An Algebraic Calculus of Database Preferences

Bernhard Möller, Patrick Roocks, and Markus Endres

Institut für Informatik, Universität Augsburg, D-86135 Augsburg, Germany
{moeller,roocks,endres}@informatik.uni-augsburg.de

Abstract. Preference algebra, an extension of the algebra of database relations, is a well-studied field in the area of personalized databases. It allows modelling user wishes by preference terms; they represent strict partial orders telling which database objects the user prefers over other ones. There are a number of constructors that allow combining simple preferences into quite complex, nested ones. A preference term is then used as a database query, and the results are the maximal objects according to the order it denotes. Depending on the size of the database, this can be computationally expensive. For optimisation, preference queries and the corresponding terms are transformed using a number of algebraic laws. So far, the correctness proofs for such laws have been performed by hand and in a point-wise fashion. We enrich the standard theory of relational databases to an algebraic framework that allows completely point-free reasoning about complex preferences. This black-box view is amenable to a treatment in first-order logic and hence to fully automated proofs using off-the-shelf verification tools. We exemplify the use of the calculus with some non-trivial laws, notably concerning so-called preference prefilters which perform a preselection to speed up the computation of the maximal objects proper.

Keywords relational algebra, preferences, preference algebra, prefilter

1 Introduction

In many database applications, the queries are based on multiple, and sometimes conflicting, goals. For example, a tourist may be interested in hotels in Nassau (Bahamas) which are *cheap*, have *reasonable ratings* (say, 3-star) and are *close to* the beach. Unfortunately, these goals are conflicting, as the hotels near the beach tend to be more expensive. Thus, there may be no single *optimal* answer: it is unlikely that there exists a single 3-star hotel that is cheapest among all 3-star hotels and closest to the beach. Still, users are looking for *satisfactory* answers. But what does "satisfactory" mean? For the same query, different users, guided by their personal preferences, may find different answers appealing. For example, a person may be willing to pay a little more to be closer to the beach; another may be contented with a cheaper hotel as long as it is convenient to reach the beach from it. Thus, in our example one would like to find a whole set of budget hotels, where those closer to the beach are slightly more expensive.

Therefore it is important for a database system to present *all interesting* answers that may fulfil a user's need. Still, most current database search engines only deal with *hard constraints*: a tuple belongs to the search result if and only if it fulfils *all* given conditions.

As a remedy, queries with *soft constraints* are investigated, especially the so-called "skyline queries" [BKS01], which combine multiple, equally important, goals. An extension of this leads to the more comprehensive approach of *preference relations* which has been investigated in [Kie02,KH03,KEW11]. A preference allows users to establish a strict partial order that expresses which database objects are better for them than others. Based on this, a query selects according to the "best matches only (BMO)" model [Kie02] those objects that are not dominated by any others in the preference relation. To give the user more flexibility, a large set of predefined operators for constructing preference relations is provided.

Depending on the size of the database, the selection of the best matches according to a complex preference relation can be computationally expensive. To improve the process, preference queries and the corresponding terms are transformed using a number of algebraic laws for heuristically driven optimisation. So far, the correctness proofs for such laws have been performed by hand and in a point-wise fashion.

The contribution of the present paper is to enrich the standard theory of relational databases with an algebraic framework that allows completely point-free reasoning about (complex) preferences and their best matches. This "black-box view" is amenable to a treatment in first-order logic and hence to fully automated proofs using off-the-shelf verification tools. We exemplify the use of the calculus with some non-trivial laws, notably concerning so-called preference prefilters (introduced in [End11]), which perform a preselection to speed up the computation of the best matches proper, in particular, for queries involving expensive join operations. It turns out that the original laws hold under much weaker assumptions; moreover, several new ones are derived.

2 Types and Tuples

In this section we present the formal framework to model database objects as tuples. We introduce typed relations whose types represent attributes, i.e. the columns of a database relation. Conceptually and notationally, we largely base on [Kan90].

2.1 Typed Tuples

Definition 2.1. Let $\mathcal{A}$ be a set of *attribute names*. For $A \in \mathcal{A}$ the set D_A is called the *domain* of A, and $(D_A)_{A\in\mathcal{A}}$ is a family of domains. We define the following notions:

- A *type* T is a subset $T \subseteq \mathcal{A}$.

– An attribute $A \in \mathcal{A}$ is also used for the type $\{A\}$, omitting the set braces.
– A T-*tuple* is a mapping

$$t : T \to \bigcup_{A \in \mathcal{A}} D_A \text{ where } \forall A \in T : t(A) \in D_A.$$

– For a T-tuple t and a sub-type $T' \subseteq T$ we define the projection $\pi_{T'}(t)$ to T' as the restriction of the mapping t to T': $\pi_{T'}(t) : T' \to \bigcup_{A \in \mathcal{A}} D_A, A \mapsto t(A)$.
– The domain D_T for a type T is the set of all T-tuples, i.e., $D_T = \prod_{A \in T} D_A$.
– The set $\mathcal{U} =_{df} \bigcup_{T \subseteq \mathcal{A}} D_T$ is called the *universe*.
– For a tuple t, and a set of tuples M we introduce the following abbreviations:

$$t :: T \Leftrightarrow_{df} t \in D_T, \qquad M :: T \Leftrightarrow_{df} M \subseteq D_T.$$

Definition 2.2 (Join). The *join* of two types T_1, T_2 is the union of their attributes:

$$T_1 \bowtie T_2 =_{df} T_1 \cup T_2.$$

For sets of tuples $M_i :: T_i$ $(i = 1, 2)$, the join is defined as the set of all consistent combinations of M_i-tuples:

$$M_1 \bowtie M_2 =_{df} \{t :: T_1 \bowtie T_2 \mid \pi_{T_i}(t) \in M_i, i = 1, 2\} .$$

We illustrate this concept with the following example.

Example 2.3. Assume a database of cars with a unique ID and further attributes for model and horsepower. Hence the attribute names, i.e. types, are ID, model and hp. The tuples are written as explicit mappings. Assume the following sets:

$$M_1 =_{df} \{\{\text{ID} \mapsto 1, \text{ model} \mapsto \text{'BMW 7'}\}, \{\text{ID} \mapsto 3, \text{ model} \mapsto \text{'Mercedes CLS'}\}\},$$
$$M_2 =_{df} \{\{\text{ID} \mapsto 2, \text{ hp} \mapsto 230\}, \{\text{ID} \mapsto 3, \text{ hp} \mapsto 315\}\}.$$

The sets have the types $M_1 :: \text{ID} \bowtie \text{model}$ and $M_2 :: \text{ID} \bowtie \text{hp}$. Now we consider the join $M_1 \bowtie M_2 :: \text{ID} \bowtie \text{model} \bowtie \text{hp}$. We have $(\text{ID} \bowtie \text{model}) \cap (\text{ID} \bowtie \text{hp}) = \text{ID}$. The only tuple $t :: \text{ID} \bowtie \text{model} \bowtie \text{hp}$ which fulfills both $\pi_{T_1}(t) \in M_1$ and $\pi_{T_2}(t) \in M_2$ is the one with $t : \text{ID} \mapsto 3$. Hence the join is given by:

$$M_1 \bowtie M_2 = \{\{\text{ID} \mapsto 3, \text{ model} \mapsto \text{'Mercedes CLS'}, \text{ hp} \mapsto 315\}\} .$$

Corollary 2.4. *The following laws hold:*

1. $\bowtie$ *is associative and commutative and distributes over* $\cup$.
2. $\bowtie$ *preserves the inclusion order, i.e.* $M \bowtie N \subseteq M' \bowtie N$ *for* $M \subseteq M'$.
3. *Assume* $M_i, N_i :: T_i$ $(i = 1, 2)$. *Then the following exchange law holds:*

$$(M_1 \cap N_1) \bowtie (M_2 \cap N_2) = (M_1 \bowtie M_2) \cap (N_1 \bowtie N_2).$$

Proof. (1) and (2) follow directly from definition. Using the definition of the join and the usual intersection of sets we show the exchange law as follows:

$$
\begin{aligned}
& x \in (M_1 \cap N_1) \bowtie (M_2 \cap N_2) \\
\Leftrightarrow\ & \pi_{T_1}(x) \in (M_1 \cap N_1) \wedge \pi_{T_2}(x) \in (M_2 \cap N_2) \\
\Leftrightarrow\ & \pi_{T_1}(x) \in M_1 \wedge \pi_{T_1}(x) \in N_1 \wedge \pi_{T_2}(x) \in M_2 \wedge \pi_{T_2}(x) \in N_2 \\
\Leftrightarrow\ & x \in M_1 \bowtie M_2 \wedge x \in N_1 \bowtie N_2 \\
\Leftrightarrow\ & x \in (M_1 \bowtie M_2) \cap (N_1 \bowtie N_2)\ .
\end{aligned}
$$

2.2 Typed Relations

Definition 2.5 (Typed homogeneous binary relations). For a type T we define the following abbreviations:

$$(t_1, t_2) :: T^2 \ \Leftrightarrow_{df}\ t_i \in D_T, \qquad R :: T^2 \ \Leftrightarrow_{df}\ R \subseteq D_T \times D_T.$$

We say that the *typed relation* R has type T. There are some special relations: The full relation $\top_T =_{df} D_T \times D_T$, the identity $1_T =_{df} \{(x, x) \mid x \in D_T\}$ and the empty relation $0_T =_{df} \emptyset$.

This concept of typed relations also appears in the relation-based logical, but not primarily algebraic, approach to database notions of [MO04]. We will generalise it in Section 3.2.

Definition 2.6 (Join of relations). Let $R_i :: T_i^2$ $(i = 1, 2)$. Then the *composition* $R_1 \bowtie R_2 :: (T_1 \bowtie T_2)^2$ is defined by

$$t\,(R_1 \bowtie R_2)\,u \Leftrightarrow_{df} \pi_{T_1}(t)\,R_1\,\pi_{T_1}(u) \wedge \pi_{T_2}(t)\,R_2\,\pi_{T_2}(u).$$

Corollary 2.7.

1. *Assume $M_i, N_i :: T_i$ $(i = 1, 2)$. Then the following exchange law holds:*
$$(M_1 \bowtie M_2) \times (N_1 \bowtie N_2) = (M_1 \times N_1) \bowtie (M_2 \times N_2).$$
2. *For types T_1, T_2 and $X \in \{0, 1, \top\}$ we have $X_{T_1 \bowtie T_2} = X_{T_1} \bowtie X_{T_1}$.*

Proof.

1. Straightforward from Definition 2.6.
2. Using part (1), $(D_{T_1} \bowtie D_{T_2}) \times (D_{T_1} \bowtie D_{T_2}) = (D_{T_1} \times D_{T_1}) \bowtie (D_{T_2} \times D_{T_2})$. By definition of the join for types we have that $T_1 \bowtie T_2 = T_1 \cup T_2$. From the definition of the join for sets we infer that $D_{T_1 \bowtie T_2} = D_{T_1} \bowtie D_{T_2}$. This shows the claim for $X = \top$. For $X = 1$ we show the equality component-wise using again the argument $D_{T_1 \bowtie T_2} = D_{T_1} \bowtie D_{T_2}$. For $X = \emptyset$ the claim is obvious.

Corollary 2.8.

1. *For $M, N :: T$ we have $M \bowtie N = M \cap N$. In particular, we have $N \bowtie N = N$.*
2. *For $R_1, R_2 :: T$ we have $R_1 \bowtie R_2 = R_1 \cap R_2$.*
3. *For $M_i :: T_i$ $(i = 1, 2)$ with disjoint T_i, i.e., with $T_1 \cap T_2 = \emptyset$, the join $M =_{df} M_1 \bowtie M_2$ is isomorphic to the cartesian product of M_1 and M_2.*

Proof.

1. By the definition of join and the typing assumptions we have

$$t \in M \bowtie N \;\Leftrightarrow\; t \in M \;\wedge\; t \in N \;.$$

2. Similarly we conclude for all $x, y :: T$:

$$x\,(R_1 \bowtie R_2)\,y \;\Leftrightarrow\; \pi_T(x)\,R_i\,\pi_T(y)\;(i = 1,2) \;\Leftrightarrow\; x\,R_1\,y \;\wedge\; x\,R_2\,y$$

3. For $x \in M$, the two join conditions $\pi_{T_i}(x) \in M_i$ are independent. Hence all elements of M_1 can be joined with all elements of M_2. Thus, by definition,

$$t \in M \;\Leftrightarrow\; \pi_{T_1}(t) \in M_1 \;\wedge\; \pi_{T_2}(t) \in M_2 \;\Leftrightarrow\; (\pi_{T_1}(t), \pi_{T_2}(t)) \in M_1 \times M_2.$$

2.3 Inverse Image and Maximal Elements

Definition 2.9 (Inverse image). For a relation $R :: T^2$ the inverse image of a set $Y :: T$ under R is formally defined as

$$\langle R \rangle\, Y \;=_{df}\; \{x :: T \mid \exists y \in Y : x\,R\,y\}\;.$$

The notation stems from the fact that in modal logic the inverse-image operator is a (forward) diamond.

Lemma 2.10. *Assume $R_i :: T_i^2$ and $Y_i :: T_i$ $(i = 1,2)$ with disjoint T_1, T_2. Then the following exchange law for the join and the inverse image holds:*

$$\langle R_1 \bowtie R_2 \rangle\,(Y_1 \bowtie Y_2) \;=\; \langle R_1 \rangle\, Y_1 \bowtie \langle R_2 \rangle\, Y_2\;.$$

Proof. Using the definition of the inverse image and the composition of relations we infer:

$$\begin{aligned}
& x \in \langle R_1 \bowtie R_2 \rangle\,(Y_1 \bowtie Y_2) \\
\Leftrightarrow\; & \exists y \in (Y_1 \bowtie Y_2) : x\,(R_1 \bowtie R_2)\,y \\
\Leftrightarrow\; & \exists y \in (Y_1 \bowtie Y_2) : \pi_{T_1}(x)\,R_1\,\pi_{T_1}(y) \;\wedge\; \pi_{T_2}(x)\,R_2\,\pi_{T_2}(y) \\
\Leftrightarrow\; & \exists y_1 \in Y_1 : \exists y_2 \in Y_2 : \pi_{T_1}(x)\,R_1\,y_1 \;\wedge\; \pi_{T_2}(x)\,R_2\,y_2 \\
\Leftrightarrow\; & \pi_{T_1}(x) \in \langle R_1 \rangle\, Y_1 \;\wedge\; \pi_{T_2}(x) \in \langle R_2 \rangle\, Y_2 \\
\Leftrightarrow\; & x \in (\langle R_1 \rangle\, Y_1 \bowtie \langle R_2 \rangle\, Y_2)\;.
\end{aligned}$$

Note that splitting y into y_1 and y_2 in the third step is justified by disjointness of the types: because of $T_1 \cap T_2 = \emptyset$ the two join conditions $\pi_{T_i}(y) \in Y_i$ for $i = 1,2$ are independent of each other, hence the substitution $y_i := \pi_{T_i}(y)$ is allowed.

Assume that R_1, R_2 are strict orders (irreflexive and transitive), which is the case in our application domain of preferences. Then, together with Corollary 2.8.3, this lemma means that, under the stated disjointness assumption, $R_1 \bowtie R_2$ behaves like the *product order* of R_1 and R_2 on the Cartesian product $D_{T_1} \times D_{T_2}$.

The inverse image of a set Y under a relation R, when viewed the other way around, consists of the objects that have an R-successor in Y, i.e., are R-related to some object in Y or, in the preference context, *dominated* by some object in Y. For this reason we can characterise the set of R-maximal objects within a set Y, as follows.

Definition 2.11 (Maximal elements). For a relation $R :: T^2$ and a set $Y :: T$ we define

$$R \rhd Y \;=_{df}\; Y - \langle R \rangle\, Y \;,$$

where "$-$" is set difference.

These are the Y-objects that do not have an R-successor in Y, i.e., are not dominated by any object in Y. The mnemonic behind this notation is that in an order diagram for a preference relation R the maximal objects within Y are the peaks in Y; rotating the diagram clockwise by 90° puts the peaks to the right. Hence $R \rhd Y$ might also be read as "R-peaks in Y".

To develop the central properties of our algebra and the maximality operator it turns out useful to abstract from the concrete setting of binary relations over sets of tuples, which will be done in the next section.

3 An Algebraic Calculus

Since we have shown how to characterise the maximal elements concisely using a diamond operation, it seems advantageous to reuse the known algebraic theory around that. This also allows us to exhibit clearly which assumptions are really necessary; it turns out that most of the development is completely independent of the properties of irreflexivity and transitivity that were originally assumed for preference relations in [Kie02], and in fact also independent of the use of relations at all.

3.1 Semirings

Definition 3.1. An *idempotent semiring* consists of a set S of elements together with binary operations $+$ of *choice* and $\cdot$ of *composition*. Both are required to be associative, choice also to be commutative and idempotent. Moreover, composition has to distribute over choice in both arguments. Finally, there have to be units 0 for choice and 1 for composition.

Binary homogeneous relations over a set form an idempotent semiring with choice $\cup$ and composition ";", which have $\emptyset$ and the identity relation as their respective units.

Definition 3.2. Every idempotent semiring induces a *subsumption order* by $x \leq y \;\Leftrightarrow\; x + y = y$. A *test* is an element $x \leq 1$ that has a complement $\neg x$ relative to 1, i.e., which satisfies

$$x + \neg x = 1 \;, \qquad x \cdot \neg x = 0 \;.$$

It is well known (e.g. [MB85]) that the complement is unique when it exists and that the set of all tests forms a Boolean algebra with $+$ as join and $\cdot$ as meet. Tests are used to represent subsets or assertions in an algebraic way. In the semiring of binary relations over a set M the tests are subidentities, i.e., subsets of the identity relation, of the form $I_N =_{df} \{(x,x) \mid x \in N\}$ for some subset $N \subseteq M$ and hence in one-to-one correspondence with the subsets of M. Because of that we will, by a slight abuse of language, say that x *lies in* I_N when $(x,x) \in I_N$.

We will use small letters $a, b, c, ...$ at the beginning of the alphabet to denote arbitrary semiring elements and $p, q, ...$ to denote tests.

Based on complementation, the difference of two tests p, q can be defined as $p - q =_{df} p \cdot \neg q$. It satisfies, among other laws,

$$(p+q)-r = (p-r)+(q-r) \;, \quad (p-q)-r = p-(q+r) \;, \quad p-(q+r) = (p-q)\cdot(p-r) \;.$$

For the interaction between the complement and the subsumption ordering we can use the *shunting rule*

$$p \cdot q \leq r \;\Leftrightarrow\; p \leq \neg q + r \;.$$

A special case of applying this rule twice with $p = 1$ is the *contraposition* rule

$$q \leq r \;\Leftrightarrow\; \neg r \leq \neg q \;.$$

Tests can be used to express domain or range restrictions. For instance, when a is a relation and p, q are tests, $p \cdot a$ and $a \cdot q$ are the subrelations of a all of whose initial points lie in p and end points in q, respectively. Hence, all initial points of a lie in p if and only if $a \leq p \cdot a$.

With these properties we can give an algebraic characterisation of the test $\langle a \rangle\, q$ that represents the inverse image under a of the set represented by q or, equivalently, the set of initial points of $a \cdot q$.

Definition 3.3. Following [DMS06], the *(forward) diamond* is axiomatised by the universal property

$$\langle a \rangle\, q \leq p \;\Leftrightarrow\; a \cdot q \leq p \cdot a \cdot q \;\Leftrightarrow\; a \cdot q \leq p \cdot a \;.$$

Following the terminology of [DMS06], it would be more accurately termed a *pre-diamond*, since we do not require the axiom $\langle a \cdot b \rangle\, q = \langle a \rangle\, \langle b \rangle\, q$, which is not needed for our application. In the relational setting of [BW93], test and diamond are called monotoype and monotype factor, respectively.

The diamond enjoys the following useful algebraic properties:

$$\langle a \rangle\, 0 \;=\; 0 \;, \quad \langle a + b \rangle\, p \;=\; \langle a \rangle\, p + \langle b \rangle\, p \;, \quad \langle a \rangle\, (p + q) \;=\; \langle a \rangle\, p + \langle a \rangle\, q \;.$$

The latter two imply that diamond is isotone (i.e., monotonically increasing) in both arguments:

$$a \leq b \;\Rightarrow\; \langle a \rangle\, p \leq \langle b \rangle\, p \;, \quad p \leq q \;\Rightarrow\; \langle a \rangle\, p \leq \langle a \rangle\, q \;.$$

A special role is played by the test

$$\ulcorner a =_{df} \langle a \rangle 1 \ .$$

It represents the set of all objects that have an a-successor at all and therefore is called the *domain* of a. From the isotony of diamond we conclude, for test p,

$$\langle a \rangle p \leq \ulcorner a \ .$$

3.2 Representing Types

There are a number of ways to represent types algebraically, among then heterogeneous relation algebras [SHW97], relational allegories [BD97] or typed Kleene algebra [Koz98]. All these involve some amount of machinery and notation, which we want to avoid here.

More simply, we now interpret the largest test 1 as representing the universe $\mathcal{U}$ and use other tests to stand for subsets of it, e.g., for the domains associated with types. With every type $T \subseteq \mathcal{A}$, we associate a test 1_T representing its domain D_T. An assertion $p :: T$ means that p is a test, representing a set of tuples, with $p \leq 1_T$. Arbitrary semiring elements $a, b, c, ...$ will stand for preference relations. A type assertion $a :: T^2$ is short for $a \leq 1_T \cdot a \cdot 1_T$. By $1_T \leq 1$ this can be strengthened to an equality. Hence, since tests are idempotent under composition, $a :: T^2$ implies $1_T \cdot a = a = a \cdot 1_T$.

This latter property entails that the diamond respects types, i.e., for $a :: T^2$ and $q :: T$ we calculate

$$\langle a \rangle q :: T \ \Leftrightarrow \ \langle a \rangle q \leq 1_T \ \Leftrightarrow \ a \cdot q \leq 1_T \cdot a \cdot q \ \Leftrightarrow \ a \cdot q \leq a \cdot q \ \Leftrightarrow \ \mathsf{TRUE} \ .$$

To express that x is either an element which represents a relation *or* a test, we introduce the following notation:

$$x :: T^{(2)} \ \Leftrightarrow \ x :: T \ \vee \ x :: T^2 \ .$$

We will also need the infimum for elements $a_1, a_2 :: T^2$, which is axiomatised as follows:

$$\forall x :: T^2 : \quad x \leq a_1 \sqcap a_2 \ \Leftrightarrow_{df} \ x \leq a_1 \ \wedge \ x \leq a_2 \ .$$

In the semiring of binary relations this coincides with the intersection of two relations. For tests p, q we have, in every semiring, $p \sqcap q = p \cdot q$.

Finally, we assume for every type T a greatest element $\top_T$ in $\{x \,|\, x :: T^{(2)}\}$, i.e. we have $\forall\, x :: T^{(2)} : x \leq \top_T$.

3.3 Join Algebras

We now deal with the central notion of join. For this, we assume the typing mechanism of the previous section.

Definition 3.4 (Join algebra). A *join algebra* is an idempotent semiring with an additional binary operator $\bowtie$ satisfying the following requirements.

1. Join is associative, commutative and idempotent and distributes over choice $+$ in both arguments. Hence $\bowtie$ is isotone in both arguments.
2. If $a_i :: T_i^{(2)}$ $(i = 1, 2)$ then $a_1 \bowtie a_2 :: (T_1 \bowtie T_2)^{(2)}$.
3. For types T_i $(i = 1, 2)$ we have

$$1_{T_1 \bowtie T_2} = 1_{T_1} \bowtie 1_{T_2} \quad \text{and} \quad \top_{T_1 \bowtie T_2} = \top_{T_1} \bowtie \top_{T_2} \ .$$

4. Join and composition satisfy, for $a_i, b_i :: T_i^{(2)}$ $(i = 1, 2)$ with disjoint T_i, the exchange law

$$(a_1 \bowtie a_2) \cdot (b_1 \bowtie b_2) = (a_1 \cdot b_1) \bowtie (a_2 \cdot b_2).$$

5. The diamond operator respects joins of elements with disjoint types: for $a :: T_1^2, p :: T_1$ and $b :: T_2^2, q :: T_2$ with $T_1 \cap T_2 = \emptyset$ we have the exchange law

$$\langle a \bowtie b \rangle \, (p \bowtie q) \ = \ \langle a \rangle \, p \bowtie \langle b \rangle \, q \ .$$

Our typed relations from Section 2.2 form a join algebra.

3.4 Representation of Preferences

Preferences introduced in [Kie02] are strict partial orders, i.e. a special kind of binary homogeneous relations. These relations are defined on domains of types, and the objects compared are "database tuples" contained in a "database relation", i.e., a set of tuples.

To avoid confusion between the two uses of the word "relation" we call tuples *database elements* here and the database relation the *basic set* of objects. This means that we consider a "static" snapshot of the database at the time of the respective preference-based query and assume that no data is deleted or inserted into the database while the query being evaluated.

Abstractly, preferences can now be modelled as typed elements $a :: T^2$ for some type T. If one wants to express transitivity or irreflexivity of a, this can be done by requiring $a \cdot a \leq a$ or $a \sqcap 1_T = 0$, respectively. However, as we will see, for the most part these assumptions are inessential for the laws we will derive.

4 Maximal Element Algebra

Now we are ready for the algebraic treatment of our central notion.

4.1 Basic Definitions and Results

Definition 4.1. The *best* or *maximal* objects w.r.t. element $a :: T^2$ and test $p :: T$ are represented by the test

$$a \rhd p \ =_{df} \ p - \langle a \rangle \, p \ .$$

In particular, the test $a \rhd 1_T$ represents the a-best objects overall.

This definition is also given, in different notation, in [DMS06]. An analogous formulation, however, with tests encoded as vectors, i.e., right-universal relations, can be found in [SS93].

To give a first impression of the algebra at work, we show a number of useful basic properties of the $\rhd$ operator. Proofs of the following two lemmas can be found in Appendix B.1 and B.2.

Lemma 4.2. *Assume $a, b :: T^2$, $p :: T$. Then the following holds:*

1. $a \rhd 1_T = \neg\ulcorner a$.
2. $\ulcorner b \leq \ulcorner a \Leftrightarrow a \rhd 1_T \leq b \rhd 1_T$.
3. $a \rhd p \leq p$.
4. $a \rhd 1_T \leq p \Leftrightarrow a \rhd 1_T \leq a \rhd p$.
5. $a \rhd 1_T \leq a \rhd (a \rhd 1_T)$.
6. $a \rhd (a \rhd p) = a \rhd p$.
7. $(a + b) \rhd p = (a \rhd p) \cdot (b \rhd p)$.
8. $b \leq a \Rightarrow a \rhd p \leq b \rhd p$.
9. $1_T \leq a \Rightarrow a \rhd p = 0_T$.

Lemma 4.3. *Let $p, q :: T$ be a disjoint decomposition of 1_T, i.e. $p + q = 1_T$, $p \cdot q = 0_T$. Then we have $\neg p = q$.*

4.2 Basic Applications

Now we want to demonstrate how the maximality operator $\rhd$ works.

Example 4.4. Let $a :: T^2$ be a preference relation and suppose $p_1, p_2 :: T$ are tests that form a disjoint decomposition of 1_T. Assume that all elements in p_2 are better than all elements in p_1, i.e.,

$$\langle a \rangle\, p_2 = p_1, \quad \langle a \rangle\, p_1 = 0_T.$$

We show that p_2 represents the maximal elements, i.e. $p_2 = a \rhd 1_T$:

$$\begin{array}{ll} & a \rhd 1_T \\ = & \quad \{\![\text{ definition }]\!\} \\ & \neg\, \langle a \rangle\, 1_T \\ = & \quad \{\![\; p_1 + p_2 = 1_T \;]\!\} \\ & \neg(\langle a \rangle\, (p_1 + p_2)) \\ = & \quad \{\![\text{ distributivity of diamond }]\!\} \\ & \neg(\langle a \rangle\, p_1 + \langle a \rangle\, p_2) \\ = & \quad \{\![\text{ assumptions of } a \;]\!\} \\ & \neg p_1 \\ = & \quad \{\![\text{ Lemma 4.3 }]\!\} \\ & p_2 \end{array}$$

By this tiny example one can see how the maximality operator works in general, because one can always decompose 1_T into tests representing the non-maximal (p_1) and the maximal (p_2) elements, where p_1 and p_2 are disjoint.

4.3 Prefilters

In practical applications, e.g., in databases, the tests, in particular the test 1 representing all objects in the database, can be quite large. Hence it may be very expensive to compute $a \rhd 1$ for a given a. However, it can be less expensive to compute $b \rhd 1$ for another element b; ideally, that set is much smaller and the a-best objects overall coincide with the a-best objects within $b \rhd 1$. This motivates the following definition.

Definition 4.5. Assume $a, b :: T^2$. We call b a *prefilter* for a, written as $b \,\mathsf{pref}\, a$, if and only if

$$a \rhd 1_T = a \rhd (b \rhd 1_T) \ .$$

Note that no connection between a and b is assumed. By Lemma 4.2.6 we have $a \,\mathsf{pref}\, a$ for all a. A concrete example of a prefilter will be given in Section 5.1.

We can give another, calculationally useful, characterisation of prefilters. The proof of the following theorem can be found in appendix B.3.

Theorem 4.6. $b \,\mathsf{pref}\, a \Leftrightarrow \ulcorner b \leq \ulcorner a \wedge \ulcorner a \leq \ulcorner b + \langle a \rangle \neg \ulcorner b$.

So far, we have not required any special properties of the elements a that represent, e.g., preference relations. Instead of transitivity or irreflexivity we need an assumption that such elements admit "enough" maximal objects. This is expressed by requiring every non-maximal object to be dominated by some maximal one. In a setting with finitely many objects, such as a database, and a preference relation on them this property is always satisfied and hence is no undue restriction for our purposes. We forego a discussion of this assumption for infinite sets of objects, since there it is related to fundamental issues such as Zorn's Lemma and Hausdorff's maximality principle, hence to the axiom of choice.

Definition 4.7. We call an element $a :: T^2$ *normal* if it satisfies $\neg(a \rhd 1_T) \leq \langle a \rangle (a \rhd 1_T)$.

This is a compact algebraic formulation of the above domination requirement. By Lemma 4.2.1 it is equivalent to $\ulcorner a \leq \langle a \rangle \neg \ulcorner a$.

First we show that any relation on a subset of the domain of a normal relation provides a prefilter.

Theorem 4.8. *Assume* $a, b :: T^2$.

1. *Let* a *be normal. Then* $\ulcorner b \leq \ulcorner a \Rightarrow b \,\mathsf{pref}\, a$.
2. *Let* $a + b$ *be normal. Then* $a \,\mathsf{pref}\, (a + b)$.

Proof.

1. The assumption about b is the first conjunct of the right hand side in Theorem 4.6. For the second conjunct we calculate

$$\begin{array}{ll} & \mathsf{TRUE} \\ \Leftrightarrow & \quad \{\![\ a \text{ normal }]\!\} \\ & \lceil a \leq \langle a \rangle \neg \lceil a \\ \Rightarrow & \quad \{\![\ \lceil b \leq \lceil a, \text{ contraposition and isotony of diamond }]\!\} \\ & \lceil a \leq \langle a \rangle \neg \lceil b \\ \Rightarrow & \quad \{\![\ x \leq x + y \text{ and transitivity of } \leq\ ]\!\} \\ & \lceil a \leq \lceil b + \langle a \rangle \neg \lceil b\ . \end{array}$$

2. Since $a \leq a + b$, isotony of diamond and hence of domain imply $\lceil a \leq \lceil (a + b)$ and the claim follows from Part 1.

Next we show that under certain conditions prefilters can be nested.

Theorem 4.9. *Assume $a, b, c :: T^2$, where $b \rhd 1_T \leq c \rhd 1_T$ and $b \,\mathsf{pref}\, a$ with normal a. Then also $c \,\mathsf{pref}\, a$.*

Proof. First, by Theorem 4.6 we have $\lceil b \leq \lceil a \wedge \lceil a \leq \lceil b + \langle a \rangle \neg \lceil b$. Second, by Lemma 4.2.1 and contraposition the assumption $b \rhd 1_T \leq c \rhd 1_T$ is equivalent to $\lceil c \leq \lceil b$. Hence by transitivity of $\leq$ we infer $\lceil c \leq \lceil a$. Now normality of a and Theorem 4.8.1 show the claim.

5 Complex Preferences

We have seen how some laws of single preference relations can be proved in point-free style in our algebra.

Now we want to *compose* preferences into *complex preferences*. To this end we will introduce some special operators. The standard semiring operations like multiplication, addition and meet also lead to some kind of complex preferences, but they are rarely used in the typical application domain of preference algebra [Kie02,KEW11]. Instead the so-called *Prioritisation* and *Pareto composition* are the most important constructors for complex preferences.

5.1 Complex Preferences as Typed Relations

To motivate our algebraic treatment we first repeat the definitions of these preference combinators in the concrete setting of typed relations [Kie02].

For basic sets M, N and preference relations $R \subseteq M^2, S \subseteq N^2$ the prioritisation $R \,\&\, S$ is defined as:

$$(x_1, x_2)\,(R \,\&\, S)\,(y_1, y_2) \;\Leftrightarrow_{df}\; x_1\, R\, y_1 \;\vee\; (x_1 = y_1 \;\wedge\; x_2\, S\, y_2)$$

where $x_i \in M, y_i \in N$. The Pareto preference is defined as:

$$(x_1, x_2)\,(R \otimes S)\,(y_1, y_2) \;\Leftrightarrow_{df}\; \begin{array}{l} x_1\,R\,y_1 \wedge (x_2\,S\,y_2 \vee x_2 = y_2) \;\vee \\ x_2\,S\,y_2 \wedge (x_1\,R\,y_1 \vee x_1 = y_1) \end{array}$$

In order theory the prioritisation is well-known as *lexicographical order*.

We now want to get rid of the point-wise notation in favour of operators on relations. The technique is mostly standard; we exemplify it for the prioritisation. We calculate, assuming first $M :: A, N :: B$ with distinct attribute names A, B,

$$\begin{array}{ll} & (x_1, x_2)\,(R \,\&\, S)\,(y_1, y_2) \\ \Leftrightarrow & \quad \{\![\text{ definition }]\!\} \\ & x_1\,R\,y_1 \;\vee\; (x_1 = y_1 \;\wedge\; x_2\,S\,y_2) \\ \Leftrightarrow & \quad \{\![\text{ logic }]\!\} \\ & (x_1\,R\,y_1 \;\wedge\; \mathsf{true}) \;\vee\; (x_1 = y_1 \;\wedge\; x_2\,S\,y_2) \\ \Leftrightarrow & \quad \{\![\text{ definitions of } \top_B \text{ and } 1_A \text{ }]\!\} \\ & (x_1\,R\,y_1 \;\wedge\; x_2\,\top_B\,y_2) \;\vee\; (x_1\;1_A\;y_1 \;\wedge\; x_2\,S\,y_2) \\ \Leftrightarrow & \quad \{\![\text{ definition of cartesian product of relations }]\!\} \\ & (x_1, x_2)\,(R \times \top_B)\,(y_1, y_2) \;\vee\; (x_1, x_2)\,(1_A \times S)\,(y_1, y_2) \\ \Leftrightarrow & \quad \{\![\text{ definition of relational union }]\!\} \\ & (x_1, x_2)\,((R \times \top_B) \cup (1_A \times S))\,(y_1, y_2)\;. \end{array}$$

A similar calculation can be done for the Pareto composition. Now we can write the point-free equations

$$\begin{array}{l} R \,\&\, S \;=\; (R \times \top_B) \cup (1_A \times S)\;, \\ R \otimes S \;=\; (R \times (S \cup 1_B)) \cup ((R \cup 1_A) \times S)\;. \end{array}$$

This is close to an abstract algebraic formulation. However, since we want to cover also the case of non-disjoint, overlapping tuples, we will replace the Cartesian product $\times$ by the join $\bowtie$. From now on a preference x has type T_x, i.e. $a :: T_a^2, b :: T_b^2, ...$

Definition 5.1. For the sake of readability we define for $x :: T^2$:

$$0_x \;=_{df}\; 0_T, \quad 1_x \;=_{df}\; 1_T, \quad \top_x \;=_{df}\; \top_T$$

Definition 5.2 (Prioritisation/Pareto composition of preferences). Assume a join algebra. For $a :: T_a^2, b :: T_b^2$ the *Prioritisation* $a \,\&\, b :: T_a \bowtie T_b$ is defined by

$$a \,\&\, b \;=_{df}\; a \bowtie \top_b + 1_a \bowtie b.$$

The *Pareto compositions* $a \lessdot\otimes b, a \otimes\gtrdot b, a \otimes b :: T_a \bowtie T_b$ are defined by

$$\begin{array}{rl} a \lessdot\otimes b & =_{df}\; a \bowtie (b + 1_b), \\ a \otimes\gtrdot b & =_{df}\; (a + 1_a) \bowtie b, \\ a \otimes b & =_{df}\; a \lessdot\otimes b + a \otimes\gtrdot b. \end{array}$$

$a \lessdot\otimes b$ and $a \otimes\gtrdot b$ are called *left* and *right Semi-Pareto compositions*, while $a \otimes b$ is the standard *Pareto composition*.

Remark 5.3. Under certain circumstances the term $\langle \top_T \rangle\, q$ occurring, for example, in $\langle a \,\&\, b \rangle\, (p \bowtie q)$ can be simplified. Call an idempotent semiring *weakly Tarskian* if for all types T and tests $q :: T$ we have

$$\langle \top_T \rangle\, q = \begin{cases} 1_T & \text{if } q \neq 0_T, \\ 0_T & \text{if } q = 0_T. \end{cases}$$

For instance, the semiring of binary relations is weakly Tarskian. This implies that in a term like $\langle a \bowtie \top_b \rangle\, (q_1 \bowtie q_2)$ with $a :: T_a^2$ the test q_2 is irrelevant as long as $q_2 \neq 0_b$. This is exactly what we want, because $q_1 \bowtie 0_b$ $(= 0_{a \bowtie b})$ is a zero element and must not have successors in any relation.

A semiring with $\top$ is called *Tarskian* when $a \neq 0 \;\Rightarrow\; \top \cdot a \cdot \top = \top$. This property was first stated for the semiring of binary relations (see, e.g., [SS93]). By the standard theory of diamond and domain [DMS06], a Tarskian semiring is also weakly Tarskian, but generally not vice versa.

In our hotel example from the introduction, the user would typically express her preference as the Pareto composition of price and distance to the beach.

The definition of the Pareto compositions immediately yields an important optimisation tool.

Corollary 5.4. *The preferences $a \lessdot\otimes b$ and $a \otimes\gtrdot b$ are prefilters for $a \otimes b$. Likewise, $a \bowtie \top_B$ is a prefilter for $a \,\&\, b$.*

Proof. By definition, $a \lessdot\otimes b, a \otimes\gtrdot b \leq a \otimes b$ and $a \bowtie \top_B \leq a \,\&\, b$; hence Theorem 4.8.1 applies.

Hence, in our hotel example from the introduction, we may prefilter by price or by distance to the beach to speed up the overall filtering. Further applications of this principle are discussed in detail in [End11].

5.2 Maximality for Complex Preferences

We first state the behaviour of the maximality operator for joins of preference elements.

Lemma 5.5. *For $a :: T_a^2, p :: T_a$ and $b :: T_b^2, q :: T_b$ with $T_a \cap T_b = \emptyset$ we have*

$$(a \bowtie b) \triangleright (p \bowtie q) \;=\; (a \triangleright p) \bowtie q + p \bowtie (b \triangleright q)\ .$$

Proof. We observe that, under the disjointness assumption, by Corollary 2.8.3 and a standard law for Cartesian products, for $r :: T_a, s :: T_b$, we have

$$(p \bowtie q) - (r \bowtie s) \;=\; (p - r) \bowtie q \;+\; p \bowtie (q - s)\ .$$

Hence, by the definitions and Lemma 2.10,

$$
\begin{aligned}
& (a \bowtie b) \triangleright (p \bowtie q) \\
= \;& (p \bowtie q) - \langle a \bowtie b) \rangle \, (p \bowtie q) \\
= \;& (p \bowtie q) - (\langle a \rangle \, p \bowtie \langle b \rangle \, q) \\
= \;& (p - \langle a \rangle \, p) \bowtie q + p \bowtie (q - \langle b \rangle \, q) \\
= \;& (a \triangleright p) \bowtie q + p \bowtie (b \triangleright q) \; .
\end{aligned}
$$

Since both prioritisation and Pareto composition are defined as sums of joins, we can now use this together with Lemma 4.2.7, 4.2.1 and the exchange axiom of Definition 3.4.4 to calculate their maximal elements.

Lemma 5.6. *For $a :: T_a^2, p :: T_a$ and $b :: T_b^2, q :: T_b$ with $T_a \cap T_b = \emptyset$ we have*

$$
\begin{aligned}
(a \lessdot\otimes b) \triangleright (p \bowtie q) &= (a \triangleright p) \bowtie q \; , \\
(a \otimes\gtrdot b) \triangleright (p \bowtie q) &= p \bowtie (b \triangleright q) \; , \\
(a \otimes b) \triangleright (p \bowtie q) &= (a \triangleright p) \bowtie (b \triangleright q) \; , \\
(a \,\&\, b) \triangleright (p \bowtie q) &= (a \triangleright p) \bowtie (b \triangleright q) \; .
\end{aligned}
$$

The proofs are straightforward and hence omitted.

Remark 5.7. It follows directly from the above lemma that

$$
(a \,\&\, b) \triangleright (p \bowtie q) \;=\; (b \,\&\, a) \triangleright (q \bowtie p) \;=\; (a \otimes b) \triangleright (p \bowtie q) \, ,
$$

i.e. Pareto composition and Prioritisation are identical on tests of the form $p \bowtie q$.

Note that this does not hold for general tests. Consider, for instance, the basic set $\{0,1\}^2$ and its subset $N =_{df} \{(0,1),(1,0)\}$, both represented by tests. Assume a preference order R_i in the i-th component which fulfills $0 \, R_i \, 1$, for $i = 1, 2$. Then $(R_1 \,\&\, R_2) \triangleright N = \{(1,0)\}$, whereas $(R_1 \otimes R_2) \triangleright N = N$. This does not contradict our above result, since N cannot be represented in the form $L \times M$ with $L, M \subseteq \{0,1\}$.

5.3 Equivalence of Preference Terms

Corollary 5.8. *Let $a :: T_a^2$ and $b, b' :: T_b^2$. Then we have:*

$$
a \,\&\, (b + b') = a \,\&\, b + a \,\&\, b' .
$$

Proof. Follows from definition of & and distributivity of $\bowtie$ over $+$.

Corollary 5.9. *For $a :: T_a^2$ we have $a \lessdot\otimes a = a \otimes\gtrdot a = a \otimes a = a$.*

Proof. $a \lessdot\otimes a =_{df} (a + 1_a) \bowtie a = (a + 1_a) \sqcap a = a$. For Right Semi-Pareto and Pareto an analogous argument shows the claim.

Theorem 5.10. *For $a :: T_a$ we have that $(a \,\&)$ distributes over $\lessdot\otimes, \otimes\gtrdot$ and $\otimes$.*

Proof. Let $b :: T_b^2, c :: T_c^2$. We use the auxiliary equation (see Appendix B.4 for a proof)

$$
a \,\&\, b + 1_{a \bowtie b} = a \,\&\, (b + 1_B). \tag{1}
$$

Now we calculate:

$$
\begin{aligned}
& (a \,\&\, b) \mathbin{\otimes\!\!>} (a \,\&\, c) \\
= \quad & \{\![\text{ definition of } \otimes\!\!> \,]\!\} \\
& (a \,\&\, b + 1_{a \bowtie b}) \bowtie (a \,\&\, c) \\
= \quad & \{\![\text{ equation (1) }]\!\} \\
& (a \,\&\, (b + 1_b)) \bowtie (a \,\&\, c) \\
= \quad & \{\![\text{ definition of } \& \,]\!\} \\
& (a \bowtie \top_b + 1_a \bowtie (b + 1_b)) \bowtie (a \bowtie \top_c + 1_a \bowtie c) \\
= \quad & \{\![\text{ distributivity of } \bowtie \,]\!\} \\
& a \bowtie \top_b \bowtie a \bowtie \top_c \quad + a \bowtie \top_b \bowtie 1_a \bowtie c + \\
& 1_a \bowtie (b + 1_b) \bowtie a \bowtie \top_c + 1_a \bowtie (b + 1_b) \bowtie 1_a \bowtie c \\
= \quad & \{\![\, a \bowtie a = a \text{ and } a \bowtie 1_a = a \sqcap 1_a, \text{ compare Corollary 2.8.2 }]\!\} \\
& a \bowtie \top_b \bowtie \top_c \quad + (a \sqcap 1_a) \bowtie \top_b \bowtie c + \\
& (a \sqcap 1_a) \bowtie (b + 1_b) \bowtie \top_c + 1_a \bowtie (b + 1_b) \bowtie c \\
= \quad & \{\![\, a \sqcap 1_a \leq a,\, c \leq \top_c, \text{ subsumption order }]\!\} \\
& a \bowtie \top_b \bowtie \top_c + 1_a \bowtie (b + 1_b) \bowtie c \\
= \quad & \{\![\, \top_{b \bowtie c} = \top_b \bowtie \top_c, \text{ definition of } \& \,]\!\} \\
& a \,\&\, ((b + 1_b) \bowtie c) \\
= \quad & \{\![\text{ definition } \otimes\!\!> \,]\!\} \\
& a \,\&\, (b \mathbin{\otimes\!\!>} c)
\end{aligned}
$$

A symmetric argument holds for $\lhd\!\!\otimes$, so that $(a \,\&)$ distributes over $\lhd\!\!\otimes$ and $\otimes\!\!>$. Using this we infer the distributivity over $\otimes$, see Appendix B.4 for details.

The proof of this theorem shows that the framework of typed relations is rich enough to prove non-trivial preference term equivalences.

We have proved this theorem using Prover9. The input for the auxiliary equation (1) can be found in Appendix A and the input for the entire theorem is given in [MR12].

Such equivalences are useful for an optimized evaluation of preferences, because the evaluation of an equivalent term may be faster.

6 Conclusion and Outlook

The present work intends to advance the state of the art in formalising preference algebra. Besides the point-wise "semi-formal" proofs by hand that had been used originally we wanted to use automatic theorem provers like Prover9 to get the theorems of preference algebra machine-checked. But we realized that there was no straightforward way to put theorems like the prefilter properties or the distributive law for Prioritisation/Pareto into a theorem prover. Especially for the latter problem, the main reason is that originally the equivalence of preference terms was defined, e.g. in [Kie02], in a very implicit manner: two preference terms are equivalent if and only if the corresponding relations are

identical on the basic set. This definition is not very useful if one tries to find (automatically) general equivalence proofs.

The presented concept of a typed join algebra makes it possible to define such equivalences explicitly: two preference terms are identical, if and only if their algebraic representations are equal in the algebra.

Other theorems for which proofs are necessary do not just involve preference terms, but properties of the maximality operator and prefilters. With the inverse image we have employed a well-known algebraic concept to define the maximality operator in quantifier-free form. This reformulation led us to point-free proofs.

The relevance of this topic stems from the demand for optimizing the evaluation of preference queries (e.g. [KH03,HK05]. The paper [REM+12] presents a practical application of preference algebra, where complex preference terms and huge data sets occur, and therefore optimisation methods are of essential interest.

Our algebra is rich enough to cover the concept of preferences and their complex compositions as well as the application of the maximality operator to them. Simultaneously, the algebra is simple enough to be encoded in theorem provers like PROVER9.

With this we have produced a framework which hopefully will be the first step for a comprehensive algebraic description of preference algebra. Our work also covers some aspects of databases in general and thus contributes to the formal description of database-related problems. A project in which our calculus is applied systematically at a larger scale, using machine assistance, is under way.

Acknowledgements. We are grateful to Jeremy Gibbons (MPC Co-Chair) and the anonymous referees for valuable comments.

References

[BW93] R. Backhouse, J. van der Woude: Demonic Operators and Monotype Factors. Mathematical Structures in Computer Science 3, 417–433 (1993)

[BD97] R. Bird, O. de Moor: Algebra of programming. Prentice Hall 1997

[BKS01] S. Börzsönyi, D. Kossmann, K. Stocker: The Skyline Operator. In Data Engineering (ICDE '01), 421–430, 2001.

[DMS06] J. Desharnais, B. Möller, G. Struth: Kleene Algebra With Domain. In ACM Transactions on Computational Logic 7, 798–833, 2006.

[End11] M. Endres: Semi-Skylines and Skyline Snippets - Theory and Applications. Fakultät für Angewandte Informatik, Universität Augsburg, Dissertation. Books on Demand GmbH, Norderstedt, ISBN: 978-3-8423-5246-9, 2011.

[Kan90] P. Kanellakis: Elements of Relational Database Theory. In Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics (B), 1073–1156, 1990.

[KEW11] W. Kießling and M. Endres and F. Wenzel: The Preference SQL System – An Overview. In IEEE Data Eng. Bull., Vol. 34 (2), 11–18, 2011.

[KH03] W. Kießling, B. Hafenrichter: Algebraic Optimization of Relational Preference Queries, Technical Report 2003-1, University of Augsburg, 2003.

[HK05] B. Hafenrichter, W. Kießling: Optimization of Relational Preference Queries. In H. Williams, G. Dobbie (eds.): Database Technologies 2005, Proc. Sixteenth Australasian Database Conference, ADC 2005, Newcastle, Australia, January 31st – February 3rd 2005. CRPIT 39 Australian Computer Society, 175–184, 2005.

[Kie02] W. Kießling: Foundations of Preferences in Database Systems. In Very Large Databases (VLDB '02), 311–322, 2002.

[Koz98] Dexter Kozen. Typed Kleene algebra. Technical Report TR98-1669, Computer Science Department, Cornell University, March 1998.

[MO04] W. MacCaull, Ewa Orłowska: A Calculus of Typed Relations. In R. Berghammer, B. Möller, G. Struth (eds.): Relational and Kleene-Algebraic Methods in Computer Science. Lecture Notes in Computer Science 3051, 191–201, 2004.

[MB85] E. Manes, D. Benson: The Inverse Semigroup of a Sum-Ordered Semiring. In Semigroup Forum 31, 129–152, 1985.

[MR12] B. Möller, P. Roocks: Proof of the Distributive Law for Prioritisation and Pareto Composition `http://www.informatik.uni-augsburg.de/lehrstuehle/dbis/pmi/staff/roocks/publications/distributivity_proof.pdf`

[REM+12] P. Roocks, M. Endres, S. Mandl, W. Kießling: Composition and Efficient Evaluation of Context-Aware Preference Queries. To appear in Database Systems for Advanced Applications, 2012.

[SS93] G. Schmidt, T. Ströhlein: Relations and Graphs: Discrete Mathematics for Computer Scientists. In EATCS Monographs on Theoretical Computer Science, 1993.

[SHW97] G. Schmidt, C. Hattensperger, M. Winter: Heterogeneous relation algebra. In C. Brink, W. Kahl, G. Schmidt (eds): Relational Methods in Computer Science. Advances in Computer Science, Springer Vienna, 39–53, 1997.

A Sample Prover Input

For $a :: T_a^2$ and $b :: T_b^2$ we show the auxiliary equation (1) from Theorem 5.10:

$$a \,\&\, b + 1_{a \bowtie b} = a \,\&\, (b + 1_b).$$

We use the following operators:

Prover-Input	mathematically
`a typed T_a`	$a :: T_a^2$
`a join b`	$a \bowtie b$
`T_a tjoin T_b`	$T_a \bowtie T_b$
`a prior b`	$a \,\&\, b$
`a + b`	$a + b$

The assumptions are given as follows:

```
% all elements are typed
exists T (x typed T).

% addition is associative, commutative and idempotent
```

```
(x + y) + z = x + (y + z).
x + y = y + x.
x + x = x.

% addition preserves type
x typed z & y typed z -> (x+y) typed z.

% subsumption order
x <= y <-> y = x + y.

% top is greatest element
x typed z -> x <= top(z).

% typing of top
top(z) typed z.
top(z1 tjoin z2) = top(z1) join top(z2).

% typing of one
one(z) typed z.
one(z1 tjoin z2) = one(z1) join one(z2).

% abbreviated typing
x typed z -> top(x) = top(z).
x typed z -> one(x) = one(z).

% distributivity of the join over addition
x join (y1 + y2) = x join y1 + x join y2.

% typing of join
x typed z1 & y typed z2 -> (x join y) typed (z1 tjoin z2).

% prioritisation (without resulting type)
x prior y = x join top(y) + one(x) join y.
```

Finally our goal is:

```
% auxiliary equation for distributive law
u prior v + one(u join v) = u prior (v + one(v)).
```

The entire input for the proof of theorem 5.10 can be found in [MR12].

B Proofs

In the proofs of section 4 we omit the type-index of 1 and assume type-compatibility.

B.1 Proof of Lemma 4.2

1. Immediate from the definitions and $1 - q = \neg q$.
2. Immediate from Part 1 and shunting.
3. $$\begin{aligned} & a \triangleright p \\ = \quad & \{\![\text{ definitions of } \triangleright \text{ and } - \,]\!\} \\ & p \cdot \neg \langle a \rangle p \\ \leq \quad & \{\![\text{ property of intersection }]\!\} \\ & p \ . \end{aligned}$$

4. $a \triangleright 1 \leq a \triangleright p$

$\Leftrightarrow$ {[definition of $\triangleright$ and Part 1]}

$\neg \ulcorner a \leq p - \langle a \rangle\, p$

$\Leftrightarrow$ {[definition of $-$ and universal property of intersection]}

$\neg \ulcorner a \leq p \quad \wedge \quad \neg \ulcorner a \leq \neg\, \langle a \rangle\, p$

$\Leftrightarrow$ {[shunting in second conjunct]}

$\neg \ulcorner a \leq p \quad \wedge \quad \langle a \rangle\, p \leq \ulcorner a$

$\Leftrightarrow$ {[second conjunct true by (3.1)]}

$\neg \ulcorner a \leq p$

$\Leftrightarrow$ {[definition of $\triangleright$]}

$a \triangleright 1 \leq p$.

5. Immediate from the previous property by setting $p = a \triangleright 1$.

6. $a \triangleright (a \triangleright p)$

$=$ {[definition of $\triangleright$]}

$(p - \langle a \rangle\, p) - \langle a \rangle\, (p - \langle a \rangle)$

$=$ {[property of difference]}

$p - (\langle a \rangle\, p + \langle a \rangle\, (p - \langle a \rangle))$

$=$ {[distributivity of $\langle\rangle$]}

$p - \langle a \rangle\, (p + (p - \langle a \rangle))$

$=$ {[since $p - \langle a \rangle \leq p$]}

$p - \langle a \rangle\, p$

$=$ {[definition of $\triangleright$]}

$a \triangleright p$.

7. $(a + b) \triangleright p$

$=$ {[definition of $\triangleright$]}

$p - \langle a + b \rangle\, p$

$=$ {[distributivity of $\langle\rangle$]}

$p - (\langle a \rangle\, p + \langle b \rangle\, p)$

$=$ {[property of difference]}

$(p - \langle a \rangle\, p) \cdot (p - \langle b \rangle\, p)$

$=$ {[definition of $\triangleright$]}

$(a \triangleright p) \cdot (b \triangleright p)$.

8. Assume $b \leq a$, i.e., $b + a = a$.

$a \triangleright p$

$=$ {[assumption]}

$b + a \triangleright p$

$=$ {[previous property]}

$$
\begin{array}{ll}
& (b \rhd p) \cdot (a \rhd p) \\
\leq & \quad \{\!\![\text{ property of intersection }]\!\!\} \\
& b \rhd p \ .
\end{array}
$$

9. By isotony of the diamond we have $p = \langle 1 \rangle\, p \leq \langle a \rangle\, p$ and hence $a \rhd p = p - \langle a \rangle\, p = 0$.

B.2 Proof of Lemma 4.3

$$
\begin{aligned}
q = (p + \neg p) \cdot q = \overbrace{p \cdot q}^{=0} + \neg p \cdot q \leq \neg p \\
\neg p = \underbrace{(p + q)}_{=1} \cdot \neg p = p \cdot \neg p + q \cdot \neg p \leq q
\end{aligned}
$$

By antisymmetry we have $\neg p = q$.

B.3 Proof of Theorem 4.6

We split the left-hand side of the claim equivalently into

$$
b \,\mathsf{pref}\, a \;\Leftrightarrow\; a \rhd 1 \leq a \rhd (b \rhd 1) \;\wedge\; a \rhd (b \rhd 1) \leq a \rhd 1 \ .
$$

By Parts 4 and 2 of Lemma 4.2 the first conjunct is equivalent to $\lceil b \leq \lceil a$. For the second conjunct we calculate

$$
\begin{array}{ll}
& a \rhd (b \rhd 1) \leq a \rhd 1 \\
\Leftrightarrow & \quad \{\!\![\text{ definition of } \rhd \text{ and Lemma 4.2.1 }]\!\!\} \\
& \neg\lceil b - \langle a \rangle\, \neg\lceil b \leq \neg\lceil a \\
\Leftrightarrow & \quad \{\!\![\text{ contraposition and De Morgan }]\!\!\} \\
& \lceil a \leq \lceil b + \langle a \rangle\, \neg\lceil b \ .
\end{array}
$$

B.4 Proof of Theorem 5.10

Auxiliary equation (1):

$$
\begin{array}{ll}
& a \,\&\, b + 1_{a \bowtie b} \\
= & \quad \{\!\![\text{ definition of } \& \ ]\!\!\} \\
& a \bowtie \top_b + 1_a \bowtie b + 1_a \bowtie 1_b \\
= & \quad \{\!\![\text{ distributivity of } \bowtie \ ]\!\!\} \\
& a \bowtie \top_b + 1_a \bowtie (b + 1_b) \\
= & \quad \{\!\![\text{ definition of } \& \ ]\!\!\} \\
& a \,\&\, (b + 1_b)
\end{array}
$$

Distributivity of $(a \,\&)$ over $\otimes$:

$$
\begin{array}{ll}
 & a \mathbin{\&} (b \otimes c) \\
= & \quad \{\![\text{ definition of } \otimes \ ]\!\} \\
 & a \mathbin{\&} (b \mathbin{\overleftarrow{\otimes}} c + b \mathbin{\overrightarrow{\otimes}} c) \\
= & \quad \{\![\text{ distributivity of } \& \text{ over } +, \text{ cor. } 5.8 \ ]\!\} \\
 & a \mathbin{\&} (b \mathbin{\overleftarrow{\otimes}} c) + a \mathbin{\&} (b \mathbin{\overrightarrow{\otimes}} c) \\
= & \quad \{\![\text{ distributivity of } (a \mathbin{\&}) \text{ over } \overleftarrow{\otimes} \text{ and } \overrightarrow{\otimes} \ ]\!\} \\
 & (a \mathbin{\&} b) \mathbin{\overleftarrow{\otimes}} (a \mathbin{\&} c) + (a \mathbin{\&} b) \mathbin{\overrightarrow{\otimes}} (a \mathbin{\&} c) \\
= & \quad \{\![\text{ definition of } \otimes \ ]\!\} \\
 & (a \mathbin{\&} b) \otimes (a \mathbin{\&} c)
\end{array}
$$