The θ-Join as a Join with θ

Jules Desharnais[1] and Bernhard Möller[2]

[1] Université Laval, Québec
[2] Universität Augsburg

Abstract. We present an algebra for the classical database operators. Contrary to most approaches we use (inner) join and projection as the basic operators. Theta joins result by representing theta as a database table itself and defining theta-join as a join with that table. The same technique works for selection. With this, (point-free) proofs of the standard optimisation laws become very simple and uniform. The approach also applies to proving join/projection laws for preference queries. Extending the earlier approach of [16], we replace disjointness assumptions on the table types by suitable consistency conditions. Selected results have been machine-verified using the CalcCheck tool.

1 Introduction

The paper deals with an algebra for the classical operators of relational algebra as used in databases. While in most approaches the join operator is defined as a combination of direct product, selection and projection, we take a different approach, using (inner) join and projection as the basic operators. Theta joins are incorporated by simply representing (mathematically) theta as a database table itself and defining theta-join as a join with that table. The same can be done with selection by representing the corresponding condition as the table of all tuples that satisfy it. With this, (point-free) proofs of the standard laws become very simple and uniform. The approach is also suitable for proving join/projection laws for preference queries.

The paper builds upon [16]. While many of the laws there required disjointness of the types of the tables involved, we are here more general and replace disjointness of types by suitable consistency conditions. Technically, we extend the techniques there by deploying variants of the split and glue operators introduced in [3,4]. This allows point-free formulations of the new conditions and corresponding point-free proofs of the ensuing laws. Selected results have been machine-verified using the CalcCheck tool [9,10].

2 Preliminaries

Our approach is based on the algebra of binary relations, see e.g. [17]. A *binary relation* between sets M and N is a subset $R \subseteq M \times N$. We denote the empty relation $\emptyset$ by 0 and the universal relation $M \times N$ by $\mathsf{T}_{M\times N}$, omitting the subscript when it is clear from the context. Domain, codomain and relational composition ; are defined as usual, the latter binding stronger than union and intersection. The *converse* of R is $R^{\smile} \subseteq N \times M$, given by $R^{\smile} = \{(y,x) \mid (x,y) \in R\}$.

If $M = N$ then R is called *homogeneous*. In this case there is the *identity relation* $1_M = \{(x,x) \mid x \in M\}$, which is neutral w.r.t. ;. If M is clear from the context we omit the subscript $_M$.

A *test* over M is a sub-identity $P \subseteq 1$ which encodes the subset $\{x \mid (x,x) \in P\}$. The *negation* $\neg P$ of test P is the complement of P relative to 1, i.e., $1 - P$, where $-$ is set difference. It encodes the complement of the set encoded by P. When convenient we do not distinguish between tests and the encoded sets.

Domain and codomain can be encoded as the tests

$$\ulcorner R = R \,;\, \mathsf{T}_{N \times M} \cap 1_M \ , \qquad R^{\urcorner} = \mathsf{T}_{M \times N} \,;\, R \cap 1_N \ . \tag{1}$$

We list a few properties of domain; symmetric ones hold for the codomain operator which, however, we do not use in this paper. For proofs see [6].

Lemma 2.1 *Consider relations R, S and test P.*

1. $\ulcorner(R \cup S) = \ulcorner R \cup \ulcorner S$. *Hence* $\ulcorner$ *is isotone, i.e., monotonically increasing, w.r.t.* $\subseteq$.
2. $\ulcorner R \,;\, R = R$ *and* $\neg \ulcorner R \,;\, R = 0$.
3. $\ulcorner P = P$ (stability)
4. $\ulcorner R = 0 \Leftrightarrow R = 0$. (full strictness)
5. $\ulcorner(P \,;\, R) = P \,;\, \ulcorner R$. (import/export)
6. $\ulcorner(R \,;\, S) = \ulcorner(R \,;\, \ulcorner S)$. (locality)
7. $R \,;\, P \cap S \,=\, (R \cap S) \,;\, P \,=\, R \cap S \,;\, P$. (restriction)

3 Typed Tuples

In this section we present the formal model of database objects as typed tuples. The types represent attributes, i.e., columns of a database relation. Conceptually and notationally, we largely base on [11].

Definition 3.1 Let $\mathcal{A}$ be a set of *attribute names* and $(D_A)_{A \in \mathcal{A}}$ be a family of nonempty sets, where for $A \in \mathcal{A}$ the set D_A is called the *domain* of A.

1. The set $\mathcal{U} =_{df} \bigcup\limits_{A \in \mathcal{A}} D_A$ is called the *universe*.
2. A *type* T is a subset $T \subseteq \mathcal{A}$.
3. A T*-tuple* is a mapping $t : T \to \mathcal{U}$ where $\forall\, A \in T : t(A) \in D_A$. For $T = \emptyset$ the only T-tuple is the empty mapping $\emptyset$.
4. The domain D_T for a type T is the set of all T-tuples, i.e., the Cartesian product $D_T = \prod\limits_{A \in T} D_A$.
5. 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'. By this $\pi_\emptyset(t) = \emptyset$. Projections π are not to be confused with the Cartesian product operator $\prod$.
6. A set of tuples of the same type is called a *table* and is relationally encoded as a test.

7. For a tuple t and a table P of T-tuples we introduce the abbreviations

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

Definition 3.2 Two tuples $t_i :: T_i$ $(i = 1, 2)$ are called *matching*, in signs $t_1 \# t_2$, iff $\pi_T(t_1) = \pi_T(t_2)$, where $T =_{df} T_1 \cap T_2$. In this case we define $t_1 \bowtie t_2 =_{df} t_1 \cup t_2$. The join of nonmatching tuples is undefined. If $T = \emptyset$, i.e., the types T_i are disjoint, then the t_i are trivially matching. The empty tuple $\emptyset$ matches every tuple and hence is the neutral element of $\bowtie$.

The *join* of two types T_1, T_2 is the union of their attributes, i.e., $T_1 \bowtie T_2 =_{df} T_1 \cup T_2$. For tables $P_i :: T_i$ $(i = 1, 2)$, the join $\bowtie$, binding stronger than union and intersection, is defined as the set of all matching combinations of P_i-tuples:

$$\begin{array}{rl} P_1 \bowtie P_2 =_{df} & \{t :: T_1 \bowtie T_2 \mid \pi_{T_i}(t) \in P_i \,(i = 1, 2)\} \\ = & \{t_1 \bowtie t_2 \mid t_i \in P_i \,(i = 1, 2), t_1 \# t_2\} \;. \end{array}$$

When we want to avoid numerical indices we use the convention that table P has type T_P, etc. The table $\{\emptyset\}$ is the neutral element of $\bowtie$ on tables.

Lemma 3.3 $D_{T_1 \bowtie T_2} = D_{T_1} \bowtie D_{T_2}$. *Hence* $T_2 \subseteq T_1 \;\Rightarrow\; D_{T_1} \bowtie D_{T_2} = D_{T_1}$.

Proof. Immediate from the definition of type join and Def. 3.1.4. □

Lemma 3.4 *Consider tables* $P :: T_P, Q :: T_Q$ *and an arbitrary type* T'.

1. *Every tuple is characterised by its projections: for* $t \in D_{T_P \bowtie T_Q}$ *we have* $t = \pi_{T_P}(t) \bowtie \pi_{T_Q}(t)$. *For* $t, u \in D_{T_P \bowtie T_Q}$ *this entails* $t = u \Leftrightarrow \pi_{T_P}(t) = \pi_{T_P}(u) \wedge \pi_{T_Q}(t) = \pi_{T_Q}(u)$.
2. *Projection sub-distributes over join:* $\pi_{T'}(P \bowtie Q) \subseteq \pi_{T'}(P) \bowtie \pi_{T'}(Q)$.
3. *If* $T_P \cap T_Q = \emptyset$ *then this strengthens to an equality.*

1. Straightforward calculation.
2. By distributivity of restriction over union, for any two matching tuples t_1, t_2 (not necessarily from P, Q) we have $\pi_{T'}(t_1 \bowtie t_2) = \pi_{T'}(t_1) \bowtie \pi_{T'}(t_2)$. Hence if we take matching $t_1 \in P, t_2 \in Q$, then $t =_{df} \pi_{T'}(t_1 \bowtie t_2) \in \pi_{T'}(P \bowtie Q)$. Because $\pi_{T'}(t_1 \bowtie t_2) = \pi_{T'}(t_1) \bowtie \pi_{T'}(t_2)$, also $t \in \pi_{T'}(P) \bowtie \pi_{T'}(Q)$.
3.
$$\begin{array}{ll} & t \in \pi_{T'}(P) \bowtie \pi_{T'}(Q) \\ \Leftrightarrow & \exists\, u, v : u \in P \wedge v \in Q \wedge t = \pi_{T'}(u) \bowtie \pi_{T'}(v) \quad \{\!\![\text{ definition of join }]\!\!\} \\ \Rightarrow & \exists\, u, v : u \in P \wedge v \in Q \wedge t = \pi_{T'}(u \bowtie v) \quad \{\!\![\; u \# v \text{ by } T_P \cap T_Q = \emptyset \;]\!\!\} \\ \Rightarrow & t \in \pi_{T'}(P \bowtie Q) \quad \{\!\![\text{ definitions }]\!\!\} \end{array}$$
□

Lemma 3.5 *For* $P_i :: T_i$ $(i = 1, 2)$ *with disjoint* T_i, *i.e., with* $T_1 \cap T_2 = \emptyset$, *the join* $P_1 \bowtie P_2$ *is isomorphic to the Cartesian product of* P_1 *and* P_2.

Proof. For $t \in P_1 \bowtie P_2$, the conditions $\pi_{T_i}(t) \in P_i (i = 1, 2)$ are independent. Hence all elements of P_1 can be joined with all elements of P_2. Thus, by definition,

$$t \in P_1 \bowtie P_2 \;\Leftrightarrow\; \pi_{T_1}(t) \in P_1 \wedge \pi_{T_2}(t) \in P_2 \;\Leftrightarrow\; (\pi_{T_1}(t), \pi_{T_2}(t)) \in P_1 \times P_2 \;. \quad \square$$

Lemma 3.6 [16] *The following laws hold:*

1. $\bowtie$ *is associative, commutative and distributes over* $\cup$.
2. $\bowtie$ *is isotone in both arguments.*
3. *Assume* $P_i, Q_i :: T_i$ $(i = 1, 2)$. *Then the following interchange law holds:*

$$(P_1 \cap Q_1) \bowtie (P_2 \cap Q_2) = (P_1 \bowtie P_2) \cap (Q_1 \bowtie Q_2) \ .$$

4. *For* $P, Q :: T$ *we have* $P \bowtie Q = P \cap Q$. *In particular,* $P \bowtie P = P$.

4 The θ-Join

For simplicity we restrict ourselves to θ-joins with binary relations θ. Assume tables $P :: T_P, Q :: T_Q$ with $T_P \cap T_Q = \emptyset$ as well as $A \in T_P, B \in T_Q$ and a binary relation $\theta \subseteq D_A \times D_B$. Note that the assumptions imply $A \neq B$[3]. We want to model an expression that in standard database theory would be written "$P \bowtie_{\theta(P.A,Q.B)} Q$". The corresponding table contains exactly those tuples t of table $P \bowtie Q$ in which the values $t(A) \in P.A$ and $t(B) \in Q.B$ (remember that t is a function from attribute names to values) are in relation θ.

The idea is to consider θ mathematically again as table of type $A \bowtie B$. Then the above expression can simply be represented as $P \bowtie \theta \bowtie Q$.

Example 4.1 Here is a simple database of persons and ages with $>$ as θ.

P:

Name1	Age1
A	50
B	55
C	60

$<$:

Age1	Age2
50	55
50	60
55	60

Q:

Age2	Name2
50	E
55	F
55	G

$P \bowtie Q$:

Name1	Age1	Age2	Name2
A	50	50	E
A	50	55	F
A	50	55	G
B	55	50	E
B	55	55	F
B	55	55	G
C	60	50	E
C	60	55	F
C	60	55	G

$P \bowtie <$:

Name1	Age1	Age2
A	50	55
A	50	60
B	55	60

$< \bowtie Q$:

Age1	Age2	Name2
50	55	F
50	55	G

$P \bowtie < \bowtie Q$:

Name1	Age1	Age2	Name2
A	50	55	F
A	50	55	G

$\square$

[3] With this we follow the SQL standard. Note, however, that $P \bowtie \theta \bowtie Q$ is defined even if this disjointness condition does not hold. It is not even necessary to require $A \neq B$, although having $A = B$ is not interesting.

We use our view of the θ-join for algebraic proofs of two standard optimisation rules for projections applied to joins.

Theorem 4.2
1. *If* $Q :: L \subseteq T_P$ *then* $\pi_L(P \bowtie Q) = \pi_L(P) \bowtie Q$.
2. *Assume* $T_P \cap T_Q = \emptyset$ *and* $\theta :: L$ *for some* $L \subseteq T_P \cup T_Q$. *This means that* θ *is to provide the "glue" between the type-disjoint* P *and* Q. *Set* $L_P =_{df} T_P \cap L$ *and* $L_Q =_{df} T_Q \cap L$. *Then we have the transformation rule*

$$\pi_L(P \bowtie \theta \bowtie Q) = \pi_{L_P}(P) \bowtie \theta \bowtie \pi_{L_Q}(Q) \qquad (\textit{push projection over join}).$$

Proof.
1. ($\subseteq$) Immediate from Lm. 3.4.2 and $\pi_L(Q) = Q$ by $Q :: L$.
($\supseteq$)

$$\begin{array}{ll}
\quad \pi_L(P) & \\
= \pi_L(P \bowtie D_P) & \{\!\![\text{ definition of } D_P \,]\!\!\} \\
= \pi_L(P \bowtie D_P \bowtie D_L) & \{\!\![\text{ assumption } L \subseteq T_P, \text{ definition of } D_L \,]\!\!\} \\
= \pi_L(P \bowtie D_L) & \{\!\![\text{ definition of } D_P \,]\!\!\} \\
= \pi_L(P \bowtie (Q \cup \overline{Q})) & \{\!\![\text{ Boolean algebra, setting} \\
& \quad \overline{X} =_{df} D_L - X \text{ for } X :: L \,]\!\!\} \\
= \pi_L(P \bowtie Q) \cup \pi_L(P \bowtie \overline{Q})) & \{\!\![\text{ distributivity of join and projection }]\!\!\} \\
\subseteq \pi_L(P \bowtie Q) \cup (\pi_L(P) \bowtie \pi_L(\overline{Q})) & \{\!\![\text{ Lm. 3.4.2 }]\!\!\} \\
= \pi_L(P \bowtie Q) \cup (\pi_L(P) \cap \pi_L(\overline{Q})) & \{\!\![\text{ Lm 3.6.4 }]\!\!\} \\
\subseteq \pi_L(P \bowtie Q) \cup \pi_L(\overline{Q}) & \{\!\![\text{ Boolean algebra }]\!\!\} \\
= \pi_L(P \bowtie Q) \cup \overline{Q} & \{\!\![\ \pi_L(\overline{Q}) = \overline{Q} \text{ by } \overline{Q} :: L \,]\!\!\}
\end{array}$$

By Lm 3.6.4 and shunting we obtain from this $\pi_L(P) \bowtie Q = \pi_L(P) \cap Q \subseteq \pi_L(P \bowtie Q)$.

2.

$$\begin{array}{ll}
\quad \pi_L(P \bowtie \theta \bowtie Q) & \\
= \pi_L(P \bowtie Q \bowtie \theta) & \{\!\![\text{ associativity and commutativity of } \bowtie \,]\!\!\} \\
= \pi_L(P \bowtie Q) \bowtie \theta & \{\!\![\text{ Part 1 }]\!\!\} \\
= \pi_L(P) \bowtie \pi_L(Q) \bowtie \theta & \{\!\![\text{ assumption } T_P \cap T_Q = \emptyset \text{ with Lm. 3.4.3 }]\!\!\} \\
= \pi_L(P) \bowtie \theta \bowtie \pi_L(Q) & \{\!\![\text{ associativity and commutativity of } \bowtie \,]\!\!\} \\
= \pi_{L_P}(P) \bowtie \theta \bowtie \pi_{L_Q}(Q) & \{\!\![\ P :: T_P, Q :: T_Q, \text{ definition of projection }]\!\!\} \quad \square
\end{array}$$

5 Selection as Join

Since the representation of the θ-join as a join with θ has proved useful, we will now treat selection $\sigma_C(P)$ for table P and condition C analogously. A *condition*, i.e., a predicate on tuples, is simply represented as a subset $C \subseteq D_L$ for some type L, which means $C :: L$. Conjunction and disjunction of $C, C' :: L$ are then represented by $C \bowtie C'$ and $C \cup C'$, resp. (see Lm. 3.6.4). For $P :: T$ and $L \subseteq T$, we can now just set $\sigma_C(P) =_{df} P \bowtie C$.

Lemma 5.1 *Assume again* $P :: T$.

1. *Selections commute, i.e.,* $\sigma_C(\sigma_{C'}(P)) = \sigma_{C'}(\sigma_C(P))$.
2. *Selections can be combined, i.e.,* $\sigma_C(\sigma_{C'}(P)) = \sigma_{C \bowtie C'}(P)$.
3. *If* C *uses only attributes from* $L \subseteq T$, *i.e.,* $C \subseteq D_L$, *then* $\pi_L(\sigma_C(P)) = \sigma_C(\pi_L(P))$.

Proof.
1. Immediate from associativity/commutativity of $\bowtie$.
2. Ditto.
3. By definitions, Th. 4.2.1 and definitions again:

$$\pi_L(\sigma_C(P)) = \pi_L(P \bowtie C) = \pi_L(P) \bowtie C = \sigma_C(\pi_L(P)) \ . \qquad \square$$

6 Inverse Image and Maximal Elements

The tools developed in the preceding sections will now be applied to a subfield of database theory, namely to preference queries. They serve to remedy a well known problem for queries with *hard constraints*, by which the objects sought in the database are clearly and sharply characterised. If there are no exact matches the empty result set is returned, which is very frustrating for users.

Instead, over the last decade queries with *soft constraints* have been studied. These arise from a formalisation of the *user's preferences* in the form of partial strict orders [12,13]. Instead of returning an empty result set, one can then present the user with the maximal or "best" tuples w.r.t. her preference order.

We now show how to express the maximality operator algebraically and then prove a sample optimisation rule for it. The idea has already been described thoroughly in the predecessor paper [16]; hence we only give a brief presentation of it. After that we develop substantially new laws for it. The main ingredient is an inverse image operator on relations.

Definition 6.1 For a type T a *T-relation* is a homogeneous binary relation R on D_T; we abbreviate this by $R :: T^2$. In analogy to the notation in Sect. 2 we also write T_T instead of $D_T \times D_T$. For a relation $R :: T^2$ the *image* of a test $P :: T$ under R is obtained using the forward diamond operator as

$$|R\rangle P \ =_{df} \{(x,x) \mid \exists y \in P : x\, R\, y\} \ = \ \ulcorner(R\, ;P) \ .$$

Two immediate consequences of the definition and Lm. 2.1 are

$$|0\rangle P = 0 \ , \qquad |\mathsf{T}\rangle P = \begin{cases} D_T & \text{if } P \neq 0 \ , \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The inverse image of a set P under a relation R consists of the elements that have an R-successor in P, i.e., are R-related to some object in P. Assume that R is a strict order (irreflexive and transitive), which is the case in our application domain of preferences. Then the inverse image of P consists of the tuples *dominated* by some tuple in P. This allows the following definition.

Definition 6.2 For a relation $R :: T^2$ and a set $P :: T$ the R-*maximal* objects of P form the relative complement of the set of R-dominated objects, viz.

$$R \rhd P \ =_{df} \ P \cap \neg |R\rangle P \ .$$

The mnemonic behind the $\rhd$ symbol is that in an order diagram for a preference relation R the maximal objects within P are the peaks in P; rotating the diagram clockwise by $90°$ puts the peaks to the right. Hence $R \rhd P$ might also be read as "R-peaks in P". From (2) we obtain

$$0 \rhd P = P \ , \qquad \mathsf{T} \rhd P = 0 \ . \tag{3}$$

A central ingredient for the preference approach is a possibility for defining complex preference relations out of simpler ones. An example would be "I prefer cars that are green and, equally important, have low fuel consumption". The following sections deal with such construction mechanisms, notably with the join of relations.

7 The Join of Relations

Definition 7.1 The *join*$R_1 \bowtie R_2 :: (T_1 \bowtie T_2)^2$ of relations $R_i :: T_i^2$ $(i = 1, 2)$ is

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

Example 7.2 We model the above simple database of cars. Consider the set $\mathcal{A} = \{\mathrm{Col}, \mathrm{Fuel}\}$ of attribute names with $D_{\mathrm{Col}} = \{\mathsf{black}, \mathsf{blue}, \mathsf{green}, \mathsf{red}, \mathsf{white}\}$ and $D_{\mathrm{Fuel}} = \{4.0, 4.1, \ldots, 9.9, 10.0\}$. The comparison relation R_{Col} is given by the Hasse diagram

$$\begin{tikzcd} & \mathsf{green} \arrow[dl, dash] \arrow[d, dash] \arrow[dr, dash] & \\ \mathsf{blue} \arrow[dr, dash] & \mathsf{red} \arrow[d, dash] & \mathsf{white} \arrow[dl, dash] \\ & \mathsf{black} & \end{tikzcd}$$

while as R_{Fuel} we choose $>$. A user uttering the preference R_{Col} does not like black at all, likes green best and otherwise is indifferent about $\mathsf{blue}, \mathsf{red}, \mathsf{white}$. Hence $s\,(R_{\mathrm{Col}} \bowtie R_{\mathrm{Fuel}})\,t$ iff the colour of t is closer to green than that of s and the fuel value of t is less than that of s. □

Definition 7.3 Based on join we can define the two standard preference constructors $\otimes$ of *Pareto* and & of *prioritised* composition as

$$\begin{aligned} R \otimes S \ &=_{df} \ (R \bowtie (1 \cup S)) \cup ((1 \cup R) \bowtie S) \ , \\ R \& S \ &=_{df} \ (R \bowtie \mathsf{T}) \cup (1 \bowtie S) \ . \end{aligned}$$

Pareto composition corresponds to the product order on pairs, with two variations: it does not consider pairs, but tuples from which parts are extracted by the projections involved in $\bowtie$; moreover, it is more liberal than the product of strict orders, since it also admits equality in one part of the tuples as long as there is a strict order relation between the other parts. Prioritised composition corresponds to the lexicographic order on pairs.

We seek a set of algebraic laws that allow proving optimisation rules similar to "push projection over join" from Th. 4.2.2. As an example consider tables $P :: T_P, Q :: T_Q$ and a preference relation $R :: T_P^2$. Then we would like to show

$$(R \bowtie \mathsf{T}_Q) \triangleright (P \bowtie Q) = (R \triangleright P) \bowtie Q \tag{4}$$

under suitable side conditions on P, Q, R. The preference $R \bowtie \mathsf{T}_Q$, which also occurs as a part of the & constructor, expresses that the user does not care about the attributes in T_Q and is only interested in the T_P part. Therefore the preference query can be pushed to that part as shown on the right hand side. This may speed up the query evaluation considerably.

To achieve the mentioned algebraic laws we need to investigate the interaction between the $\bowtie$ and $\triangleright$ operators involved. Of particular importance are so-called interchange laws: the above rule can, by (3), be written as

$$(R \bowtie \mathsf{T}_Q) \triangleright (P \bowtie Q) = (R \triangleright P) \bowtie (0 \triangleright Q) \ ;$$

a maximum between joins is equal to a join between maxima[4].

8 Split, Glue and Pair Relations

To formulate and prove rules about the join of relations in an algebraic style we bring the pointwise definition into a more manageable point-free form. For this we deploy techniques from [3,4]. First we introduce relations for connecting tuples and pairs of tuples.

Definition 8.1 For types T_1, T_2 we define *split* $\prec$ and its converse *glue* $\succ$ with the functionalities

$$\begin{aligned} {}_{T_1 \bowtie T_2}\!\prec_{T_1 \times T_2} &\subseteq D_{T_1 \bowtie T_2} \times (D_{T_1} \times D_{T_2}) \ , \\ {}_{T_1 \times T_2}\!\succ_{T_1 \bowtie T_2} &\subseteq (D_{T_1} \times D_{T_2}) \times D_{T_1 \bowtie T_2} \ . \end{aligned}$$

Again we suppress the type indices for readability. The behaviour is given by

$$t \prec (t_1, t_2) \ \Leftrightarrow_{df} \ (t_1, t_2) \succ t \ \Leftrightarrow_{df} \ t_1 = \pi_{T_1}(t) \wedge t_2 = \pi_{T_2}(t) \ .$$

Hence $\prec$ relates every tuple to all its possible splits into matching pairs of subtuples. The definition is stronger than the corresponding one in [3,4], and this results in more useful laws which are detailed below: [3,4] allow arbitrary splittings on the left and right of $\succ ; \prec$, whereas ours are "synchronised" by the projections so that the same splits are used on the left and right. By the difference in approach the forward interchange rule of Th. 9.2 does not hold in their setting. For the purposes of database algebra, however, the stronger definition is quite adequate.

While split and glue tell us how to decompose or recompose tuples or tuple parts, we also want to want to relate corresponding parts "in parallel".

[4] We use this example only for motivation; strictly speaking an interchange law needs to have the same variables on both sides.

Definition 8.2 A *pair relation* over types T_1, T_2 is a subset of $(D_{T_1} \times D_{T_2}) \times (D_{T_1} \times D_{T_2})$. The *parallel product* $R_1 \times R_2$ of relations $R_i :: T_i^2$ is the pair relation

$$(t_1, t_2)\,(R_1 \times R_2)\,(u_1, u_2) \;\Leftrightarrow_{df}\; t_1\,R_1\,u_1 \,\wedge\, t_2\,R_2\,u_2 \;.$$

By $1_{T_1 \times T_2}$ we denote the identity pair relation. When the T_i are clear from the context we omit the type index.

The parallel product is a standard construct in relation algebra; it occurs, for instance, in [2] and [7] and is also called a Kronecker product [8]. With its help we can express the lifting of join to relations in Def. 7.1 more compactly.

Lemma 8.3 *The join of relations $R_i :: T_i^2$ can be expressed point-free as*

$$R_1 \bowtie R_2 \;=_{df}\; \prec ; (R_1 \times R_2) ; \succ \;.$$

The proof is immediate from the definitions. From this relational representation it follows that join is strict w.r.t. 0 and distributes through union in both arguments. We note that for relational tests P, Q the lifting $P \bowtie Q$ is a test in the algebra of relations. Details are given in Lm. 10.5.

Next to this, we also use the concept of tests for pair relations. These are again sub-identities, i.e., subsets of $1_{T_1 \times T_2}$; as usual they are idempotent and commute under ; (e.g. [5]). The parallel product of tests is a test in the set of pair relations.

Definition 8.4 Another test in the set of pair relations is the lifted *matching check* ${}_{T_1}\textcircled{\#}_{T_2}$: for tuples $t_i, u_i :: T_i$

$$(t_1, t_2)\;{}_{T_1}\textcircled{\#}_{T_2}\;(u_1, u_2) \;\;\Leftrightarrow_{df}\;\; t_1 = u_1 \,\wedge\, t_2 = u_2 \,\wedge\, t_1 \# t_2 \;.$$

To ease notation, we suppress the type indices.

We now present the essential laws for all these constructs.

Lemma 8.5
1. $\succ \,=\, \prec^{\smile}$.
2. $\succ ; \prec \,=\, \textcircled{\#}$ *and hence* $\succ ; \prec \,\subseteq 1$.
3. $\prec ; \succ \,=\, 1$.
4. $\textcircled{\#} ; \succ \,=\, \succ$ *and symmetrically* $\prec ; \textcircled{\#} \,=\, \prec$.
5. $\prec$ *and* $\succ$ *are deterministic and injective; in addition* $\prec$ *is total and* $\succ$ *is surjective.*
6. $\prec ; C ; \succ \,\subseteq\, R \;\Leftrightarrow\; \textcircled{\#} ; C ; \textcircled{\#} \,\subseteq\, \succ ; R ; \prec$. *In particular,* $\prec ; C ; \succ \,\subseteq\, \prec ; D ; \succ \;\Leftrightarrow\; \textcircled{\#} ; C ; \textcircled{\#} \,\subseteq\, \textcircled{\#} ; D ; \textcircled{\#}$.
7. $\prec ; \mathsf{T} ; \succ \,=\, \mathsf{T}$.
8. $\prec ; C ; \textcircled{\#} ; \mathsf{T} ; \succ \,=\, \prec ; C ; \succ ; \mathsf{T}$.

Proof. The proofs of Parts 1—3 are straightforward pointwise calculations.
4. By 2 and 3, $\textcircled{\#} ; \succ \,=\, \succ ; \prec ; \succ \,=\, \succ ; 1 \,=\, \succ$.

5. These are standard relation-algebraic consequences of Parts 1—3.
6. By isotony, Part 2, isotony and Parts 4 and 3,

$$\begin{aligned} & \prec ; C ; \succ \subseteq R \;\Rightarrow\; \succ ; \prec ; C ; \succ ; \prec \subseteq \succ ; R ; \prec \\ \Leftrightarrow\; & \circledast ; C ; \circledast \subseteq \succ ; R ; \prec \;\Rightarrow\; \prec ; \circledast ; C ; \circledast ; \succ \subseteq \prec ; \succ ; R ; \prec ; \succ \\ \Leftrightarrow\; & \prec ; C ; \succ \subseteq R \,. \end{aligned}$$

For $R = \prec ; D ; \succ$ the second claim results again by Part 2.
7. This is direct by totality of $\prec$ and surjectivity of $\succ$ (Part 5).
8. By Parts 2 and 7, $\prec ; C ; \circledast ; \mathsf{T} ; \succ \;=\; \prec ; C ; \succ ; \prec ; \mathsf{T} ; \succ \;=\; \prec ; C ; \succ ; \mathsf{T}$. □

Lemma 8.6
1. $1_{T_1 \times T_2} = 1_{T_1} \times 1_{T_2}$.
2. $\mathsf{T}_{T_1} \times \mathsf{T}_{T_2}$ *is the universal pair relation.*
3. The operators $\times$ *and* $\cap$ *satisfy an equational interchange law:*

$$(R_1 \cap R_2) \times (S_1 \cap S_2) \;=\; (R_1 \times S_1) \cap (R_2 \times S_2) \;.$$

4. The operators $\times$ *and* $;$ *satisfy an equational interchange law:*

$$(R_1 ; R_2) \times (S_1 ; S_2) \;=\; (R_1 \times S_1) ; (R_2 \times S_2) \;.$$

Again, the proofs are straightforward calculations. In addition, we have the following result.

Lemma 8.7 *Identity and top behave nicely w.r.t.* $\bowtie$*, i.e.,* $1_{T_1} \bowtie 1_{T_2} = 1_{T_1 \bowtie T_2}$*. Similarly,* $\mathsf{T}_{T_1} \bowtie \mathsf{T}_{T_2} = \mathsf{T}_{T_1 \bowtie T_2}$*; equivalently,* $\prec ; \mathsf{T}_{T_1 \times T_2} ; \succ = \mathsf{T}_{T_1 \bowtie T_2}$*.*

Proof. For the first claim we calculate, using Lms. 8.3, 8.6.1 and 8.5.3,

$$1_{T_1} \bowtie 1_{T_2} \;=\; \prec ; (1_{T_1} \times 1_{T_2}) ; \succ \;=\; \prec ; 1_{T_1 \times T_2} ; \succ \;=\; \prec ; \succ \;=\; 1_{T_1 \bowtie T_2} \;.$$

The second claim was shown in Lm. 8.5.7. □

9 Interchange Laws for Join

We have already seen some interchange laws. It turns out that join inherits many of them, sometimes as inclusions rather than equations.

Lemma 9.1 *Relations* $R_i, S_i :: T_i^2$ *satisfy the equational interchange law*

$$(R_1 \bowtie R_2) \cap (S_1 \bowtie S_2) \;=\; (R_1 \cap S_1) \bowtie (R_2 \cap S_2) \;.$$

Proof.

$$\begin{aligned} & (R_1 \bowtie R_2) \cap (S_1 \bowtie S_2) \\ = \; & \prec ; (R_1 \times R_2) ; \succ \cap \prec ; (S_1 \times S_2) ; \succ \quad \{\!\![\text{ Lm. 8.3 }]\!\!\} \end{aligned}$$

$$
\begin{array}{ll}
= \prec ; ((R_1 \times R_2) \cap (S_1 \times S_2)) ; \succ & \{\!\![\text{ determinacy of } \prec \text{ and} \\
& \quad \text{injectivity of } \succ \text{ (Lm. 8.5.5) }]\!\!\} \\
= \prec ; ((R_1 \cap S_1) \times (R_2 \cap S_2)) ; \succ & \{\!\![\times\text{-}\cap\text{-interchange (Lm. 8.6) }]\!\!\} \\
= (R_1 \cap S_1) \bowtie (R_2 \cap S_2) & \{\!\![\text{ Lm. 8.3 }]\!\!\} \qquad \square
\end{array}
$$

Theorem 9.2 (Forward Interchange) *Relations $R_i, S_i :: T_i^2$ satisfy the inclusional interchange law*

$$(R_1 \bowtie R_2) ; (S_1 \bowtie S_2) \;\subseteq\; (R_1 ; S_1) \bowtie (R_2 ; S_2) \ .$$

Proof. We calculate as follows.

$$
\begin{array}{ll}
\quad (R_1 \bowtie R_2) ; (S_1 \bowtie S_2) & \\
= \prec ;(R_1 \times R_2) ; \succ ; \prec ;(S_1 \times S_2) ; \succ & \{\!\![\text{ Lm. 8.3 }]\!\!\} \\
\subseteq \prec ;(R_1 \times R_2) ; 1 ; (S_1 \times S_2) ; \succ & \{\!\![\text{ Lm. 8.5.2 }]\!\!\} \\
= \prec ;(R_1 \times R_2) ; (S_1 \times S_2) ; \succ & \{\!\![\text{ neutrality of } 1 \ ]\!\!\} \\
= \prec ;((R_1 ; S_1) \times (R_2 ; S_2)) ; \succ & \{\!\![\ ;\text{-}\times\text{-interchange (Lm. 8.6.3) }]\!\!\} \\
= (R_1 ; S_1) \bowtie (R_2 ; S_2) & \{\!\![\text{ Lm. 8.3 }]\!\!\} \qquad \square
\end{array}
$$

Using this we can show a subdistribution law for domain over join.

Theorem 9.3 *For $R_i :: T_i^2$ $(i = 1, 2)$ the domain of their join satisfies*

$$\ulcorner(R_1 \bowtie R_2) \subseteq \ulcorner R_1 \bowtie \ulcorner R_2 \ .$$

Proof. By (1) and Lm. 8.7, Th. 9.2, $\bowtie$-$\cap$-interchange (Lm. 9.1) and (1):

$$
\begin{array}{rl}
\ulcorner(R_1 \bowtie R_2) & = (R_1 \bowtie R_2) ; (\top_{T_1} \bowtie \top_{T_2}) \cap (1_{T_1} \bowtie 1_{T_2}) \\
& \subseteq ((R_1 ; \top_{T_1}) \bowtie (R_2 ; \top_{T_2})) \cap (1_{T_1} \bowtie 1_{T_2}) \\
& = (R_1 ; \top_{T_1} \cap 1_{T_1}) \bowtie (R_2 ; \top_{T_2} \cap 1_{T_2}) = \ulcorner R_1 \bowtie \ulcorner R_2 \qquad \square
\end{array}
$$

Next we present conditions under which these inclusions become equations.

10 Compatibility and Matching

Definition 10.1

1. We call R_1, R_2 *weakly matching* if for all $x_i \in \ulcorner R_i$ $(i = 1, 2)$ with $x_1 \# x_2$ there are $y_i :: T_i$ $(i = 1, 2)$ with $y_1 \# y_2$ and $x_i \, R_i \, y_i$. This means that starting from matching tuples one can always reach corresponding matching tuples via the R_i.
2. R_1, R_2 are *strongly matching* if for all $x_i \in \ulcorner R_i$ $(i = 1, 2)$ with $x_1 \# x_2$ all tuples $y_i :: T_i$ $(i = 1, 2)$ with $x_i \, R_i \, y_i$ satisfy $y_1 \# y_2$. This means that starting from matching tuples all corresponding tuples reachable via the R_i are matching again.

We want to find a algebraic characterisations of these forms of matching.

Definition 10.2 Relations R_1, R_2 are *forward compatible* iff

$$\text{\textcircled{\#}}\,;(R_1 \times R_2) \;\subseteq\; (R_1 \times R_2)\,;\text{\textcircled{\#}}\;,$$

and *backward compatible* iff $(R_1 \times R_2)\,;\text{\textcircled{\#}} \subseteq \text{\textcircled{\#}}\,;(R_1 \times R_2)$. Finally, R_1 and R_2 are *compatible* iff they are forward and backward compatible.

We can now give point-free characterisations of matching.

Lemma 10.3

1. *All test relations are compatible with each other.*
2. *Two relations are strongly matching iff they are forward compatible.*
3. *R_1, R_2 are weakly matching iff $\text{\textcircled{\#}}\,;({}^{\ulcorner}R_1 \times {}^{\ulcorner}R_2) \subseteq {}^{\ulcorner}((R_1 \times R_2)\,;\text{\textcircled{\#}})$ iff ${}^{\ulcorner}(\text{\textcircled{\#}}\,;(R_1 \times R_2)) \subseteq {}^{\ulcorner}((R_1 \times R_2)\,;\text{\textcircled{\#}})$.*
4. *Strongly matching relations are also weakly matching.*

Proof.

1. For test relations P, Q the relation $P \times Q$ is a test in the algebra of pair relations. Since $\text{\textcircled{\#}}$ is a test there too, they commute, which means forward and backward compatibility of P and Q.
2. Straightforward predicate calculus with the definitions.
3. Ditto for the first inclusion. The second one results from the first by distributivity of domain over $\times$ and the import/export law of Lm. 2.1.5.
4. Immediate from the second inclusion of Part 3 and isotony of domain. $\square$

Now we can show a reverse interchange law between $\bowtie$ and ;.

Theorem 10.4 (Backward Interchange) *Let $R_i, S_i :: T_i^2$. If R_1, R_2 are forward compatible or S_1, S_2 are backward compatible then*

$$(R_1\,;S_1) \bowtie (R_2\,;S_2) \;\subseteq\; (R_1 \bowtie R_2)\,;(S_1 \bowtie S_2)\;.$$

In particular, if R_1, R_2 or S_1, S_2 are tests then the inclusion holds.

Proof. We assume R_1, R_2 to be forward compatible.

$$
\begin{array}{ll}
\quad (R_1\,;S_1) \bowtie (R_2\,;S_2) & \\
= \prec;(R_1\,;S_1) \times (R_2\,;S_2)\,;\succ & \{\![\ \text{Lm. 8.3}\ ]\!\} \\
= \prec;(R_1 \times R_2)\,;(S_1 \times S_2)\,;\succ & \{\![\ ;\text{-}\times\text{-interchange (Lm. 8.6.4)}\ ]\!\} \\
= \prec;\text{\textcircled{\#}}\,;(R_1 \times R_2)\,;(S_1 \times S_2)\,;\succ & \{\![\ \text{Lm. 8.5.4}\ ]\!\} \\
\subseteq \prec;(R_1 \times R_2)\,;\text{\textcircled{\#}}\,;(S_1 \times S_2)\,;\succ & \{\![\ \text{forward compatibility}\ ]\!\} \\
= \prec;(R_1 \times R_2)\,;\succ;\prec;(S_1 \times S_2)\,;\succ & \{\![\ \text{Lm. 8.5.2}\ ]\!\} \\
= (R_1 \bowtie R_2)\,;(S_1 \bowtie S_2) & \{\![\ \text{Lm. 8.3}\ ]\!\}
\end{array}
$$

The proof under backward compatibility of S_1, S_2 is symmetric. $\square$

Finally, we show the announced result on the join of tests.

Lemma 10.5 *If $P_i :: T_i$ $(i = 1, 2)$ are tests then $P_1 \bowtie P_2 :: T_1 \bowtie T_2$ is a test with $\neg(P_1 \bowtie P_2) = \neg P_1 \bowtie 1_{T_2} \cup 1_{T_1} \bowtie \neg P_2$, where $\neg P = 1 - P$.*

Proof. First,

$$
\begin{array}{cl}
 & (P_1 \bowtie P_2) ; (\neg P_1 \bowtie 1_{T_2} \cup 1_{T_1} \bowtie \neg P_2) \\
= & \quad \{\!\![\text{ distributivity }]\!\!\} \\
 & (P_1 \bowtie P_2) ; (\neg P_1 \bowtie 1_{T_2}) \cup (P_1 \bowtie P_2) ; (1_{T_1} \bowtie \neg P_2) \\
\subseteq & \quad \{\!\![\text{ forward interchange (Th. 9.2) }]\!\!\} \\
 & (P_1 ; \neg P_1) \bowtie (P_2 ; 1_{T_2}) \cup (P_1 ; 1_{T_1}) \bowtie (P_2 ; \neg P_2) \\
= & \quad \{\!\![\ P_i \text{ tests and strictness of join }]\!\!\} \\
 & 0_{T_1 \bowtie T_2}
\end{array}
$$

Second,

$$
\begin{array}{cl}
 & P_1 \bowtie P_2 \cup \neg P_1 \bowtie 1_{T_2} \cup 1_{T_1} \bowtie \neg P_2 \\
= & \quad \{\!\![\text{ Boolean algebra and distributivity of join }]\!\!\} \\
 & P_1 \bowtie P_2 \cup \neg P_1 \bowtie P_2 \cup \neg P_1 \bowtie \neg P_2 \cup P_1 \bowtie \neg P_2 \cup \neg P_1 \bowtie \neg P_2 \\
= & \quad \{\!\![\text{ distributivity of join and Boolean algebra }]\!\!\} \\
 & 1_{T_1} \bowtie P_2 \cup 1_{T_1} \bowtie \neg P_2 \\
= & \quad \{\!\![\text{ distributivity of join and Boolean algebra }]\!\!\} \\
 & 1_{T_1} \bowtie 1_{T_2} \\
= & \quad \{\!\![\text{ Lm. 8.7 }]\!\!\} \\
 & 1_{T_1 \bowtie T_2}
\end{array}
$$

□

11 About Weak Matching

We have seen that strong matching turns $\bowtie$-;-interchange from inclusion to equation form (Lm. 10.3, Ths. 9.2 and 10.4). We now show that weak matching does the same for distributivity of domain over join.

Theorem 11.1 *Weakly matching $R_i :: T_i^2$ satisfy $\ulcorner R_1 \bowtie \ulcorner R_2 \subseteq \ulcorner(R_1 \bowtie R_2)$.*

Proof. By Lm. 8.3, Lm. 8.5.4, weak matching with Lm. 10.3.3, domain representation (1), isotony, Lm. 8.5(8,3) and Lm. 8.3 with domain representation (1):

$$
\begin{aligned}
\ulcorner R_1 \bowtie \ulcorner R_2 &= \prec;(\ulcorner R_1 \times \ulcorner R_2) ; \succ\, = \prec ; \text{\#} ;(\ulcorner R_1 \times \ulcorner R_2) ; \succ \\
&\subseteq \prec ;\ulcorner((R_1 \times R_2) ; \text{\#}) ; \succ\, = \prec ;((R_1 \times R_2) ; \text{\#} ; \mathsf{T} \cap 1) ; \succ \\
&\subseteq \prec ;(R_1 \times R_2) ; \text{\#} ; \mathsf{T} ; \succ \cap \prec ;1 ; \succ\, = \prec ;(R_1 \times R_2) ; \succ ;\mathsf{T} \cap 1 \\
&= \ulcorner(R_1 \bowtie R_2)
\end{aligned}
$$

□

Weak matching is even equivalent to distributivity of domain.

Theorem 11.2 *If $\ulcorner R_1 \bowtie \ulcorner R_2 \subseteq \ulcorner(R_1 \bowtie R_2)$ then R_1, R_2 are weakly matching.*

Proof. We first prove that an injective relation S and an arbitrary relation R satisfy $S ; \ulcorner(S^{\smile} ; R) = \ulcorner R ; S$. By domain representation (1), then by $R ; (S ; \mathsf{T} \cap 1) =$

$R \cap \mathsf{T} \,;\, S^{\smile}$ and laws of ${}^{\smile}$, right distributivity due to injectivity of S, $P^{\smile} = P$ for any test P and domain representation (1):

$$\begin{aligned} S \,;\, \ulcorner(S^{\smile} \,;\, R) &= S \,;\, (S^{\smile} \,;\, R \,;\, \mathsf{T} \cap 1) = S \cap \mathsf{T} \,;\, R^{\smile} \,;\, S \\ &= (1 \cap \mathsf{T} \,;\, R^{\smile}) \,;\, S = (1 \cap R \,;\, \mathsf{T}) \,;\, S = \ulcorner R \,;\, S \ . \end{aligned}$$

To prove the lemma, we assume $\ulcorner R_1 \bowtie \ulcorner R_2 \subseteq \ulcorner(R_1 \bowtie R_2)$ and prove $\textcircled{\#} \,;\, (\ulcorner R_1 \times \ulcorner R_2) \subseteq \ulcorner((R_1 \times R_2) \,;\, \textcircled{\#})$ (see Lm. 10.3.3).

$$\begin{array}{ll} \quad \textcircled{\#} \,;\, (\ulcorner R_1 \times \ulcorner R_2) & \\ = \textcircled{\#} \,;\, (\ulcorner R_1 \times \ulcorner R_2) \,;\, \textcircled{\#} & \{\!\![\ \textcircled{\#} \text{ and } \ulcorner R_1 \times \ulcorner R_2 \text{ are tests, idempotence} \\ & \quad \text{and commutativity of tests }]\!\!\} \\ = \succ \,;\, \prec \,;(\ulcorner R_1 \times \ulcorner R_2) \,;\, \succ \,;\, \prec & \{\!\![\ \text{Lm. 8.5.2 }]\!\!\} \\ = \succ \,;(\ulcorner R_1 \bowtie \ulcorner R_2) \,;\, \prec & \{\!\![\ \text{Lm. 8.3 }]\!\!\} \\ \subseteq \succ \,;\ulcorner(R_1 \bowtie R_2) \,;\, \prec & \{\!\![\ \text{assumption and isotony }]\!\!\} \\ = \succ \,;\ulcorner(\prec \,;(R_1 \times R_2) \,;\, \succ) \,;\, \prec & \{\!\![\ \text{Lm. 8.3 }]\!\!\} \\ = \ulcorner((R_1 \times R_2) \,;\, \succ) \succ \,;\, \prec & \{\!\![\ \text{Lm. 8.5(1,5) and preliminary result }]\!\!\} \\ = \ulcorner((R_1 \times R_2) \,;\, \textcircled{\#} \,;\, \succ) \,;\, \textcircled{\#} & \{\!\![\ \text{Lm. 8.5(2,4) }]\!\!\} \\ \subseteq \ulcorner((R_1 \times R_2) \,;\, \textcircled{\#}) & \{\!\![\ \ulcorner(R \,;\, S) \subseteq \ulcorner R,\ \textcircled{\#} \text{ is a test and isotony }]\!\!\} \qquad \Box \end{array}$$

12 Join and Maximal Elements

We now study how join and the maximum operator interact. First we show an interchange law for join and diamond.

Lemma 12.1 *For $R_i :: T_i^2$ and $P_i :: T_i$ $(i = 1, 2)$,*

$$|R_1 \bowtie R_2\rangle(P_1 \bowtie P_2) \ \subseteq\ |R_1\rangle P_1 \bowtie |R_2\rangle P_2 \ .$$

If the $R_i \,;\, P_i$ are weakly matching then this strengthens to an equality.

Proof. By definition of inverse image, Th. 9.2 with Lm. 10.3.1 and Th. 10.4, Th. 9.3 and definition of inverse image:

$$\begin{aligned} |R_1 \bowtie R_2\rangle(P_1 \bowtie P_2) &= \ulcorner((R_1 \bowtie R_2) \,;\, (P_1 \bowtie P_2)) = \ulcorner((R_1 \,;\, P_1) \bowtie (R_2 \,;\, P_2)) \\ &\subseteq \ulcorner(R_1 \,;\, P_1) \bowtie \ulcorner(R_2 \,;\, P_2) = |R_1\rangle P_1 \bowtie |R_2\rangle P_2) \end{aligned}$$

The claim when the $R_i \,;\, P_i$ are weakly matching follows by using Th. 11.1 in the third step. $\Box$

This is used to derive an interaction law for join and maximum.

Lemma 12.2 *Consider tables $P :: T_P, Q :: T_Q$ and relations $R :: T_P^2$ and $S :: T_Q^2$ such that $R \,;\, P$ and $S \,;\, Q$ are weakly matching. Then*

$$(R \bowtie S) \triangleright (P \bowtie Q) = (R \triangleright P) \bowtie Q \ \cup\ P \bowtie (S \triangleright Q) \ .$$

Proof.

$$
\begin{array}{ll}
\quad (R \bowtie S) \triangleright (P \bowtie Q) & \\
= (P \bowtie Q) - |R \bowtie S\rangle(P \bowtie Q) & \{\!\![\ \text{definition of } \triangleright\ ]\!\!\} \\
= (P \bowtie Q) - (|R\rangle P \bowtie |S\rangle Q) & \{\!\![\ \text{Lm. 12.1}\ ]\!\!\} \\
= (P \bowtie Q)\,;\neg(|R\rangle P \bowtie |S\rangle Q) & \{\!\![\ \text{definition of } -\ ]\!\!\} \\
= (P \bowtie Q); & \{\!\![\ \text{complement of test (Lm. 10.5)}\ ]\!\!\} \\
\quad (\neg|R\rangle P \bowtie 1_Q \cup 1_P \bowtie \neg|S\rangle Q) & \\
= (P\,;\neg|R\rangle P) \bowtie (Q\,;1_Q) & \{\!\![\ \text{distributivity and interchange laws of} \\
\quad \cup\ (P\,;1_P) \bowtie Q\,;\neg|S\rangle Q) & \quad \text{Ths. 9.2 and 10.4, since } P, Q \text{ are tests}\ ]\!\!\} \\
= (R \triangleright P) \bowtie Q \cup P \bowtie (S \triangleright Q) & \{\!\![\ \text{neutrality of 1 and definition of } \triangleright\ ]\!\!\} \quad \square
\end{array}
$$

Corollary 12.3 *Consider tables $P :: T_P, Q :: T_Q$ and a relation $R :: T_P^2$ such that $R\,;P$ and $\mathsf{T}_Q\,;Q$ are weakly matching. Then*

$$(R \bowtie \mathsf{T}_Q) \triangleright (P \bowtie Q) = (R \triangleright P) \bowtie Q\ .$$

Proof. Immediate from Lm. 12.2, (3), strictness of $\bowtie$ and neutrality of 0. □

This shows (4) — the only question is how to establish weak matching. For this we introduce a sufficient condition.

Definition 12.4 Assume tables $P :: T_P, Q :: T_Q$. We call P *joinable* with Q if $P \subseteq |\#\rangle Q$, where $\#$ is the matching relation between tuples. Pointwise, P is joinable with Q iff $\forall p \in P : \exists q \in Q : p \# q$. Informally this means that every tuple in P has a join partner in Q.

Lemma 12.5 *If P is joinable with Q then $R;P$ and $\mathsf{T}_Q;Q$ are weakly matching.*

Since the proof needs additional notions we defer it to the Appendix.

Now we can state an optimisation rule involving a θ-join.

Theorem 12.6 *Consider $P :: T_P, Q :: T_Q$, $R :: T_P^2$ as well as $\theta :: \{A\} \bowtie \{B\}$ with $A \in T_P, B \in T_Q$ with $T_P \cap T_Q = \emptyset$. If P is joinable with $\theta \bowtie Q$ then*

$$(R \bowtie \mathsf{T}_{\theta\bowtie Q}) \triangleright (P \bowtie \theta \bowtie Q) = (R \triangleright P) \bowtie \theta \bowtie Q\ .$$

This is immediate from Lm. 12.5 and Cor. 12.3.

Without the premise of joinability the theorem need not hold.

Example 12.7 Choose, for instance, θ as equality and $T_P = \{A\}$, $P = \{1,2\}$, $T_Q = \{B\}$, $Q = \{1\}$ as well as $D_A = D_B = \{1,2\}$. Here $\{2\}$ has no join partner in $\theta \bowtie Q$. Now for a preference R with $1\,R\,2$ we have the differing expressions

$$
\begin{array}{l}
(R \bowtie \mathsf{T}_{\theta\bowtie Q}) \triangleright (P \bowtie \theta \bowtie Q) = (R \bowtie \mathsf{T}_{\theta\bowtie Q}) \triangleright \{(1,1)\} = \{(1,1)\}\ , \\
(R \triangleright P) \bowtie \theta \bowtie Q = \{2\} \bowtie \theta \bowtie \{1\} = 0\ .
\end{array}
$$
□

13 Conclusion and Outlook

We have presented a new and simple approach to an algebraic treatment of the theta join in databases. This is a piece that was missing in the predecessor paper [16], because there mostly only joins of tables with disjoint attribute sets were treated. However, overlapping types are mandatory for coping with theta joins. And so other important outcomes of the present paper are the more liberal notions of weak and strong matching of binary relations over database tuples.

With the help of the developed tools we have algebraically proved the correctness of two sample optimisation rules, namely "push projection over join" and "push preference over join".

Further work will be to treat the large catalogue of preference optimisation rules in [14] with these techniques. This also concerns the complex preference relation constructors of Pareto and prioritised composition. In fact, the relation $R \bowtie \mathsf{T}_U$ in Th. 12.6 is equal to the prioritised preference $R \,\&\, 0$.

The present treatment was performed in the setting of concrete binary relations. While mostly point-free, some of the basic lemmas in Sect. 3 still were proved in a pointwise fashion. A next step to a more abstract view would be to axiomatise the projections and then reason point-free in terms of these. Another more abstract approach could be based on the concept of typed join algebras from the predecessor paper [16].

Acknowledgement Helpful comments were provided by Patrick Roocks, Andreas Zelend and the anonymous referees.

References

1. R. Berghammer, A. Haeberer, G. Schmidt, P. Veloso: Comparing two different approaches to products in abstract relation algebra. Proceedings of the Third International Conference on Algebraic Methodology and Software Technology (AMAST '93), University of Twente, Enschede, The Netherlands, 1993, Springer, 167–176
2. R. Berghammer, B. von Karger: Relational Semantics of Functional Programs. In C. Brink, W. Kahl, G. Schmidt (eds.): Relational Methods in Computer Science. Advances in Computing Science, Springer 1997, 115–130
3. H.-H. Dang, P. Höfner, B. Möller: Algebraic separation logic. Journal of Logic and Algebraic Programming 80(6), 221–247 (2011)
4. H.-H. Dang, B. Möller: Reverse exchange for concurrency and local reasoning. In: J. Gibbons, P. Nogueira (eds.): MPC. LNCS 7342. Springer 2012, 177–197. Revised and extended version: H.-H. Dang, B. Möller: Concurrency and local reasoning under reverse exchange. Science of Computer Programming 85, Part B, 204–223 (2013)
5. J. Desharnais, B. Möller, G. Struth: Modal Kleene algebra and applications — a survey. Journal on Relational Methods in Computer Science 1, 93–131 (2004)
6. J. Desharnais, B. Möller, G. Struth: Kleene algebra with domain. ACM Transactions on Computational Logic 7, 798–833 (2006)
7. A. Haeberer, M. Frias, G. Baum, P. Veloso: Fork algebras. In C. Brink, W. Kahl, G. Schmidt (eds.): Relational Methods in Computer Science. Advances in Computing Science. Springer 1997, 54–69
8. R. Horn, C. Johnson: Topics in matrix analysis, Cambridge University Press 1991

9. W. Kahl: CalcCheck: A proof checker for teaching the "Logical Approach to Discrete Math". In J. Avigad, A. Mahboubi (eds.): ITP 2018 — Interactive Theorem Proving. LNCS 10895. Springer 2018, 324–341
10. W. Kahl: CalcCheck — A proof-checker for Gries and Schneider's *Logical Approach to Discrete Math*. `http://calccheck.mcmaster.ca/`
11. P. Kanellakis: Elements of relational database theory. In J. van Leeuwen (ed.): Handbook of Theoretical Computer Science. Volume B: Formal Models and Semantics. Elsevier 1990, 1073–1156
12. W. Kießling: Preference queries with SV-semantics. In International Conference on Management of Data (COMAD '05), 15–26, 2005.
13. W. Kießling, M. Endres, F. Wenzel: The Preference SQL system — An overview. Bull. Technical Committee on Data Engineering, IEEE Computer Society 34(2), 11–18 (2011). Details under `http://www.markusendres.de/preferencesql/`
14. W. Kießling, B. Hafenrichter: Algebraic optimization of relational preference queries. Technical Report No. 2003-01. University of Augsburg, Institute of Computer Science, February 2003
15. R. Maddux: On the derivation of identities involving projection functions. In L. Csirmaz, D. Gabbay, M. de Rijke (eds.): Logic Colloquium '92. Studies in Logic, Languages, and Information. CSLI Publications, 143–163 (1995)
16. B. Möller, P. Roocks: An algebra of database preferences. Journal of Logical and Algebraic Methods in Programming 84(3): 456-481 (2015)
17. G. Schmidt, T. Ströhlein: Relations and Graphs: Discrete Mathematics for Computer Scientists. EATCS Monographs on Theoretical Computer Science. Springer 1993
18. H. Zierer: Programmierung mit Funktionsobjekten: Konstruktive Erzeugung semantischer Bereiche und Anwendung auf die partielle Auswertung. Institut für Informatik, Technische Universität München. Report TUM-I8803, February 1988

14 Appendix

For types T_P, T_Q we use the notion of a *direct product* of D_P and D_Q (e.g. [17]). This is a pair (ρ_P, ρ_Q) of relations with $\rho_P \subseteq (D_P \times D_Q) \times D_P$ and $\rho_Q \subseteq (D_P \times D_Q) \times D_Q$ such that

$$\begin{array}{ll} \breve{\rho_P} \,;\, \rho_P = 1 \ , & \breve{\rho_Q} \,;\, \rho_Q = 1 \ , \\ \rho_P \,;\, \breve{\rho_P} \cap \rho_Q \,;\, \breve{\rho_Q} = 1 \ , & \breve{\rho_P} \,;\, \rho_Q = \mathsf{T} \ . \end{array}$$

Using this concept the parallel product can be represented as

$$P \times Q \;=\; \rho_P \,;\, P \,;\, \breve{\rho_P} \cap \rho_Q \,;\, Q \,;\, \breve{\rho_Q} \ . \tag{5}$$

The following properties of direct products are used in the main proof[5]:

$$\rho_P \,;\, \mathsf{T} = \mathsf{T} = \rho_Q \,;\, \mathsf{T} \ , \tag{6}$$

$$(R_1 \,;\, \breve{\rho_P} \cap R_2 \,;\, \breve{\rho_Q}) \,;\, (\rho_P \,;\, S_1 \cap \rho_Q \,;\, S_2) \;=\; R_1 \,;\, S_1 \,\cap\, R_2 \,;\, S_2 \ . \tag{7}$$

[5] Equation (7) is valid for concrete relations. For abstract relations, only $\subseteq$ holds. This phenomenon is called *unsharpness* in the literature (an early mention is [18], a further elaboration [1]). The situation is similar with Lm. 8.6.4. The paper [15] constructs an RA that does not satisfy sharpness.

Proof of **Lemma 12.5**. The proof consists in showing $\textcircled{\#} ; (\ulcorner(R;P) \times \ulcorner(\mathsf{T}_Q ; Q)) \subseteq \ulcorner(((R ; P) \times (\mathsf{T}_Q ; Q)) ; \textcircled{\#})$ (see Lm. 10.3.3). We do this by showing the stronger property $\ulcorner(R;P) \times \ulcorner(\mathsf{T}_Q ; Q) \subseteq \ulcorner(((R;P) \times (\mathsf{T}_Q ; Q)) ; \textcircled{\#})$, from which the original claim follows by $\textcircled{\#} \subseteq 1$ and isotony of $;$.

Since "joinable" is defined with $\#$ and the formula to prove uses $\textcircled{\#}$, we have to make a connection between the two:

$$\textcircled{\#} \ = \ulcorner(\rho_P ; \# \cap \rho_Q) \ . \tag{8}$$

This is analogous to the conversion of a relation to a vector explained in [17], which would give $\textcircled{\#} ; \mathsf{T} = (\rho_P ; \# \cap \rho_Q) ; \mathsf{T}$. The inverse transformation is $\# = \breve{\rho}_P ; (\textcircled{\#} ; \mathsf{T} \cap \rho_Q)$. Both equations are easily verified. Using restriction (Lm. 2.1.7) and Boolean algebra, the second one can be simplified to $\# = \breve{\rho}_P ; \textcircled{\#} ; \rho_Q$. Then by Def. 12.4 and the definition of diamond (Def. 6.1) P is joinable with Q iff

$$P \subseteq \ulcorner(\breve{\rho}_P ; \textcircled{\#} ; \rho_Q ; Q) \ . \tag{9}$$

Now we calculate as follows.

$$\begin{aligned}
& \ulcorner(R ; P) \times \ulcorner(\mathsf{T}_Q ; Q) \\
= \quad & \{\!\![\text{ distributivity of domain over } \times \ ]\!\!\} \\
& \ulcorner((R ; P) \times (\mathsf{T}_Q ; Q)) \\
= \quad & \{\!\![\ (5) \ ]\!\!\} \\
& \ulcorner(\rho_P ; R ; P ; \breve{\rho}_P \cap \rho_Q ; \mathsf{T}_Q ; Q ; \breve{\rho}_Q) \\
\subseteq \quad & \{\!\![\text{ Boolean algebra and isotony of } \ulcorner \ ]\!\!\} \\
& \ulcorner(\rho_P ; R ; P ; \breve{\rho}_P) \\
= \quad & \{\!\![\text{ locality (Lm. 2.1.6) }]\!\!\} \\
& \ulcorner(\rho_P ; R ; P ; \ulcorner(\breve{\rho}_P)) \\
= \quad & \{\!\![\ \rho_P \text{ is surjective, hence } \breve{\rho}_P \text{ is total }]\!\!\} \\
& \ulcorner(\rho_P ; R ; P ; 1) \\
= \quad & \{\!\![\text{ neutrality of 1 and (9) with Boolean algebra }]\!\!\} \\
& \ulcorner(\rho_P ; R ; P ; \ulcorner(\breve{\rho}_P ; \textcircled{\#} ; \rho_Q ; Q)) \\
= \quad & \{\!\![\text{ locality (Lm. 2.1.6) twice }]\!\!\} \\
& \ulcorner(\rho_P ; R ; P ; \breve{\rho}_P ; \ulcorner(\textcircled{\#} ; \rho_Q ; Q)) \\
= \quad & \{\!\![\text{ domain representation (1) }]\!\!\} \\
& \ulcorner(\rho_P ; R ; P ; \breve{\rho}_P ; (\textcircled{\#} ; \rho_Q ; Q ; \mathsf{T} \cap 1)) \\
= \quad & \{\!\![\ \textcircled{\#} \text{ is a test, restriction (Lm. 2.1.7) and neutrality of 1 }]\!\!\} \\
& \ulcorner(\rho_P ; R ; P ; \breve{\rho}_P ; (\rho_Q ; Q ; \mathsf{T} \cap \textcircled{\#})) \\
= \quad & \{\!\![\ R_1 ; (R_2 ; \mathsf{T} \cap R_3) = (R_1 \cap \mathsf{T} ; \breve{R}_2) ; R_3 \text{ for all } R_1, R_2, R_3, \\
& \quad \text{laws of converse and } Q \text{ is a test }]\!\!\} \\
& \ulcorner((\rho_P ; R ; P ; \breve{\rho}_P \cap \mathsf{T} ; Q ; \breve{\rho}_Q) ; \textcircled{\#}) \\
= \quad & \{\!\![\ (6) \ ]\!\!\} \\
& \ulcorner((\rho_P ; R ; P ; \breve{\rho}_P \cap \rho_Q ; \mathsf{T}_Q ; Q ; \breve{\rho}_Q) ; \textcircled{\#}) \\
= \quad & \{\!\![\ (5) \ ]\!\!\} \\
& \ulcorner(((R ; P) \times (\mathsf{T}_Q ; Q)) ; \textcircled{\#})
\end{aligned}$$

$\square$