \text{Assuming in addition that } f(x) < f(y): \\
& x \parallel_{Pd_1} y \text{ iff } \lceil f(x)/d_1 \rceil = \lceil f(y)/d_1 \rceil \\
& \text{implies } 0 \leq (f(y) - f(x)) / d_1 \leq 1 \\
& \text{implies } 0 \leq (f(y) - f(x)) / d_2 \leq 1 \\
& \text{implies } \lceil f(x)/d_2 \rceil \leq \lceil f(y)/d_2 \rceil \text{ iff } x <_{Pd_2} y \vee x \parallel_{Pd_2} y
\end{aligned}$$

As a net effect we have a non-monotonic behavior: If x is not in the BMO-set for d_1 , it may get into it for some $d_2 > d_1$. On the other hand, if x is in the BMO-set for d_1 , then there is no guarantee that x stays in it for $d_2 > d_1$. qed

For illustration we study $P_d := \text{SCORE}_d(A, f)$ for a relation $R(A) = \{a_1, a_2, a_3\}$, where $f(a_1) = 2.5$, $f(a_2) = 3.2$ and $f(a_3) = 3.5$. Let's define $\text{BMO-size}(d) := \text{card}(\sigma[P_d](R))$.

$$\begin{aligned}
d_1 = 1.0: & \quad \lceil f(a_1)/d_1 \rceil = \lceil 2.5/1.0 \rceil = \lceil 2.5 \rceil = 3, \\
& \quad \lceil f(a_2)/d_1 \rceil = \lceil 3.2/1.0 \rceil = \lceil 3.2 \rceil = 4, \\
& \quad \lceil f(a_3)/d_1 \rceil = \lceil 3.5/1.0 \rceil = \lceil 3.5 \rceil = 4, \text{ yielding} \\
& \quad \text{BMO-size}(d_1) = 2 \\
d_2 = 1.7: & \quad \lceil f(a_1)/d_2 \rceil = \lceil 2.5/1.7 \rceil = \lceil 1.59 \rceil = 2, \\
& \quad \lceil f(a_2)/d_2 \rceil = \lceil 3.2/1.7 \rceil = \lceil 1.88 \rceil = 2, \\
& \quad \lceil f(a_3)/d_2 \rceil = \lceil 3.5/1.7 \rceil = \lceil 2.06 \rceil = 3, \text{ yielding} \\
& \quad \text{BMO-size}(d_2) = 1 \\
d_3 = 2.0: & \quad \lceil f(a_1)/d_3 \rceil = \lceil 2.5/2.0 \rceil = \lceil 1.25 \rceil = 2, \\
& \quad \lceil f(a_2)/d_3 \rceil = \lceil 3.2/2.0 \rceil = \lceil 1.60 \rceil = 2, \\
& \quad \lceil f(a_3)/d_3 \rceil = \lceil 3.5/2.0 \rceil = \lceil 1.75 \rceil = 2, \text{ yielding} \\
& \quad \text{BMO-size}(d_3) = 3
\end{aligned}$$

Thus $d_1 \leq d_2$ doesn't imply $\text{BMO-size}(d_1) \leq \text{BMO-size}(d_2)$.

Theorem 5 (see Section 6.2)

Consider $P_1 = (A_1, <_{p_1}, \cong_1)$, $P_1^* = (A_1, <_{p_1}, \cong_1^*)$, differing only wrt the SV-relation, and similarly $P_2 = (A_2, <_{p_2}, \cong_2)$, $P_2^* = (A_2, <_{p_2}, \cong_2^*)$.

- a) $\sigma[P_1 \otimes P_2](R) \subseteq \sigma[P_1^* \otimes P_2^*](R)$
if $\cong_1 \succcurlyeq_{p_1} \cong_1^*$ and $\cong_2 \succcurlyeq_{p_2} \cong_2^*$
- b) $\sigma[P_1 \& P_2](R) \subseteq \sigma[P_1^* \& P_2](R)$ if $\cong_1 \succcurlyeq_{p_1} \cong_1^*$
- c) $\sigma[P_1 \& P_2](R) \subseteq \sigma[P_1 \otimes P_2](R)$

Proof:

- a) If $\cong_1 \succcurlyeq_{p_1} \cong_1^*$ and $\cong_2 \succcurlyeq_{p_2} \cong_2^*$, then from Definition 10a it is clear that for all $x, y \in \text{dom}(A_1 \cup A_2)$:
 $x <_{p_1^* \otimes p_2^*} y$ implies $x <_{p_1 \otimes p_2} y$
Then according to [7], Theorem 5.5, the proof is immediate.
- b) If $\cong_1 \succcurlyeq_{p_1} \cong_1^*$, then from Definition 10b it is clear that for all $x, y \in \text{dom}(A_1 \cup A_2)$:
 $x <_{p_1^* \& p_2} y$ implies $x <_{p_1 \& p_2} y$
Again according to [7], Theorem 5.5, the proof is immediate.
- c) In [13] the so-called *non-discrimination theorem* for Pareto preferences was proved:
 $P_1 \otimes P_2 \equiv (P_1 \& P_2) \blacklozenge (P_2 \& P_1)$
This immediately gives us:
 $(x_1, x_2) <_{p_1 \otimes p_2 \text{ new}} (y_1, y_2)$
iff $(x_1, x_2) <_{p_1 \& p_2 \text{ new}} (y_1, y_2) \wedge$
 $(x_1, x_2) <_{p_2 \& p_1 \text{ new}} (y_1, y_2)$
implies $(x_1, x_2) <_{p_1 \& p_2 \text{ new}} (y_1, y_2)$ qed