

Lossless Decompositions in Complex-Valued Databases

Henning Koehler and Sebastian Link*

Massey University, Palmerston North, New Zealand
{h.koehler,s.link}@massey.ac.nz

Abstract. When decomposing database schemas, it is desirable that a decomposition is lossless and dependency preserving. A well-known and frequently used result for the relational model states that a functional dependency preserving decomposition is lossless if and only if it contains a key. We will show that this result does not always hold when domains are allowed to be finite, but provide conditions under which it can be preserved. We then extend our work to a complex-valued data model based on record, list, set and multiset constructor, where finite domains occur naturally for subattributes, even if the domains of flat attributes are infinite.

1 Introduction

In order to obtain well-designed databases, it is often necessary to decompose a schema into smaller subschemas. When doing so, we must take care not to lose any information - decompositions which preserve information are called *lossless*. In addition, it is important to preserve dependencies specified over the schema, which restrict what data can be stored in the database, thereby avoiding inconsistencies. Thus one typically wants decompositions that are not only lossless, but also *dependency preserving* [2,3,8,9,11,12,16,19].

For the relational data model, lossless and dependency preserving decompositions have often been studied under the (sometimes implicit) assumption that attribute domains are infinite. A well-known and frequently used result (Theorem 1) states that a dependency preserving decomposition of a schema R with a set Σ of functional dependencies (FDs) as constraints is lossless if and only if it contains a key [3]. It is the basis for the well-known synthesis approach introduced by Biskup, Dayal and Bernstein in [3] for finding dependency preserving decompositions into 3NF. Their work has since been extended to find different lossless and dependency preserving decompositions, e.g. into EKNF [19] and BCNF [4,10,18]. In order to extend the synthesis approach to other, e.g. complex-valued data models, it is thus vital to investigate whether Theorem 1 still holds in these models.

* This research is supported by the Marsden fund council from Government funding, administered by the Royal Society of New Zealand.

We will demonstrate in this paper that the characterization of lossless and dependency preserving decompositions is only valid when domains of attributes are assumed to be infinite. This fact is particularly relevant in practice since many domains naturally only carry a finite number of elements. Moreover, when studying lossless and dependency preserving decompositions in complex-valued data models, then finite domains can often not be avoided as well.

More precisely, we will show that Theorem 1 is still valid if LHS-attributes, i.e., attributes which appear in the left hand side X of FDs $X \rightarrow Y \in \Sigma$, have infinite domains while domains of other attributes may be finite. This result is then extended to complex-valued databases that can be generated by a finite number of recursive applications of record, list, set and multiset constructor.

The rest of the paper is organized as follows. We first give a brief introduction to the relational model in Section 2, then examine the relational case in the presence of finite domains in Section 3. In Section 4 we introduce the complex-valued model which we shall use for our investigation. We then show how Theorem 1 can be extended to this complex model in Section 5.

2 The Relational Model of Data

We begin by introducing some basic terms for the relational model. More details can be found e.g. in [13,15,17].

A *relational database schema* consists of a set of relation schemas. A *relation schema* $R = \{A_1, A_2, \dots, A_n\}$ is a finite set of *attributes*. Each attribute A_i has a *domain* $dom(A_i)$ associated with it. Domains are arbitrary sets, but unless explicitly stated otherwise, we will assume that domains are countably infinite.

A *relation* r over a relation schema R is a finite set of tuples, while a tuple t on R is a mapping $t : R \rightarrow \bigcup_{A \in R} dom(A)$ with $t(A) \in dom(A)$. Relations over a schema are also commonly referred to as schema instances or tables. Sets of schema instances, one for each relation schema in a database schema, are called database instances. Note that one could also consider relations which are infinite. Since we will only consider functional dependencies (see below), this would not affect our results.

With each relation schema we associate a set Σ of *functional dependencies* (FD). A functional dependency on R is an expression of the form $X \rightarrow Y$ (read “ X determines Y ”) where X and Y are subsets of R . For attribute sets X, Y and attribute A we will write XY short for $X \cup Y$ and A short for $\{A\}$. We say that an FD $X \rightarrow Y$ *holds* on a relation r over R if every pair of tuples in r that coincides on all attributes in X also coincides on all attributes in Y .

A set Σ of FDs over R *implies* an FD (or set of FDs) Σ' , written $\Sigma \models \Sigma'$, if Σ' holds on every relation r over R for which all FDs in Σ hold. If two sets of FDs Σ and Σ' imply each other, we call Σ a *cover* of Σ' (and vice versa) and write $\Sigma \equiv \Sigma'$. We write Σ^* for the set of all FDs on R implied by Σ .

The *closure* X^* of a set $X \subseteq R$ w.r.t. a set Σ of FDs is the set of all attributes determined by X :

$$X^* := \{A \in R \mid \Sigma \models X \rightarrow A\}$$

We call X *closed* under Σ if $X = X^*$.

A set $X \subseteq R$ is a *key* of R w.r.t. a set Σ of FDs on R , if Σ implies $X \rightarrow R$. Note that some authors use the term ‘key’ only for minimal keys, and call keys which may not be minimal ‘superkeys’. For us, keys need not be minimal.

The *projection* of a set Σ of FDs onto a subschema $R_i \subseteq R$ is

$$\Sigma[R_i] := \{X \rightarrow Y \in \Sigma \mid XY \subseteq R_i\}$$

For a set $X \subseteq R$ we denote the *projection* of r onto the attributes in X by $r[X]$. The *join* of two relations $r[X]$ and $r[Y]$ is a relation on $X \cup Y$:

$$r[X] \bowtie r[Y] := \left\{ t \mid \begin{array}{l} \exists t_1 \in r[X], t_2 \in r[Y]. \\ t[X] = t_1 \wedge t[Y] = t_2 \end{array} \right\}$$

A decomposition $\mathcal{D} = \{R_1, \dots, R_n\}$ of R is a set of subschemas $R_i \subseteq R$ of R such that

$$\bigcup \mathcal{D} = R$$

To ensure that the schemas R_1, \dots, R_n of a decomposition \mathcal{D} of R can hold the same data as the original schema R , we must ask for a decomposition that is *lossless*, i.e., that for all relations r on R for which Σ holds we have

$$r[R_1] \bowtie \dots \bowtie r[R_n] = r$$

Furthermore, the decomposition should be dependency preserving, i.e., the dependencies on the schemas R_i which are implied by Σ should form a cover of the original functional dependencies Σ :

$$\Sigma \equiv \bigcup \Sigma^*[R_i]$$

This allows a database management system to check constraints for individual relations only, without having to compute their join. In practice FDs *are* only checked for individual relations, so for a given schema (R, Σ) , losslessness and dependency preservation of a decomposition ensure that the semantics of (R, Σ) are adequately preserved.

In the relational model there is a nice characterization when a dependency-preserving decomposition is lossless. Note again that we are currently assuming all domains to be infinite.

Theorem 1. [3] *A dependency-preserving decomposition of R is lossless if and only if it contains a subschema which forms a key of R .*

As a result, the task of finding a dependency preserving and lossless decomposition becomes easier: We can first concentrate on finding a dependency preserving decomposition. If this decomposition is not lossless, Theorem 1 shows that we only need to add a minimal key to the decomposition to make it lossless.

Example 1. Let $R = \{ABC\}$ with FDs $\Sigma = \{A \rightarrow C, B \rightarrow C\}$. Then the decomposition $\mathcal{D} = \{AC, BC\}$ is dependency preserving, but not lossless: for the relation r on R

$$r = \begin{array}{|c|c|c|} \hline A & B & C \\ \hline 0 & 1 & 0 \\ \hline 1 & 0 & 0 \\ \hline \end{array}$$

the FDs in Σ hold, but we get

$$r[AC] \bowtie r[BC] = \begin{array}{|c|c|} \hline A & C \\ \hline 0 & 0 \\ \hline 1 & 0 \\ \hline \end{array} \bowtie \begin{array}{|c|c|} \hline B & C \\ \hline 1 & 0 \\ \hline 0 & 0 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline A & B & C \\ \hline 0 & 1 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 1 & 1 & 0 \\ \hline \end{array} \neq r$$

We can turn \mathcal{D} into a lossless decomposition \mathcal{D}' by adding the key schema AB . Hence,

$$\mathcal{D}' := \mathcal{D} \cup \{AB\} = \{AB, AC, BC\}$$

is both lossless and dependency-preserving.

In order to extend the synthesis approach to other, complex-valued data models, it is thus vital to investigate whether Theorem 1 still holds in these models.

3 Lossless Decompositions - The Relational Case

As shown in [3], Theorem 1 holds when all attribute domains are infinite. If domains are allowed to be finite, the “if” direction still holds - this becomes obvious when we consider that instances over finite domains are also instances over infinite supersets of these domains.

To see where the “only if” direction fails, we first look into the corresponding proof [3], adapting the correctness proof for the chase procedure [1].

Lemma 1. *Let R be a relational schema and Σ a set of FDs on R . Then every lossless decomposition \mathcal{D} of R contains a key of R .*

Proof. Let $\mathcal{D} = \{R_1, \dots, R_n\}$ not contain a key of R . We show that \mathcal{D} is not lossless by constructing a sample relation r on R as follows. Let t be an arbitrary tuple on R . Then for every schema $R_i \in \mathcal{D}$ add a tuple t_i to r which is identical to t on the closure R_i^* of R_i w.r.t. Σ , and contains unique values for attributes outside R_i^* .

Then for every FD $X \rightarrow Y \in \Sigma$, two tuples t_i, t_j are identical on X if and only if $X \subseteq R_i^* \cap R_j^*$. If $X \subseteq R_i^* \cap R_j^*$ holds then Y is also included in $R_i^* \cap R_j^*$, since $R_i^* \cap R_j^*$ is closed under Σ . Thus $X \rightarrow Y$ holds on r .

Since \mathcal{D} contains no key, the tuple t does not lie in r . It does however lie in

$$\bowtie r[\mathcal{D}] := r[R_1] \bowtie \dots \bowtie r[R_n]$$

which makes \mathcal{D} not lossless.

We first note that the proof requires a sufficient number of distinct attribute values for the construction. In this, we implicitly assume that attribute domains are infinite. For the remainder of this paper we will allow attribute domains to be finite, although we still require them to contain at least two elements.

Example 2. Let $R = ABCD$ with FDs $\Sigma = \{A \rightarrow BC, BC \rightarrow A\}$ and decomposition $\mathcal{D} = \{ABC, BD, CD\}$. Let furthermore the domain of A contain only two values, say $dom(A) = \{0, 1\}$. Then \mathcal{D} contains no key of R , but is lossless.

Proof (of Example 2). Assume \mathcal{D} was not lossless, i.e., for some relation r on R there exists a tuple t with

$$t \in (\bowtie r[\mathcal{D}]) \setminus r$$

Then for every $R_i \in \mathcal{D}$ there must be a tuple $t_i \in r$ with $t_i[R_i] = t[R_i]$. We may assume w.l.o.g. that $t = (0, 0, 0, 0)$. This gives us the following subset of r :

$$r' = \begin{array}{|c|c|c|c|} \hline A & B & C & D \\ \hline 0 & 0 & 0 & d \\ \hline a_1 & 0 & c & 0 \\ \hline a_2 & b & 0 & 0 \\ \hline \end{array}$$

for some values a_1, a_2, b, c, d with $a_1, a_2 \in dom(A) = \{0, 1\}$. If $a_1 = 0$ or $a_2 = 0$ then the FD $A \rightarrow BC$ implies $c = 0$ or $b = 0$, respectively, and thus $t \in r$ which contradicts the assumption. If $a_1 = a_2 = 1$ then $A \rightarrow BC$ again implies $b = 0$ and $c = 0$, which gives us the following subset of r :

$$r' = \begin{array}{|c|c|c|c|} \hline A & B & C & D \\ \hline 0 & 0 & 0 & d \\ \hline 1 & 0 & 0 & 0 \\ \hline \end{array}$$

But then the FD $BC \rightarrow A$ would be violated on r' and thus on r .

Note that this example can easily be generalized to work for arbitrary finite domains. For $dom(A) = \{1, \dots, k\}$ chose

$$\begin{aligned} R &= AB_1 \dots B_k C \\ \Sigma &= \{A \rightarrow B_1 \dots B_k, B_1 \dots B_k \rightarrow A\} \\ \mathcal{D} &= \{AB_1 \dots B_k, R \setminus AB_1, \dots, R \setminus AB_k\} \end{aligned}$$

Obviously we need some restrictions on the domains for Lemma 1 to work. However, requiring all domains of attributes occurring in R to be infinite is too strong. While finite domains could perhaps be ignored in relational databases, we will see that they arise naturally in complex-valued databases for subattributes such as $\{\lambda\}$, for which the domain contains only the two values \emptyset and $\{ok\}$.

In the example above, the attribute A occurred in the LHS of an FD in Σ . It turns out that requiring that these attributes have infinite domains is sufficient. Recall that even if we allow the domain of an attribute to be finite, we still require it to contain at least two different values.

Lemma 2. *Let R be a relational schema and Σ a set of FDs on R . If all attributes which occur in the LHS of an FD in Σ have infinite domains, then every lossless decomposition \mathcal{D} of R contains a key of R .*

Proof. As for Lemma 1, except that for attributes not occurring in the LHS of an FD in Σ we do not require the t_i to have unique values on these attributes, but only values different from t .

Note that instead of working with Σ , we could use a cover of Σ instead. This can affect which attributes appear in the LHS of FDs, and thus give us different requirements for the domains in Lemma 2. However, it can easily be shown (although we will not do that here) that all *reduced* covers [15] - these are covers in which no attributes can be removed from FDs while maintaining a cover - share the same set of LHS-attributes. Clearly these form a subset of the LHS-attributes for any cover Σ' of Σ , since we can transform Σ' into a reduced cover by removing attributes.

4 The Complex-Valued Data Model

A number of complex-valued data models have been suggested. We will follow the approach of Hartmann, Link and Schewe [5,6,7], since it provides a general and unifying framework for the study of many existing data models. In particular, one can focus on the main data structures under consideration and, thereby, appreciate the direct impact of the type constructors on the results.

Instead of dealing with relation schemas we utilize nested attributes. These can be generated from flat attributes (which are the same as attributes in the relational model) by an arbitrary finite number of recursive applications of record, list, set and multiset constructors.

Definition 1. [7] *A universe is a finite set \mathcal{U} together with domains (i.e., sets of values) $\text{dom}(A)$ for all $A \in \mathcal{U}$. The elements of \mathcal{U} are called flat attributes.*

For the relational data model a universe was sufficient. That is, a relation schema is defined as a finite and non-empty subset $R \subseteq \mathcal{U}$. For data models supporting complex objects, however, nested attributes are needed. In the following definition we use a set \mathcal{L} of labels, and assume that the symbol λ is neither a flat attribute nor a label, i.e., $\lambda \notin \mathcal{U} \cup \mathcal{L}$. Moreover, flat attributes are not labels and vice versa, i.e., $\mathcal{U} \cap \mathcal{L} = \emptyset$.

Definition 2. [7] *Let \mathcal{U} be a universe and \mathcal{L} a set of labels. The set $\mathcal{NA}(\mathcal{U}, \mathcal{L})$ of nested attributes over \mathcal{U} and \mathcal{L} is the smallest set satisfying the following conditions:*

1. $\lambda \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$,
2. $\mathcal{U} \subseteq \mathcal{NA}(\mathcal{U}, \mathcal{L})$,
3. for $L \in \mathcal{L}$ and $N_1, \dots, N_k \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$ with $k \geq 1$ we have $L(N_1, \dots, N_k) \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$,

4. for $L \in \mathcal{L}$ and $N \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$ we have $L[N] \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$,
5. for $L \in \mathcal{L}$ and $N \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$ we have $L\{N\} \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$,
6. for $L \in \mathcal{L}$ and $N \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$ we have $L\langle N \rangle \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$.

We call λ null attribute, $L(N_1, \dots, N_k)$ record-valued attribute, $L[N]$ list-valued attribute, $L\{N\}$ set-valued attribute, and $L\langle N \rangle$ multiset-valued attribute.

From now on we will assume that a universe \mathcal{U} and a set \mathcal{L} of labels are fixed. Instead of writing $\mathcal{NA}(\mathcal{U}, \mathcal{L})$ we simply write \mathcal{NA} .

A relation schema $R = \{A_1, \dots, A_n\}$ can be viewed as the record-valued attribute $R(A_1, \dots, A_n)$ using the name R as a label. The null attribute λ must not be confused with a null value, which is a distinguished element of a certain domain. The null attribute rather indicates that some information of the underlying nested attribute, i.e. some information on the schema level, has been left out. Further explanations follow.

The mapping *dom* can be extended from flat to nested attributes, i.e., we define a set $dom(N)$ of values for every nested attribute $N \in \mathcal{NA}$. We denote empty set, empty multiset, and empty list by $\emptyset, \langle \rangle, []$, respectively.

Definition 3. [7] For a nested attribute $N \in \mathcal{NA}$ we define the domain $dom(N)$ as follows:

1. $dom(\lambda) = \{ok\}$,
2. $dom(A)$ as above (Definition 1) for all $A \in \mathcal{U}$,
3. $dom(L(N_1, \dots, N_k)) = \{(v_1, \dots, v_k) \mid v_i \in dom(N_i) \text{ for } i = 1, \dots, k\}$, i.e., the set of all k -tuples (v_1, \dots, v_k) with $v_i \in dom(N_i)$ for all $i = 1, \dots, k$,
4. $dom(L[N]) = \{[v_1, \dots, v_n] \mid v_i \in dom(N) \text{ for } i = 1, \dots, n\} \cup \{[]\}$, i.e., $dom(L[N])$ is the set of all finite lists with elements in $dom(N)$,
5. $dom(L\{N\}) = \{\{v_1, \dots, v_n\} \mid v_i \in dom(N) \text{ for } i = 1, \dots, n\} \cup \{\emptyset\}$, i.e., $dom(L\{N\})$ is the set of all finite subsets of $dom(N)$,
6. $dom(L\langle N \rangle) = \{\langle v_1, \dots, v_n \rangle \mid v_i \in dom(N) \text{ for } i = 1, \dots, n\} \cup \{\langle \rangle\}$, i.e., $dom(L\langle N \rangle)$ is the set of all finite multisets with elements in $dom(N)$.

The domain of the record-valued attribute $R(A_1, \dots, A_n)$ is a set of n -tuples, i.e., an n -ary relation. The value *ok* can be interpreted as the null value “some information exists, but is currently omitted”.

The replacement of flat attribute names by the null attribute λ within a nested attribute decreases the amount of information that is modelled by the corresponding attributes. This fact allows us to introduce an order between nested attributes.

Definition 4. [7] The subattribute relation \leq on the set of nested attributes \mathcal{NA} over \mathcal{U} and \mathcal{L} is defined by the following rules, and the following rules only:

1. $N \leq N$ for all nested attributes $N \in \mathcal{NA}$,
2. $\lambda \leq A$ for all flat attributes $A \in \mathcal{U}$,
3. $\lambda \leq N$ for all set-valued, multiset-valued and list-valued attributes $N \in \mathcal{NA}$,
4. $L(N_1, \dots, N_k) \leq L(M_1, \dots, M_k)$ whenever $N_i \leq M_i$ for all $i = 1, \dots, k$,

5. $L[N] \leq L[M]$ whenever $N \leq M$,
6. $L\{N\} \leq L\{M\}$ whenever $N \leq M$,
7. $L\langle N \rangle \leq L\langle M \rangle$ whenever $N \leq M$.

For $N, M \in \mathcal{NA}$ we say that M is a subattribute of N if and only if $M \leq N$ holds. We write $M \not\leq N$ if and only if M is not a subattribute of N .

The finite set of all subattributes of N is denoted by $Sub(N)$. It has a smallest element with respect to the subattribute relation, called the bottom element.

Lemma 3. [5, Lemma 38] *The bottom element λ_N of $Sub(N)$ is given by $\lambda_N = L(\lambda_{N_1}, \dots, \lambda_{N_k})$ whenever $N = L(N_1, \dots, N_k)$, and $\lambda_N = \lambda$ whenever N is not a record-valued attribute.*

The binary operators *join* \sqcup and *meet* \sqcap are the equivalent to union \cup and intersection \cap in the relational case. For $X, Y \leq N$ we get:

- $X \sqcup Y$ is the minimal subattribute of N with $X, Y \leq X \sqcup Y$
- $X \sqcap Y$ is the maximal subattribute of N with $X, Y \geq X \sqcap Y$

It has been shown in [7] that join and meet always exist.

Given the relation schema $R = \{A, B, C\}$, the attribute set $\{A, C\}$ can be viewed as the subattribute $R(A, \lambda, C)$ of the record-valued attribute $R(A, B, C)$. The occurrence of the null attribute λ in $R(A, \lambda, C)$ indicates that the information about the attribute B has been masked out. The inclusion order \subseteq on attribute sets in the relational data model is now generalized to the subattribute relation \leq in complex-valued data models.

Lemma 4. *The subattribute relation is a partial order on nested attributes.* \square

Informally, $M \leq N$ for $N, M \in \mathcal{NA}$ if and only if M comprises at most as much information as N does. The informal description of the subattribute relation is formally documented by the existence of a projection function $\pi_M^N : dom(N) \rightarrow dom(M)$ in case $M \leq N$ holds.

Definition 5. [7] *Let $N, M \in \mathcal{NA}$ with $M \leq N$. The projection function $\pi_M^N : dom(N) \rightarrow dom(M)$ is defined as follows:*

1. if $N = M$, then $\pi_M^N = id_{dom(N)}$ is the identity on $dom(N)$,
2. if $M = \lambda$, then $\pi_\lambda^N : dom(N) \rightarrow \{ok\}$ is the constant function that maps every $v \in dom(N)$ to ok ,
3. if $N = L(N_1, \dots, N_k)$ and $M = L(M_1, \dots, M_k)$, then $\pi_M^N = \pi_{M_1}^{N_1} \times \dots \times \pi_{M_k}^{N_k}$ which maps every tuple $(v_1, \dots, v_k) \in dom(N)$ to $(\pi_{M_1}^{N_1}(v_1), \dots, \pi_{M_k}^{N_k}(v_k)) \in dom(M)$,
4. if $N = L[N']$ and $M = L[M']$, then $\pi_M^N : dom(N) \rightarrow dom(M)$ maps every list $[v_1, \dots, v_n] \in dom(N)$ to the list $[\pi_{M'}^{N'}(v_1), \dots, \pi_{M'}^{N'}(v_n)] \in dom(M)$,
5. if $N = L\{N'\}$ and $M = L\{M'\}$, then $\pi_M^N : dom(N) \rightarrow dom(M)$ maps every set $S \in dom(N)$ to the set $\{\pi_{M'}^{N'}(s) : s \in S\} \in dom(M)$, and

6. if $N = L\langle N' \rangle$ and $M = L\langle M' \rangle$, then $\pi_M^N : \text{dom}(N) \rightarrow \text{dom}(M)$ maps every multiset $S \in \text{dom}(N)$ to the multiset $\langle \pi_{M'}^{N'}(s) : s \in S \rangle \in \text{dom}(M)$.

It follows, in particular, that $\emptyset, \langle \rangle, []$ are always mapped to themselves, except when projected on the null attribute λ in which each of them is mapped to *ok*. Note that for $Y \leq X$ we have $\pi_Y^N = \pi_Y^X \circ \pi_X^N$ where \circ denotes the composition of functions.

We are now ready to repeat the syntax and semantics of functional dependencies in complex-valued databases.

Definition 6. [7] Let $N \in \mathcal{NA}$ be a nested attribute. A functional dependency on N is an expression of the form $\mathcal{X} \rightarrow \mathcal{Y}$ where $\mathcal{X}, \mathcal{Y} \subseteq \text{Sub}(N)$ are non-empty. A set $r \subseteq \text{dom}(N)$ satisfies the functional dependency $\mathcal{X} \rightarrow \mathcal{Y}$ on N , denoted by $\models_r \mathcal{X} \rightarrow \mathcal{Y}$, if and only if $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ holds for all $Y \in \mathcal{Y}$ whenever $\pi_X^N(t_1) = \pi_X^N(t_2)$ holds for all $X \in \mathcal{X}$ and any $t_1, t_2 \in r$.

The requirement that \mathcal{X}, \mathcal{Y} are sets of subattributes cannot be weakened without losing expressiveness. In fact, if a set of subattributes in an FD is replaced by its join, then this may result in an FD with a different semantics. We illustrate this fact by the following example.

Example 3. [14] Suppose we store sets of tennis matches using the nested attribute

$$\text{Tennis}\{\text{Match}(\text{Winner}, \text{Loser})\}.$$

Consider the following instance r over $\text{Tennis}\{\text{Match}(\text{Winner}, \text{Loser})\}$:

$$\{ \{(\text{Becker}, \text{Agassi}), (\text{Stich}, \text{McEnroe})\}, \\ \{(\text{Becker}, \text{McEnroe}), (\text{Stich}, \text{Agassi})\} \}.$$

The second element of this set results from the first by simply switching opponents. We can see that $\models_r \text{Tennis}\{\text{Match}(\text{Winner}, \lambda)\} \rightarrow \text{Tennis}\{\text{Match}(\lambda, \text{Loser})\}$ holds. In fact, the set of winners $\{\text{Becker}, \text{Stich}\}$ is the same for both elements and so is the set of losers $\{\text{Agassi}, \text{McEnroe}\}$.

However, $\not\models_r \text{Tennis}\{\text{Match}(\text{Winner}, \lambda)\} \rightarrow \text{Tennis}\{\text{Match}(\text{Winner}, \text{Loser})\}$ since the matches stored in both elements are different from one another. The instance r is therefore a prime example for the failure of the extension rule

$$\frac{X \rightarrow Y}{X \rightarrow X \sqcup Y}$$

in the presence of sets. The same is true for multisets as a set is just a multiset in which every element occurs exactly once.

Sufficient and necessary conditions when projections on subattributes X and Y do determine the projection on $X \sqcup Y$ have been identified.

Definition 7. [7] Let $N \in \mathcal{NA}$. The subattributes $X, Y \in \text{Sub}(N)$ are reconcilable if and only if one of the following conditions is satisfied

- $Y \leq X$ or $X \leq Y$,
- $N = L(N_1, \dots, N_k), X = L(X_1, \dots, X_k), Y = L(Y_1, \dots, Y_k)$ where X_i and Y_i are reconcilable for all $i = 1, \dots, k$,
- $N = L[N'], X = L[X'], Y = L[Y']$ where X' and Y' are reconcilable.

In Example 3 the subattributes $\text{Tennis}\{\text{Match}(\text{Winner}, \lambda)\}$, $\text{Tennis}\{\text{Match}(\lambda, \text{Loser})\}$ are not reconcilable. However, if we use $\text{Tennis}[\text{Match}(\text{Winner}, \text{Loser})]$ to store (sets of) lists of tennis matches, rather than (sets of) sets, then the subattributes $\text{Tennis}[\text{Match}(\text{Winner}, \lambda)]$ and $\text{Tennis}[\text{Match}(\lambda, \text{Loser})]$ are reconcilable.

In fact, projections of complex data tuples to reconcilable subattributes X, Y uniquely determine the projection on the join $X \sqcup Y$.

Lemma 5. [7, Lemmas 16 and 21] *Let $N \in \mathcal{NA}$, $X, Y \in \text{Sub}(N)$. Then the following are equivalent:*

- (i) X and Y are reconcilable
- (ii) for all $t_1, t_2 \in \text{dom}(N)$ with $\pi_X^N(t_1) = \pi_X^N(t_2)$ and $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ we have $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$

Lemma 5 can be illustrated using Example 3: $\text{Tennis}\{\text{Match}(\text{Winner}, \lambda)\}$ and $\text{Tennis}\{\text{Match}(\lambda, \text{Loser})\}$ are not reconcilable, and the two different elements in

$$\{ \{(\text{Becker}, \text{Agassi}), (\text{Stich}, \text{McEnroe})\}, \{(\text{Becker}, \text{McEnroe}), (\text{Stich}, \text{Agassi})\} \}.$$

are identical on $\text{Tennis}[\text{Match}(\text{Winner}, \lambda)]$ and $\text{Tennis}[\text{Match}(\lambda, \text{Loser})]$.

In order to simplify the implication problem for FDs, attributes are split into maximal reconcilable subattributes.

Definition 8. [5] *Let $N \in \mathcal{NA}$. A nested attribute $N_i \in \mathcal{NA}$ is a unit of N if and only if*

1. $N_i \leq N$,
2. $\forall X, Y \leq N_i$, if X and Y are reconcilable, then $X \leq Y$ or $Y \leq X$,
3. N_i is \leq -maximal with properties 1. and 2.

The set of all units of N is denoted by $\mathcal{U}(N)$.

We will be interested in all subattributes with properties 1. and 2. of Definition 8, not just maximal ones.

Definition 9. *We call a nested attribute N unitary if it is a unit of itself. We write N^\downarrow for the set of all unitary subattributes of N .*

Note that N^\downarrow can be shown to be exactly the extended subattribute basis $\mathcal{E}(N)$ of N as defined in [5], though we will not need that here.

Example 4. In Example 3 where $N = \text{Tennis}\{\text{Match}(\text{Winner}, \text{Loser})\}$, the subattribute $\text{Tennis}\{\text{Match}(\text{Winner}, \lambda)\}$ meets conditions 1. and 2. of Definition 8, but not 3. Thus it is unitary but not a unit of N . The only unit of N is N itself.

For the nested attribute $N' = \text{Tennis}[\text{Match}(\text{Winner}, \text{Loser})]$ we get the units $\text{Tennis}[\text{Match}(\text{Winner}, \lambda)]$ and $\text{Tennis}[\text{Match}(\lambda, \text{Loser})]$. Further unitary subattributes of N' are $\text{Tennis}[\text{Match}(\lambda, \lambda)]$ and λ .

When representing FDs, we will want to replace non-unitary attributes in FDs with their units. As a result, we will only need to deal with sets of unitary attributes, which makes it easier to formulate and prove some lemmas.

In [5] a characterization for the units of a nested attribute is given.

Lemma 6. [5, Lemma 26] *Let $N \in \mathcal{NA}$. Then*

$$\mathcal{U}(N) = \bigcup_{i=1}^k \{L(\lambda_{N_1}, \dots, M, \dots, \lambda_{N_k}) : M \in \mathcal{U}(N_i) \text{ and } N_i \neq \lambda_{N_i}\}$$

if $N = L(N_1, \dots, N_k)$ and $N \neq \lambda_N$,

$$\mathcal{U}(N) = \{L[M'] : M' \in \mathcal{U}(M)\}$$

if $N = L[M]$ holds and $\mathcal{U}(N) = \{N\}$ in any other case.

Furthermore, it is clear from the results in [5], that replacing attributes with their units does not change the semantics of FDs. This means that the FDs $\mathcal{X} \rightarrow \mathcal{Y}$ and $\mathcal{X}' \rightarrow \mathcal{Y}'$ imply each other, where

$$\mathcal{X}' := \bigcup_{X \in \mathcal{X}} \mathcal{U}(X), \quad \mathcal{Y}' := \bigcup_{Y \in \mathcal{Y}} \mathcal{U}(Y)$$

For ease of reading, we will not always distinguish between an attribute and the singleton set containing that attribute, and leave out brackets, commas, labels and ‘ λ ’s where this does not cause ambiguities. Thus we would write e.g. the nested attribute $\text{Tennis}\{\text{Match}(\text{Winner}, \text{Loser})\}$ short as $\{\text{Winner}, \text{Loser}\}$, and the subattribute $\text{Tennis}\{\text{Match}(\text{Winner}, \lambda)\}$ as $\{\text{Winner}\}$.

For a more thorough discussion of the complex-valued model introduced here see e.g. [5,7].

5 Lossless Decomposition

When decomposing a nested attribute we need to decide what we decompose it into. Here we will be more general than in [6] where decompositions contain only subattributes.

Definition 10. *We call a set S of unitary subattributes of N an extended subattribute of N if no attribute in S is a subattribute of any other attribute in S (i.e., S is a \leq -antichain of unitary subattributes). We say that an extended subattribute S' of N is an extended subattribute of S if every attribute in S' is a subattribute of some attribute in S .*

From now on, when talking about decompositions of nested attributes, we will mean decompositions into extended subattributes. Note that, unlike substituting attributes by their units, using extended subattributes rather than just subattributes for decomposition changes the semantics. An extended subattribute corresponds to a set of (not necessarily unitary) subattributes rather than just a single subattribute. This is different to the relational case where all “subattributes” (subsets of attributes) are reconcilable.

Example 5. Let $N = \{AB\}C$ with FDs $\Sigma = \{\{A\}\{B\} \rightarrow C\}$. Then we can decompose N into $N_1 = \{AB\}$ and $N_2 = \{A\}\{B\}C$. This decomposition is dependency preserving and lossless (by Lemma 7). If we were to restrict ourselves to decompositions into subattributes, we could not decompose N any further without loss of information and dependencies.

The requirement that subattributes in S be unitary identifies semantically equivalent sets, such as the following sets of subattributes of $N = \{AB\}CD$:

$$\{\{A\}, \{B\}, C\}, \{\{A\}C, \{B\}\}, \{\{A\}, \{B\}C\} \text{ and } \{\{A\}C, \{B\}C\}$$

The first set $\{\{A\}, \{B\}, C\}$ is an extended subattribute of N , but the others are not. E.g. the second set contains the subattribute $\{A\}C$ which is not unitary as its units are $\{A\}$ and C .

Prohibiting subattributes of another attribute in a tuple prevents obvious redundancy.

Definition 11. For two extended subattributes $\mathcal{X}, \mathcal{Y} \subseteq N^\perp$ we call \mathcal{X} an extended subattribute of \mathcal{Y} , written $\mathcal{X} \leq_{ext} \mathcal{Y}$, if for every $X \in \mathcal{X}$ there exists a $Y \in \mathcal{Y}$ with $X \leq Y$. Join \sqcup and meet \sqcap between extended subattributes are then defined w.r.t. \leq_{ext} . In particular we get

$$\mathcal{X} \sqcup \mathcal{Y} = \max(\mathcal{X} \cup \mathcal{Y})$$

i.e., we take the union but then remove (redundant) non-maximal elements.

Definition 12. The domain of an extended subattribute $S \subseteq N^\perp$ consists of all functions which map each subattribute $s \in S$ to an element in $\text{dom}(s)$, in a consistent manner:

$$\text{dom}(S) := \left\{ f : S \rightarrow \bigcup_{s \in S} \text{dom}(s) \mid \begin{array}{l} f(s) \in \text{dom}(s) \wedge \forall X, Y \in S. \\ \pi_{X \sqcap Y}^X(f(X)) = \pi_{X \sqcap Y}^Y(f(Y)) \end{array} \right\}$$

We can now define projection, join and losslessness as one would expect.

Definition 13. Let N be a nested attribute, $S \subseteq N^\perp$ an extended subattribute of N . Then the projection $\pi_S^N : \text{dom}(N) \rightarrow \text{dom}(S)$ is defined as follows:

$$\pi_S^N(t) := \bigcup_{s \in S} \{s \mapsto \pi_s^N(t)\}$$

Definition 14. *The join of two relations r_1, r_2 on the extended subattributes $S_1, S_2 \subseteq N^\downarrow$ is a relation on $S_1 \sqcup S_2$:*

$$r_1 \bowtie r_2 := \{t \in \text{dom}(S_1 \sqcup S_2) \mid \exists t_i \in r_i. \pi_{S_i}^{S_1 \sqcup S_2}(t) = t_i, i = 1, 2\}$$

As in the relational case, we call a decomposition $\mathcal{D} = \{N_1, \dots, N_n\}$ of (N, Σ) into extended subattributes N_i lossless, if for every relation r on N for which Σ holds we have¹

$$r[N_1] \bowtie \dots \bowtie r[N_n] = r$$

We are now ready to investigate how Theorem 1 behaves in the complex-valued model. One direction (the “if” part) is straight forward.

Lemma 7. *Every dependency preserving decomposition of N containing an extended subattribute which forms a key of N is lossless.*

Proof. As in the relational case [3].

Extending Lemma 2 to the complex-valued case turns out to be more challenging. We will do so next.

Definition 15. *Let N be a nested attribute and S a unitary subattribute of N . We call S restricted if $\text{dom}(S)$ is finite. A FD $X \rightarrow Y$ is LHS-restricted if some element in X is restricted, and a set Σ of FDs over N is LHS-restricted if any of its FDs are.*

To prove Lemma 2 for the complex-valued data model, we need to be able to construct tuples t_i which are identical to some tuple t on an extended subattribute (corresponding to R_i^* in the relational case), but unique (or at least different when domains are finite) for subattributes “outside” these extended subattributes. For this we use and adapt some lemmas from [7,14].

Note that in [7] a set $\mathcal{X} \subseteq \text{Sub}(N)$ of subattributes is an *ideal* w.r.t. \leq if and only if the following holds for all $Y \in \text{Sub}(N)$:

$$X \in \mathcal{X} \text{ and } Y \leq X \quad \Rightarrow \quad Y \in \mathcal{X}$$

We will adopt this definition of ideal as well.

Lemma 8. [7, Lemma 21] *Let $N \in \mathcal{NA}$, and $\emptyset \neq \mathcal{X} \subseteq \text{Sub}(N)$ an ideal with respect to \leq with the property that for reconcilable $X, Y \in \mathcal{X}$ also $X \sqcup Y \in \mathcal{X}$ holds. Then there are $t_N, t'_N \in \text{dom}(N)$ with $\pi_W^N(t_N) = \pi_W^N(t'_N)$ if and only if $W \in \mathcal{X}$.*

Note that in the last lemma we could also consider only unitary subattributes of N , and thus no longer need to worry about reconcilable subattributes. We will take that view when formulating our main lemma in the following.

Intuitively, Lemma 9 ensures the existence of tuples similar to those used in the proof of Lemmas 1 and 2 for constructing a counter example in the relational case. Recall that N^\downarrow denotes the set of all unitary subattributes of N , and that we allow the domains of flat attributes to be finite.

¹ We identify N with the extended subattribute $\mathcal{U}(N)$.

Lemma 9. *Let $N \in \mathcal{N}A$, and $\emptyset \neq \mathcal{X}_i \subseteq N^\downarrow$ for $i = 1, \dots, n$ be ideals with respect to \leq . Then there are $t, t_1, \dots, t_n \in \text{dom}(N)$ with the following properties (for all $W \in N^\downarrow$):*

- (i) $\pi_W^N(t_i) = \pi_W^N(t)$ if $W \in \mathcal{X}_i$
- (ii) $\pi_W^N(t_i) \neq \pi_W^N(t)$ if $W \notin \mathcal{X}_i$ and W unrestricted or a unit of N
- (iii) $\pi_W^N(t_i) \neq \pi_W^N(t_j)$ if $W \notin \mathcal{X}_i \cap \mathcal{X}_j$ and W unrestricted

Proof. We distinguish several cases, depending on the form of N . The numbering of these cases follows the one given in Definition 2. The case (1) where $N = \lambda$ is trivial.

(2) For $N = A$ chose $t = a$ and

$$t_i = \begin{cases} a & \text{if } \mathcal{X}_i = \{\lambda, A\} \\ a_i \neq a & \text{if } \mathcal{X}_i = \{\lambda\} \end{cases}$$

and the a_i pairwise different if $\text{dom}(A)$ is infinite. This obviously meets the conditions (i)-(iii).

(3) For $N = (M_1, \dots, M_k)$ we construct t, t_1, \dots, t_n inductively on the structure of N . Let $t^{M_j}, t_i^{M_j}$ denote the tuples constructed on the nested attribute M_j w.r.t. $\pi_{M_j}^N(\mathcal{X}_i)^2$. We chose $t = (t^{M_1}, \dots, t^{M_k})$ and $t_i = (t_i^{M_1}, \dots, t_i^{M_k})$. It is easy to see that the tuples t, t_i meet conditions (i)-(iii) if the tuples $t^{M_j}, t_i^{M_j}$ do.

(4) For $N = [M]$, we have $\mathcal{X}_i = \{[X] \mid X \in \mathcal{Y}_i\} \cup \{\lambda\}$ for some ideal $\mathcal{Y}_i \subseteq M^\downarrow$. If $\mathcal{Y}_i \neq \emptyset$ then by Lemma 8 there exist tuples $t_{M,i}, t'_{M,i} \in \text{dom}(M)$ which are identical exactly on \mathcal{Y}_i . Otherwise let $t_{M,i}$ be arbitrary. We then construct t, t_1, \dots, t_n as follows:

$$t = [t_{M,1}, \dots, t_{M,n}]$$

$$t_i = \begin{cases} [t_{M,1}, \dots, t'_{M,i}, \dots, t_{M,n}] & \text{if } \mathcal{Y}_i \neq \emptyset \\ \text{arbitrary of length } n + i & \text{otherwise, i.e., if } \mathcal{X}_i = \{\lambda\} \end{cases}$$

With this definition, tuples t_i with $\mathcal{X}_i = \{\lambda\}$ are of unique length, and thus differ from tuples t, t_j with $j \neq i$ on all subattributes except λ . For other tuples t_i, t_j we have

$$\pi_W^N(t_i) = \pi_W^N(t) \quad \text{if and only if} \quad \pi_W^N(t'_{M,i}) = \pi_W^N(t_{M,i})$$

which shows conditions (i) and (ii), as well as

$$\pi_W^N(t_i) = \pi_W^N(t_j) \quad \text{if and only if} \quad \pi_W^N(t'_{M,i}) = \pi_W^N(t_{M,i}) \wedge \pi_W^N(t'_{M,j}) = \pi_W^N(t_{M,j})$$

from which (iii) follows.

² Strictly speaking we would have to distinguish between M_j and the subattribute $(\lambda, \dots, M_j, \dots, \lambda) \leq N$. We will neglect this subtlety for ease of readability.

(6) For $N = \langle M \rangle$ let $t_{N,i}, t'_{N,i} \in \text{dom}(N)$ be tuple pairs with $\pi_W^N(t_{N,i}) = \pi_W^N(t'_{N,i})$ if and only if $W \in \mathcal{X}_i$, which exist by Lemma 8. We then define

$$t = c \cdot t_{N,1} \cup \dots \cup c^n \cdot t_{N,n}$$

$$t_i = c \cdot t_{N,1} \cup \dots \cup c^i \cdot t'_{N,i} \dots \cup c^n \cdot t_{N,n}$$

with $c \in \mathbf{N}$ sufficiently large, i.e., larger than the multiplicity of any element in the multisets $t_{N,1}, \dots, t_{N,n}, t'_{N,1}, \dots, t'_{N,n}$. Here $c \cdot t_{N,1}$ denotes the multiset obtained by multiplying the multiplicity of all elements of $t_{N,1}$ with c . This allows us to determine the “origin” of an element e of t or t_i from its multiplicity, i.e. the number of times the element e occurs in the multiset t or t_i . We shall denote this number by $\text{mult}_e(t)$ or $\text{mult}_e(t_i)$, respectively. For every tuple t, t_1, \dots, t_n we have

$$\begin{aligned} \text{mult}_e(t/t_i) &= \text{mult}_e(c \cdot M_1 \cup \dots \cup c^n \cdot M_n) \\ &= c \cdot \text{mult}_e(M_1) + \dots + c^n \cdot \text{mult}_e(M_n) \\ &= c \cdot a_1 + \dots + c^n \cdot a_n \end{aligned} \tag{1}$$

for some multisets M_1, \dots, M_n and some values $a_1, \dots, a_n \in \{0, \dots, c - 1\}$. Given a multiplicity $\text{mult}_e(t/t_i)$ there exists only one set of values $a_1, \dots, a_n \in \{0, \dots, c - 1\}$ such that (1) holds. Consequently, we can uniquely determine the multisets M_1, \dots, M_n given t or some t_i . Conditions (i)-(iii) can now be shown as in the case of lists.

(5) What remains is the case of sets. For $N = \{M\}$ with N restricted, we use that by Lemma 8 there exist two tuples $t_N \neq t'_N$ which are identical on $N^\downarrow \setminus N$. We then set $t = t_N$ and

$$t_i = \begin{cases} t_N & \text{if } \mathcal{X}_i = N^\downarrow \\ t'_N & \text{otherwise} \end{cases}$$

which clearly meet condition (i)-(iii).

For $N = \{M\}$ with N unrestricted, we adapt the proof of Lemma 25 in [7], which shows Lemma 8 for the case of sets. For every index $i = 1, \dots, n$ we define different identifying terms:

- $\tau_\lambda^i(\lambda) = ok$,
- $\tau_A^i(\lambda) = a, \tau_A^i(A) = a_i$ with $a, a_i \in \text{dom}(A), a \neq a_i$ and a_1, \dots, a_n pairwise different if $\text{dom}(A)$ is infinite
- $\tau_{L(N_1, \dots, N_n)}^i(L(M_1, \dots, M_n)) = (\tau_{N_1}^i(M_1), \dots, \tau_{N_n}^i(M_n))$,
- $\tau_{L\{N\}}^i(L\{M\}) = \{\tau_N^i(M)\}$ and $\tau_{L\{N\}}^i(\lambda) = \emptyset$,
- $\tau_{L(N)}^i(L\langle M \rangle) = \langle \tau_N^i(M), \dots, \tau_N^i(M) \rangle$ of cardinality i and $\tau_{L(N)}^i(\lambda) = \langle \rangle$,
- $\tau_{L[N]}^i(L[M]) = [\tau_N^i(M), \dots, \tau_N^i(M)]$ of length i and $\tau_{L[N]}^i(\lambda) = []$.

Note that with this definition the identifying terms $\tau_N^i(M)$ of a subattribute M with infinite domain are pairwise different, i.e., $\tau_N^i(M) \neq \tau_N^j(M)$ for $i \neq j$.

We then create pairs of tuples $t_{N,i}, t'_{N,i} \in \text{dom}(N)$ with $\pi_W^N(t_{N,i}) = \pi_W^N(t'_{N,i})$ if and only if $W \in \mathcal{X}_i$ as in [7, Lemma 25], but using the corresponding identifying

terms $\tau_N^i(\dots)$. That is, we have $\mathcal{X} = \{L\{X\} : X \in \mathcal{Y}\} \cup \{\lambda\}$ for some $\mathcal{Y} \subseteq \text{Sub}(M)$, and define

$$t_{N,i} = \{\tau_M^i(X) : X \leq M\}$$

$$t'_{N,i} = \{\tau_M^i(X) : X \in \mathcal{Y}\}$$

The tuple pairs $t_{N,i}, t'_{N,i}$ are used to construct the tuples t, t_1, \dots, t_k as follows:

$$t = t_{N,1} \cup \dots \cup t_{N,n}$$

$$t_i = t_{N,1} \cup \dots \cup t_{N,i-1} \cup t'_{N,i} \cup t_{N,i+1} \cup \dots \cup t_{N,n}$$

Condition (i) clearly holds, since equivalence of $t_{N,i}$ and $t'_{N,i}$ on W implies equivalence of t and t_i on W . Now let $W = \{V\}$ meet the 'if' condition of (ii). By construction we have

$$\tau_M^i(V) \in \pi_W^N(t_{N,i}) \setminus \pi_W^N(t'_{N,i})$$

Furthermore, W must be unrestricted, since N is unrestricted and the only unit of N . As the identifying terms of unrestricted subattributes are all different, $\tau_M^i(V)$ does not lie in any set $\pi_W^N(t_{N,j})$ for $j \neq i$ either. Thus $\pi_W^N(t_i) \neq \pi_W^N(t)$, which shows (ii). Condition (iii) is proven analogous to (ii).

The following example illustrates the construction for multisets and sets.

Example 6. [Multiset] Let $N = \langle \{\{A\}\} \rangle$ and

$$\mathcal{X}_1 = \{\lambda, \langle \lambda \rangle, \langle \{\lambda\} \rangle, \langle \{\{\lambda\}\} \rangle\}$$

$$\mathcal{X}_2 = \{\lambda, \langle \lambda \rangle, \langle \{\lambda\} \rangle\}$$

$$\mathcal{X}_3 = \{\lambda, \langle \lambda \rangle\}$$

$$\mathcal{X}_4 = \{\lambda\}$$

Lemma 8 might give us the following tuples:

$$t_{N,1} = \langle \{\{\{a_1\}\}\} \rangle, t'_{N,1} = \langle \{\{\{a_2\}\}\} \rangle$$

$$t_{N,2} = \langle \{\{\{a_1\}\}\} \rangle, t'_{N,2} = \langle \{\{\emptyset\}\} \rangle$$

$$t_{N,3} = \langle \{\{\emptyset\}\} \rangle, t'_{N,3} = \langle \{\emptyset\} \rangle$$

$$t_{N,4} = \langle \rangle, t'_{N,4} = \langle \{\emptyset\} \rangle$$

Since all multisets above contain only a single element (or less), it suffices to choose $c = 2$. Using our construction we obtain

$$t = 2 \cdot t_{N,1} \cup 4 \cdot t_{N,2} \cup 8 \cdot t_{N,3} \cup 16 \cdot t_{N,4}$$

$$= 2 \cdot \langle \{\{\{a_1\}\}\} \rangle \cup 4 \cdot \langle \{\{\{a_1\}\}\} \rangle \cup 8 \cdot \langle \{\{\emptyset\}\} \rangle \cup 16 \cdot \langle \rangle$$

$$= \langle \{\{\{a_1\}\}\}^6, \{\emptyset\}^8 \rangle$$

$$t_1 = \langle \{\{\{a_2\}\}\}^2, \{\{\{a_1\}\}\}^4, \{\emptyset\}^8 \rangle$$

$$t_2 = \langle \{\{\{a_1\}\}\}^2, \{\emptyset\}^{12} \rangle$$

$$t_3 = \langle \{\{\{a_1\}\}\}^6, \emptyset^8 \rangle$$

$$t_4 = \langle \{\{\{a_1\}\}\}^6, \{\emptyset\}^8, \emptyset^{16} \rangle$$

where $\langle E^n \rangle$ means that element E occurs n times in the multiset.

[Set] Now let $N = \{AB\}$ with $\text{dom}(A), \text{dom}(B)$ infinite, and

$$\begin{aligned}\mathcal{X}_1 &= \{\lambda, \{\lambda\}, \{A\}, \{B\}\} \\ \mathcal{X}_2 &= \{\lambda, \{\lambda\}, \{A\}\} \\ \mathcal{X}_3 &= \{\lambda, \{\lambda\}, \{B\}\}\end{aligned}$$

Using our construction for sets we get

$$\begin{aligned}t_{N,1} &= \{(a, b), (a_1, b), (a, b_1), (a_1, b_1)\}, t'_{N,1} = \{(a, b), (a_1, b), (a, b_1)\} \\ t_{N,2} &= \{(a, b), (a_2, b), (a, b_2), (a_2, b_2)\}, t'_{N,2} = \{(a, b), (a_2, b)\} \\ t_{N,3} &= \{(a, b), (a_3, b), (a, b_3), (a_3, b_3)\}, t'_{N,3} = \{(a, b), (a, b_3)\}\end{aligned}$$

and from this

$$\begin{aligned}t &= t_{N,1} \cup t_{N,2} \cup t_{N,3} \\ &= \{(a, b), (a_1, b), (a, b_1), (a_1, b_1), (a_2, b), (a, b_2), (a_2, b_2), (a_3, b), (a, b_3), (a_3, b_3)\} \\ t_1 &= \{(a, b), (a_1, b), (a, b_1), (a_2, b), (a, b_2), (a_2, b_2), (a_3, b), (a, b_3), (a_3, b_3)\} \\ t_2 &= \{(a, b), (a_1, b), (a, b_1), (a_1, b_1), (a_2, b), (a_3, b), (a, b_3), (a_3, b_3)\} \\ t_3 &= \{(a, b), (a_1, b), (a, b_1), (a_1, b_1), (a_2, b), (a, b_2), (a_2, b_2), (a, b_3)\}\end{aligned}$$

In both cases (multiset and set) it is easy to check that the t, t_i constructed meet the conditions (i)-(iii) of Lemma 9.

Comparing the last lemma to Lemma 8, one may wonder why we require in condition (ii) that W is unrestricted or a unit, and whether this requirement can be omitted. The following example shows that it is really needed.

Example 7. Let $N = \{AB\}$ and $\mathcal{X}_1 = \mathcal{X}_2 = \{\lambda\}, \mathcal{X}_3 = \{\lambda, \{\lambda\}, \{B\}\}$. Let further t, t_1, t_2 be tuples on N which meet the conditions of Lemma 9. If $t = \emptyset$ then by (i) it follows that $t_3 = \emptyset$, which violates condition (ii) for $W = \{A\}$. So $t \neq \emptyset$. If $t_1 = t_2 = \emptyset$ then condition (iii) is violated for $W = \{A\}$. So $t_1 \neq \emptyset$ or $t_2 \neq \emptyset$, say $t_1 \neq \emptyset$. But then we have

$$\pi_{\{\lambda\}}^N(t_1) = \{ok\} = \pi_{\{\lambda\}}^N(t)$$

which shows that the restriction on W in condition (ii) is necessary.

We are now ready to prove Lemma 2 in the complex-valued model.

Lemma 10. *Let N be a nested attribute with FDs Σ . If Σ is not LHS-restricted, then every lossless decomposition of N contains an extended subattribute which forms a key of N .*

Proof. We will proceed as in the proof of Lemma 1. Let $\mathcal{D} = \{N_1, \dots, N_n\}$ be a decomposition of N not containing a key schema, i.e., an extended subattribute which forms a key of N . For $N_i \in \mathcal{D}$ define $\mathcal{X}_i := (N_i^*)^\perp$, where N_i^* is the closure of N_i w.r.t. Σ , and \mathcal{X}_i is the ideal generated by it. Then there exist

tuples t, t_1, \dots, t_n on N which meet the conditions (i)-(iii) of Lemma 9. We define $r = \{t_1, \dots, t_n\}$, and start by showing that Σ holds on r .

So let $X \rightarrow Y \in \Sigma$ and $t_i, t_j \in r$ be two tuples with $\pi_X^N(t_i) = \pi_X^N(t_j)$. Then X is unrestricted by assumption, so from condition (iii) it follows that $X \subseteq \mathcal{X}_i \cap \mathcal{X}_j$. Thus, by definition of $\mathcal{X}_i, \mathcal{X}_j$, we have $Y \subseteq \mathcal{X}_i \cap \mathcal{X}_j$, which gives us $\pi_Y^N(t_i) = \pi_Y^N(t_j)$ by condition (i). It follows that $X \rightarrow Y$ holds on r .

Also by condition (i), the tuple t lies in $\bowtie r[\mathcal{D}]$. It remains to be shown that $t \notin r$. By assumption N_i is not a key of N , so \mathcal{X}_i does not contain all units of N . Thus $t_i \neq t$ by condition (ii), which shows that \mathcal{D} is not lossless.

Together with Lemma 7 this gives us an extension of Theorem 1 for complex-valued data bases.

Theorem 2. *Let N be a nested attribute with FDs Σ . If Σ is not LHS-restricted, then a dependency preserving decomposition of N is lossless if and only if it contains an extended subattribute which forms a key of N .*

Proof. Follows from Lemmas 7 and 10.

6 Conclusion

We have shown that in the relational model decompositions can be lossless without containing a key if domains are allowed to be finite. However, we found that this only occurs if attributes with finite domains appear in the left hand side of functional dependencies.

In complex-valued data models containing sets, finite domains occur naturally for subattributes such as $\{\lambda\}$, with $dom(\{\lambda\}) = \{\emptyset, \{ok\}\}$. It can be argued though that subattributes with finite domains rarely occur in the left hand side of functional dependencies. We were able to extend our results from the relational model, showing that when such LHS-restricted functional dependencies do not exist, then every lossless decomposition must contain a key (Lemma 10).

As a result, Theorem 2, which corresponds to Theorem 1 in the relational case, can usually be applied in our complex-valued data model as well. This simplifies the task of finding lossless (and dependency preserving) decompositions, since it is possible to apply methods similar to those used in the relational case, as e.g. in [3,10,18].

We wish to thank the anonymous reviewers of this paper for many helpful comments and suggestions.

References

1. Aho, A.V., Beeri, C., Ullman, J.D.: The theory of joins in relational databases. ACM Trans. Database Syst. 4(3), 297–314 (1979)
2. Arenas, M., Libkin, L.: An information-theoretic approach to normal forms for relational and XML data. In: PoDS, pp. 15–26 (2003)

3. Biskup, J., Dayal, U., Bernstein, P.A.: Synthesizing independent database schemas. In: SIGMOD Conference, pp. 143–151 (1979)
4. Codd, E.: Recent investigations in relational data base systems. In: IFIP Congress, pp. 1017–1021 (1974)
5. Hartmann, S., Link, S.: Deciding implication for functional dependencies in complex-value databases. *Theor. Comput. Sci.* 364(2), 212–240 (2006)
6. Hartmann, S., Link, S.: The nested list normal form for functional and multivalued dependencies. In: Dix, J., Hegner, S.J. (eds.) FoIKS 2006. LNCS, vol. 3861, pp. 137–158. Springer, Heidelberg (2006)
7. Hartmann, S., Link, S., Schewe, K.-D.: Axiomatisations of functional dependencies in the presence of records, lists, sets and multisets. *Theor. Comput. Sci.* 355(2), 167–196 (2006)
8. Hegner, S.J.: Characterization of desirable properties of general database decompositions. *Ann. Math. Artif. Intell.* 7(1–4), 129–195 (1993)
9. Hegner, S.J.: Unique complements and decomposition of database schemata. *J. Comput. Syst. Sci.* 48(1), 9–57 (1994)
10. Koehler, H.: Finding faithful Boyce-Codd normal form decompositions. In: Cheng, S.-W., Poon, C.K. (eds.) AAIM 2006. LNCS, vol. 4041, pp. 102–113. Springer, Heidelberg (2006)
11. Kolahi, S.: Dependency-preserving normalization of relational and XML data. *J. Comput. Syst. Sci.* 73(4), 636–647 (2007)
12. Levene, M., Loizou, G.: Semantics for null extended nested relations. *ACM Trans. Database Syst.* 18(3), 414–459 (1993)
13. Levene, M., Loizou, G.: *A Guided Tour of Relational Databases and Beyond*. Springer, Heidelberg (1999)
14. Link, S.: *Dependencies in Complex-valued Databases*. PhD Thesis (2004)
15. Maier, D.: *The Theory of Relational Databases*. Computer Science Press (1983)
16. Maier, D., Mendelzon, A.O., Sadri, F., Ullman, J.D.: Adequacy of decompositions of relational databases. *J. Comput. Syst. Sci.* 21(3), 368–379 (1980)
17. Mannila, H., Rähkä, K.-J.: *The Design of Relational Databases*. Addison-Wesley, Reading (1987)
18. Osborn, S.L.: Testing for existence of a covering Boyce-Codd normal form. *Information Processing Letters* 8(1), 11–14 (1979)
19. Zaniolo, C.: A new normal form for the design of relational database schemata. *ACM Trans. Database Syst.* 7(3), 489–499 (1982)