# Re-grouping information in a domain theoretic data model

Hermann Puhlmann
Fachbereich Mathematik
Technische Hochschule Darmstadt
D–64289 Darmstadt
Germany
puhlmann@mathematik.th-darmstadt.de

October 1995

## 1 Introduction

The success of relational databases is largely due to the easy and intuitive way of putting queries. One does not destinguish between a database relation and the real-world set which is to be described but identifies the one with the other. In this setting, all operations of nested relational algebra or query languages such as SQL can be interpreted in an ad-hoc manner, and in fact that is what the average user does. One drawback of these database systems however is that they do not allow for structured data types nor for partial information.

An attempt of overcoming this whilst keeping as many advantages from the relational model as possible, in particular that of a sound mathematical foundation, was made by Buneman et al. [BJO91] when they suggested to base a database model on Scott-domains. These are ordered sets in which the ordering can be regarded as an order of information contents. A least element incorporates the idea of an element of no information, and order theoretic properties model ways of combining pieces of consistent information.

The concept of a *snack*, invented by Buneman and investigated in [Nga92, Puh93, JP95], was a breakthrough towards including set-valued types into the domain theoretic setting. As it allows for repeated application of a set-type constructor, the domain theoretic database model is not only a generalization of the relational model, as initially intended, but also a generalization of nested relational databases. These were introduced in the early 1980s by Schek et al. (a nice overview as well as detailed results are given in [Sch88]) who implemented the model in the "Darmstädter Datenbanksystem". They also enriched relational algebra by two operations "nest"

($\nu$) and "unnest" ($\mu$) [JS82] which allow the passage between nested relations of different nesting levels.

While the use of relational algebra for simple queries is straightforward, nested relational algebra appears to be less comprehendable. As argued in [Puh91, Puh95] this is due to the lack of a formal semantics of nested databases, and it becomes evident in the composite application of $\nu$ and $\mu$. While $\mu \circ \nu$ equals the identity, we do not have this for $\nu \circ \mu$. Though this cannot be expected to be satisfied, one would expect $\nu \circ \mu$ to be less than the identity if a suitable information ordering is given.

This paper shows how nested database relations can be represented in the domain theoretic database model and operations $\mathsf{nest}$ and $\mathsf{unnest}$ can be defined within that model such that $\mathsf{unnest} \circ \mathsf{nest} = \mathsf{id}$ and $\mathsf{nest} \circ \mathsf{unnest} \leq \mathsf{id}$. We thus get a formal semantics of nested databases as well as operations for nesting and unnesting which keep as much information as possible and therefore build the ideal passage between relations of different nesting levels.

The organisation of the paper is as follows: In Section 1 we give a short introduction to nested relational databases. Section 2 illustrates the domain theoretic approach. In particular, the snack-powerdomain which models set-types is defined, and the interpretation of database relations in the new setting is given. We begin Section 3 with giving an intuitive, picture-oriented account of nested snacks (these will model nested relations). After that we turn to defining $\mathsf{nest}$ and $\mathsf{unnest}$ and proving the result claimed above. Finally we show how the snack representation of a nested relation behaves under $\mathsf{unnest}$ and $\mathsf{nest}$ in Section 4.

# 2 Nested Relational Databases

In [JS82], Jaeschke and Schek dismiss the principle that relational databases must be in so-called first-normal-form, i. e. that the domains connected with the attributes must be sets without any further structure. Jaeschke and Schek rather allow an entry in a database relation to be a relation itself. This construction may be applied in a nested manner, leading to the nested relations database model (see [Sch88]).

While in the standard relational model a relational schema is determined by the set of attribute names of a relation, the nested relations model allows for different types of relations with the same attributes. These types differ in the structure in which attributes are nested to sub-relations, and transformations between them can be performed by the operations $\nu$ (nest) and $\mu$ (unnest).

In order to explain how these operators work, it suffices to consider two attributes $A$ and $B$ and the transformation between relations of the types $\{A \times B\}$ and $\{A \times \{B\}\}$. Here we use $\{\cdot\}$ for building relations, i. e. $\{A \times B\}$ is the type of a (traditional) relation with the two attributes $A$ and $B$ while in $\{A \times \{B\}\}$ the second attribute is relation-valued with relations with the single attribute $B$.

So far "relation" meant "database relation". We may however think of the "relations" in the mathematical sense as well if we keep in mind that database relations are always finite. In this light, objects of type $\{A \times B\}$ are finite subsets of $A \times B$ and objects of type $\{A \times \{B\}\}$ are finite subsets of the product of $A$ and the set of finite subsets of $B$. Denoting the projection onto $A$ by $\pi_A$ and writing pairs in angle brackets, the functions $\nu$ and $\mu$ between these two types are given as follows:

$$
\begin{aligned}
\nu \quad &: \quad \{A \times B\} \rightarrow \{A \times \{B\}\} \\
\nu(R) \quad &:= \quad \{\langle a, \hat{b} \rangle \mid a \in \pi_A(R) \text{ and } \hat{b} = \{b \in B \mid \langle a, b \rangle \in R\}\}
\end{aligned}
$$

and

$$
\begin{aligned}
\mu \quad &: \quad \{A \times \{B\}\} \rightarrow \{A \times B\} \\
\mu(S) \quad &:= \quad \{\langle a, b \rangle \mid \exists \langle a, \hat{b} \rangle \in S.\ b \in \hat{b}\} \ .
\end{aligned}
$$

The transition $\nu : \{A \times B\} \rightarrow \{A \times \{B\}\}$ pairs each $A$-value with the set of $B$-values associated with it. The operation $\mu : \{A \times \{B\}\} \rightarrow \{A \times B\}$ reverses this by taking an $A$-value and the associated set of $B$-values and producing those pairs which have the $A$-value in the first component and one of the $B$-values in the second component. In fact, we always have $\mu \circ \nu = \mathsf{id}$ [JS82]. The reverse composition $\nu \circ \mu$, however, does not equal the identity. This does not come as a surprise since in a nested relation of type $\{A \times \{B\}\}$ one $a \in A$ may occur together with different sets of $B$-values. This grouping is not respected by $\nu \circ \mu$ as the sample relation $R$ illustrates.

$R$ :

| $A$ | $\{B\}$ |
|---|---|
| $a_1$ | $b_1$<br>$b_2$ |
| $a_1$ | $b_2$<br>$b_3$ |
| $a_2$ | $\emptyset$ |

Applying the unnest operator yields

$\mu(R)$ :

| $A$ | $B$ |
|---|---|
| $a_1$ | $b_1$ |
| $a_1$ | $b_2$ |
| $a_1$ | $b_3$ |

It emerges that the information on how $a_1$ was linked to subsets of $B$ has vanished. Moreover, the information about $a_2$ which was linked to the empty set is gone as well. Neither of this information can be recovered by nesting the relation again:

$\nu\mu(R)$ :

| $A$ | $\{B\}$ |
|---|---|
| $a_1$ | $b_1$<br>$b_2$<br>$b_3$ |

The mere fact that the composition $\nu \circ \mu$ does not equal the identity is not worrying. In fact, operations such as the projection are specifically made for losing information and thus do not equal the identity as well. However, we believe that in a setting of relations ordered by their information contents, operations which transform relations should always be below (or equal to) the identity function. This corresponds to the idea that by merely operating on the given data we can never gain information which was not present before.

There are several possibilities of endowing relational databases with an information ordering, for instance one could compare relations with the same scheme by "$\subseteq$" or "$\supseteq$" (for a discussion of this, see [BJO91]). The different possibilities reflect different ways of thinking about the information contents of a database relation. If, however, we adopt one of these ways, we should use it in a unique fashion for relations on all nesting levels. With this in mind, it seems odd that our sample relation loses lines under the process of unnesting and nesting while the remaining relation on the inner level (sets of $B$-values) increases. Thus, under no suitable interpretation we will have $\nu \circ \mu \leq \mathsf{id}$. Variants of the operations $\nu$ and $\mu$ as introduced in [Sch88] do not improve this situation.

The underlying problem is the unclear semantics of nested relations. If we use the domain theoretic approach to a semantics of databases which was suggested by Buneman et. al [BJO91] and if we interpret relations as elements of the snack powerdomain [Puh93, JP95, Puh95] we are much better off. In this case it is possible to define functions $\mathsf{nest}$ and $\mathsf{unnest}$ with $\mathsf{nest} \circ \mathsf{unnest} \leq \mathsf{id}$ and $\mathsf{unnest} \circ \mathsf{nest} = \mathsf{id}$. Thus, the two functions form an embedding-projection pair, a connection between objects of different nesting levels which loses as little information as possible.

# 3 The domain theoretic database model

## 3.1 Froms sets to domains

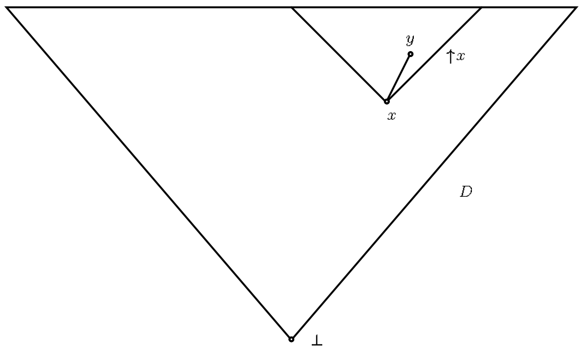
The domain theoretic approach to modelling databases was introduced by Buneman et al. [BJO91]. Their idea was to replace unstructured sets by Scott-domains to serve as database domains. Scott-domains are algebraic ordered sets with least element in which every directed subset and every (upward) bounded set has a supremum.[1] Algebraicity means that every element is the supremum of compact elements. They are those elements which are finitely representable. Hence, Scott-domains allow for calculations with finite representations — an indispensible prerequisite of an implementation of the database model on a computer.

The ordering of a Scott-domain is interpreted as an order of information: The larger elements contain more information, the maximal elements are those of total information whereas non-maximal elements incorporate partially specified objects. These are approximations to those maximal elements which are above them.

[1]See e. g. [AJ95] for an introduction into domain theory.

Database tuples will then be replaced by compact elements of a domain, allowing us to represent partial information in the database by taking non-maximal elements of the domain.

To provide the reader with a better intuition of the more complicated issues which we deal with later, we will offer pictorial representations of (Scott-)domains in the sequel. As usual, a domain $D$ is drawn as an upside-down triangle. The bottom corner corresponds to the bottom (least) element of the domain, and $x \leq y$ means that we can reach $y$ from $x$ by somehow going up within the triangle. The maximal elements of the domain thus correspond to the top side of the triangle, and the upper set $\uparrow x$ generated by one element can be thought of as a little triangle with its top side being part of the top side of the triangle which stands for the domain:



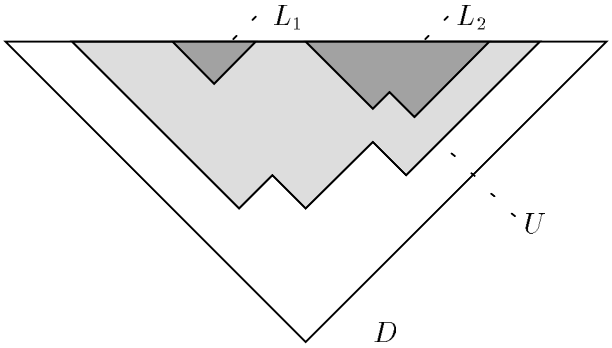

## 3.2 Snacks as approximations of sets

By using domains rather than unstructured sets it is possible to work with partial information in a database. However, as the picture stands, we only know how to represent partiality with respect to individual elements. But database relations are sets of such elements, and the spirit of [BJO91] calls for a domain of approximations to sets. To this end, Buneman invented the notion of a *snack*, and various aspects of the snack-powerdomain were studied in [Nga92, Puh93, JP95, Puh95]. Snacks formalize the idea of approximating sets of maximal elements of a given domain by giving an outer bound and specific information about individual elements. Let us first give the precise definition:

**Definition 3.1** *A* snack *on a domain $D$ is a pair $(U, \mathcal{L})$ where $U$ is a compact open subset of $D$ and $\mathcal{L}$ is a finite set of nonempty compact open subsets of $D$ such that $L \subseteq U$ for all $L \in \mathcal{L}$ and $\mathcal{L}$ is an antichain w.r.t. subset-inclusion.*

Note that the compact open subsets of a domain are exactly those upper sets which are generated by finitely many compact elements of the domain. Because of this, the building blocks of a snack can be represented by a finite amount of information; this is essential if we think of an implementation on a computer.

A typical snack looks like the following:




We think of the set $U$ as an outer bound of the approximated set $S$, thus $S$ being a subset of $U \cap D_{\max}$. The sets $L \in \mathcal{L}$ (in our case $L_1$ and $L_2$) specify individual elements of $S$ in the sense that in each $L \in \mathcal{L}$ there must be an element of $S$. From this it is clear that $L \subseteq U$ since this element cannot be outside the outer bound $U$.

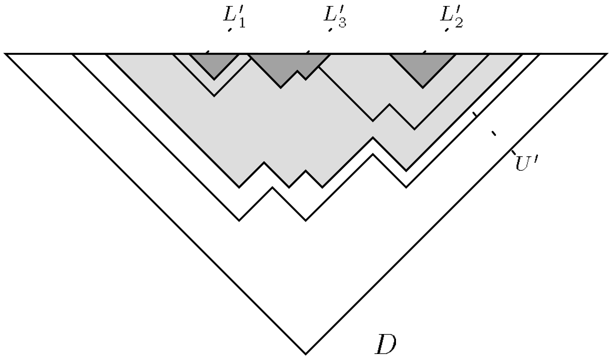
With this interpretation in mind the following order relation on snacks is easy to understand:

**Definition 3.2** *The set of snacks on a domain is ordered by*

$$
\begin{aligned}
&(U, \mathcal{L}) \sqsubseteq (U', \mathcal{L}') \\
&\quad \Longleftrightarrow \quad U \supseteq U' \text{ and for all } L \in \mathcal{L} \text{ there is some } L' \in \mathcal{L}' \text{ such that } L \supseteq L' \ .
\end{aligned}
$$

The definition simply says that in order to get a better approximation one has to shrink the outer bound and each of the sets $L$ which specify individual elements of the approximated set. Note that the introduction of "new" sets $L' \in \mathcal{L}'$ which are not related to the sets $L \in \mathcal{L}$ is allowed so that one may add information about further elements of the approximated set.

The following picture illustrates the snack-ordering: The snack $(U', \{L'_1, L'_2, L'_3\})$ is better than the previously shown snack $(U, \{L_1, L_2\})$.




In the domain theoretic model, database relations will be replaced by snacks (the next passage will say how), but with this construction we can do more, namely we can include a type-constructor "set of" into the database type system. This is the basis of interpreting nested relations, as we will see later. To lay the foundation of this we cite from [Puh93]:

**Definition 3.3** *Let $D$ be a domain. The* snack powerdomain *of $D$ is the ideal completion of the ordered set $\mathcal{S}(D)$ of snacks on $D$.*

**Theorem 3.4** *The snack powerdomain of a Scott-domain is a Scott-domain.* ▮

This is fine since it ensures that we do not leave the world of Scott-domains by applying the snack-construction. In the following, however, we do not need to use the ideals of snacks themselves. We take advantage of the fact that an algebraic domain is completely determined by its compact elements. Hence we can stick to the ordered set of snacks $\mathcal{S}(D)$ to do all relevant calculations.

## 3.3 Interpreting database relations

Just as database tuples are replaced by elements of a domain, we now replace database relations by snacks on the ground domain. There is a standard interpretation of database relations in this new setting, such that we can regard the domain theoretic database model as an extension to the relational model.

For this, database relations are interpreted as being the sets which are to be approximated, i. e. they are considered as totally specified. The set of potential entries of a relation $R$ is enriched with a bottom element, yielding a flat domain. If $R$ consists of the tuples $a_1, \ldots, a_n$ then we identify $R$ with the snack $(\{a_1, \ldots, a_n\}, \{\{a_1\}, \ldots, \{a_n\}\})$. The only set $S$ which is approximated by this snack is $R$ itself, since $S$ must be a subset of $R = \{a_1, \ldots, a_n\}$ and each of the elements $a_1, \ldots, a_n$ must be in $S$.

## 3.4 Interpreting nested relations

In order to interpret nested relations within the domain theoretic database model we can restrict ourselves to the two type-constructors "product" ($\times$) and "set of" ($\{\cdot\}$).[2] On the domain theoretic side these constructors are interpreted as cartesian product resp. the snack powerdomain. As the category of Scott domains is closed under both constructions, we do not leave the scope of our model by these means.

With $\times$ and $\{\cdot\}$ we are able to build the types of nested relations. For this exposition we restrict ourselves to relations with two attributes $A$ and $B$. The flat relation with both attributes will have entries out of $A \times B$, and in the domain theoretic model, the relation is a snack on $A \times B$, i. e. an element of $\mathcal{S}(A \times B)$. The nested relation with $B$-nests is of type $\{A \times \{B\}\}$ i. e. an approximation of a set whose elements are pairs, consisting of an element of $A$ and an approximation of a subset of $B$. In case we start with a (traditional) nested relation where we take all sets which occur as totally specified sets, we may wish to omit the word "approximation". This is reflected by the fact that the snack representation of these relations will be maximal in the associated powerdomain. This representation is gained by applying

[2] A richer type system is discussed in [JP95].

the construction which we gave above in a nested manner. We illustrate this by the sample relation from Section 2:

$R:$
<table>
<tr><th>$A$</th><th>$\{B\}$</th></tr>
<tr><td>$a_1$</td><td><table><tr><td>$b_1$</td></tr><tr><td>$b_2$</td></tr></table></td></tr>
<tr><td>$a_1$</td><td><table><tr><td>$b_2$</td></tr><tr><td>$b_3$</td></tr></table></td></tr>
<tr><td>$a_2$</td><td>$\emptyset$</td></tr>
</table>

At first, we give the snacks that correspond to the sets of $B$-values:

| subset of $B$ | corresponding snack |
|---|---|
| $\{b_1, b_2\}$ | $(\{b_1, b_2\}, \{\{b_1\}, \{b_2\}\})$ |
| $\{b_2, b_3\}$ | $(\{b_2, b_3\}, \{\{b_2\}, \{b_3\}\})$ |
| $\emptyset$ | $(\emptyset, \emptyset)$ |

These snacks are now paired with the respective $A$-values. Out of the resulting pairs we then build the snack that corresponds to the nested relation. Its $U$-part is

$$\Big\{\Big\langle a_1, (\{b_1, b_2\}, \{\{b_1\}, \{b_2\}\})\Big\rangle, \Big\langle a_1, (\{b_2, b_3\}, \{\{b_2\}, \{b_3\}\})\Big\rangle, \Big\langle a_2, (\emptyset, \emptyset)\Big\rangle\Big\} \ .$$

For the $\mathcal{L}$-part we have to make a set $L \in \mathcal{L}$ out of each of these pairs. Hence, the $\mathcal{L}$-part is

$$\Big\{\Big\{\langle a_1, (\{b_1, b_2\}, \{\{b_1\}, \{b_2\}\})\rangle\Big\}, \Big\{\langle a_1, (\{b_2, b_3\}, \{\{b_2\}, \{b_3\}\})\rangle\Big\}, \Big\{\langle a_2, (\emptyset, \emptyset)\rangle\Big\}\Big\}.$$

We also give the definition for the general case:

**Definition 3.5** *The* snack representation $O^S$ *of an object $O$ (i. e. a tuple or relation) of a nested relational database is recursively defined as follows:*

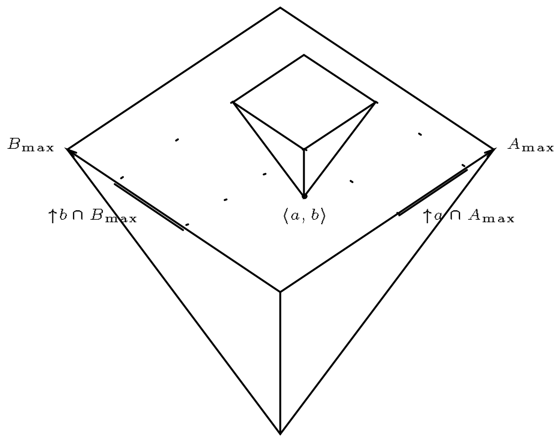
- *If $O$ is a relation $R$ then* $R^S := (\{r^S \mid r \in R\}, \{\{r^S\} \mid r \in R\})$,
- *if $O$ is a tuple $t = \langle t_1, \ldots, t_n\rangle$ then* $t^S := \langle t_1^S, \ldots, t_n^S\rangle$,
- *if $O$ is an element $g$ of a database ground domain, then* $g^S := g$.

# 4 Nesting and unnesting snacks

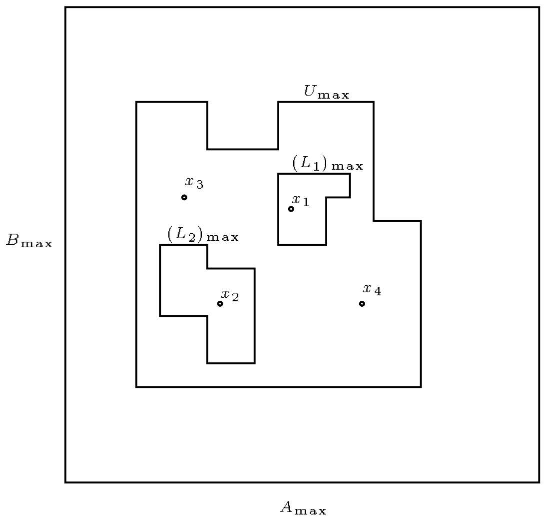
## 4.1 Understanding nested snacks

In the previous section we saw how nested relations are represented in the domain theoretic model. We used snacks for the construction of set-valued types and got representations of nested relations as maximal elements in the snack-powerdomain. We now turn from maximal to arbitrary elements in the snack-powerdomain. They incorporate database entries of partial information. The aim of the following is to give an intuitive account of the quite involved structures which occur.

First of all, we do the step from the triangle picture for a single domain to a picture of the product of two domains $A$ and $B$. This is given by an upside-down four-sided pyramid. The base of the pyramid comes on top and corresponds to the maximal elements of $A \times B$. It can be thought of as coordinatized in the two directions by maximal elements of $A$ resp. $B$. Without investigating the more complex interior structure of the pyramid (if seen as the product of the triangles for $A$ and $B$) we can think of the elements $\langle a, b\rangle \in A \times B$ as points in the pyramid, and the upper set $\uparrow\langle a, b\rangle$ generated by a point as a little four-sided pyramid within the large pyramid for $A \times B$. Its base is a rectangular part of the base of that pyramid, and one side of the rectangle corresponds to $\uparrow a \cap A_{\max}$, the other to $\uparrow b \cap B_{\max}$:

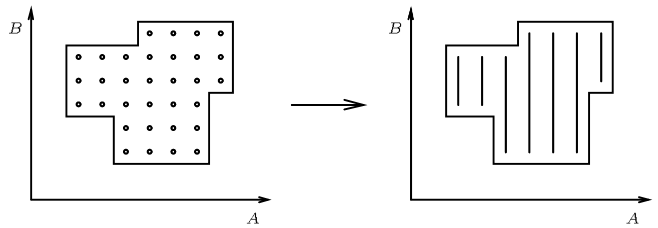



Finitely generated upper sets as they are used in the construction of snacks are the union of finitely many four-sided pyramids and therefore form some kind of an upside-down roof (or a collection of roofs). The shadow they cast on the base of the pyramid (i. e. their intersection with the base) is a finite union of rectangular polygons. In order to make statements about a set $M \subseteq A_{\max} \times B_{\max}$ which is approximated by a snack $(U, \mathcal{L})$ it suffices to look at the pyramid's base and these shadows. Here is an example:




The set $M = \{x_1, \ldots, x_4\}$, for example, is approximated by $(U, \{L_1, L_2\})$ since all elements of $M$ are in $U$ and for each $L \in \mathcal{L} = \{L_1, L_2\}$ there is an element of $M$ which is in $L$, viz. $x_1$ for $L_1$ and $x_2$ for $L_2$.

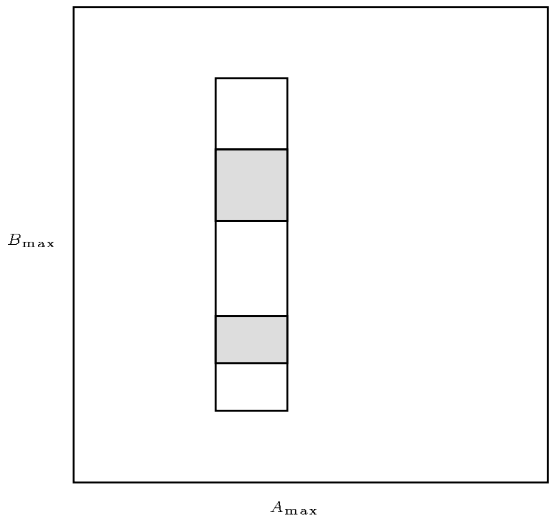
The pictures drawn and thought of so far stem from simultaneously considering the $A$- and the $B$-component of the elements. In nested relational databases, nesting along $B$ (i. e. making $B$-nests) results in a different way of looking at the situation. Now each $A$-value is given together with the *set of* $B$-values associated with it. Intuitively, we change from a set of points to a set of lines:




In the presence of partiality, things become more involved. In a first step, we introduce partiality for $B$-values, i. e. every $A$-value will be given together with a snack that approximates the set of $B$-values that go with it. However, also the $A$-values are only known approximatively. Again, we express this by using a snack. It will give upper and lower bounds for the set of pairs $\langle a, (U^B, \mathcal{L}^B)\rangle$ that approximate lines (rather than elements) of the approximated subset of $(A \times B)_{\max}$.

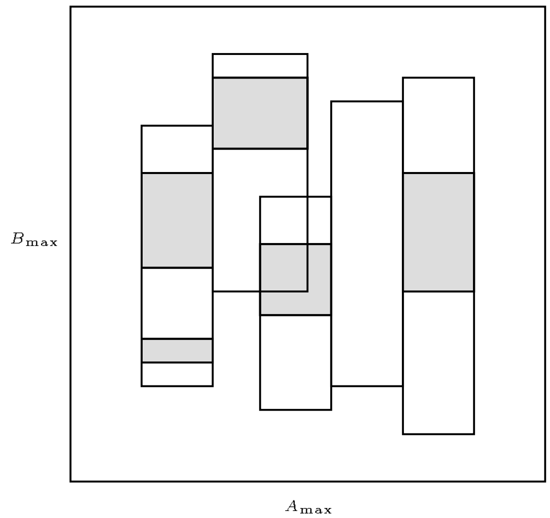
In order to clarify these ideas, let us look at the structure and the interpretation of a nested snack. It will be of the form $(U^{A\{B\}}, \mathcal{L}^{A\{B\}})$ where $\mathcal{L}^{A\{B\}}$ consists of finitely many sets $L^{A\{B\}}$. These and the set $U^{A\{B\}}$ are upper sets in $A \times \mathcal{S}(B)$ which are generated by finitely many pairs of the form $\langle a, (U^B, \mathcal{L}^B)\rangle$, i. e. an element $a \in A$ together with a snack on $B$.

For a pictorial presentation of this situation we will draw $U^{A\{B\}}$ and each $L^{A\{B\}} \in \mathcal{L}^{A\{B\}}$ in a separate picture. Each of the generating pairs of these sets casts a shadow of the following form on $(A \times B)_{\max}$:




The $A$-direction of the shadow corresponds to $\uparrow a \cap A_{\max}$, the $B$-direction of the large rectangle to $(U^B)_{\max}$. The shaded rectangles which are within the large rectangle correspond to the sets $L^B \in \mathcal{L}^B$. For each such set $L^B$ we get one shaded region where the $A$-direction is again $\uparrow a \cap A_{\max}$ and the $B$-direction is $(L^B)_{\max}$. We call this picture a "stripe".

The sets $U^{A\{B\}}$ and $L^{A\{B\}}$ are gotten by composing the stripes of the generating pairs into one picture:




As the example shows, the stripes of the generating pairs may intersect, and the sets $\mathcal{L}^B$ may be empty, resulting in a stripe without shaded subregions. Another effect which may occur (but cannot be drawn) is that there may be pairs $\langle a, (\emptyset, \emptyset)\rangle$. In this case, the "stripe" vanishes because there is no intersection of $\emptyset$ with $B_{\max}$.

The interpretation of a union of stripes depends on whether it stems from the set $U^{A\{B\}}$ or a set $L^{A\{B\}}$.

We first investigate the case where $U^{A\{B\}}$ is the underlying set. The nested snack is a two-step approximation of a set $M$ of lines $\{a\} \times B'$ (with $a \in A_{\max}$ and

$B' \subseteq B_{\max}$). In the first step, given by the snack on the outer level, the possible $A$-values of these lines are determined. Our picture of the shadow cast by $U^{A\{B\}}$ says that all lines which are in $M$ must have one of the $A$-values covered by projection of the union of stripes onto $A_{\max}$. It remains to find the sets $B' \subseteq B_{\max}$ which are possibly going with the $A$-value. This second step of approximation is made by the individual stripes. Every possible $A$-value stems from one of the stripes in the picture. The set $B'$ must be within the $B$-extension of that stripe, and within the $B$-extension of each shaded region of the stripe there must be an element of $B'$. We can summarize this by saying that the $U^{A\{B\}}$-picture does not tell us anything about the existence of lines. It only restricts the set of possible lines in two ways: Firstly, with respect to the possible $A$-values, and secondly with respect to the sets $B'$ which are allowed for an $A$-value.

We now look at the case where our picture stems from a set $L^{A\{B\}}$. Now the picture gives information about a line that must exist in $M$. Its $A$-value must be within the $A$-extension of one of the stripes. Once we choose the stripe, this determines the set $B'$ which is associated to the $A$-value in the same way as in our first case: The set $B'$ must be within the $B$-extension of the stripe and it must meet each of the shaded regions.

Thus, the collection $\mathcal{L}^{A\{B\}}$ of sets $L^{A\{B\}}$ determines a lower bound of the set $M$ by giving an approximation to a collection of indiviual lines which are guaranteed to be in $M$.

## 4.2 Nesting snacks

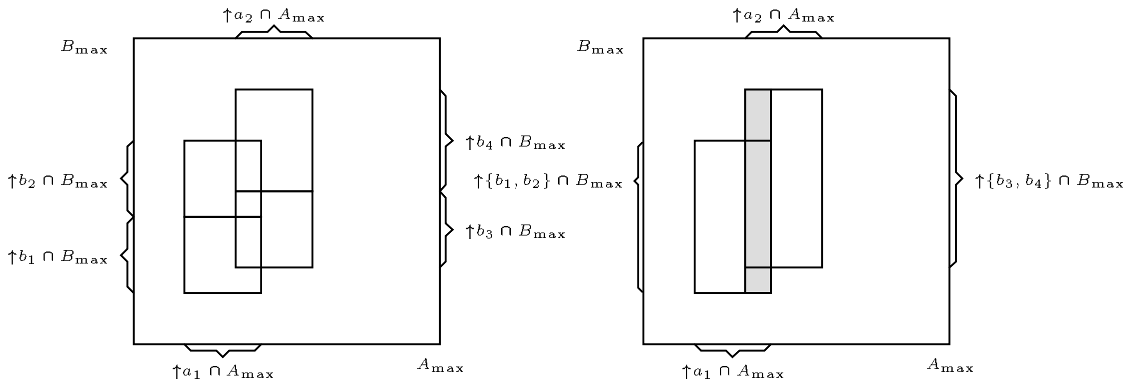
Hopefully, you now have a good intuition of nested snacks so that we can go on and discuss how to nest a snack on a product $A \times B$ to get $B$-nests. We start with a snack $(U, \mathcal{L})$ on $A \times B$. This is to be converted to a snack $(U^{A\{B\}}, \mathcal{L}^{A\{B\}})$ of the kind which we discussed in the previous section.

Before we go on we fix a piece of notation:

**Definition 4.1** *For a subset $R \subseteq A \times B$ and an element $a \in A$ we define*

$$[a]R := \{b \in B \mid \langle a, b\rangle \in R\} \ .$$

At first we determine $U^{A\{B\}}$. The snack $(U, L)$ approximates a set $M \subseteq (A \times B)_{\max}$ in its "point"-representation, and each $m \in M$ must be in $U \cap (A \times B)_{\max}$. If we change to the "line"-representation $\bar{M}$ of $M$, we must make sure that all the lines are within $U \cap (A \times B)_{\max}$. This means that the union of the stripes which correspond to elements of $U^{A\{B\}}$ must be $U \cap (A \times B)_{\max}$. Hence, given an $A$-value $a$, we take those $B$-values as elements of $U^B$ which are in $U$-relation with $a$, i. e. those $b \in B$ with $\langle a, b\rangle \in U$. We do not impose any restriction in the sense of shaded regions. Any information concerning guaranteed elements will be dealt with later. Thus, for an $a \in A$, we get the pair $\langle a, ([a]U, \emptyset)\rangle$ as an element of $U^{A\{B\}}$. Note that we do not need to restrict the $A$-values. If there is no matching $B$-value for some $a \in A$, we will simply get the pair $\langle a, (\emptyset, \emptyset)\rangle$, the shadow of which is empty. You may check yourself that this construction works in general, however here is an example for a set $U$ and its transformation to the set $U^{A\{B\}}$:




We have

$$U = \uparrow\{\langle a_1, b_1\rangle, \langle a_1, b_2\rangle, \langle a_2, b_3\rangle, \langle a_2, b_4\rangle\} \ .$$

The generators of $U^{A\{B\}}$ then are

$$\langle a_1, (\uparrow\{b_1, b_2\}, \emptyset)\rangle \text{ and } \langle a_2, (\uparrow\{b_3, b_4\}, \emptyset)\rangle$$

but also

$$\langle \bot_A, (\emptyset, \emptyset)\rangle \text{ and } \langle a_1 \vee a_2, (\uparrow\{b_1, b_2, b_3, b_4\}, \emptyset)\rangle \ .$$

While the first two generators do not come as a surprise, the third and the fourth generator need some explanation. The pair $\langle \bot_A, (\emptyset, \emptyset)\rangle$ expresses that there is no $B$-values which is paired with the least element of $A$. As far as the shadows are concerned it is not needed, but in the spirit of the approximation of lines which we gave above, we have to include it. The shadow of the pair $\langle a_1 \vee a_2, (\uparrow\{b_1, b_2, b_3, b_4\}, \emptyset)\rangle$ seems to be covered by the first two generators. However, they do not allow for really long lines which extend over the whole of the shaded region. If we did not include this fourth generator, we would have to split up these long lines artificially into two shorter lines which are covered by the first two generating pairs.

Let us now turn to determining $\mathcal{L}^{A\{B\}}$. It will consist of finitely many sets $L^{A\{B\}}$ each of which is generated by finitely many "stripes". Each $L \in \mathcal{L}$ informs us of an element that is guaranteed to belong to the approximated set $M$. We can now turn this into information about a line that belongs to the line representation $\bar{M}$ of $M$. Thus, for each $L \in \mathcal{L}$ we get one set $L^{A\{B\}} \in \mathcal{L}^{A\{B\}}$, and now we will explicate how this transformation is done.

Given an $L \in \mathcal{L}$, the line which we guarantee may actually be a very short line, consisting of one point only. As we have only approximative information of the whereabouts of this point, the set $L^{A\{B\}}$ may nevertheless have to deal with larger regions. The first question is: For which $A$-values do we include a stripe into $L^{A\{B\}}$? As we are speaking of guaranteed elements of $M$ resp. $\bar{M}$, we only want those $a \in A$

to appear which — together with some $b \in B$ — form a pair $\langle a, b\rangle \in L$. Given an $a$ of that kind, we ask for the maximal extension of $B$-values that go with $a$. This is determined by the set of $B$-values which are connected with $a$ in the set $U$. Hence, the $B$-extension of an $a$-stripe is given by $[a]U$. We now have to determine the shaded regions of the $a$-stripe. As the set $L$ gives rise to just one guaranteed point, we can at most guarantee one $B$-value to go with $a$ and hence have only one shaded region. In fact, this is to be thought of as a guaranteed $B$-value only in case that the guaranteed $A$-value happens to be the one whose stripe we are about to construct. The $B$-values of this single shaded region are gotten out of $L$: They are those $b \in B$ which are in $L$-relation with $a$, i. e. the stripe we want to construct is cast as a shadow by the pair $\langle a, ([a]U, \{[a]L\})\rangle$. As said above, we have to do this for all $a \in \pi_1(L)$, such that $L \in \mathcal{L}$ is translated into

$$\{\langle a, ([a]U, \{[a]L\})\rangle \mid a \in \pi_1(L)\} \ .$$

This completes our construction of the nesting operator, which we summarize by giving the function definition:

$$\begin{aligned}
&\mathsf{nest} : \mathcal{S}(A \times B) \to \mathcal{S}(A \times \mathcal{S}(B))\\
&\mathsf{nest}(U, \{L_1, \ldots, L_k\})\\
&\quad := \ \Big(\uparrow\{\langle a, \big([a]U, \emptyset\big)\rangle \mid a \in A\},\\
&\qquad\quad \{\uparrow\{\langle c, \big([c]U, \{[c]L_i\}\big)\rangle \mid c \in \pi_1(L_i)\} \mid i = 1, \ldots, k\}\Big) \ .
\end{aligned}$$

For the more critical reader who does not believe in the intuitive approach, we now give a proof that the nest-operation is well formed. We start with collecting a number of useful results. By $\mathsf{fst}(\cdot)$ and $\mathsf{snd}(\cdot)$ we denote the first resp. second component of a snack.

**Proposition 4.2** *If $U \subseteq A \times B$ is an upper set and $a$ an element of $A$, then $[a]U$ is an upper set in $B$.* ▌

**Proposition 4.3** *If $a \leq a'$ in $A$ and $U \subseteq A \times B$ is an upper set then $[a]U \subseteq [a']U$.*

**Proof:** With $\langle a, b\rangle \in U$ we also have $\langle a', b\rangle \in U$ because $U$ is an upper set. ▌

**Proposition 4.4** *If $U$ is an open set in $A \times B$ then so is $[a]U$ in $B$.*

**Proof:** A directed supremum $\bigvee^{\uparrow}_{i\in I} b_i$ is in $[a]U$ if and only if $\bigvee^{\uparrow}_{i\in I}\langle a, b_i\rangle$ is in $U$. ▌

**Proposition 4.5** *For every $a \in A$ and every finitely generated upper set $U \subseteq A \times B$ the set $M := [a]U$ is a finitely generated upper set in $B$. If the generators of $U$ are $\{\langle a_1, b_1\rangle, \ldots, \langle a_n, b_n\rangle\}$ then $M = \uparrow\{b_i \mid a \geq a_i,\ i \in \{1, \ldots, n\}\}$.*

**Proof:** If $\langle a, b\rangle \in U$ then there is some $i \in \{1, \dots, n\}$ with $\langle a, b\rangle \geq \langle a_i, b_i\rangle$. Therefore $M = \{b \mid \langle a, b\rangle \in U\} \subseteq \uparrow\{b_i \mid a \geq a_i, i \in \{1, \dots, n\}\}$. Of course we have $b_i \in M$ for all $i$ with $a \geq a_i$. As $M$ is an upper set (Lemma 4.2), the desired equality is shown. ▮

**Proposition 4.6** *If $U \subseteq A \times B$ is a compact upper set and $a$ is an element of $A$, then $[a]U$ is a compact upper set in $B$.*

**Proof:** The Hofmann-Mislove Theorem [HM81, KP94] tells us that $U$ is the filtered intersection of its open neighbourhoods $O_i$ $(i \in I)$. The set $[a]U = [a](\bigcap_{i\in I} O_i) = \bigcap_{i\in I}([a]O_i)$ is again a filtered intersection of open sets, i. e. a compact upper set. ▮

**Proposition 4.7** *If $U \subseteq A \times B$ is a finitely generated upper set then there are only finitely many sets $[a]U$. If $U$ is generated by the finite set $M$ and $\{a_1, \dots, a_n\} = \pi_1(M)$ then the sets $[a]U$ are those subsets of $B$ which are in relation to a supremum of a subset of $\{a_1, \dots, a_n\}$.*

**Proof:** Let $a \in A$ and $\check{a} := \bigvee\{a_i \mid a_i \leq a, i \in \{1, \dots, n\}\}$. We have $\check{a} \leq a$ and $[\check{a}]U \subseteq [a]U$ because of Proposition 4.3. If $b$ is in $[a]U$, that is if $\langle a, b\rangle \in U$ then there is some $\langle a_i, b_i\rangle \in M$ which is below $\langle a, b\rangle$. Then $\langle \check{a}, b\rangle \geq \langle a_i, b\rangle \geq \langle a_i, b_i\rangle$ from which we learn$\langle \check{a}, b\rangle \in U$. This shows $[\check{a}]U = [a]U$. ▮

**Proposition 4.8** *If $L$ and $L'$ are subsets of $A \times B$ and if there is some $\langle a, b\rangle \in L \setminus L'$ then*

$$[a]L \not\subseteq [a]L' \ . ▮$$

**Proposition 4.9** *If $L$ and $L'$ are compact opens in $A \times B$ satisfying $L \not\subseteq L'$ then there is some $a \in \pi_1(L)$ such that $[a]L \not\subseteq [a']L'$ for all $a' \leq a$.*

**Proof:** There is an element $\langle a, b\rangle \in L \setminus L'$. As $L'$ is an upper set, there is no $a' \leq a$ such that the pair $\langle a', b\rangle$ belongs to $L'$. Consequently, $b \in [a]L \setminus [a']L'$. ▮

**Proposition 4.10** *Let $L$ and $L'$ be compact open subsets of $A \times B$ which are not in $\subseteq$-relation and which are contained in the compact open set. Then the subsets*

$$\uparrow\{\langle c, \big([c]U, \{[c]L\}\big)\rangle \mid c \in \pi_1(L)\} \qquad (\mathbf{1})$$

*and*

$$\uparrow\{\langle c, \big([c]U, \{[c]L'\}\big)\rangle \mid c \in \pi_1(L')\} \qquad (\mathbf{2})$$

*of $A \times \mathcal{S}(B)$ are not in $\subseteq$-relation.*

**Proof:** Following Proposition 4.9, there is some $\hat{c} \in \pi_1(L)$ such that $[\hat{c}]L \not\subseteq [c']L'$ for all $c' \leq \hat{c}$. Hence there cannot be an element in $(\mathbf{2})$ which is less or equal $\langle \hat{c}, \big([\hat{c}]U, \{[\hat{c}]L\}\big)\rangle$. Therefore $(\mathbf{1}) \not\subseteq (\mathbf{2})$. Our claim follows because of the symmetry of the situation. ▮

We now come to the main result about the function $\mathsf{nest}$.

**Lemma 4.11** *The function* $\mathsf{nest}$ *maps each snack* $(U, \{L_1, \ldots, L_k\})$ *on* $A \times B$ *to a snack on* $A \times \mathcal{S}(B)$.

**Proof:** Using Propositions 4.4 and 4.6 we see that $[a]U$ is compact open for all $a \in A$ (and hence for all $a \in \pi_1(L_i)$, $\quad i = 1, \ldots, k$) and $[c]L_i$ is compact open for all $c \in \pi_1(L_i)$ $(i = 1, \ldots, k)$. Because of $\emptyset \neq L_i \subseteq U$ we have $\emptyset \neq [c]L_i \subseteq [c]U$ for $c \in \pi_1(L_i)$ for $i = 1, \ldots, k$. As the snacks on the inner level of the result under $\mathsf{nest}$ have an empty or one-element second component, we do not need to check whether these components are antichains. Hence, the snacks on the inner level are well formed.

On the outer level, we do not need all $a \in A$ resp. $c \in \pi_1(L_i)$ for generation. Using Proposition 4.7 we see that finitely many suffice. Their compactness follows from the compactness of the building components.

It is easy to see that the sets in the second component of $\mathsf{nest}(U, \{L_1, \ldots, L_k\})$ are nonempty and contained in the first component. They are nonempty since so are the sets $L_i$ $(i = 1, \ldots, k)$ and they are contained in the first component because $\pi_1(L_i) \subseteq A$ and $\langle c, ([c](U), \emptyset)\rangle \leq \langle c, ([c]U, \{[c]L_i\})\rangle$ for all $i = 1, \ldots, k$ and $c \in \pi_1(L_i)$.

Finally, Proposition 4.10 shows that the second component of the result of $\mathsf{nest}$ is an antichain w. r. t. $\subseteq$.

This proves that $\mathsf{nest}$ maps snacks on $A \times B$ to well-formed snacks on $A \times \mathcal{S}(B)$. ∎

**Lemma 4.12** *The function* $\mathsf{nest}$ *is monotone.*

**Proof:** Let $(U, \mathcal{L})$ and $(U', \mathcal{L}')$ be snacks on $A \times B$ such that $(U, \mathcal{L}) \sqsubseteq (U', \mathcal{L}')$. For every $a \in A$ we have $[a]U \supseteq [a]U'$. Hence it follows that $\mathsf{fst}(\mathsf{nest}(U, \mathcal{L})) \supseteq \mathsf{fst}(\mathsf{nest}(U', \mathcal{L}'))$. For every $L \in \mathcal{L}$ there is some $L' \in \mathcal{L}'$ with $L' \subseteq L$ and for all $c \in \pi_1(L')$ we have

$$([c]U, \{[c]L\}) \sqsubseteq ([c]U', \{[c]L'\}) \ .$$

Furthermore, $\pi_1(L) \supseteq \pi_1(L')$, i. e. $c \in \pi_1(L)$ for all $c \in \pi_1(L')$. Therefore we can conclude

$$\uparrow\{\langle c, ([c]U, \{[c]L\})\rangle \mid c \in \pi_1(L)\} \supseteq \uparrow\{\langle c, ([c]U', \{[c]L'\})\rangle \mid c \in \pi_1(L')\} \ .$$

For every set in $\mathsf{snd}(\mathsf{nest}(U, \mathcal{L}))$ we thus found a smaller set in $\mathsf{snd}(\mathsf{nest}(U', \mathcal{L}'))$. This proves $\mathsf{nest}(U, \mathcal{L}) \sqsubseteq \mathsf{nest}(U', \mathcal{L}')$. ∎

## 4.3 Unnesting snacks

In order to nest a traditional database relation, one has to know the entire relation that is to be nested. In contrast to this, the unnest operation can be performed stepwise. One takes the tuples of the nested relation seperately. A pair that consists of some $a \in A$ and a set of $B$-values is transformed into the set of pairs $\langle a, b\rangle$ with

$b$ being one of the $B$-values. The result of the unnest operation then is the union of all sets which we obtain in this way.

In case of domain theoretic databases we can proceed in a very similar way. For this we use the characterization of the snack-powerdomain as a free algebra [Puh93]. We start with introducing the relevant operations on snacks.

**Definition 4.13** *The* formal union *on the set $\mathcal{S}(D)$ of snacks on a domain $D$ is given by*

$$\begin{aligned} \uplus \quad &: \quad \mathcal{S}(D) \times \mathcal{S}(D) \to \mathcal{S}(D) \ , \\ (U, \mathcal{L}) \uplus (U', \mathcal{L}') \quad &:= \quad (U \cup U', \mathit{MIN}^{\subseteq}(\mathcal{L} \cup \mathcal{L}')) \end{aligned}$$

*where $\mathit{MIN}^{\subseteq}$ selects those sets which are minimal under subset inclusion.*

The snack $(\emptyset, \emptyset)$ is a neutral element w. r. t. $\uplus$.

Any two snacks on a domain have an infimum. We include this explicitly as an operation:

**Lemma 4.14** *The operation*

$$\begin{aligned} \sqcap \quad &: \quad \mathcal{S}(D) \times \mathcal{S}(D) \to \mathcal{S}(D) \\ (U, \mathcal{L}) \sqcap (U', \mathcal{L}') \quad &= \quad (U \cup U', \mathit{MIN}^{\subseteq}(\{L \cup L' \mid L \in \mathcal{L}, L' \in \mathcal{L}'\})) \end{aligned}$$

*builds the infimum of two snacks.* ∎

We incorporate the algebraic properties of these operations in the following definitions.

**Definition 4.15** *An* ordered algebra *is an algebra with an order relation on the carrier set and operations which are monotone w. r. t. this order.*

**Definition 4.16** *A* snack algebra *is an ordered algebra $(M, \sqsubseteq)$ with two semilattice operations $\uplus, \sqcap : M \times M \to M$ and a constant $e \in M$ such that for all $r, s, t \in M$:*

$$\begin{aligned} r \sqcap s \quad &\sqsubseteq \quad r \\ e \uplus r \quad &= \quad r \\ (r \uplus s) \sqcap t \quad &= \quad (r \sqcap t) \uplus (s \sqcap t) \end{aligned}$$

*A* snack-algebra homomorphism *is a monotone function that preserves $\uplus$, $\sqcap$ and $e$.*

One easily checks that the set of snacks on a domain together with $\uplus$, $\sqcap$ and $(\emptyset, \emptyset)$ (playing the role of $e$) is a snack algebra.

The key to our unnest operation is that every snack can be built out of primitive snacks by means of $\uplus$ and $\sqcap$. The primitive snacks are called "singletons":

**Definition 4.17** *Let $x$ be a compact element of the domain $D$. The* singleton *of $x$ is the snack $\{\!|x|\!\} := (\uparrow x, \{\uparrow x\})$.*

**Lemma 4.18** *Every snack $s$ on a domain $D$ can be built out of singletons by means of $\uplus$, $\sqcap$, and $(\emptyset, \emptyset)$.*

*If $s = (U, \{L_1, \ldots, L_k\})$, $U = \uparrow\{u_1, \ldots, u_n\}$ and $L_i = \uparrow\{l_{i1}, \ldots, l_{im_i}\}$ ($i = 1, \ldots, k$; $m_i \in \mathbb{N}$ for $i = 1, \ldots, k$), then*

$$\begin{aligned} s \quad = \quad & \Big(\{\!|u_1|\!\} \sqcap \ldots \sqcap \{\!|u_n|\!\} \sqcap (\emptyset, \emptyset)\Big) \ \uplus \\ & \Big(\{\!|l_{11}|\!\} \sqcap \ldots \sqcap \{\!|l_{1m_1}|\!\}\Big) \ \uplus \ \ldots \ \uplus \ \Big(\{\!|l_{k1}|\!\} \sqcap \ldots \sqcap \{\!|l_{km_k}|\!\}\Big) \ . \blacksquare \end{aligned}$$

The algebraic characterization of the snack-powerdomain can be stated as follows:

**Theorem 4.19** *The set $\mathcal{S}(D)$ of snacks on a domain $D$ together with $\uplus$, $\sqcap$, and $(\emptyset, \emptyset)$ is the free snack algebra generated by the ordered set $\mathsf{K}D$ of compact elements of $D$, i. e. for every snack algebra $A$ and every monotone mapping $f : \mathsf{K}D \to A$, there is a uniquely determined snack-algebra homomorphism $f^+ : \mathcal{S}(D) \to A$ such that $f^+ \circ \{\!|\cdot|\!\} = f$.* $\blacksquare$

This enables us to define an unnest operation stepwise. Rather than unnesting a snack on $A \times \mathcal{S}(B)$ right away, we take an element of $\mathsf{K}(A) \times \mathcal{S}(B)$ and transform it into a snack on $A \times B$ by pairing the generating elements of the snack on $B$ with the compact element from $A$ (hence the name $\mathsf{pw}$, pairwith):

**Definition 4.20** *Let $\langle a, (U, \{L_1, \ldots, L_k\})\rangle$ be in $\mathsf{K}A \times \mathcal{S}(B)$. We define*

$$\mathsf{pw}(\langle a, (U, \{L_1, \ldots, L_k\})\rangle) := (\uparrow a \times U, \{\uparrow a \times L_1, \ldots \uparrow a \times L_k\}) \ .$$

**Proposition 4.21** *The function $\mathsf{pw}$ maps elements of $\mathsf{K}A \times \mathcal{S}(B)$ to snacks on $A \times B$ and is monotone.*

**Proof:** As $a$ is in $\mathsf{K}A$, the set $\uparrow a$ is compact open in $A$. The products of $\uparrow a$ with the compact opens $U, L_1, \ldots, L_k$ from $B$ are compact open in $A \times B$. The remaining requirements for snacks on $A \times B$ follow from the fact that $(U, \{L_1, \ldots, L_k\})$ is a snack on $B$.

Monotonicity of $\mathsf{pw}$ is also easily seen. $\blacksquare$

Using Theorem 4.19 we get a snack-algebra homomorphism $\mathsf{pw}^+ : \mathcal{S}(A \times \mathcal{S}(B)) \to \mathcal{S}(A \times B)$ which we call $\mathsf{unnest}$:

**Definition 4.22** *The function $\mathsf{unnest}$ is the uniquely determined extension of $\mathsf{pw}$ to a snack-algebra homomorphism from $\mathcal{S}(A \times \mathcal{S}(B))$ to $\mathcal{S}(A \times B)$, as given by Theorem 4.19:*

$$\begin{aligned} & \mathsf{unnest}\Big(\uparrow\{\langle a_1, B_1\rangle, \ldots, \langle a_n, B_n\rangle\}, \Big\{\uparrow\{\langle c_{i1}, D_{i1}\rangle, \ldots, \langle c_{im_i}, D_{im_i}\rangle\} \mid i = 1, \ldots, k\Big\}\Big) \\ = \quad & \Big(\mathsf{pw}(\langle a_1, B_1\rangle) \sqcap \ldots \sqcap \mathsf{pw}(\langle a_n, B_n\rangle) \sqcap (\emptyset, \emptyset)\Big) \\ & \quad \uplus \biguplus_{i=1}^{k} \Big(\mathsf{pw}(\langle c_{i1}, D_{i1}\rangle) \sqcap \ldots \sqcap \mathsf{pw}(\langle c_{im_i}, D_{im_i}\rangle)\Big) \ . \end{aligned}$$

In order to unnest a concretely given snack, the following representation is more useful:

**Lemma 4.23** *The function* unnest *satisfies*

$$\begin{aligned}&\mathsf{unnest}(\uparrow\{\langle a_1, B_1\rangle, \ldots, \langle a_n, B_n\rangle\}, \{\uparrow\{\langle c_{i1}, D_{i1}\rangle, \ldots, \langle c_{im_i}, D_{im_i}\rangle\} \mid i = 1, \ldots, k\})\\ &= (\bigcup_{j=1}^{n}(\uparrow a_j \times \mathsf{fst} B_j), \mathit{MIN}^{\subseteq} \bigcup_{i=1}^{k}\{\bigcup_{j=1}^{m_i}(\uparrow c_{ij} \times \ell(j)) \mid \ell{:}\{1, \ldots, m_i\} \xrightarrow{\odot} \bigcup_{j=1}^{m_i} \mathsf{snd} D_{ij}\}),\end{aligned}$$

*where* $\ell{:}\{1, \ldots, m_i\} \xrightarrow{\odot} \bigcup_{j=1}^{m_i} \mathsf{snd} D_{ij}$ *is a choice function, i. e.* $\ell(j) \in \mathsf{snd} D_{ij}$.

**Proof:** We have

$$\begin{aligned}&\Big(\mathsf{pw}(\langle a_1, B_1\rangle) \sqcap \ldots \sqcap \mathsf{pw}(\langle a_n, B_n\rangle) \sqcap (\emptyset, \emptyset)\Big)\\ &\qquad \uplus \biguplus_{i=1}^{k} \Big(\mathsf{pw}(\langle c_{i1}, D_{i1}\rangle) \sqcap \ldots \sqcap \mathsf{pw}(\langle c_{im_i}, D_{im_i}\rangle)\Big)\\ =\;& \Big(\bigcup_{j=1}^{n}(\uparrow a_j \times \mathsf{fst} B_j), \emptyset\Big) \uplus \biguplus_{i=1}^{k} \Big(\bigcup_{j=1}^{m_i}(\uparrow c_{ij} \times \mathsf{fst} D_{ij}),\\ &\qquad\qquad \mathsf{MIN}^{\subseteq}\{\bigcup_{j=1}^{m_i}(\uparrow c_{ij} \times \ell(j)) \mid \ell{:}\{1, \ldots, m_i\} \xrightarrow{\odot} \bigcup_{j=1}^{m_i} \mathsf{snd} D_{ij}\}\Big)\\ =\;& \Big(\bigcup_{j=1}^{n}(\uparrow a_j \times \mathsf{fst} B_j), \mathsf{MIN}^{\subseteq} \bigcup_{i=1}^{k}\{\bigcup_{j=1}^{m_i}(\uparrow c_{ij} \times \ell(j)) \mid \ell{:}\{1, \ldots, m_i\} \xrightarrow{\odot} \bigcup_{j=1}^{m_i} \mathsf{snd} D_{ij}\}\Big)\ .\end{aligned}$$

The last step is correct since $\bigcup_{j=1}^{m_i}(\uparrow c_{ij} \times \mathsf{fst} D_{ij}) \subseteq \bigcup_{j=1}^{n}(\uparrow a_j \times \mathsf{fst} B_j)$ for $i = 1, \ldots, k$. ▌

## 4.4 Nest and unnest combined

Now we prove that nest and unnest form, as promised, an embedding-projection pair.

**Lemma 4.24** *For the functions* nest *and* unnest *as defined above we have the identity* $\mathsf{unnest} \circ \mathsf{nest} = \mathsf{id}$.

**Proof:** Let $(U, \{L_1, \ldots, L_k\}) \in \mathcal{S}(A \times B)$. Then

$$\begin{aligned}&\mathsf{unnest}(\mathsf{nest}(U, \{L_1, \ldots, L_k\}))\\ &\quad = \mathsf{unnest}\Big(\uparrow\{\langle a, \big([a]U, \emptyset\big)\rangle \mid a \in A\},\\ &\qquad\qquad \{\uparrow\{\langle c, \big([c]U, [c]L_i\}\big)\rangle \mid c \in \pi_1(L_i)\} \mid i = 1, \ldots, k\}\Big).\end{aligned}$$

For the unnest operation, we can handle the two components of the outer snack seperately. Proposition 4.7 guarantees the existence of a finite subset $\check{A}$ of $A$ with

$$\uparrow\{\langle a, \big([a]U, \emptyset\big)\rangle \mid a \in A\} = \uparrow\{\langle a, \big([a]U, \emptyset\big)\rangle \mid a \in \check{A}\}\ .$$

Moreover, for every $a \in A$ there is some $\check{a} \in \check{A}$ with $\check{a} \leq a$ and $[a]U \subseteq [\check{a}]U$. The first component of the snack that is to be unnested is therefore transformed to

$$\bigcup_{a \in \check{A}} \Big(\uparrow a \times [a]U\Big) = \bigcup_{a \in A} \Big(\uparrow a \times [a]U\Big) \supseteq \bigcup_{a \in A} \Big(\{a\} \times [a]U\Big) = U.$$

As $U$ is an upper set, the reverse inclusion is also seen to hold.

Just as for the first component we find a finite set $\check{C}_i$ for the $i^{\text{th}}$ set in the second component such that $\check{C}_i$ generates $\pi_1(L_i)$ and that this $i^{\text{th}}$ set is generated by the pairs $\langle c, \big([c]U, \{[c]L_i\}\big)\rangle$ with $c \in \check{C}_i$. Note that the second components of the occuring snacks have only one element. Hence, there is only one choice function and this set is transformed to

$$\Big\{ \bigcup_{c \in \check{C}_i} (\uparrow c \times [c]L_i) \Big\} \ .$$

That this equals $L_i$ is seen with the same argumentation as in the case of $U$.

Therefore the second component of the result of $\mathsf{unnest}$ is $\mathsf{MIN}^{\subseteq}(\{L_1, \ldots, L_k\})$. This equals $\{L_1, \ldots, L_k\}$ because the sets $L_1, \ldots, L_k$ form an antichain w. r. t subset-inclusion.

This completes the proof. ▮

In order to prepare the next result we show:

**Proposition 4.25** *Let $U_1, \ldots, U_n$ be subsets of $B$ and $a_1, \ldots, a_n$ elements of $A$. Furthermore, let $c \geq a_i$ and $V \subseteq U_i$ for some $i \in \{1, \ldots, n\}$. Then*

$$[c]\Big(\bigcup_{j=1}^{n} (\uparrow a_j \times U_j)\Big) \supseteq V \ .$$

**Proof:** We have

$$[c]\Big(\bigcup_{j=1}^{n} (\uparrow a_j \times U_j)\Big) \supseteq [c](\uparrow a_i \times U_i) = U_i \supseteq V \ . ▮$$

**Lemma 4.26** *For the functions* $\mathsf{nest}$ *und* $\mathsf{unnest}$ *we have*

$$\mathsf{nest} \circ \mathsf{unnest} \leq \mathsf{id} \ .$$

**Proof:** We apply $\mathsf{nest} \circ \mathsf{unnest}$ to an element of $\mathcal{S}(A \times \mathcal{S}B))$ and introduce abbreviations for some sub-terms:

$$\mathsf{nest}(\mathsf{unnest}(\uparrow\{\langle a_1, B_1\rangle, \ldots, \langle a_n, B_n\rangle\}, \{\uparrow\{\langle c_{i1}, D_{i1}\rangle, \ldots, \langle c_{i_{mi}}, D_{i_{mi}}\rangle\} \mid i = 1, \ldots, k\}))$$

$$= \mathsf{nest}(\underbrace{\bigcup_{j=1}^{n} (\uparrow a_j \times \mathsf{fst} B_j)}_{\mathsf{U}}, \underbrace{\mathsf{MIN}^{\subseteq}(\bigcup_{i=1}^{k} \{\bigcup_{l=1}^{m_i} (\uparrow c_{il} \times \ell(l)) \mid \ell{:}\{1, \ldots, m_i\} \xrightarrow{\odot} \bigcup_{l=1}^{m_i} \mathsf{snd} D_{il}\}}_{\mathsf{L}})) \ .$$

The proof is now done in several steps.
**Step 1**: We show that $\mathsf{fst}(\mathsf{nest}(\mathsf{U}, \mathsf{MIN}^{\subseteq}(\mathsf{L}))) \supseteq \uparrow\{\langle a_1, B_1\rangle, \ldots, \langle a_n, B_n\rangle\}$.

Because of $\mathsf{fst}(\mathsf{nest}(\mathsf{U}, \mathsf{L})) = \uparrow\{\langle a, [a]\mathsf{U}), \emptyset)\rangle \mid a \in A\}$ it suffices to prove

$$([a_i]\mathsf{U}, \emptyset) \sqsubseteq B_i$$

for $i = 1, \ldots, n$. For the second component there is nothing to do since this is empty for the snack on the left hand side. It remains to show

$$[a_i]\Big(\bigcup_{j=1}^{n}(\uparrow a_j \times \mathsf{fst}B_j)\Big) \supseteq \mathsf{fst}(B_i) \ .$$

But this is an immediate consequence of proposition 4.25.
**Step 2**: For every $M \in \mathsf{snd}(\mathsf{nest}(\mathsf{U}, \mathsf{MIN}^{\subseteq}(\mathsf{L})))$ we have to find an $i \in \{1, \ldots, k\}$ such that $M \supseteq \uparrow\{\langle c_{i1}, D_{i1}\rangle, \ldots, \langle c_{im_i}, D_{im_i}\rangle\}$. This means that for every $\mathsf{L}_\bullet \in \mathsf{MIN}^{\subseteq}(\mathsf{L})$ we have to find an $i \in \{1, \ldots, k\}$ with

$$\uparrow\{\langle c, \big([c]\mathsf{U}, \{[c]\mathsf{L}_\bullet\}\big)\rangle \mid c \in \pi_1(\mathsf{L}_\bullet)\} \supseteq \uparrow\{\langle c_{i1}, D_{i1}\rangle, \ldots, \langle c_{im_i}, D_{im_i}\rangle\} \ . \qquad (*)$$

We do not restrict ourselves to those elements of $\mathsf{L}$ which are minimal w. r. t. $\subseteq$. Instead we show the existence of such a number $i$ for all $\mathsf{L}_\bullet \in \mathsf{L}$.

For every $\mathsf{L}_\bullet \in \mathsf{L}$, there is a natural candidate for the $i \in \{1, \ldots, k\}$ which we are looking for. The set $\mathsf{L}$ is the union of certain sets which are distinguished by a parameter between 1 and $k$. The parameter of that subset of $\mathsf{L}$ that contains $\mathsf{L}_\bullet$ is the number we are looking for.

We now assume that $i \in \{1, \ldots, k\}$ is such that

$$\mathsf{L}_\bullet \in \{\bigcup_{l=1}^{m_i}(\uparrow c_{il} \times \ell(l)) \mid \ell{:}\{1, \ldots, m_i\} \xrightarrow{\odot} \bigcup_{l=1}^{m_i} \mathsf{snd}D_{il}\} \ ,$$

and $\ell_\bullet{:}\{1, \ldots, m_i\} \xrightarrow{\odot} \bigcup_{l=1}^{m_i} \mathsf{snd}D_{il}$ is given such that $\mathsf{L}_\bullet = \bigcup_{l=1}^{m_i}(\uparrow c_{il} \times \ell_\bullet(l))$.

In order to show $(*)$, we first verify $c_{ij} \in \pi_1(\mathsf{L}_\bullet)$ for all $j \in \{1, \ldots, m_i\}$.

For all $j \in \{1, \ldots, m_i\}$ we have

$$\pi_1(\mathsf{L}_\bullet) = \pi_1(\bigcup_{l=1}^{m_i} \uparrow c_{il} \times \ell_\bullet(l)) \supseteq \pi_1(\uparrow c_{ij} \times \ell_\bullet(l)) \ .$$

Because of $\ell_\bullet(j) \neq \emptyset$ it follows that $c_{ij} \in \pi_1(\uparrow c_{ij} \times \ell_\bullet(j))$. This yields $c_{ij} \in \pi_1(\mathsf{L}_\bullet)$. To reach our aim of proving $(*)$ it now remains to show

$$\big([c_{ij}]\mathsf{U}), \{[c_{ij}]\mathsf{L}_\bullet)\}\big) \sqsubseteq D_{ij}$$

for all $j \in \{1, \ldots, m_i\}$.

We do this seperately for the two components of the snacks.

Step 2a: We show: $[c_{ij}]\mathsf{U} \supseteq \mathsf{fst}(D_{ij})$.

There is some $\langle \check{a}, \check{B} \rangle \in \{\langle a_1, B_1 \rangle, \ldots, \langle a_n, B_n \rangle\}$ with $\langle \check{a}, \check{B} \rangle \leq \langle c_{ij}, D_{ij} \rangle$ since the first component of a snack is a superset of each element of its second component. Therefore $\check{a} \leq c_{ij}$ and $\mathsf{fst} D_{ij} \subseteq \mathsf{fst} \check{B}$. Consequently, $\mathsf{fst}(B_1), \ldots, \mathsf{fst}(B_n)$, $a_1, \ldots, a_n$, $c_{ij}$ and $\mathsf{fst}(D_{ij})$ meet the requirements of proposition 4.25, and the desired result follows.

Step 2b: We have to show that there is some $L_D \in \mathsf{snd}(D_{ij})$ with

$$[c_{ij}]\mathsf{L}_\bullet \supseteq L_D \ .$$

We have

$$\mathsf{L}_\bullet = \bigcup_{l=1}^{m_i} (\uparrow c_{il} \times \ell_\bullet(l)) \ ,$$

where $\ell_\bullet(l) \in \mathsf{snd}(D_{il})$ for each $l \in \{1, \ldots, m_i\}$. If we choose $L_D = \ell_\bullet(j)$ then the desired result follows directly with Proposition 4.25.

This completes Step 2 and the entire proof. ▌

# 5 A comparison with nested relational algebra

In this section we investigate how snack-representations of nested relations are transformed by means of $\mathsf{nest}$ and $\mathsf{unnest}$. Because of $\mathsf{unnest} \circ \mathsf{nest} = \mathsf{id}$, no snack-representation of a nested relation will be altered by this composition of the two operations. We thus concentrate on the other composition, i. e. $\mathsf{nest} \circ \mathsf{unnest}$, and proceed as follows: We first give a new presentation of snacks which is easier to read. After that we show how nested snacks which represent a nested relation are unnested. From that we can finally move to nesting the flat snacks again and comparing the result with the effects of nested relational algebra. As throughout the paper, we will consider relations with attributes $A$ and $B$ where in the nested relations the nested attribute is $B$.

In order to obtain a more readable presentation of snacks, we introduce a new notation: For a snack $(U, \mathcal{L})$ we tag the set $U$ by a $\Box$ and each $L \in \mathcal{L}$ by a $\Diamond$. This is in line with the interpretation of $\Box$ and $\Diamond$ in modal logic: We take the elements of the approximated set as possible worlds, then $\Box U$ says that each element must satisfy $U$, i. e. be in $U$, and $\Diamond L$ says that there must be an element which is in $L$.[3] We do not require the tagged sets to be upper sets but take the given elements as a (not necessarily minimal) set of generators. Thus, a snack $(U, \{L_1, \ldots, L_k\}$ with generators $u_1, \ldots, u_n$ (for $U$) and $l_{i1}, \ldots, l_{im_i}$ (for $L_i$) can be represented as a collection of tagged relations:

$$\Box\!: \begin{array}{|c|}\hline u_1 \\ \hline \vdots \\ \hline u_n \\ \hline\end{array} \qquad \Diamond\!: \begin{array}{|c|}\hline l_{11} \\ \hline \vdots \\ \hline l_{1m_1} \\ \hline\end{array} \ \cdots \ \Diamond\!: \begin{array}{|c|}\hline l_{k1} \\ \hline \vdots \\ \hline l_{km_k} \\ \hline\end{array}$$

[3]A more detailed account of the use of $\Box$ and $\Diamond$ in a logical representation of the snack powerdomain can be found in [JP95, Puh95].

In nested snacks, each tuple of a tagged relation will have a part which itself is a collection of tagged relations. When we go about to unnest a nested snack of this kind, we may proceed stepwise, as shown in Section 4.3.

The first step is to apply the function pw to the individual tuples of the tagged relations of the outer level. Generally, these tuples look like the following:

$$\begin{array}{|c|c|}\hline a & \Box\colon \begin{array}{|c|}\hline u_1 \\ \hline \vdots \\ \hline u_n \\ \hline \end{array} \quad \Diamond\colon \begin{array}{|c|}\hline l_{11} \\ \hline \vdots \\ \hline l_{1m_1} \\ \hline \end{array} \;\cdots\; \Diamond\colon \begin{array}{|c|}\hline l_{k1} \\ \hline \vdots \\ \hline l_{km_k} \\ \hline \end{array} \\ \hline \end{array}$$

By means of pw these are transformed to the following snack:

$$\Box\colon \begin{array}{|c|c|}\hline a & u_1 \\ \hline \multicolumn{2}{|c|}{\vdots} \\ \hline a & u_n \\ \hline \end{array} \qquad \Diamond\colon \begin{array}{|c|c|}\hline a & l_{11} \\ \hline \multicolumn{2}{|c|}{\vdots} \\ \hline a & l_{1m_1} \\ \hline \end{array} \;\cdots\; \Diamond\colon \begin{array}{|c|c|}\hline a & l_{k1} \\ \hline \multicolumn{2}{|c|}{\vdots} \\ \hline a & l_{km_k} \\ \hline \end{array}$$

The result of unnesting a nested snack is now obtained by putting these pieces together as in Definition 4.22. In doing so, it matters where the tuple comes from. If it stems from a relation tagged with $\Box$ then it adds to the first component of the resulting snack, i. e. the relation that will also be tagged with $\Box$. In this case, the tuples $\langle a, u_1\rangle, \ldots, \langle a, u_n\rangle$ will belong to this first component. All the tuples which come from a sub-relation with a $\Diamond$ are not used.

If the tuple however comes from one of the relations that have a $\Diamond$ then the $\Box$-subrelations will not be needed. Instead, those parts which also have a $\Diamond$ on the inner level will contribute to the result of unnesting. Fortunately, we do not have to discuss how this is done in general. The nested snacks which we will be interested in, viz. those which are a representation of a nested database relation, have a very special structure. In these snacks, all relations which are tagged with a $\Diamond$ have exactly one element because we interpret each element of a database relation as giving information about an element that is guaranteed to be in the set which is described by the relation. So each element makes a singleton $\Diamond$-set. By analyzing the formula in Lemma 4.23, one verifies that in this situation each pair $\langle a, l_{ij}\rangle$ from the above snack will finally form a one-tuple $\Diamond$-tagged relation in the result of unnest.

This leads to the following result:

**Lemma 5.1** *If $\{\langle a_1, B_1\rangle, \ldots, \langle a_n, B_n\rangle\}$ is a nested relation of type $\{A \times \{B\}\}$ then its snack representation is unnested to*

$$\Box\Big(\bigcup_{i=1}^{n}(\{a_i\} \times B_i)\Big), \quad \Big\{\Diamond\,\{\langle a, b\rangle\} \mid \langle a, b\rangle \in \bigcup_{i=1}^{n}(\{a_i\} \times B_i)\Big\}\ . \blacksquare$$

Note that, just as with nested relational algebra, those tuples with empty subrelations will vanish when the nested snack is unnested.

To illustrate the effect of the unnest operation, let us once again look at the sample relation from Section 2.

| $R:$ $A$ | $\{B\}$ |
|---|---|
| $a_1$ | $\begin{array}{\|c\|}\hline b_1\\ \hline b_2\\ \hline\end{array}$ |
| $a_1$ | $\begin{array}{\|c\|}\hline b_2\\ \hline b_3\\ \hline\end{array}$ |
| $a_2$ | $\emptyset$ |

We already gave its snack representation in Section 3.4. In our new notation it looks as follows:

$$\Box:\ \begin{array}{|c|l|}\hline a_1 & \Box: \begin{array}{|c|}\hline b_1\\ \hline b_2\\ \hline\end{array} \qquad \Diamond: \boxed{b_1} \qquad \Diamond: \boxed{b_2} \\ \hline a_1 & \Box: \begin{array}{|c|}\hline b_2\\ \hline b_3\\ \hline\end{array} \qquad \Diamond: \boxed{b_2} \qquad \Diamond: \boxed{b_3} \\ \hline a_2 & \Box : \emptyset \\ \hline\end{array}$$

$$\Diamond:\ \begin{array}{|c|l|}\hline a_1 & \Box: \begin{array}{|c|}\hline b_1\\ \hline b_2\\ \hline\end{array} \qquad \Diamond: \boxed{b_1} \qquad \Diamond: \boxed{b_2} \\ \hline\end{array}$$

$$\Diamond:\ \begin{array}{|c|l|}\hline a_1 & \Box: \begin{array}{|c|}\hline b_2\\ \hline b_3\\ \hline\end{array} \qquad \Diamond: \boxed{b_2} \qquad \Diamond: \boxed{b_3} \\ \hline\end{array}$$

$$\Diamond:\ \begin{array}{|c|l|}\hline a_2 & \Box : \emptyset \\ \hline\end{array}$$

The result of unnesting this snack is:

$$\Box:\ \begin{array}{|c|c|}\hline a_1 & b_1\\ \hline a_1 & b_2\\ \hline a_1 & b_3\\ \hline\end{array} \qquad \Diamond:\ \begin{array}{|c|c|}\hline a_1 & b_1\\ \hline\end{array} \qquad \Diamond:\ \begin{array}{|c|c|}\hline a_1 & b_2\\ \hline\end{array} \qquad \Diamond:\ \begin{array}{|c|c|}\hline a_1 & b_3\\ \hline\end{array}$$

This is exactly the snack representation of $\mu(R)$, the result of unnesting $R$ by means of Schek's unnest operation $\mu$. In fact, one easily checks that this is true in general:

**Theorem 5.2** *Let $R$ be a nested relation of type $\{A \times \{B\}\}$ and $R^S$ be its snack representation. Then* $\mathsf{unnest}(R^S)$ *is the snack representation of* $\mu(R)$. ▮

The difference is made in nesting the relation again. A closer look at the $\mathsf{nest}$ operation reveals that there is no snack $(U, \mathcal{L})$ such that $\mathsf{nest}(U, \mathcal{L})$ is the snack representation of a nested database relation.

For our example, $\mathsf{nest}(\mathsf{unnest}(R))$ is given by

$\Box$:
<table>
<tr><td>$a_1$</td><td>$\Box$: <table><tr><td>$b_1$</td></tr><tr><td>$b_2$</td></tr><tr><td>$b_3$</td></tr></table></td></tr>
<tr><td>$\perp$</td><td>$\Box : \emptyset$</td></tr>
</table>

$\Diamond$:
<table>
<tr><td>$a_1$</td><td>$\Box$: <table><tr><td>$b_1$</td></tr><tr><td>$b_2$</td></tr><tr><td>$b_3$</td></tr></table></td><td>$\Diamond$: <table><tr><td>$b_1$</td></tr></table></td></tr>
</table>

$\Diamond$:
<table>
<tr><td>$a_1$</td><td>$\Box$: <table><tr><td>$b_1$</td></tr><tr><td>$b_2$</td></tr><tr><td>$b_3$</td></tr></table></td><td>$\Diamond$: <table><tr><td>$b_2$</td></tr></table></td></tr>
</table>

$\Diamond$:
<table>
<tr><td>$a_1$</td><td>$\Box$: <table><tr><td>$b_1$</td></tr><tr><td>$b_2$</td></tr><tr><td>$b_3$</td></tr></table></td><td>$\Diamond$: <table><tr><td>$b_3$</td></tr></table></td></tr>
</table>

Let us conclude with a few remarks on this result. First of all, the $\Box$-component contains the tuple that we get as a result of $\nu\mu(R)$ by means of the nested relational algebra operations. So it might seem that nested relations have to be equipped with the $\Box$-meaning in order to satisfy $\nu \circ \mu \leq \mathsf{id}$. But if we did so, the problem of vanishing tuples with empty sub-relations would remain. In order to capture these we have to include the tuple $\langle \perp, \Box : \emptyset \rangle$. In the absence of a least element $\perp$ we would have to include the tuples $\langle a, \Box : \emptyset \rangle$ for all $a \in A$. If $A$ happened to be infinite then we would leave the realm of finite relations. In so far, the order structure with a least element is crucial for a definition of nesting and unnesting operations which build an embedding-projection pair.

The $\Diamond$-components, finally, practically do not build nests at all since each of the elements $b_1$, $b_2$ and $b_3$ gives rise to a seperate $\Diamond$-component. But this is sound if we think of each database entry as a piece of information about objects which are guaranteed to exist.

The result of our operations is only natural if we recall that unnesting is bound to destroy all information on how the $B$-values were grouped in the original relation. When we nest the unnested relation again, the most we can say is that the $B$-groups going with $a_1$ are nonempty subsets of the set $\{b_1, b_2, b_3\}$, and that each of these elements must occur in one $B$-group. Clearly, this is what the nested snack from above says.

# 6 Conclusion

There are two main aspects one has to focus on when using domains for generalizing relational databases. The first is that the domain theoretic model must allow for the datatypes of databases. For relational databases, these are certain ground types and the two type-constructors "finite product" and "set of". While the ground types can be modelled as flat domains, the type-constructors are given by the product of domains and the snack powerdomain. As these can be applied repeatedly, we are even able to model the nested relations model. This was already mentioned in [JP95].

The second aspect is to find meaningful operations for the generalized model. It is desirable that these operations include conservative extensions of (nested) relational algebra. There is no difficulty in generalizing the relational projection. As observed by Buneman [BJO91], this can be done by domain theoretic projection functions. In this paper we defined the operations $\mathsf{nest}$ and $\mathsf{unnest}$. The unnesting operation turns out to be a conservative extension to Jaeschke and Schek's unnesting operator $\mu$. However, we cannot expect $\mathsf{nest}$ to be an extension of their nesting operator $\nu$ since we argued that $\nu$ has no satisfactory semantics. Instead we define $\mathsf{nest}$ and $\mathsf{unnest}$ such that they preserve as much information as possible: $\mathsf{nest}$ as a domain theoretic embedding preserves all information contained in its argument whereas $\mathsf{unnest}$ loses only those pieces of information which need the richer structure of nested snacks in order to be represented.

Domain theoretic databases originally came along with the promise that they would allow for a representation and processing of partial information. Although we did not discuss the effect of $\mathsf{nest}$ and $\mathsf{unnest}$ on snack representations of partial data, it is now possible to study this with respect to domain-theoretic representations of various sorts of null values and otherwise structured data.

What remains is to find suitable generalizations of the relational "select" and "join" operations. While we believe that selections may successfully be modelled as suprema with snacks that incorporate the filter formula of the selection, the generalization of natural join seems to be a more demanding task. For this one will at first have to find a formal semantics of the join operation such as to check the definition of the operation with respect to this semantics. Another approach could be to define a function that, in composition with certain projections, satisfies the various conditions of losless joins. If that were successful, a possible semantics of natural join might emerge from the definition.

# References


[AJ95] S. Abramsky and A. Jung. Domain theory. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 3. Clarendon Press, 1995.

[BJO91] P. Buneman, A. Jung, and A. Ohori. Using powerdomains to generalize relational databases. *Theoretical Computer Science*, 91:23–55, 1991.

[HM81] K. H. Hofmann and M. Mislove. Local compactness and continuous lattices. In B. Banaschewski and R.-E. Hoffmann, editors, *Continuous Lattices, Proceedings Bremen 1979*, volume 871 of *Lecture Notes in Mathematics*, pages 209–248. Springer Verlag, 1981.

[JP95] A. Jung and H. Puhlmann. Types, logic, and semantics for nested databases. *Electronic Notes in Theoretical Computer Science*, (1), 1995. Proceedings of the 11th Conference on Mathematical Foundations of Programming Semantics.

[JS82] G. Jaeschke and H.-J. Schek. Remarks on the algebra of non first normal form relations. In *Proceedings of the ACM SIGACT/SIGMOD Symposium on Principles of DBS*, 1982.

[KP94] K. Keimel and J. Paseka. A direct proof of the Hofmann-Mislove theorem. *Proceedings of the AMS*, 120:301–303, 1994.

[Nga92] Teow-Hin Ngair. *Convex Spaces as an Order-theoretic Basis for Problem Solving*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, July 1992.

[Puh91] H. Puhlmann. A Semantics for Generalized Database Relations. Master's thesis, Imperial College of Science, Technology and Medicine, London, 1991.

[Puh93] H. Puhlmann. The snack powerdomain for database semantics. In A. M. Borzyszkowski and S. Sokołowski, editors, *Mathematical Foundations of Computer Science*, volume 711 of *Lecture Notes in Computer Science*, pages 650–659, 1993.

[Puh95] H. Puhlmann. *Die Snack-Potenzkonstruktion: Grundlage einer Semantik genesteter Datenbanken*. PhD thesis, Technische Hochschule Darmstadt, July 1995. 88pp.

[Sch88] M. H. Scholl. Das Modell geschachtelter Relationen. Dissertation, Fachbereich Informatik, Technische Hochschule Darmstadt, 1988.