On Fast and Space-Efficient Database Normalization

A dissertation presented in partial fulfilment
of the requirements for the degree of

Doctor of Philosophy
in
Information Systems

at Massey University, Palmerston North, New Zealand

Henning Koehler

2007

Abstract

A common approach in designing relational databases is to start with a relation schema, which is then decomposed into multiple subschemas. A good choice of subschemas can often be determined using integrity constraints defined on the schema.

Two central questions arise in this context. The first issue is what decompositions should be called "good", i.e., what normal form should be used. The second issue is how to find a decomposition into the desired form.

These question have been the subject of intensive research since relational databases came to life. A large number of normal forms have been proposed, and methods for their computation given. However, some of the most popular proposals still have problems:

- algorithms for finding decompositions are inefficient
- dependency preserving decompositions do not always exist
- decompositions need not be optimal w.r.t. redundancy/space/update anomalies

We will address these issues in this work by

- designing efficient algorithms for finding dependency preserving decompositions
- proposing a new normal form which minimizes overall storage space

This new normal form is then characterized syntactically, and shown to extend existing normal forms.

Acknowledgement

I would like to thank my supervisor Sven Hartmann for his constant support, ranging from fruitful discussions and extensive proofreading to help with administrative hurdles and moral support.

Furthermore, my thanks go to my co-supervisor Klaus-Dieter Schewe, and to my colleagues Sebastian Link and Markus Kirchberg, who helped and supported me in various forms.

I dedicate this thesis to my parents, Klaus and Waltraud Koehler, and to my partner Jane Zhao.

Contents

1 Introduction **3**
1.1 Relational Databases and Dependencies . . . 3
1.2 Normal Forms . . . 5
1.3 Contributions and Outline . . . 7

2 Linear Resolution and Faithful BCNF Decomposition **10**
2.1 Linear Resolution . . . 11
2.1.1 The Basic Algorithm . . . 13
2.1.2 Improvements and Complexity Analysis . . . 17
2.1.3 Polynomial Cases . . . 20
2.1.4 Updating the Atomic Closure . . . 22
2.1.5 Other Applications . . . 24
2.1.6 Related work . . . 25
2.2 Faithful BCNF Decomposition . . . 26
2.2.1 The Basic Algorithm . . . 26
2.2.2 Improvements and Complexity Analysis . . . 29
2.2.3 Partial Determination Cycles . . . 31
2.2.4 Related Work . . . 36
2.3 Complex-Valued Databases . . . 36
2.3.1 Introduction . . . 36
2.3.2 Representation Basis . . . 40
2.3.3 Linear Resolution . . . 42
2.3.4 Faithful BCNF-Decomposition . . . 46
2.3.5 Lossless Decomposition . . . 48
2.3.6 Testing for BCNF . . . 54

3 Canonical Covers **55**
3.1 Hypergraph Decomposition . . . 57
3.1.1 Autonomous Sets . . . 57
3.1.2 Superedges and Partial Superedges . . . 62
3.1.3 Computing Autonomous Sets . . . 63
3.2 Computing all Canonical Covers . . . 65
3.2.1 Partial Covers . . . 65
3.2.2 Relative Covers . . . 67
3.2.3 Implication Dependencies . . . 69
3.2.4 The Algorithm . . . 72
3.2.5 Improvements and Complexity Analysis . . . 74

3.2.6 LR-reduced Covers 75
3.2.7 Related Work 75
3.3 Size and Number of Non-redundant Covers 76
3.4 Partial Implication Cycles 78
3.5 Essential FDs 82
3.5.1 Deriving essential FDs 83
3.5.2 Testing essentiality 86

4 Domination Normal Form **91**
4.1 Minimization as Normal Form 92
4.1.1 Ordering by Size of Instances 92
4.1.2 Ordering by Attribute Count of Instances 95
4.1.3 Ordering by Containing Schema Closures 96
4.2 Equivalence of Orderings 97
4.2.1 Size vs. Attribute Count 98
4.2.2 Attribute Count vs. Containing Schema Closures - Part I 99
4.2.3 Subset Construction 99
4.2.4 Attribute Count vs. Containing Schema Closures - Part II 107
4.3 Relationship to other Normal Forms 110
4.3.1 A Detailed Example 111
4.4 Computing Domination Normal Form 114
4.4.1 Dependency Preserving DNF 116
4.5 Combining Normal Forms 119
4.5.1 DNF and BCNF 119
4.5.2 DNF and EKNF 127

5 Summary **129**
5.1 Main Results 129
5.2 Open Problems 130

Chapter 1

Introduction

In this chapter we will give a brief introduction to relational database theory, outlining challenges and how we will deal with them in this work.

1.1 Relational Databases and Dependencies

A number of different database models have been suggested, such as relational, complex-valued or object-oriented databases. In particular, XML-Databases (which are a certain type of complex-valued databases) have received much attention in recent years. While the relational model is one of the oldest, relational databases are still the most common ones.

The main advantage of the relational model over more powerful models is its simplicity. By approaching a problem, which is relevant in different data models, in the relational model first, we can concentrate on the core of the problem. Once solved for the relational case, the results can often be extended to more complex database models, with the relational model serving as common denominator. This approach seems more suitable than tackling issues in a complex model directly. We therefore will mainly focus on the relational model in this work.

We begin by introducing some basic terms. A *relational database schema* consists of a set of relation schemas. A *relation schema* $R = \{A_1, A_2, \ldots, A_n\}$ 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 or infinite set of tuples, and each element $e_i \in dom(A_i)$ of the tuple corresponds to one attribute $A_i \in R$. 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.

A vital tool for managing data are *integrity constraints*. They describe, usually in a syntactic manner, what instances of a database schema are acceptable or valid. Formally, an integrity constraint on R is a function mapping relations r on R to $\{True, False\}$. We say that a constraint holds on r if it maps r to $True$.

Constraints are usually used to avoid storing instances with inconsistent data. By only accepting updates which lead to valid instances, inconsistencies can be detected immediately. Many different types of integrity constraints have been suggested in the literature. In this work we will focus on the most common type, that is on functional dependencies, and in some occasions also consider multivalued and join dependencies.

With each relation schema we associate a set Σ of integrity constraints, in particular *functional dependencies* (FD), *multivalued dependencies* (MVD) and *join dependencies* (JD). These restrict which relations over R we may store. We say that a set Σ of constraints over R *implies* a constraint (or set of constraints) c, written $\Sigma \models c$, if c holds on every relation r over R for which all constraints in Σ hold. If two sets of constraints Σ and Σ' imply each other, we call Σ a *cover* of Σ' (and vice versa). We say that a FD $X \to Y \in \Sigma$ is *redundant* in Σ, if $\Sigma \setminus \{X \to Y\}$ implies $X \to Y$ (and thus is still a cover of Σ).

At this point we need to consider two options: If we allow relations to be infinite, we get a different notion of implication than we get when considering only finite relations. In the latter case, implication is commonly referred to as *finite implication*, and these notions of implications can be different [11]. In this work we shall consider only finite relations. However, as implication and finite implication are actually the same for functional and join dependencies [33], all our results from chapters 2 and 3 will hold for infinite relations as well. In chapter 4 we will compare relations w.r.t. their size, which is best done for finite relations only. Alternatively, we could allow infinite relations in principal, but only consider finite ones in our comparison, which would again lead to the same results.

A functional dependency on R is an expression of the form $X \to Y$ (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 a FD $X \to Y$ *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. We call a FD $X \to Y$ *trivial* if $Y \subseteq X$. Trivial FDs are the only FDs which hold on every relation. A set $X \subseteq R$ is a *key* of R w.r.t. a set Σ of integrity constraints on R, if Σ implies $X \to R$. Note that some authors use the term 'key' only for minimal keys, and call keys which may not be minimal 'superkeys'.

Implication of FDs can be characterized syntactically by the following derivation rules, known as the Armstrong Axioms:

$$\frac{}{X \to Y}\,Y \subseteq X, \qquad \frac{X \to Y}{XW \to YW}, \qquad \frac{X \to Y \quad Y \to Z}{X \to Z} \tag{1.1}$$

Here the FDs on the top imply the FD at the bottom, and $Y \subseteq X$ is a side condition which needs to hold for the first rule to be applicable. The Armstrong Axioms are known to be *sound*, meaning that only implied FDs can be derived (which is easy to see), and also *complete*, i.e., every implied FD can be derived [2]. We write Σ^* for the set of all FDs on R implied by Σ.

Note that it is often necessary to apply these derivation rules multiple times, as implied FDs usually cannot be derived in one step. This leads to a sequence of derivation steps, which can be written as *derivation trees*.

Example 1.1. Consider the set of FDs

$$\Sigma = \{A \to B, AB \to C, BC \to D\}$$

The FD $A \to D$ is implied by Σ, and can be derived using the Armstrong Axioms as follows:

$$\dfrac{\dfrac{A \to B}{A \to AB} \qquad \dfrac{\dfrac{AB \to C}{AB \to ABC} \qquad \dfrac{\dfrac{}{ABC \to BC} \quad BC \to D}{ABC \to D}}{AB \to D}}{A \to D}$$

While such derivation trees could in principal be used to show that a FD $X \to Y$ is implied by a set Σ of other FDs, this is not very efficient. Instead we compute the *closure* X^* of the left hand side X, which is the set of all attributes determined by X:

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

In the rare case where the set Σ is not clear from the context, we write $X^{*\Sigma}$.

Once computed, we only need to check whether the right hand side Y is a subset of X^*. Computing X^* can be done quickly using the well-known closure algorithm, which can be implemented to run in linear time, as shown by Beeri and Bernstein in [5].

Algorithm "closure"

```
INPUT: set of FDs $\Sigma$, attribute set $X$
OUTPUT: $X^*$, the closure of $X$ w.r.t. $\Sigma$


$X^* := X$
while $\exists X' \to Y \in \Sigma$ with $X' \subseteq X^*, Y \not\subseteq X^*$ do
    $X^* := X^*Y$
end
```

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 $X \cup Y$:

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

A join dependency on R is an expression of the form $\bowtie [R_1, \ldots, R_n]$ where the R_i are subsets of R with $\bigcup R_i = R$. We say that the JD $\bowtie [R_1, \ldots, R_n]$ holds on r if the decomposition $\{R_1, \ldots, R_n\}$ is *lossless* for r, i.e., if

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

A multivalued dependency on R is a join dependency $\bowtie [R_1, R_2]$ with only two subschemas. It is usually written as $X \twoheadrightarrow Y$ where $X = R_1 \cap R_2$ and $Y = R_1 \setminus R_2$ or $Y = R_2 \setminus R_1$. Note that while these two forms of writing are technically equivalent for a fixed relation R, their "natural interpretations" (i.e., how a designer would understand them) can be quite different. This has lead researchers to consider different notions of implication for MVDs in undetermined universes [8, 30]. These will not be relevant for our work though, and we stick with the standard notion of implication.

For a more thorough introduction to relational database theory see e.g. [27, 33, 36].

1.2 Normal Forms

A common approach in designing relational databases is to start with a universal relation schema, which is then decomposed into multiple subschemas. A good choice of subschemas can often be determined using integrity constraints defined on the original schema.

To ensure that the schemas of a decomposition $\mathcal{D} = \{R_1, \ldots, R_n\}$ with $R_i \subseteq R$ can hold the same data as the original schema R, we must ask for a decomposition that has the *lossless join* property, i.e., the join dependency $\bowtie [R_1, \ldots, R_n]$ must be implied by Σ.

Furthermore, the decomposition should be dependency preserving or *faithful*, i.e. the dependencies on the schemas R_i which are implied by Σ should form a cover of Σ. This allows a database management system to check constraints for individual relations only, without having to compute their join. The *projection* of a set Σ of FDs onto a subschema $R_i \subseteq R$ is

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

Thus a decomposition $\mathcal{D} = \{R_1, \ldots, R_n\}$ is dependency preserving if

$$\left(\bigcup \Sigma^*[R_i]\right)^* = \left(\bigcup \Sigma_i\right)^* = \Sigma^*$$

where Σ_i is a cover for $\Sigma^*[R_i]$ (when describing the decomposition, we usually want to represent $\Sigma^*[R_i]$ by a smaller cover for it). Note that it is not sufficient to only project Σ onto the R_i, rather than Σ^*.

Example 1.2. Let $R = ABC$ with constraints $\Sigma = \{A \to B, B \to C\}$. Then $\Sigma[AC] = \emptyset$, although $A \to C$ is a FD on AC which is implied by Σ.

Normal forms are syntactic descriptions of "good" relation or database schemas. A number of normal forms have been proposed, depending on the types of integrity constraints used. For functional dependencies, the two most important normal forms are Third Normal Form (3NF) and Boyce-Codd Normal Form (BCNF).

A relation schema R is in *Boyce-Codd Normal Form* w.r.t. a set Σ of FDs on R if and only if for every non-trivial FD $X \to Y \in \Sigma$ the left hand side (LHS) X is a key for R, i.e., $X \to R \in \Sigma^*$, where

$$\Sigma^* := \{X \to Y \mid X, Y \subseteq R, \Sigma \vDash X \to Y\}$$

This normal form is desirable as it prevents redundancy and update anomalies caused by such redundancy [36].

Third Normal Form is weaker than BCNF, i.e., a relation schema in BCNF is also in 3NF. A relation schema R is in *Third Normal Form* if and only if for every non-trivial FD $X \to A \in \Sigma^*$ we have that

- the left hand side (LHS) X is a key for R, or
- attribute A lies in a minimal key of R

Attributes which lie in some minimal key are called *prime attributes*.

While 3NF does not prevent redundancy, it does prevent certain types of update anomalies [36]. The advantage of 3NF over the more powerful BCNF is that a schema can always be decomposed into 3NF while preserving dependencies [10]. This is not the case for BCNF [5].

When decomposing a schema into BCNF (or at least 3NF), we are looking for a decomposition $\mathcal{D} = \{R_1, \ldots, R_n\}$ such that each R_i is in BCNF (3NF) w.r.t. to the projected constraints $\Sigma^*[R_i]$. The following decomposition algorithm, originally suggested by Codd in [12] finds a lossless BCNF decomposition:

Algorithm "BCNF Decomposition"

```
INPUT: schema $R$ with FDs $\Sigma$
OUTPUT: lossless BCNF decomposition $\mathcal{D}$

$\mathcal{D} := \{R\}$
while $\mathcal{D}$ contains schema $R_i$ not in BCNF do
        pick non-trivial $X \to A$ on $R_i$ with $\Sigma \models X \to A$ and $\Sigma \not\models X \to R_i$
        $\mathcal{D} := \mathcal{D} \setminus \{R_i\} \cup \{R_i \setminus A, XA\}$
end
```

We call Σ *LHS-minimal* or *LHS-reduced* if the left hand sides (LHSs) of FDs $X \to Y \in \Sigma$ cannot be reduced further, i.e., for every $X' \subsetneq X$ we have $\Sigma \not\models X' \to Y$. It was shown by Biskup, Dayal and Bernstein in [10] that the following algorithm produces a lossless 3NF decomposition.

Algorithm "3NF Synthesis"

```
INPUT: schema $R$ with LHS-minimal FDs $\Sigma$
OUTPUT: dependency preserving 3NF decomposition $\mathcal{D}$

$\mathcal{D} := \emptyset$
for all $X \to Y \in \Sigma$ do
    $\mathcal{D} := \mathcal{D} \cup \{XY\}$
end
if $\mathcal{D}$ contains no key of $R$ then
   add a minimal key of $R$ to $\mathcal{D}$
```

A large number of other normal forms have been proposed, mostly for different classes of dependencies, e.g. 4NF for FDs and MVDs, and 5NF, PJ/NF, KCNF for FDs and JDs [33, 44]. However, most of them are not directly relevant to the work presented here. Those that are will be introduced where we need them.

1.3 Contributions and Outline

In this work, we will address the problem of finding a "good" database schema through decomposition. Here we face two different issues. The first issue is what decompositions should be called "good", i.e., what normal form should be used. The second issue is how to find a decomposition into the desired form.

We will first tackle the problem of how to find decompositions into the existing normal form BCNF. For 3NF this has already been solved in a satisfactory manner, as a dependency preserving 3NF decomposition can be found in polynomial time [10]. Finding dependency preserving BCNF decompositions, however, is a more challenging issue. The problem of deciding whether a dependency preserving BCNF decomposition exist is known to be co-NP-hard [5, 42]. The algorithm proposed in [5] for finding BCNF decompositions requires exponential time, but still does not always guarantee preservation of dependencies whenever possible. The approach described in [42] improves on that by finding a BCNF decomposition in polynomial time, but it does not generate dependency preserving decompositions either. Only the algorithm suggested in [38] always finds a dependency preserving BCNF decomposition if one exists, but uses a brute force approach which always requires exponential time.

We will address this problem in chapter 2 by developing an efficient algorithm called *linear resolution* for computing the *atomic closure*, which consists of all functional dependencies implied by a given set Σ of functional dependencies, which are minimal in some sense. While the size of the atomic closure can be exponential in the size of Σ, it often turns out to be rather small, and our linear resolution algorithm computes it in output-linear time. Furthermore, we identify polynomial subclasses based on the form of Σ, and describe an efficient method for updating the atomic closure when functional dependencies get added to Σ. Other applications for linear resolution, such as computing all minimal keys or computing covers on subschemas, are considered as well.

In section 2.2 the atomic closure is then used to find a suitable cover to be used with the synthesis algorithm, resulting in a BCNF decomposition if one exists. For this we improve the algorithm by Osborn [38]. In addition, we show that for finding faithful BCNF decompositions we only require a subset of the atomic closure. This subset consists of all functional dependencies which participate in cycles (e.g. $A \to B, B \to A$), and we show that our linear resolution approach can easily be adapted to compute only this subset.

In section 2.3 we then extend the core of our work from sections 2.1 and 2.2 to a complex-valued data model described in [29], which is based on record, list, set and multiset constructor. It turns out that our approach works in this complex-valued model as well, but special care has to be taken to combat an explosion in complexity, which arises due to an exponential number of subattributes. Another complication arises from the fact that a vital theorem for characterizing lossless and dependency-preserving decompositions in the relational model does not extend to the complex-valued model directly. We solve this by placing some suitable restrictions on the set Σ, and show that the theorem holds under these restrictions.

For the work of chapter 2, the choice which set Σ of functional dependencies to preserve is central. Instead of Σ we could use a cover Σ' of functional dependencies equivalent to Σ instead, and the choice of the cover determines the decomposition. This reduces the problem of finding the "right" decomposition to finding the "right" cover. As a result of working with covers instead of decompositions directly, we can ensure the preservation of dependencies more easily. It turns out that for many problems, including that of finding faithful BCNF decompositions, the "right" cover is in canonical form. Roughly speaking, canonical covers are minimal sets of minimal FDs - a precise definition is given in section 2.1.

In chapter 3 we approach the problem of finding the "right" cover (be it for BCNF decomposition or other purposes) by developing an algorithm for computing the set of all canonical covers. However, the number of canonical covers can be huge, so computing or representing them directly becomes quickly infeasible. To counter this, we partition canonical covers into partial canonical covers, which can reduce their number dramatically, and at the same time makes them easier to compute as well. Since the set of all canonical covers forms a hypergraph, with functional dependencies as vertices and covers as edges, we develop a general approach for hypergraph decomposition in section 3.1, based on the notion of *autonomous set*.

This is then used in section 3.2, where we characterize some (non-minimal) autonomous sets for the hypergraph of all canonical covers. Furthermore, we make implications between functional dependencies explicit in the form of *implication dependencies*, which are functional dependencies over other functional dependencies. This allows us to compute (partial) canonical covers as minimal keys w.r.t. a set of implication dependencies.

We derive some simple bounds on the size and number of non-redundant covers (which includes canonical covers) in section 3.3, and develop a polynomial-time method for finding finer (but still non-minimal) autonomous sets in section 3.4. As we only require so-called *essential* functional dependencies which actually appear in some canonical cover, we wish to avoid inessential ones when computing the atomic closure. In section 3.5 we reduce this to the problem of identifying essential functional dependencies. It turns out that this problem, as well as the problem of identifying autonomous sets are NP-complete and co-NP-hard, respectively. Consequently we provide some necessary, but not sufficient, criteria for testing essentiality, which can be checked in polynomial time.

Finally, we address the question of what we should call a "good" decomposition in chapter 4. We propose a new normal form (more precisely, a number of similar normal forms) called *Domination Normal Form* in section 4.1, motivated by semantic properties. Roughly speaking, we say that a decomposition is in Domination Normal Form if it is "minimal" among a set of "suitable" decompositions. The main advantages of this approach, compared to existing normal forms, are the following:

- Domination Normal Form has a clear semantic motivation, optimizing the whole decomposition rather than only considering individual schemas.

- Requirements for decompositions, such as preservation of dependencies, can easily be integrated into Domination Normal Form by restricting the set of "suitable" decompositions to those which meet the requirements.

- A decomposition into Domination Normal Form always exists, provided the set of all "suitable" decompositions is not empty.

For this we introduce two semantic notions of minimality orders, based on the size (storage space) and attribute count (how often attribute values appear) of instances. We also present a syntactical characterization in the form of a third ordering, and prove that all three orderings are equivalent in section 4.2. These new normal forms are then compared to other normal forms in section 4.3, and it turns out that they are identical to BCNF and other existing normal forms for a single relation schema, but extend them to multiple schemas in a different, and perhaps (as we will argue) more suitable way. Using the results from chapters 2 and 3, we describe algorithms for computing decompositions into Domination Normal Form efficiently in sections 4.4 and 4.5.

We conclude in chapter 5 where we summarize our results briefly and discuss some problems which remain open. Experimental results where we implemented some of our algorithms are presented in an appendix.

Some parts of this thesis have already been published. Results from sections 2.1 and 2.2 were presented in [24], while parts of chapter 4 are described in [25].

Chapter 2

Linear Resolution and Faithful BCNF Decomposition

It is well known that lossless and faithful (i.e., dependency preserving) decompositions of relational database schemas into Boyce-Codd Normal Form (BCNF) do not always exist, depending on the set of functional dependencies given, and that the corresponding decision problem is NP-hard [5, 42].

As the next lemma shows, finding a lossless and faithful BCNF decomposition is easy once we have found a faithful BCNF decomposition. This allows us to concentrate on the latter problem.

Lemma 2.1. *[10] A faithful decomposition of R is lossless iff it contains a subschema which forms a key of R.*

Every minimal key is in BCNF, as the projection of Σ^* onto it contains only trivial FDs. Thus we can easily make a faithful BCNF decomposition $\mathcal{D}$ lossless by adding a minimal key as additional subschema if $\mathcal{D}$ does not contain a key of R, such that the new decomposition is still in BCNF (and obviously faithful).

To see that there does not always exist a faithful BCNF decomposition consider e.g. the schema $R = ABC$ with FDs

$$\Sigma = \{AB \to C, C \to B\}.$$

The well-known decomposition algorithm "BCNF Decomposition" from section 1.2 produces a lossless BCNF decomposition, which however need not be faithful, even if a faithful decomposition exists (example 2.1).

Example 2.1. Consider the schema $CLRT$ containing the attributes

$$C = Course, L = Lecturer, R = Room, T = Time$$

with the functional dependencies

$$\Sigma = \{C \to L, CT \to R, LT \to C, RT \to C\}$$

The only FD in Σ for which the left hand side is not a key is $C \to L$, so algorithm "BCNF Decomposition" produces the decomposition

$$\left\{ \begin{array}{l} (CL, \{C \to L\}), \\ (CRT, \{CT \to R, RT \to C\}) \end{array} \right\}$$

The missing FD $LT \to C$ is not implied by

$$\{C \to L\} \cup \{CT \to R, RT \to C\}$$

thus the decomposition is not faithful. □

On the other hand, the popular synthesis algorithm "3NF-Synthesis" produces a faithful decomposition, but the resulting relations R_i need not be in BCNF. And again, there are cases where a faithful BCNF decomposition exists, but the synthesis algorithm does not find one:

Example 2.2. Consider again the schema $CLRT$ with the FDs

$$\Sigma = \{C \to L, CT \to R, LT \to C, RT \to C\}$$

If we synthesize a decomposition by projecting on the attributes involved in each FD in Σ and eliminate non-maximal sets, we get the decomposition

$$\left\{ \begin{array}{c} (CLT, \{C \to L, LT \to C\}), \\ (CRT, \{CT \to R, RT \to C\}) \end{array} \right\}$$

While this decomposition is clearly faithful, the subschema

$$(CLT, \{C \to L, LT \to C\})$$

is not in BCNF as the left hand side C of $C \to L$ is not a key for CLT. □

Since the schema CLT is not in BCNF, the information who is lecturer of a course is stored multiple times (once for each lecture time). Thus, in order to change the lecturer of a course, multiple tuples need to be updated. These problems are avoided by the BCNF decomposition CL, CRT from example 2.2, but there we lost $LT \to C$ which prevented us from creating tables where a lecturer is supposed to give different courses at the same time.

The only algorithm to guarantee both faithfulness and BCNF (if possible) proposed so far by Osborn in [38] is a brute-force approach which always requires exponential time. To be useful in practice, e.g. in automated design tools, we require more efficient means.

However, given that the problem we are trying to solve is NP-hard, we will not be able to find a polynomial time algorithm (unless P=NP). A more modest, but realistic goal would be an algorithm which requires exponential time in some cases, but performs much better in many of the cases we are interested in. In this chapter we develop such an algorithm (illustrated in Example 2.11 on page 28), which always finds a faithful BCNF decomposition if one exists, and computes a faithful decomposition into 3NF otherwise. The advantage over the approach in [38] is that in most cases our algorithm is much faster. Some experimental results can be found in the appendix.

2.1 Linear Resolution

The central idea of our approach is to compute all "minimal" FDs in Σ^*. While the number of such FDs can grow exponentially in the number of attributes and FDs in Σ, it often turns out to be reasonably small. In this section we develop a fast algorithm for computing the set of all "minimal" FDs, more precisely an algorithm who's runtime is linear in the size of the output and polynomial in the input.

In the following R denotes a relation schema with FD set Σ. We will use letters at the end of the alphabet $(\ldots, X, Y, Z)$ to denote subsets of R, while letters at the beginning $(A, B, C, \ldots)$ denote single attributes.

Definition 2.2. We use the following terminology:

(i) A FD $X \to A$ is called *singular*.

(ii) A non-trivial singular FD $X \to A \in \Sigma^*$ is called *atomic*, if and only if for all $Y \subsetneq X$ we have $Y \to A \notin \Sigma^*$.

(iii) The *atomic closure* Σ^{*a} of Σ is the set of all atomic FDs in Σ^*

(iv) A set $G \subseteq \Sigma^{*a}$ of atomic FDs is called *canonical cover* if it is a cover of Σ which is minimal w.r.t. set inclusion, i.e., for all $H \subsetneq G$ the set H is not a cover of Σ.

We will focus on computing the atomic closure Σ^{*a}, since it is needed for finding a dependency preserving BCNF decomposition, based on the approach of Osborn [38]. A detailed description of this will be provided in section 2.2.

Note that atomic FDs have also been called "elemental" FDs [46]. Osborn [38] uses the atomic closure as well, but without defining any name for it.

Example 2.3. Consider the set of FDs

$$\Sigma = \{A \to B, AB \to C, BC \to AD\}$$

(i) The FDs $A \to B$ and $AB \to C$ are singular but $BC \to AD$ is not.
(ii) While $A \to B$ is atomic, $AB \to C$ is not, since $A \to C \in \Sigma^*$.
(iii) For the atomic closure of Σ we get

$$\Sigma^{*a} = \{A \to B, A \to C, BC \to A, BC \to D, A \to D\}$$

(iv) The canonical covers of Σ are

$$\begin{array}{l}\{A \to B, A \to C, BC \to A, BC \to D\},\\ \{A \to B, A \to C, BC \to A, A \to D\}\end{array}$$

□

It is well-known that every set of FDs Σ has a canonical cover, and one can easily compute one by splitting non-singular FDs into singular ones, minimizing their left hand sides (LHS) and removing redundant FDs [33].

Algorithm for computing a canonical cover

INPUT: set of FDs Σ
OUTPUT: canonical cover Σ' of Σ

```
$\Sigma' := \emptyset$
for all $X \to Y \in \Sigma$ do
    $\Sigma' := \Sigma' \cup \{LHS-minimize(X \to A, \Sigma) \mid A \in Y\}$
end
for all $X \to A \in \Sigma'$ do
    if $\Sigma' \setminus \{X \to A\} \models X \to A$ then
        $\Sigma' := \Sigma' \setminus \{X \to A\}$
end
```

```
proc LHS-minimize($X \to Y, \Sigma$)
for all $A \in X$ do
    if $\Sigma \models (X \setminus A) \to Y$ then
        $X := X \setminus A$
end
```

The existence of a canonical cover immediately implies that Σ^{*a} is a cover for Σ:

Lemma 2.3. *For every set Σ of FDs, the atomic closure Σ^{*a} of Σ is a cover for Σ.*

Proof. Σ has a canonical cover Σ', which must be a subset of Σ^{*a}, thus $\Sigma^{*a} \models \Sigma' \models \Sigma$. □

2.1.1 The Basic Algorithm

We wish to compute Σ^{*a}. As we have seen in section 1.1, implication of FDs can be decided in linear time. However, creating all possible singular FDs on R and testing whether they are implied by Σ and atomic is inefficient, as the number of singular FDs on R grows exponentially in the size of R. In order to compute Σ^{*a} efficiently, we need some method for deriving new FDs.

A first candidate for this could be the Armstrong axioms from section 1.1:

$$\frac{}{X \to Y} Y \subseteq X, \quad \frac{X \to Y}{XW \to YW}, \quad \frac{X \to Y \quad Y \to Z}{X \to Z}$$

However, the first two rules rule can only derive trivial or non-minimal FDs, which seems counter-productive to our goal of obtaining minimal, non-trivial FDs. When deriving minimal FDs, they are only used to bring FDs into a form which allows us to apply the third (transitivity) rule. Thus, instead of using all three Armstrong axioms, we will discard the first two rules, but make the transitivity rule more flexible.

In our approach we use a single derivation rule, namely the resolution rule

$$\frac{X \to A \quad AY \to B}{XY \to B}$$

which has already been used successfully by Gottlob in [18], and is easily checked to be sound. This simplifies the derivation of FDs significantly. Consider e.g. the derivation tree from example 1.1, which derives $A \to D$ from the FDs $\Sigma = \{A \to B, AB \to C, BC \to D\}$:

$$\dfrac{\dfrac{A \to B}{A \to AB} \quad \dfrac{\dfrac{AB \to C}{AB \to ABC} \quad \dfrac{\dfrac{}{ABC \to BC} \quad BC \to D}{ABC \to D}}{AB \to D}}{A \to D}$$

Using the resolution rule, the derivation becomes much easier:

$$\dfrac{A \to B \quad \dfrac{AB \to C \quad BC \to D}{AB \to D}}{A \to D}$$

Note that the derivation tree (when read from bottom to top) above is *right-linear*, i.e., the left branch always ends in a leaf (a FD in Σ, rather than an arbitrary sub-tree, in this case $A \to B$ and $AB \to C$). We refer to such derivation trees as *linear resolution trees.*

Definition 2.4. For any application of the derivation rule

$$\frac{X \to A \quad AY \to B}{XY \to B}$$

we call $\left\{ \begin{array}{r} X \to A \\ AY \to B \\ XY \to B \end{array} \right\}$ the $\left\{ \begin{array}{l} \textit{substituting} \\ \textit{base} \\ \textit{derived} \end{array} \right\}$ FD.

The following theorem, in which we show that such derivations are possible in general, is central as it allows us to create a fast algorithm for computing Σ^{*a}.

Theorem 2.5. *Let Σ be a set of singular FDs. Then every atomic FD in Σ^{*a} can be derived from Σ using the resolution rule*

$$\frac{X \to A \quad AY \to B}{XY \to B} \tag{2.1}$$

This result still holds if we restrict ourselves to derivations where the substituting FDs $X \to A$ lie in Σ, i.e., for every atomic FD $X_i \to A_i \in \Sigma^{*a}$ there exists a linear resolution tree deriving $X_i \to A_i$ from Σ.

Proof. Let $X \to A \in \Sigma^{*a}$, and thus $A \in X^* \setminus X$. We use the known fact that the "closure" algorithm works. For any run of the "closure" algorithm let $X_i \to A_i \in \Sigma, i = 1 \dots k$ be the FDs $X' \to Y$ used to compute X^* up to the point where $A = A_k$ is added, in that order. We may assume that the set of FDs used is minimal, i.e., for every $X_i \to A_i$ with $i < k$ the attribute A_i lies in some X_j (with $i < j <= k$).

We start our derivation with $X_k \to A (= A_k)$, and then successively use $X_i \to A_i$ for $i = k-1, \dots, 1$ in the resolution rule (2.1):

$$\frac{X_i \to A_i \quad U_{i+1} \to A}{U_i \to A}$$

In this, the derived left hand sides U_i have the form

$$U_k = X_k, U_{i<k} = X_i(U_{i+1} \setminus A_i)$$

It is easy to see that $U_j \subseteq X \cup \{A_1, \dots, A_{j-1}\}$, and in particular $U_1 \subseteq X$. Since $X \to A$ is atomic, we get $U_1 = X$, thus we have indeed constructed the derivation we wanted. □

Note that the intermediate FDs $U_i \to A$ during the derivation need not be atomic, since the U_i need not be minimal. As example 2.4 shows, this is unavoidable when using the resolution rule.

Example 2.4. Consider the set

$$\Sigma = \{A \to B, A \to C, BC \to D\}$$

The atomic FD $A \to D$ cannot be derived using (2.1) without intermediate non-atomic FDs: The only possible applications of the resolution rule are

$$\frac{A \to B \quad BC \to D}{AC \to D} \text{ and } \frac{A \to C \quad BC \to D}{AB \to D},$$

and neither $AC \to D$ nor $AB \to D$ are atomic. □

Indeed, no derivation rule of the form considered, which are commonly referred to as Hilbert-Style derivation rules, always produces atomic FDs. This is because these rules only consider a subset $S \subseteq \Sigma$ as premises, but FDs which are atomic w.r.t. S need not be atomic w.r.t. Σ.

Since we are only interested in atomic FDs, we reduce the left hand side of any FD we derive by removing attributes that are not needed, or *extraneous*:

Definition 2.6. Let Σ be a set of FDs and $X \to Y \in \Sigma^*$. We say that an attribute $A \in X$ is *extraneous* in $X \to Y$ w.r.t. Σ if $(X \setminus A) \to Y \in \Sigma^*$. A FD $X \to Y$ without extraneous attributes in X is called *LHS-minimal.* For an attribute set X we call $A \in X$ extraneous if it is extraneous in $X \to X$, i.e., if $(X \setminus A) \to X \in \Sigma^*$.

This leads to the following derivation rule:

$$\frac{X \to A \quad AY \to B}{LM_\Sigma(XY \to B)}$$

where LM_Σ denotes LHS-minimization w.r.t. Σ. Note that this is not a Hilbert-Style derivation rule, as it relies on the whole set Σ rather than just the premises. We refer to the derivation rule above as *LM-resolution.*

Corollary 2.7. *Let Σ be a set of singular FDs. Then every atomic FD in Σ^{*a} can be derived from Σ using the LM-resolution rule*

$$\frac{X \to A \quad AY \to B}{LM_\Sigma(XY \to B)} \tag{2.2}$$

where the substituting FDs $X \to A$ lie in Σ.

Proof. Theorem 2.5 states the same result for the resolution rule 2.1. The FD $LM_\Sigma(XY \to B)$ derived by rule 2.2 implies $XY \to B$, and can replace it in any derivation which uses $XY \to B$ as base FD. □

LHS-minimizing FDs immediately can reduce the number of possible derivation sequences considerably. This provides us with an efficient algorithm, which we name *linear resolution*, that computes the atomic closure Σ^{*a} of any set of functional dependencies Σ.

Algorithm "linear resolution"

```
    INPUT: set of FDs $\Sigma$
    OUTPUT: atomic closure $\Sigma^{*a}$

    compute a canonical cover $\Sigma'$ of $\Sigma$
    $\Sigma^{*a} := \Sigma'$
    for all $Y \to B \in \Sigma^{*a}$ do
        for all $X \to A \in \Sigma'$ with $A \in Y, B \notin X$ do
            $\Sigma^{*a} := \Sigma^{*a} \cup \{LM_\Sigma((XY \setminus A) \to B)\}$
        end
    end
```

Here the function LM_Σ can be computed using the function "LHS-minimize" described earlier in section 2.1.

Example 2.5. Starting with the canonical cover

$$\Sigma = \{C \to L, CT \to R, LT \to C, RT \to C\}$$

from examples 2.1 and 2.2, we use resolution:

$$\frac{RT \to C \quad C \to L}{RT \to L}, \frac{LT \to C \quad CT \to R}{LT \to R}$$

The newly found FDs $RT \to L$ and $LT \to R$ are already atomic, so we add them to Σ^{*a}. We then test whether new resolution steps have become possible:

$$\frac{CT \to R \quad RT \to L}{[CT \to L]}, \frac{C \to L \quad LT \to R}{[CT \to R]}$$

The FD $CT \to L$ can be LHS-minimized to $C \to L$, which has already been found. The FD $CT \to R$ is already contained in Σ^{*a} as well, so no further atomic FDs can be derived. We therefore obtain:

$$\Sigma^{*a} = \Sigma \cup \{RT \to L, LT \to R\}$$

□

Theorem 2.8. *The "linear resolution" algorithm computes Σ^{*a} correctly.*

Proof. Follows from Corollary 2.7. □

We remark that Corollary 2.7 immediately gives us a sound and complete derivation system for atomic FDs:

$$\frac{X \to A \quad AY \to B}{LM_\Sigma(XY \to B)} \quad B \notin X$$

However, this system is not a Hilbert-Style axiomatization, and according to the discussion earlier no such axiomatization exists. For singular FDs an axiomatization exists though.

Corollary 2.9. *The following axiom system is sound and complete for singular FDs:*

$$\frac{X \to A \quad Y \to B}{X(Y \setminus A) \to B}, \quad \frac{}{XA \to A}$$

Proof. It is obvious that the rules are sound. To show completeness, let $W \to A$ be any singular FD in Σ^*. If $W \to A$ is trivial it can be derived directly from the second rule. Otherwise there exists an atomic FD $X \to A$ with $X \subseteq W$, and $X \to A$ can be derived using the first rule by Theorem 2.5. If $W \neq X$ (and thus $W \neq \emptyset$) then we derive $W \to C$ for some $C \in W$ by the second rule. Now $W \to A$ can be derived from $W \to C$ and $X \to A$ by another application of the first rule. □

Note that the set X in the proof could be empty, so that the resolution rule (2.1) cannot be applied in the last step. Thus we had to modify it slightly.

2.1.2 Improvements and Complexity Analysis

We now discuss possible improvements and implementation issues for the linear resolution algorithm. Based on these improvements we present a brief complexity analysis.

In the linear resolution algorithm presented, we select the substituting FDs from a canonical cover of Σ rather than Σ itself. This cover can be larger than Σ, since splitting FDs into singular FDs increases the number of FDs. To avoid this, we use the original set Σ and replace the singular LM-resolution rule (2.2) by the generalized LM-resolution rule

$$\frac{X \to Z \quad Y \to B}{LM_\Sigma(X(Y \setminus Z) \to B)} \quad Z \cap Y \neq \emptyset$$

This generalized rule is then used in the same fashion as rule (2.2) in our linear resolution approach.

Lemma 2.10. *Let Σ be a set of FDs and Σ' a singular cover of Σ. Then every atomic FD in Σ^{*a} can be derived from Σ and Σ' using the generalized resolution rule*

$$\frac{X \to Z \quad Y \to B}{X(Y \setminus Z) \to B} \quad Z \cap Y \neq \emptyset \tag{2.3}$$

where the substituting FDs $X \to Z$ lie in Σ, and the base FDs $Y \to B$ are derived or lie in Σ'.

Proof. Let $X \to A \in \Sigma^{*a} = \Sigma'^{*a}$. Then there exists a FD $Y \to A \in \Sigma'$ with $\Sigma' \models X \to Y$, and thus $\Sigma \models X \to Y$. We can then construct a derivation of $X \to A$ from $Y \to A$ as in the proof of Theorem 2.5, where the substituting FDs are those used in the closure computation of X under Σ, up to the point where Y is included, in reverse order. □

Corollary 2.11. *Let Σ be a set of FDs and Σ' a singular cover of Σ. Then every atomic FD in Σ^{*a} can be derived from Σ and Σ' using the generalized LM-resolution rule*

$$\frac{X \to Z \quad Y \to B}{LM_\Sigma(X(Y \setminus Z) \to B)} \quad Z \cap Y \neq \emptyset \tag{2.4}$$

where the substituting FDs $X \to Z$ lie in Σ, and the base FDs $Y \to B$ are derived or lie in Σ'.

Proof. As for Corollary 2.7. □

To speed up LHS-minimization of $X(Y \setminus Z) \to B$, we replace each FD $X \to Z \in \Sigma$ by $X \to X^*$. As this can make FDs in Σ redundant, we then remove such redundant FDs. Note that this turns Σ into a *minimal cover*, i.e., a cover with a minimal number of FDs [40]. This is clearly a bonus, since lowering the number of potential substituting FDs reduces the number of rule applications.

For our next optimization, we partition the set Σ^{*a} into sets Σ^{*a}_A, where

$$\Sigma^{*a}_A := \{W \to A \in \Sigma^{*a}\}$$

The set Σ^{*a}_A of all atomic FDs with right hand side A can be computed independently from atomic FDs with other RHSs, as the derivation (using linear resolution) of a FD in Σ^{*a}_A only uses substituting FDs in Σ and base FDs in Σ^{*a}_A.

Computation of Σ^{*a}_A can now be optimized, by reducing the set $\Sigma_A \subseteq \Sigma$ from which substituting FDs need to be chosen. For this we first initialize Σ_A with Σ, and then remove all FDs $X \to Z \in \Sigma_A$ with $A \in X^*$. If $A \notin X$ (otherwise $X \to A$ would be trivial), then we add $LM_\Sigma(X \to A)$ to the initial set Σ^{*a}_A of base FDs. This still allows us to derive all FDs in Σ^{*a}_A, since any application of (2.4) using $X \to Z$ with $A \in X^*$ as substituting FD could result in the FD $LM_\Sigma(X \to A)$, which is either trivial or already contained in the initial set.

Our optimized linear resolution algorithm is given next.

Algorithm “linear resolution” revised

```
INPUT: set of FDs $\Sigma$
OUTPUT: atomic closure $\Sigma^{*a}$

$\Sigma := \{X \to X^* \mid X \to Z \in \Sigma\}$
for all $X \to Z \in \Sigma$ do
    if $\Sigma \setminus \{X \to Z\} \vDash X \to Z$ then
        $\Sigma := \Sigma \setminus \{X \to Z\}$
end
for all $A \in RHS(\Sigma) := \bigcup_{X \to Z \in \Sigma} Z$ do
    $\Sigma^{*a}_A := \emptyset, \Sigma_A := \emptyset$
    for all $X \to Z \in \Sigma$ do
        if $A \notin X^*$ then
            $\Sigma_A := \Sigma_A \cup \{X \to Z\}$
        else if $A \notin X$ then
            $\Sigma^{*a}_A := \Sigma^{*a}_A \cup \{LM_\Sigma(X \to A)\}$
    end
    for all $Y \to A \in \Sigma^{*a}_A$ do
        for all $X \to Z \in \Sigma_A$ with $Y \cap Z \neq \emptyset$ do
            $\Sigma^{*a}_A := \Sigma^{*a}_A \cup \{LM_\Sigma(X(Y \setminus Z) \to A)\}$
        end
    end
end
$\Sigma^{*a} := \bigcup \Sigma^{*a}_A$
```

Theorem 2.12. *The revised "linear resolution" algorithm computes Σ^{*a} correctly.*

Proof. Follows from Corollary 2.11 and the arguments above. □

Before we give a complexity analysis, we need to point out an implementation issue which has been completely ignored so far. When adding a new FD $U \to A$ to Σ_A^{*a}, we need to check whether $U \to A$ is already contained in Σ_A^{*a}. Since Σ_A^{*a} can potentially be large, we require an implementation for sets which allows us to add or remove elements and check containment quickly. While a simple hash table implementation would be sufficient for practical purposes, it can potentially degenerate, resulting in poor worst-case behavior. For the sake of a proper theoretical analysis, we present a data structure for storing the LHSs of FDs in Σ_A^{*a} which always allows for fast updates and containment checking. Note that it suffices to store the LHSs of FDs in Σ_A^{*a}, since we know that their RHS is A.

Data Structure. Let $R = \{A_1, \dots, A_k\}$. We will represent a subset $S \subseteq \mathcal{P}(R)$ of the powerset of R by a binary tree $T(S)$ of height $k+1$ (or 0 if $S = \emptyset$). To describe this representation, we identify each of the $2^{k+1} - 1$ potential nodes of this tree with a binary string of length at most k, in such a way that the string of a node's parent is obtained by removing the last bit from the child node's string. Similarly, we identify each subset $s \in \mathcal{P}(R)$ of R with a binary string of length k, where the i-th bit is 1 iff $A_i \in s$. Then the binary tree $T(S)$ representing S is constructed as follows: For each $s \in S$ add the node with the same identifying string as s to $T(s)$, as well as all its ancestor nodes.

With this representation, adding or removing a set s to S, or checking whether s lies in S already, can be performed in $O(k)$. This is done by starting at the root, and following the path described by the identifying string of s, which leads to the corresponding potential node.

We will now give a brief complexity analysis.

Lemma 2.13. *The revised "linear resolution" algorithm runs in time $O(f \cdot k^2 n^2)$, where*

$$\begin{aligned} k &= \text{number of attributes in } R \\ n &= \text{number of FDs in } \Sigma \\ f &= \text{number of FDs in } \Sigma^{*a} \end{aligned}$$

Proof. Computing X^* for each $X \to A \in \Sigma$ can be performed in $O(kn^2)$, given that each X^* can be computed in $O(kn)$ using the "closure" algorithm [5]. Checking whether a FD is implied also requires only one closure computation, so that the first five lines, which turn Σ into a minimal cover, only require $O(kn^2)$ steps.

Each test whether an attribute is extraneous requires one run of the "closure" algorithm. Thus each LHS-minimization takes at most $O(k^2 n)$ operations. These LHS-minimizations are the most time consuming steps, so that it suffices to count how often they need to be performed.

Initializing the sets Σ_A^{*a}, Σ_A takes at most n calls of LM_Σ. Computing Σ_A^{*a} requires at most $f_A \cdot n$ resolution steps, where f_A is the number of FDs in the final set Σ_A^{*a}, and each resolution step calls LM_Σ once. Thus LM_Σ is called at most

$$\sum f_A \cdot n = f \cdot n$$

times in total, leading to an overall complexity of $O(f \cdot k^2 n^2)$. □

Note that the algorithm is not polynomial in the size of the input (which lies between $k+n$ and $k \cdot n$), since f can be exponential in n and k. However, for outputs of moderate size, i.e., whenever f does not grow too large, our algorithm performs well. Experimental results can be found in the appendix.

We will establish some simple bounds on the size of f next.

2.1.3 Polynomial Cases

While the size of the atomic closure can be potentially exponential in the size of Σ, it is often much smaller. A reason for this is that the left hand sides (LHSs) of functional dependencies are often small. It is estimated that the majority of FDs occurring in practice contain only a single attribute in their LHS.

Definition 2.14. We call a FD $X \to Y$ *unitary* if X consists of a single attribute.

We now establish some upper bounds for the size of Σ^{*a}, depending on the number and size of non-unitary FDs in Σ. For this purpose, the size of a FD or set of FDs is the number of attributes appearing in it, counting each attribute as often as it occurs.

Lemma 2.15. *If Σ contains only unitary FDs, then Σ^{*a} is polynomial in the size of Σ.*

Proof. Since all FDs in Σ are unitary, all FDs in Σ^{*a} are unitary as well, as they can all be derived using the linear resolution rule. The number of FDs in Σ^{*a} is thus bounded by k^2, where k is the number of attributes occurring in Σ. □

Note however, that allowing for even a single non-unitary FD in Σ can make Σ^{*a} exponential in Σ again.

Example 2.6. Consider the set of FDs

$$\Sigma = \{A_1 \to B_1, \dots, A_n \to B_n, B_1 \dots B_n \to C\}$$

Σ contains only a single non-unitary FD, but Σ^{*a} contains over 2^n FDs. □

The exponential growth of Σ^{*a} in example 2.6 was due to the FD $B_1 \dots B_n \to C$ with large LHS. Instead of restricting the number of non-unitary FDs, we could restrict the size of their LHSs. However, as the next example shows, this alone is not sufficient.

Example 2.7. Let

$$\Sigma = \left\{ \begin{array}{c} A_{(1,v)}A_{(2,v)} \to A_{(1,v-1)}, \dots, A_{(2^v-1,v)}A_{(2^v,v)} \to A_{(2^{v-1},v-1)}, \\ A_{(1,v-1)}A_{(2,v-1)} \to A_{(1,v-2)}, \dots, A_{(2^{v-1}-1,v-1)}A_{(2^{v-1},v-1)} \to A_{(2^{v-2},v-2)}, \\ \dots \\ A_{(1,1)}A_{(2,1)} \to A_{(1,0)}, \\ B_1 \to A_{(1,v)}, \dots, B_{2^v} \to A_{(2^v,v)} \end{array} \right\}$$

Then all FDs in Σ have at most 2 attributes in their LHS. However, they imply the atomic FD

$$A_{(1,v)}A_{(2,v)} \dots A_{(2^v,v)} \to A_{(1,0)}$$

with 2^v attributes in its LHS. Using the FDs

$$B_1 \to A_{(1,v)}, \dots, B_{2^v} \to A_{(2^v,v)}$$

we can derive 2^{2^v} different atomic FDs, showing that Σ^{*a} is exponential in Σ (whose size is merely of order 2^v). □

To guarantee that Σ^{*a} is only of polynomial size, we must not only limit the size of the LHSs of FDs in Σ, but also of those in Σ^{*a}. We will see that this can be done by restricting both the number of non-unitary FDs in Σ and the size of their LHSs.

Definition 2.16. We define the *width* of a FD $X \to Y$ with $X \neq \emptyset$ as

$$width(X \to Y) := |X| - 1$$

In particular all unitary FDs have width zero. We define the width of a FD $\emptyset \to Y$ as zero as well. Furthermore, the *width* of a set Σ of FDs is

$$width(\Sigma) := \sum_{X \to Y \in \Sigma} width(X \to Y)$$

Note that limiting the number of non-unitary FDs (not considering FDs with empty LHS) in Σ and their width is the same as limiting their total width, i.e., the width of Σ. If Σ has width W, then it contains at most W non-unitary FDs, and no FD in Σ has width greater than W. On the other hand, if Σ contains no more than n non-unitary FDs, and no FD in Σ has width greater than W, then the width of Σ is bounded by $n \cdot W$.

Note also that the width Σ is a lower bound for its size, though no function of the width is an upper bound for the size. This is because unitary FDs do not add to the width, but do add to the size of Σ.

Lemma 2.17. *Let Σ be a set of FDs. Then for all FDs $X \to A \in \Sigma^{*a}$ we have*

$$width(X \to A) \leq width(\Sigma)$$

Proof. Every atomic FD in Σ^{*a} can be derived from Σ by applying the generalized resolution rule (2.3):

$$\frac{X \to Z \quad Y \to A}{X(Y \setminus X^*) \to A} \quad Z \cap Y \neq \emptyset$$

in a linear fashion. The widths of the FDs involved in an application of the generalized resolution rule can be related as follows:

$$width(X(Y \setminus X^*) \to A) \leq width(X \to Z) + width(Y \to A)$$

As any FD in Σ is used in deriving a FD in Σ^{*a} at most once, the lemma follows. □

Lemma 2.18. *Let R be a schema with k attributes. Then the number of non-trivial singular FDs on R of width at most W is less than k^{W+2}.*

Proof. Consider the function

$$f : \begin{array}{ccc} R^{W+2} & \to & \{\text{non-trivial singular FDs on } R \text{ of width at most } W\} \\ (A_1, \ldots, A_{W+2}) & \mapsto & \{A_1, \ldots, A_{W+1}\} \setminus \{A_{W+2}\} \to A_{W+2} \end{array}$$

Clearly f is surjective but not injective, thus the target domain contains less elements than the source domain R^{W+2}, which contains k^{W+2} elements. □

We can now generalize Lemma 2.15.

Theorem 2.19. *Consider all sets Σ of FDs with width$(\Sigma) \leq W$ for some fixed constant W. Then the size of Σ^{*a} is polynomial in the size of Σ.*

Proof. By Lemma 2.17 all FDs in Σ^{*a} have width at most W. Let k be the number of attributes appearing in Σ. By Lemma 2.18 there exist less than k^{W+2} non-trivial singular FDs of width at most W, using only those k attributes. They form a superset of Σ^{*a} and thus bound its size. Since k^{W+2} is polynomial in the size of Σ, this shows the theorem. □

Corollary 2.20. *Consider all sets Σ of FDs with width$(\Sigma) \leq W$ for some fixed constant W. The (revised) "linear resolution" algorithm operates on them in polynomial time.*

Proof. Follows immediately from Lemma 2.13 and Theorem 2.19. □

Note that, while the upper bound of k^{W+2} on the cardinality of Σ^{*a} has been useful for establishing some theoretical results, it is usually far too large to be used as an estimate (say, for estimating runtime in advance). Consider e.g. the set of FDs

$$\Sigma = \{C \to L, CT \to R, LT \to C, RT \to C\}$$

from examples 2.1, 2.2 and 2.5. We get $k = 4$ and $W = 3$, which provides us with an upper bound of $4^{3+2} = 1024$, although we have seen that Σ^{*a} only contains 6 FDs.

Finally, we want to stress that Corollary 2.20 does *not* state that computing Σ^{*a} is *fixed parameter tractable* (FPT) in the width of Σ, as introduced by Downey and Fellows in [13]. To be FPT in W, the runtime of an algorithm would need to be bounded by $f(W) \cdot P(size(\Sigma))$ for some function f and some polynomial P. This is not the case for the bound $k^{W+2} \cdot k^2 n^2$ we established, and the following variation of example 2.6 shows that Σ^{*a} can grow too large to allow computations to be FPT in the width.

Example 2.8. Consider the set of FDs

$$\Sigma = \left\{ \begin{array}{c} B_1 \dots B_n \to C, \\ A_{1,1} \to B_1, A_{2,1} \to B_1, \dots, A_{k,1} \to B_1, \\ \vdots \\ A_{1,n} \to B_n, A_{2,n} \to B_n, \dots, A_{k,n} \to B_n \end{array} \right\}$$

The width of Σ is $W = n - 1$, and the size $S = 2nk + n + 1$, but Σ^{*a} contains over k^n FDs. It is easy to see that k^n is not bounded by any $f(W) \cdot P(S)$. □

2.1.4 Updating the Atomic Closure

In this section we address the question how to update Σ^{*a}, once computed, when Σ changes. If possible we would like to avoid recomputing Σ^{*a} from scratch. We begin by noting that small changes to Σ can cause large changes in Σ^{*a}.

Example 2.9. Consider the set of atomic FDs

$$\Sigma = \{A_1 \to B_1, \dots, A_n \to B_n\}$$

Clearly $\Sigma^{*a} = \Sigma$. However, if we add the FD $B_1 \dots B_n \to C$ to Σ, then the new Σ^{*a} contains a total of 2^n new atomic FDs. These are obtained by substituting any number of attributes B_i in $B_1 \dots B_n \to C$ by the corresponding A_i. □

The same can happen when removing FDs from Σ.

Example 2.10. Consider the set of atomic FDs

$$\Sigma = \{AB_1 \to C_1, \ldots, AB_n \to C_n, C_1 \ldots C_n \to D, A \to D\}$$

Again, $\Sigma^{*a} = \Sigma$. If we remove $A \to D$ from Σ, it also gets removed from Σ^{*a}, but Σ^{*a} now contains $2^n - 1$ new FDs. These are obtained by substituting any positive number of attributes C_i in $C_1 \ldots C_n \to D$ by the corresponding AB_i. As $A \to D$ is no longer implied by Σ, the resulting FDs are now atomic. □

We cannot hope to update Σ^{*a} quickly if the changes required to it are large, but if Σ^{*a} changes only by a few FDs, we might be able to do better. We consider two types of changes to Σ, depending on whether FDs get added to or removed from Σ. To distinguish Σ before and after the changes, we will denote the changed set by Σ_+ or Σ_-, respectively.

If we only add FDs to Σ, i.e., $\Sigma \subseteq \Sigma_+$, then all FDs in Σ^{*a} will still lie in Σ_+^*, though their left hand side may not be minimal. It is however easy to LHS-minimize them to atomic FDs in Σ_+^{*a}, using the algorithm from section 2.1. Let $\Sigma_{LM} := LM_{\Sigma_+}(\Sigma^{*a})$ denote some set we might get when LHS-minimizing all FDs in Σ^{*a} w.r.t. Σ_+. By performing this LHS-minimization we ensure that for every FD $X \to A$ which can be derived by resolution from a base FD in Σ^{*a} and a substituting FD in Σ, the set Σ_{LM} contains a "stronger" FD $U \to A$ with $U \subseteq X$. Therefore we do not have to apply any of those resolution steps again. We can obtain Σ_+^{*a} by linear resolution starting with $\Sigma_{LM} \cup (\Sigma_+ \setminus \Sigma)$ as base FDs, but using a FD in Σ_{LM} as base FD only in combination with a substituting FD in $\Sigma_+ \setminus \Sigma$.

For the sake of simplicity, we adapt only the basic "linear resolution" algorithm to perform this update, although the improvements used in the revised version could be applied here as well.

Algorithm "update atomic closure"

```
INPUT: sets of singular FDs $\Sigma \subseteq \Sigma_+$, atomic closure $\Sigma^{*a}$
OUTPUT: atomic closure $\Sigma_+^{*a}$

$\Sigma_{LM} := LM_{\Sigma_+}(\Sigma^{*a})$
$\Sigma_+^{*a} := \Sigma_{LM} \cup LM_{\Sigma_+}(\Sigma_+ \setminus \Sigma)$
for all $Y \to B \in \Sigma_+^{*a}$ do
    for all $X \to A \in \left\{ \begin{array}{ll} \Sigma_+ & \text{if } Y \to B \notin \Sigma_{LM} \\ \Sigma_+ \setminus \Sigma & \text{if } Y \to B \in \Sigma_{LM} \end{array} \right\}$ with $A \in Y, B \notin X$ do
        $\Sigma_+^{*a} := \Sigma_+^{*a} \cup \{LM_{\Sigma_+}((XY \setminus A) \to B)\}$
    end
end
```

The terms $LM_{\Sigma_+}(\ldots)$, which denote LHS-minimization w.r.t. Σ_+, can be computed using the function "LHS-minimize" from section 2.1. To analyze the complexity of this update algorithm, let

$$\begin{aligned} & k := |R|, \\ & n := |\Sigma|,\ n_+ := |\Sigma_+|,\ n_\Delta := n_+ - n, \\ & f := |\Sigma^{*a}|,\ f_+ := |\Sigma_+^{*a}|,\ f_\Delta := |\Sigma_+^{*a} \setminus \Sigma^{*a}| \end{aligned}$$

A single FD can be LHS-minimized w.r.t. Σ_+ in $O(k^2 \cdot n_+)$, so Σ_{LM} can be computed in $O(f \cdot k^2 \cdot n_+)$. In return, the number of resolution steps to perform is bounded by $f \cdot n_\Delta + f_\Delta \cdot n_+$ (rather than $f_+ \cdot n_+$ as we would get by computing Σ_+^{*a} from scratch), which is small if f_Δ and n_Δ are small. Each of the derived FDs needs to be LHS-minimized, which leads to a total complexity of

$$O((f \cdot n_\Delta + f_\Delta \cdot n_+) \cdot k^2 \cdot n_+)$$

Note that if $\Sigma^{*a} \subseteq \Sigma_+^{*a}$, the only extra work performed by computing Σ^{*a} first and then updating rather than computing Σ_+^{*a} directly is the computation of Σ_{LM} (which would not even be necessary if we knew beforehand that $\Sigma^{*a} \subseteq \Sigma_+^{*a}$).

This approach does not work, however, when FDs get removed from Σ. To explain this, let us compare the cases of Σ_+, where FDs get added, and Σ_-, where they get removed. The essential problem in reusing Σ^{*a} is that we LHS-minimized the derived FDs w.r.t. the wrong set Σ, instead of Σ_+ or Σ_-. In the first case this could easily be corrected by LHS-minimizing again, this time w.r.t. Σ_+. In the latter case, however, we would need to "undo" some LHS-minimization steps. It is not clear how this could be done efficiently, considering that for a FD $U \to A \in \Sigma^{*a}$ we would need to compute *all* $X \supset U$ for which $X \to A$ could have been derived using linear resolution.

However, one would expect that in practice it is more common that new FDs are added, which have been overlooked previously. At other times, a designer may be given a set of FDs Σ_1 which are known to hold, whereas a second set of FD Σ_2 are expected to hold, but whether they really do is not certain. In such a case, we could first compute Σ_1^{*a}, and then update it to $(\Sigma_1 \cup \Sigma_2)^{*a}$, storing both sets. This way, if some FDs from Σ_2 later turn out to be invalid, one can compute $(\Sigma_1 \cup (\Sigma_2 \setminus \{\ldots\}))^{*a}$ from Σ_1^{*a}, which might save at least some work. Obviously this approach can be generalized if we have more than two sets of FDs, which are ordered w.r.t. confidence in them.

2.1.5 Other Applications

The linear resolution algorithm can also be used to compute all minimal left hand sides X for any given right hand side Y, i.e., all minimal X with $X \to Y \in \Sigma^*$.

Theorem 2.21. *Let Σ be any set of singular FDs over R and $Y \subseteq R$. Then any FD $X \to Y \in \Sigma^*$ with minimal left hand side X can be derived from $Y \to Y$ using linear resolution.*

Proof. Let X be minimal with $X \to Y \in \Sigma^*$. For every $A_i \in Y \setminus X$ there exists a minimal $X_i \subseteq X$ with $X_i \to A_i \in \Sigma^{*a}$. By Theorem 2.5 every $X_i \to A_i \in \Sigma^{*a}$ has a linear resolution tree in Σ. Combining these resolution trees to substitute all $A_i \in Y \setminus X$ in the LHS of $Y \to Y$ (possibly skipping some if the attribute A_i has been eliminated in an earlier step) yields a linear resolution of $X' \to Y$ for some $X' \subseteq \bigcup X_i \subseteq X$. Since X is minimal $X' = X$. □

In particular, linear resolution can be used to compute all minimal keys of a relation, and thus all prime attributes.

As well, the atomic closure Σ^{*a} can be used to compute the atomic closure of the projection of Σ^* over some subschema S, using the following simple lemma:

Lemma 2.22. *Let $S \subseteq R$ be a subschema and Σ a set of FDs over R. Then*

$$(\Sigma^*[S])^{*a} = \Sigma^{*a}[S]$$

Proof. A non-trivial singular FD $X \to A$ on S is atomic w.r.t. Σ if and only if it is contained in Σ^* (and thus in $\Sigma^*[S]$), and for any true subset $U \subsetneq X$ the FD $U \to A$ is not contained in Σ^* (equivalently, in $\Sigma^*[S]$), i.e., if and only if it is atomic w.r.t. $\Sigma^*[S]$. □

Since the atomic closure $(\Sigma^*[S])^{*a}$ is a cover for $\Sigma^*[S]$ by Lemma 2.3, we can use Lemma 2.22 to compute covers for subschemas. For this we first compute Σ^{*a} using linear resolution, and then project it onto the subschema. Note that this approach is particularly efficient if we have already computed Σ^{*a} for other purposes.

2.1.6 Related work

A similar method for deriving all minimal keys has been described by Lucchesi and Osborn in [31]. However, while they derive keys in the same manner, i.e., by using the resolution rule (2.1) in a linear fashion, they do not consider linear resolution as a general method for deriving minimal LHSs. Furthermore, their algorithm includes an extra step which checks for inclusion between derived keys, which makes the approach less efficient for large sets Σ^{*a}.

Another similar algorithm was published even earlier by Fadous and Forsyth in [14], although their description is not always clear, and the algorithm does not avoid non-minimal intermediate keys. It is similar to linear resolution without minimization of derived LHSs.

In [18] Gottlob presents a different algorithm for computing covers on subschemas, which also uses the resolution rule for deriving new FDs. This is done in a non-linear fashion, which results in more resolution steps to be checked. In exchange, other optimizations are applied, which however do not work with linear resolution.

The approach taken by Mannila and Räihä in [37] is in some ways similar to ours. For a set $X \subseteq R$ and an attribute $A \in R$ they define the set

$$max(X, A) := \{Y \subseteq X \setminus A \mid Y \text{ is a maximal set such that } Y \nrightarrow A\}$$

of all maximal attribute sets which do not determine A. The sets $max(R, A)$ and $max(X, A)$ can be used to test whether A is prime, and whether a subschema X is in BCNF or 3NF. Instead of computing $max(X, A)$ directly for every subschema X, they compute $max(R, A)$ using an algorithm given in [35], and then obtain $max(X, A)$ as follows:

$$max(X, A) = \text{maximal sets in } \{Y \cap X \mid Y \in max(R, A)\}$$

To compare this with our work, let us define

$$\begin{aligned} min(X, A) &:= \{Y \subseteq X \setminus A \mid Y \text{ is a minimal set such that } Y \to A\} \\ &= \{Y \subseteq X \mid Y \to A \in \Sigma^{*a}\} \end{aligned}$$

so that the sets $min(R, A)$ together form an equivalent description of Σ^{*a}.

The sets $max(X, A)$ and $min(X, A)$ are simple hypergraphs on X (see section 3.1), and it is easy to show (as in [22]) that they are related as follows:

$$min(X, A) = Tr(\overline{max(X, A)})$$

or equivalently

$$max(X, A) = Tr(\overline{min(X, A)})$$

where $\overline{H} = \{X \setminus Y \mid Y \in H\}$ and $Tr(H)$ is the transversal hypergraph of H.

For every simple hypergraph H on $R \setminus A$ there exists a set Σ of FDs on R such that $min(R, A) = H$ and another one such that $max(R, A) = H$. Thus, neither set is larger or smaller than the other in general, and due to the symmetry in their relationship we would expect them to be equally large "on average".

However, we want to highlight some advantages of our approach:

- The set Σ^{*a} and correspondingly $min(R, A)$ is not dependent on R but only on Σ, that is, we can add extra attributes to R without changing $min(R, A)$. In contrast, extra attributes get added to all sets in $max(R, A)$. However, this is just a matter of elegance, not of efficiency.
- Obtaining $min(X, A)$ from $min(R, A)$ is easier than obtaining $max(X, A)$ from $max(R, A)$. In the latter case, all sets $X \cap Y$ need to be checked for maximality.
- We can use Σ^{*a} for computing a cover on subschemas, and for synthesizing decompositions (see section 2.2).
- While $min(R, A)$ can be computed in time polynomial (even linear) in the size of the output $min(R, A)$ using linear resolution, the algorithm given in [35] for computing $max(R, A)$ does not have this property. As far as we know, it is still an open question whether an output-polynomial algorithm for computing $max(R, A)$ exists.

2.2 Faithful BCNF Decomposition

In the following we will use the atomic closure Σ^{*a} to construct a faithful BCNF decomposition $\mathcal{D}$, provided it exists. In this we take a similar approach as was taken by Osborn in [38]. The main difference to the algorithm given in [38] is that we compute Σ^{*a} efficiently using linear resolution, rather than using brute force as was done in [38].

2.2.1 The Basic Algorithm

When synthesizing a decomposition from a set of FDs Σ, we create a schema XY for every FD $X \to Y \in \Sigma$. This ensures that the decomposition is faithful, though it might not be in BCNF. To obtain a BCNF decomposition, we must use a cover of Σ with the "right" FDs, so that the subschemas created are in BCNF.

Definition 2.23. A FD $X \to Y \in \Sigma^*$ is called *critical* w.r.t. Σ if XY is not in BCNF w.r.t. $\Sigma^*[XY]$. A cover G of Σ is called *critical* if it contains a critical FD. A FD or cover that is not critical is called *uncritical.*

Theorem 2.24. *The following are equivalent:*

(i) A schema (R, Σ) has a faithful and lossless decomposition into BCNF

(ii) Σ has an uncritical cover $\Sigma' \subseteq \Sigma^*$

(iii) Σ has an uncritical atomic cover $\Sigma'' \subseteq \Sigma^{*a}$

Proof. $(ii) \rightarrow (i)$: If Σ has an uncritical cover Σ', then we can synthesize a faithful BCNF decomposition $\mathcal{D}$ by creating a schema XY for every $X \rightarrow Y \in \Sigma'$, and if necessary add a minimal key to $\mathcal{D}$ to make the decomposition lossless (Lemma 2.1).

$(i) \rightarrow (ii)$ Now let $\mathcal{D} = \{R_1, \ldots, R_n\}$ be a faithful BCNF decomposition of (R, Σ), and Σ_i be the FD set associated with R_i. Since $\mathcal{D}$ is faithful, the union $\Sigma' := \bigcup \Sigma_i$ of all FDs on the subschemas forms a cover of Σ. For each FD $X \rightarrow Y \in \Sigma'$ the schema XY is a subschema of some R_i. Since R_i is in BCNF, so are all of its subschemas (in particular XY), which means $X \rightarrow Y$ is uncritical. This makes Σ' an uncritical cover of Σ as required.

$(ii) \rightarrow (iii)$ We can construct an uncritical atomic cover Σ'' of Σ', which is then a cover of Σ as well, as follows: For each FD $X \rightarrow Y \in \Sigma'$ and each $A_i \in Y \setminus X$ there exists a minimal $U_i \subseteq X$ such that $U_i \rightarrow A_i \in \Sigma^*$. Each such FD $U_i \rightarrow A_i$ is atomic, and since $U_i A_i \subseteq XY$, it is uncritical. Furthermore, $\Sigma''_{X \rightarrow Y} := \{U_i \rightarrow A_i \mid A_i \in Y \setminus X\}$ implies $X \rightarrow Y$, therefore $\Sigma'' := \bigcup\limits_{X \rightarrow Y \in \Sigma'} \Sigma''_{X \rightarrow Y}$ is a cover of Σ', and thus an uncritical atomic cover of Σ. □

Testing whether a FD $X \rightarrow A$ is critical by computing and examining all FDs in $\Sigma^*[XA]$ as the definition suggests is tedious. Instead, we use the well-known fact (e.g. [36]) that for testing BCNF it suffices to use a cover of $\Sigma^*[XA]$:

Lemma 2.25. *A schema R is in BCNF w.r.t. Σ if and only if it is in BCNF w.r.t. every cover Σ' of Σ.*

Once we have computed the atomic closure of Σ, we can easily find a cover for $\Sigma^*[S]$ using Lemma 2.22, as described in section 2.1.5. This gives us the following test whether a FD is critical:

Lemma 2.26. *A FD $X \rightarrow A$ is critical w.r.t. Σ if and only if there exists a FD $Y \rightarrow B \in \Sigma^{*a}$ with $YB \subseteq XA$ and $XA \not\subseteq Y^*$.*

Proof. $X \rightarrow A$ is critical if and only if XA is not in BCNF w.r.t. $\Sigma^*[XA]$. By Lemma 2.25 this holds for every cover of $\Sigma^*[XA]$. By Lemma 2.3 $(\Sigma^*[XA])^{*a}$ is a cover for $\Sigma^*[XA]$, and by Lemma 2.22 we have

$$(\Sigma^*[XA])^{*a} = \Sigma^{*a}[XA] = \{Y \rightarrow B \in \Sigma^{*a} \mid YB \subseteq XA\}.$$

By definition XA is not in BCNF w.r.t. $(\Sigma^*[XA])^{*a}$ if and only if $(\Sigma^*[XA])^{*a}$ contains a (non-trivial) FD $Y \rightarrow B$ for which Y is not a key, i.e., $XA \not\subseteq Y^*$. □

Guided by Theorem 2.24, we try to find an uncritical canonical cover Σ' of Σ as follows: for each $X \rightarrow A \in \Sigma^{*a}$ we check whether it is critical, and if so whether it is redundant w.r.t. the remaining FDs in Σ^{*a}. If it is redundant we discard it, otherwise we can be sure that there exists no uncritical atomic cover, and thus no faithful and lossless BCNF decomposition. If all critical FDs in Σ^{*a} are redundant, i.e., the uncritical FDs in Σ^{*a} form a cover of Σ, we can create Σ' from the remaining FDs by eliminating redundant ones.

We then apply the "3NF Synthesis" algorithm from section 1.2 using this cover, i.e., our decomposition will consist of those subschemas XA where $X \rightarrow A \in \Sigma'$, plus a minimal key if needed. Thus, even if no uncritical cover is found, we obtain a decomposition into 3NF [10].

Algorithm "least critical cover synthesis"

```
INPUT: set of FDs $\Sigma$ on schema $R$
OUTPUT: decomposition $\mathcal{D}$ of $R$

$\mathcal{D} := \emptyset$
compute $\Sigma^{*a}$ using linear resolution
$\Sigma' := \Sigma^{*a}$
for all $X \to A \in \Sigma'$ do
    if critical($X \to A, \Sigma^{*a}$) then
        if $X \to A \in (\Sigma' \setminus \{X \to A\})^*$ then
            remove $X \to A$ from $\Sigma'$
end
for all $X \to A \in \Sigma'$ do
    if $X \to A \in (\Sigma' \setminus \{X \to A\})^*$ then
        remove $X \to A$ from $\Sigma'$
    else
        add schema $XA$ to $\mathcal{D}$
end
remove all schemas $R_i \in \mathcal{D}$ with $R_i \subsetneq R_j \in \mathcal{D}$
if $\mathcal{D}$ contains no key of $R$, add a minimal key

function critical($X \to A, \Sigma^{*a}$)
for all $Y \to B \in \Sigma^{*a}$ do
    if $YB \subseteq XA$ and $XA \not\subseteq Y^*$ then
        return true
end
return false
```

Note that in the algorithm given above we combined the removal of redundant uncritical FDs in Σ' and the synthesis of schemas from FDs not removed into one loop. The algorithm "3NF Synthesis" is thus reflected in the lines

```
add schema $XA$ to $\mathcal{D}$
...
if $\mathcal{D}$ contains no key of $R$, add a minimal key
```

We illustrate the algorithm by continuing our example.

Example 2.11. Consider again the schema $CLRT$ from examples 2.1, 2.2 and 2.5 with

$$\Sigma = \{C \to L, CT \to R, LT \to C, RT \to C\}$$

In example 2.5 we have already computed the atomic closure of Σ:

$$\Sigma^{*a} = \Sigma \cup \{RT \to L, LT \to R\}$$

Checking all FDs in Σ^{*a}, we find that only $LT \to C$ is critical (due to $C \to L$), and that it is redundant. Eliminating further redundant FDs leads to the uncritical canonical cover

$$\{C \to L, RT \to C, LT \to R\}$$

which gives us the faithful BCNF decomposition:

$$\{CL, CRT, LRT\}$$

□

Theorem 2.27. *The "least critical cover synthesis" algorithm computes a faithful, lossless BCNF decomposition $\mathcal{D}$ of R iff such a decomposition exists.*

Proof. Initially, the set Σ' forms a cover for Σ. During the computation, only redundant FDs are removed from Σ', thus Σ' is still a cover in the end. Each FD in Σ' is contained in some schema $R_i \in \mathcal{D}$, which makes the decomposition faithful. Since $\mathcal{D}$ contains a subschema which forms a key of R, the decomposition is lossless by Lemma 2.1.

Assume now that $\mathcal{D}$ is not in BCNF. The only schema in $\mathcal{D}$ which might not be induced by a FD from Σ' forms a minimal key, which is automatically in BCNF. Thus Σ' must contain some critical FD. But since we removed as many critical FDs as possible while maintaining a cover, *every* atomic cover of Σ must contain critical FDs. Therefore no faithful, lossless BCNF decomposition exists by Theorem 2.24. □

2.2.2 Improvements and Complexity Analysis

While the improvements which we shall present in the following reduce the runtime of our "least critical cover synthesis" algorithm considerably, they do not improve its worst-case complexity behavior. We therefore begin with a brief complexity analysis.

Computation of Σ^{*a} can be done in $O(f \cdot k^2 n^2)$ as described earlier. Pre-computing Y^* for each $Y \to B \in \Sigma^{*a}$ can be done in $O(f \cdot kn)$ using the "closure" algorithm, so that the condition

$$Y \subseteq XA \text{ and } XA \nsubseteq Y^*$$

can be tested in $O(k)$. The number of such tests is at most f^2, leading to a complexity of $O(f^2 \cdot k)$. The redundancy test

$$X \to A \in (\Sigma' \setminus \{X \to A\})^*$$

can be performed in $O(f \cdot k)$ using the "closure" algorithm. At most f such tests are performed (both loops combined), which again gives us $O(f^2 \cdot k)$ as bound. Since $\mathcal{D}$ contains at most f schemas, removal of included schemas from $\mathcal{D}$ can be performed in $O(f^2 \cdot k)$ as well. Checking whether $\mathcal{D}$ contains a key can be done in $O(f \cdot kn)$. A minimal key can be found in $O(k^2 n)$ by starting with the trivial key R and testing for each attribute whether it can be removed while maintaining a key of R. Adding these complexities up leads to an overall bound of

$$O(f \cdot k^2 n^2 + f^2 \cdot k)$$

We want to reduce the number of FDs for which we check for criticality and redundancy. Furthermore, we often do not start with a single relation schema, but rather with a decomposition from an earlier design. This decomposition is reflected by the form of the FDs given. It is usually desirable to keep the cover Σ' produced as close to the original cover Σ as possible, as this will lead to a decomposition which is similar to the original design.

We do so by maintaining a cover Σ_R, which gets initialized with an atomic cover of Σ, which (if Σ is not already atomic) we obtain by splitting FDs into singular ones and minimizing their LHSs, as described in section 2.1. We then check all FDs in Σ_R for being critical. If we find a critical one, we need to check whether it is redundant, and if so, substitute it for one or more non-critical ones from Σ^{*a} (or from a "suitable" subset $\Sigma' \subseteq \Sigma^{*a}$).

These two tasks can be combined: when checking whether $X \to A \in \Sigma_R$ is redundant by computing the closure X^* (up to the point where A is added), we check any FD to be used for criticality (discarding FDs tested critical to avoid double-testing) and use only non-critical ones. If $X \to A$ turns out to be redundant, we replace it by the FDs used in computing the closure.

Algorithm "substitute FD"

```
INPUT: sets of FDs $\Sigma_R, \Sigma' \subseteq \Sigma^{*a}$, FD $X \to A \in \Sigma_R$
OUTPUT: modified $\Sigma_R, \Sigma'$: $X \to A$ substituted by uncritical set of FDs $Subst \subseteq \Sigma'$
(if possible), $Subst$ and critical FDs get removed from $\Sigma'$

$X^* := X, Subst := \emptyset$
while $A \notin X^*$ do
    if $\exists Y \to B \in \Sigma_R \setminus \{X \to A\}$ with $Y \subseteq X^*, B \notin X^*$ then
      add $B$ to $X^*$
    else
      if $\exists Y \to B \in \Sigma'$ with $Y \subseteq X^*, B \notin X^*$ then
        if $Y \to B$ critical then
          remove $Y \to B$ from $\Sigma'$
        else
          add $B$ to $X^*$
          add $Y \to B$ to $Subst$
      else
        return "$X \to A$ not redundant in $\Sigma_R$"
end
$\Sigma_R := (\Sigma_R \setminus \{X \to A\}) \cup Subst$
$\Sigma' := \Sigma' \setminus Subst$
```

Note that the data structure described in [5] to speed up the "closure" algorithm can also be used to quickly perform the check

$$\exists Y \to B \in \Sigma_R \setminus \{X \to A\} \text{ with } Y \subseteq X^*, B \notin X^*$$

and is thus a desirable implementation.

The remaining task is the same as before: We remove redundant FDs and create schemas to hold the remaining ones. This gives us the following algorithm, where "substitute(...)" denotes a call of algorithm "substitute FD", and "critical" is the function defined in algorithm "least critical cover synthesis".

Algorithm “least critical cover synthesis” revised

```
INPUT: set of atomic FDs $\Sigma$ on schema $R$
OUTPUT: decomposition $\mathcal{D}$ of $R$

$\Sigma_R := \Sigma$
compute $\Sigma' := \Sigma^{*a} \setminus \Sigma_R$
for all $X \to A \in \Sigma_R$ do
    if critical($X \to A$) then
       substitute($\Sigma_R, \Sigma', X \to A$)
end
for all $X \to A \in \Sigma_R$ do
    if $X \to A \in (\Sigma_R \setminus \{X \to A\})^*$ then
       remove $X \to A$ from $\Sigma_R$
    else
       add schema $XA$ to $\mathcal{D}$
end
remove all schemas $R_i \in \mathcal{D}$ with $R_i \subsetneq R_j \in \mathcal{D}$
if $\mathcal{D}$ contains no key of $R$, add a minimal key
```

Having constructed $\mathcal{D}$, we can attempt to merge schemas, meaning that we replace two schemas $R_i, R_j \in \mathcal{D}$ by $R_i \cup R_j$, if this is desired (e.g. to avoid unnecessary duplication of attributes). To ensure that we still get a faithful and lossless BCNF decomposition, we only need to check whether $R_i \cup R_j$ is in BCNF, since merging schemas clearly preserves FDs and keeps the decomposition lossless. This check can be done using Lemma 2.22.

Note that we only need to check whether R_i and R_j can be merged if they have the same closure, i.e., if $R_i^* = R_j^*$, since otherwise we can be sure that $R_i \cup R_j$ is not in BCNF.

2.2.3 Partial Determination Cycles

Sometimes the atomic closure is large, although it is easy to see that a faithful BCNF decomposition exists.

Example 2.12. Consider the set of FDs

$$\Sigma = \{A_1 \to B_1, \ldots, A_n \to B_n, B_1 \ldots B_n \to C\}$$

Clearly Σ is uncritical, but Σ^{*a} contains $2^n + n$ elements. The $2^n - 1$ elements of Σ^{*a} not already contained in Σ can be obtained by substituting any number of attributes B_i in $B_1 \ldots B_n \to C$ with the corresponding A_i. □

In such cases, it would be nice if we could quickly decide that Σ is already uncritical and not compute the atomic closure. At other times we would not really need all of Σ^{*a} in order to decide whether a faithful BCNF decomposition exists.

Example 2.13. Let Σ be as in example 2.12, and Σ' be any set of FDs on attributes not appearing in Σ. Then $\Sigma \cup \Sigma'$ is critical iff Σ' is critical, and $\Sigma \cup \Sigma'$ has an uncritical cover iff Σ' has an uncritical cover. □

In the following we will develop some criteria that sometimes allow us to avoid computing all of Σ^{*a}. The basic idea is the following: If an atomic FD $X \rightarrow A$ is critical, it must contain some other atomic FD $Y \rightarrow B$, with $A \in Y \subset XA, B \in X$. Thus there is a cycle between A and B in the sense that A is used to determine B, and vice versa. If an atomic FD participates in no such cycle, then it is never contained in another atomic FD, and thus not needed for checking criticality. We will show that it will not be needed to substitute a critical FD either, so that we can avoid computing FDs with this property entirely.

A similar approach was taken by Majster-Cederbaum in [34], where the complete absence of cycles was used as a sufficient but not necessary condition for the existence of a faithful BCNF decomposition. This condition can be checked in polynomial time.

Definition 2.28. Let Σ be a set of FDs and A, B be attributes. We say that A *partially determines* B, written $A \xrightarrow{p} B$, if Σ contains a FD $X \rightarrow Y$ with $A \in X, B \in Y$. We denote the "partially determines" relation w.r.t. Σ by $\mathcal{R}_\Sigma$.

As relations can be regarded as directed graphs, we use terminology from graph theory. For us a *cycle* is a directed path which starts and ends at the same vertex. An important point to note is that we allow vertices and arcs to appear multiple times in a cycle. The reason for using such a general notion of cycle may not be obvious, since critical atomic FDs always participate in a cycle which visits vertices and arcs only once (even stronger: a cycle of length two). However, it turns out that it is easier to check whether two vertices participate in a general cycle, because this property is transitive: If A, B lie in a cycle, and B, C lie in a different cycle, then A, C also lie in some cycle.

Lemma 2.29. *Let Σ be a set of atomic FDs, and $Y \rightarrow B, X \rightarrow A \in \Sigma$ be different. If $YB \subseteq XA$, then A and B participate in a cycle in $\mathcal{R}_\Sigma$.*

Proof. Assume $B \notin X$ and thus $B = A$. Then $Y \subseteq X$, and since $Y \rightarrow B$ and $X \rightarrow A$ are different, $Y \subsetneq X$. But this is not possible since $Y \rightarrow B$ and $X \rightarrow A$ are both atomic.

Assume now $A \notin Y$, and thus $Y \subseteq X \setminus B$. This would imply $(X \setminus B) \rightarrow A \in \Sigma^*$, which is a contradiction since $X \rightarrow A$ is atomic.

We therefore have $B \in X$ and $A \in Y$, and thus $B \xrightarrow{p} A$ and $A \xrightarrow{p} B$, which is a cycle in $\mathcal{R}_\Sigma$. □

In Lemma 2.29 we just show that A and B partially determine each other w.r.t. Σ, and it depends on the choice of cover whether A partially determines B or not, even if we only consider atomic covers.

Example 2.14. Consider the set of FDs $\Sigma = \{A \rightarrow B, B \rightarrow C\}$. Then A does not partially determine C w.r.t. Σ, but it does w.r.t. $\Sigma^{*a} = \Sigma \cup \{A \rightarrow C\}$. □

However, the transitive closure of the "partially determines" relation $\xrightarrow{p}$ is independent of the choice of atomic cover, as we show next.

Theorem 2.30. *Let Σ be a set of atomic FDs, and let $\mathcal{R}_\Sigma$ and $\mathcal{R}_{\Sigma^{*a}}$ be the "partially determines" relations w.r.t. Σ and Σ^{*a}. Then the transitive closures $\mathcal{R}^+_\Sigma$ and $\mathcal{R}^+_{\Sigma^{*a}}$ of $\mathcal{R}_\Sigma$ and $\mathcal{R}_{\Sigma^{*a}}$ are identical.*

Proof. By Theorem 2.5 every atomic FD in Σ^{*a} can be derived from Σ using the resolution rule (2.1):

$$\frac{X \to A \quad AY \to B}{XY \to B}$$

Let G be the set of all FDs derivable this way. Since $\Sigma \subseteq \Sigma^{*a} \subseteq G$ and thus

$$\mathcal{R}^+_\Sigma \subseteq \mathcal{R}^+_{\Sigma^{*a}} \subseteq \mathcal{R}^+_G$$

it suffices to show that $\mathcal{R}^+_G$ is included in $\mathcal{R}^+_\Sigma$. For this we only need to show that the derived FD $XY \to B$ does not add to $\mathcal{R}^+_G$. The only partial determinations that are added by $XY \to B$ though are of the form $C \xrightarrow{p} B$ with $C \in XY$. If $C \in X$ then $C \xrightarrow{p} A, A \xrightarrow{p} B \in \mathcal{R}_G$ and thus $C \xrightarrow{p} B$ was already contained in $\mathcal{R}^+_G$. If $C \in Y$ then $C \xrightarrow{p} B$ was already contained in $\mathcal{R}_G$. □

Note that, since all covers of Σ have the same atomic closure, Theorem 2.30 shows that all atomic covers of Σ generate the same "partially determines" relation after taking the transitive closure. This does not hold for non-atomic covers Σ', which can add arbitrary attributes to the LHSs of FDs and thus generate arbitrary extra arcs in $\mathcal{R}_{\Sigma'}$.

Most importantly, Theorem 2.30 assures us that, instead of testing whether an attribute participates in any cycle in $\mathcal{R}_{\Sigma^{*a}}$, we only need to test whether it participates in a cycle in $\mathcal{R}_\Sigma$. This is because taking the transitive closure does not affect whether an attribute participates in a cycle.

Definition 2.31. In the following we call attributes that participate in some cycle in $\mathcal{R}_\Sigma$ *cyclic*. We call a FD *cyclic* if at least one of its LHS attributes participates in a cycle with one of its RHS attributes. An attribute or FD that is not cyclic is called *acyclic*.

We use the atomic closure for two purposes. The first use is to test whether a FD is critical. We have seen that we only need cyclic FDs for this task, since acyclic atomic FDs are never contained in other atomic FDs. The second use is to replace critical FDs. To avoid computing acyclic FDs, we need to show that we can replace critical FDs without them.

Definition 2.32. The *cyclic closure* Σ^{*c} of a set Σ of FDs is defined as

$$\Sigma^{*c} := \{X \to A \in \Sigma^{*a} \mid X \to A \text{ is cyclic}\}.$$

Note that taking the cyclic closure is not a closure operation in the mathematical sense. We chose the name since it contains the cyclic FDs in the atomic closure, which in turn is not actually a closure either, but rather contains the atomic elements in the closure.

Lemma 2.33. *Let $X \to A \in \Sigma^{*a}$ be critical and implied by a minimal set $S \subset \Sigma^{*a}$. Then every FD $Y \to B \in S$ with $X^* = Y^*$ is cyclic.*

Proof. Since S is minimal, B is used in deriving $X \to A$, and thus B precedes A in $\mathcal{R}^+_\Sigma$. By Lemma 2.29 every critical FD is cyclic, in particular $X \to A$, so there exists an attribute $C \in X$ which is preceded by A, and thus by B. Either $C \in Y$, or, as $X \to Y$ is LHS-minimal, C is used to derive some attribute $C' \in Y$ and thus precedes it. In the latter case B precedes C'. Since Y is atomic, C or C' precedes B as well, so B participates in a cycle with C or C', which makes $Y \to B$ cyclic. □

The above lemma does not allow us to replace critical FDs, which are implied by an uncritical subset of Σ^{*a}, with uncritical FDs in Σ^{*c}. This is indeed not always possible:

Example 2.15. Consider the set of atomic FDs

$$\Sigma = \left\{ \begin{array}{c} ABC \to D, D \to C, \\ ABC \to F, EF \to D, \\ AB \to G, G \to B, G \to E \end{array} \right\}$$

with the atomic closure

$$\Sigma^{*a} = \Sigma \cup \left\{ \begin{array}{c} ABF \to C, ABF \to D, ABD \to F, \\ AGC \to D, AGC \to F, AGD \to F, \\ AB \to E, EF \to C, GF \to C, GF \to D \end{array} \right\}$$

Note that Σ does not possess an uncritical cover: This is because the critical FD $AB \to G$ cannot be replaced. While this means that we cannot get a faithful BCNF decomposition, we could still try reduces the number of schemas which violate BCNF by replacing the FD $ABC \to D$, which is also critical.

It is easy to see that $ABC \to D$ is implied by the uncritical FDs

$$S = \{ABC \to F, AB \to E, EF \to D\} \subseteq \Sigma^{*a}$$

Indeed all the FDs in S are needed, i.e., every uncritical set $S' \subseteq \Sigma^{*a}$ that implies $ABC \to D$ is a superset of S. However, the FD $AB \to E$ is not contained in

$$\Sigma^{*c} = \Sigma^{*a} \setminus \{G \to E, AB \to E\}$$

which we can find by computing the strongly connected components of $\mathcal{R}_\Sigma$:

$$SCC(\mathcal{R}_\Sigma) = \{A, BG, CDF, E\}$$

This means that $ABC \to D$ is not implied by any uncritical subset of $\Sigma \cup \Sigma^{*c}$. However, $ABC \to D$ is implied by, and can be replaced with

$$\{ABC \to F, AB \to G, G \to E, EF \to D\} \subset \Sigma \cup \Sigma^{*c}$$

in which only $AB \to G$ is critical. Since we need to keep $AB \to G$ anyway, this replacement is just as good as a replacement with uncritical FDs. $\square$

We generalize this idea next.

Definition 2.34. For a set $G \subseteq \Sigma^{*a}$ of FDs we denote by

$$crit(G) := \{X \to A \in G \mid X \to A \text{ is critical}\}$$

the set of all critical FDs in G. Similarly

$$uncrit(G) := G \setminus crit(G)$$

Note again that critical FDs are always cyclic by Lemma 2.29.

Lemma 2.35. *Let Σ be a minimal set of FDs such that $\Sigma \models X \to Y$. Then for all $S \to T \in \Sigma$ we have $\Sigma \models X \to S$.*

Proof. We have $\Sigma \models X \to Y$ iff $Y \subseteq X^*$, and X^* can be constructed using only FDs $S \to T \in \Sigma$ with $S \subseteq X^*$. Since Σ is minimal, all FDs in Σ must be used in the construction. Clearly $S \subseteq X^*$ is equivalent to $\Sigma \models X \to S$. □

Theorem 2.36. *Let Σ be a set of atomic FDs and $G \subseteq \Sigma^{*a}$ be a cover for Σ. Then there exists a cover $H \subseteq \Sigma \cup \Sigma^{*c}$ of Σ with $crit(G) = crit(H)$.*

Proof. Choose H maximal as

$$H := uncrit(\Sigma \cup \Sigma^{*c}) \cup crit(G)$$

By Lemma 2.29 all critical FDs are cyclic, and thus $H \subseteq \Sigma \cup \Sigma^{*c}$. The equality $crit(G) = crit(H)$ holds by definition, so we only need to show that H is a cover for Σ.

Clearly $\Sigma \models H$ and $H \models uncrit(\Sigma)$, so let $X \to A$ be a minimal (w.r.t. LHS-implication) critical FD in Σ which is not implied by H. Since G is a cover of Σ, there exist a minimal subset $S \subseteq G$ with $S \models X \to A$. We show that H implies S.

By Lemma 2.33 every FD $Y \to B \in S$ is either cyclic or, since $X \to Y \in \Sigma^*$ by Lemma 2.35, smaller than $X \to A$ w.r.t. LHS-implication. In the former case, $Y \to B$ is contained in H and thus implied. In the latter case it is implied by a minimal subset $T \subseteq \Sigma$ which, again by Lemma 2.35, contains only FDs smaller that $X \to A$. Since we chose $X \to A \in \Sigma$ minimal, all FDs in T are implied by H. □

If we take the set of critical FDs as a measure of how "good" a cover is for BCNF decomposition, the above theorem assures us that we can always find an equally good cover by restricting our search to $\Sigma \cup \Sigma^{*c}$ rather than Σ^{*a}. In particular we get the following:

Corollary 2.37. *Σ has an uncritical cover (and thus a faithful BCNF decomposition) iff it has an uncritical cover in $\Sigma \cup \Sigma^{*c}$.*

The question remaining is whether we can compute the set Σ^{*c} efficiently without computing all of Σ^{*a}. This is possible.

Lemma 2.38. *Let $X \to A, AY \to B \in \Sigma^{*a}$. If $AY \to B$ is acyclic then so is the FD $XY \to B \in \Sigma^*$, derived by the resolution rule (2.1):*

$$\frac{X \to A \quad AY \to B}{XY \to B}$$

Proof. Let $XY \to B$ be cyclic. Then $B \xrightarrow{p} C$ lies in $\mathcal{R}_\Sigma^+$ for some $C \in XY$. If $C \in Y$ then $AY \to B$ is clearly cyclic. If $C \in X$ then $C \xrightarrow{p} A$ and $A \xrightarrow{p} B$, and again $AY \to B$ is cyclic. □

Since LHS-minimization does not make an acyclic FD cyclic, all FDs derived from an acyclic FD by linear resolution (with LHS-minimization) will be acyclic as well. Thus we need not use any acyclic FD as base FD and can safely discard it instead (note though that acyclic FDs in Σ are still needed as substituting FDs).

Testing whether a FD is cyclic can be done efficiently by pre-computing the maximal strongly connected components (MSCs) of $\mathcal{R}_\Sigma$, which only requires linear time [41]. The time complexity for computing Σ^{*c} with an adapted "linear resolution" algorithm is thus

$$O\left(f_c \cdot k^2 n^2\right)$$

where f_c is the number of FDs in Σ^{*c}. Similarly, the time complexity for the "least critical cover synthesis" algorithm using Σ^{*c} instead of Σ^{*a} becomes

$$O\left(f_c \cdot k^2 n^2 + f_c^2 \cdot k\right)$$

In conclusion we observe that restricting our computations to cyclic FDs can lead to great speed improvements in cases where Σ^{*c} is much smaller than Σ^{*a}, while generating only a small computational overhead (for testing cyclicity) when all or most FDs in Σ^{*a} are cyclic.

2.2.4 Related Work

As mentioned before, the algorithm for finding faithful BCNF decompositions in section 2.2.1 is an improvement of an algorithm by Osborn [38].

Kandzia and Mangelmann present an algorithm which always finds faithful BCNF decompositions in [23]. Their approach extends that of Osborn [38] by introducing extra attributes and FDs to generate a faithful BCNF decomposition in cases where none exists normally. While the introduction of extra attributes for the sake of obtaining a BCNF decomposition may not always be desirable (indeed it is counter productive to our work in chapter 4), it is fully compatible with our improvements of Osborn's approach.

Gottlob, Pichler and Wei describe an algorithm for testing whether a subschema is in BCNF (or 3NF) in [20], and show that its complexity is linear for sets Σ of bounded treewidth.

Our work on partial determination cycles extends the work of Majster-Cederbaum [34]. Partial determination cycles were also used by Saiedian and Spencer in [39], though for different purposes.

2.3 Complex-Valued Databases

We extend our approach for normalization to complex-valued database schemas which are constructed using record, list, set and multiset constructors.

2.3.1 Introduction

A number of complex-valued data models have been suggested. We will follow the approach of Link in [29], since it focuses on the main data structures and avoids unnecessary complications which do not enrich the model.

Instead of dealing with relation schemas, nested attributes are introduced as extensions of flat attributes (which are the same as attributes in the relational model) using a number of type constructors.

Definition 2.39. [29, Def. 2.12] A *universe* is a finite set $\mathcal{U}$ together with domains (i.e., sets of values) $dom(A)$ for all $A \in \mathcal{U}$. The elements of $\mathcal{U}$ are called *flat attributes.*

Definition 2.40. [29, Def. 2.13] Let $\mathcal{U}$ be a universe and $\mathcal{L}$ a set of labels. The set $\mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ of *nested attributes over* $\mathcal{U}$ *and* $\mathcal{L}$ is the smallest set satisfying the following conditions:

1. $\lambda \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$,

2. $\mathcal{U} \subseteq \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$,

3. for $L \in \mathcal{L}$ and $N_1, \dots, N_k \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ with $k \geq 1$ we have $L(N_1, \dots, N_k) \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$,

4. for $L \in \mathcal{L}$ and $N \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ we have $L[N] \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$,

5. for $L \in \mathcal{L}$ and $N \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ we have $L\{N\} \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$,

6. for $L \in \mathcal{L}$ and $N \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ we have $L\langle N\rangle \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$.

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*.

The domains of nested attributes are defined as one would expect.

Definition 2.41. [29, Def. 2.14] For a nested attribute $N \in \mathcal{N}\mathrm{A}$ we define the *domain* $dom(N)$ as follows:

1. $dom(\lambda) = \{ok\}$,

2. $dom(A)$ as above 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 $dom(N)$.

Note that a relation schema $R = \{A_1, \dots, A_n\}$ is captured by the record-valued attribute $R(A_1, \dots, A_n)$ with label R, i.e., by a single application of the record constructor. Instead of relation schemata R we will now consider a nested attribute N. An R-relation r is then replaced by some set $r \subseteq dom(N)$.

In the relational model we had to deal with subschemas. In the complex valued approach we obtain subattributes instead.

Definition 2.42. [29, Def. 2.15] The *subattribute relation* $\leq$ on the set of nested attributes $\mathcal{N}\mathrm{A}$ 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 $N \in \mathcal{N}\mathrm{A}$,

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 $N \in \mathcal{N}\mathrm{A}$,

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 $N, M \in \mathcal{N}\mathrm{A}$ 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.

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 $R(A, B, C)$.

In the following we will abbreviate subattributes by dropping labels, e.g. the nested attribute $R(A, ST(B, C))$ will be abbreviated as $(A, (B, C))$.

Definition 2.43. [29, Def. 2.17] Let $N, M \in \mathcal{N}\mathrm{A}$ with $M \leq N$. The *projection function* $\pi^N_M : dom(N) \to dom(M)$ is defined as follows:

1. if $N = M$, then $\pi^N_M = id_{dom(N)}$ is the identity on $dom(N)$,
2. if $M = \lambda$, then $\pi^N_\lambda : dom(N) \to \{ok\}$ 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 $\pi^N_M = \pi^{N_1}_{M_1} \times \cdots \times \pi^{N_k}_{M_k}$ which maps every tuple $(v_1, \dots, v_k) \in dom(N)$ to $(\pi^{N_1}_{M_1}(v_1), \dots, \pi^{N_k}_{M_k}(v_k)) \in dom(M)$,
4. if $N = L[N']$ and $M = L[M']$, then $\pi^N_M : dom(N) \to dom(M)$ maps every list $[v_1, \dots, v_n] \in dom(N)$ to the list $[\pi^{N'}_{M'}(v_1), \dots, \pi^{N'}_{M'}(v_n)] \in dom(M)$,
5. if $N = L\{N'\}$ and $M = L\{M'\}$, then $\pi^N_M : dom(N) \to dom(M)$ maps every set $S \in dom(N)$ to the set $\{\pi^{N'}_{M'}(s) \,:\, s \in S\} \in dom(M)$, and
6. if $N = L\langle N'\rangle$ and $M = L\langle M'\rangle$, then $\pi^N_M : dom(N) \to dom(M)$ maps every multiset $S \in dom(N)$ to the multiset $\langle \pi^{N'}_{M'}(s) \,:\, s \in S\rangle \in dom(M)$.

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 $X, Y \geq X \sqcap Y$

Functional Dependencies can now be defined as follows:

Definition 2.44. [29, Def. 5.1] Let $N \in \mathcal{N}\mathrm{A}$ be a nested attribute. A *functional dependency on* N is an expression of the form $\mathcal{X} \to \mathcal{Y}$ where $\mathcal{X}, \mathcal{Y} \subseteq Sub(N)$ are non-empty. A set $r \subseteq dom(N)$ *satisfies* the functional dependency $\mathcal{X} \to \mathcal{Y}$ on N, denoted by $\models_r \mathcal{X} \to \mathcal{Y}$, if and only if $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ holds for all $Y \in \mathcal{Y}$ whenever $\pi^N_X(t_1) = \pi^N_X(t_2)$ holds for all $X \in \mathcal{X}$ and any $t_1, t_2 \in r$.

Note that we cannot always replace a set of subattributes by their join.

Example 2.16. [29] 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 Tennis{Match(Winner,Loser)}:

$$\begin{aligned}\{\ \ &\{(\text{Becker, Agassi}), (\text{Stich, McEnroe})\},\\ &\{(\text{Becker, McEnroe}), (\text{Stich, Agassi})\}\ \}.\end{aligned}$$

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

However, $\not\models_r$ Tennis{Match(Winner)} $\rightarrow$ Tennis{Match(Winner, 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. $\square$

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

Definition 2.45. [29, Def. 5.2] Let $N \in \mathcal{N}$A. The subattributes $X, Y \in Sub(N)$ 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.

Lemmas 5.3, 5.14 and 5.16 in [29] show the following:

Lemma 2.46. *Let $N \in \mathcal{N}A$, $X, Y \in Sub(N)$. Then the following are equivalent:*

(i) X and Y are reconcilable

(ii) for all $t_1, t_2 \in dom(N)$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$ and $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ we have $\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$

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

Definition 2.47. [29, Def. 5.27] Let $N \in \mathcal{N}$A. A nested attribute $N_i \in \mathcal{N}$A 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 $\mathcal{U}(N)$.

When representing FDs, we will want to replace non-unitary attributes in FDs with their units. In [29] a characterization for the units of a nested attribute is given, and it is shown that this replacement does not change the semantics of the FDs:

Lemma 2.48. *[29, Lemma 5.28] Let $N \in \mathcal{NA}$. Then*

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

if $N = L(N_1, \ldots, N_k)$ 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.

Lemma 2.49. *[29, Lemma 5.29] Let $N \in \mathcal{NA}$. Then $N = \bigsqcup\{M \mid M \in \mathcal{U}(N)\}$ and for $N_1, N_2 \in \mathcal{U}(N)$ with $N_1 \neq N_2$ and $U \leq N_1, V \leq N_2$ follows that U and V are reconcilable.*

For a more thorough discussion of the complex-valued model introduced here see [29].

2.3.2 Representation Basis

Unlike in [29] we will be interested in all subattributes with properties 1. and 2. of Definition 2.47, not just maximal ones.

Definition 2.50. 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.

Unitary attributes can also be characterized as follows:

Corollary 2.51. *An attribute N is unitary iff the FD $N^{\downarrow} \setminus N \to N$ is non-trivial.*

Proof. If N is unitary then by definition its subattributes cannot be reconciled to N, which makes $N^{\downarrow} \setminus N \to N$ non-trivial. If N is not unitary, then $U(N) \subseteq N^{\downarrow} \setminus N$, and by Lemma 2.49 N is the join of its reconcilable units, making $N^{\downarrow} \setminus N \to N$ trivial. □

We will use unitary attributes for representing FDs. However, not all of them will actually be needed for this task.

Definition 2.52. We define the *representation basis* (RB) $rb(N)$ of a nested attribute N as the set of its units, and the RB of a FD $X \to Y$ is the union of the RBs of all attributes appearing in X or Y, while $rb(\Sigma)$ of a set Σ of FDs is the union of the RBs of all FDs in Σ.

Example 2.17. Let $\Sigma = \{\{(A, \{(B, C)\})\} \to \{(D)\}, \{(\{(B)\}, D)\} \to \{(A)\}\}$ be a set of FDs over the nested attribute $(A, \{(B, C)\}, D)$. Then

$$\begin{aligned} rb((A, \{(B, C)\})) &= \{(A), (\{(B, C)\})\} \\ rb(\{(A, \{(B, C)\})\} \to \{D\}) &= \{(A), (\{(B, C)\}), (D)\} \\ rb(\Sigma) &= \{(A), (\{(B, C)\}), (D), (\{(B)\})\} \end{aligned}$$

□

Clearly an attribute can be represented equivalently by its representation basis. The benefit of this representation is that we can represent FDs derived from Σ using only the representation basis of Σ, rather than the possibly exponential set of all unitary subattributes. While this does not hold for all FDs implied by Σ, it does for all FDs that we will be interested in (i.e., for all FDs occurring during our derivation process).

Definition 2.53. We call an attribute A *RB-representable* if $rb(A) \subseteq RB$. Similar for FDs or sets of attributes/FDs.

We will want to talk about the representation basis of the left- and right hand sides of FDs in Σ separately.

Definition 2.54. Let Σ be a set of FDs. We define $lhs(\Sigma) := \bigcup_{X \to Y \in \Sigma} X$ and similarly $rhs(\Sigma) := \bigcup_{X \to Y \in \Sigma} Y$, and say that Σ is $RB_L \to RB_R$*-representable* if $lhs(\Sigma)$ is RB_L-representable and $rhs(\Sigma)$ is RB_R-representable.

Definition 2.55. We call a FD $X \to Y$ in *standard form* if X and Y contain only unitary elements. We call a $X \to Y$ *singular* if it is in standard form and Y contains only a single element.

Clearly every FD can be replaced by an equivalent FD in standard form, or an equivalent set of singular FDs. For ease of reading, we will not always distinguish between an attribute and the singleton set containing that attribute, and leave out brackets and commas where this does not cause ambiguities. Thus we write the set Σ from example 2.17 shorter as $\Sigma = \{A\{BC\} \to D, \{B\}D \to A\}$.

Definition 2.56. Let X be a set of unitary attributes. We define the *basis* $\lceil X \rceil$ of X as the maximal elements in X. Conversely, we define the *completion* $A^{\downarrow RB}$ of an attribute A w.r.t. a set RB of unitary attributes as the set of all subattributes of A in RB, and the completion $X^{\downarrow RB}$ of X as the union of the completions of the attributes in X.

Definition 2.57. A non-trivial FD $X \to Y \in \Sigma^*$ is *atomic* w.r.t. a set Σ of FDs if

(i) $X \to Y$ is singular

(ii) X is minimal with $X \to Y \in \Sigma^*$, i.e. there is no $Z \to Y \in \Sigma^*$ with $Z \subsetneq X$ or $Z^{\downarrow} \subsetneq X^{\downarrow}$

We call $X \to Y$ *maximal atomic* if the following holds as well:

(iii) $A \in Y$ is maximal among unitary attributes with $X \to A \in \Sigma^*$

The definition of minimality for X in (ii) says that we cannot weaken X, neither by removing attributes from it nor by replacing an attribute by one or more of its subattributes. Thus for the set $\Sigma = \{A\{BC\} \to D, \{B\}D \to A\}$ from example 2.17, the FD $\{BC\}D \to A$ would not be atomic.

2.3.3 Linear Resolution

We can now derive atomic FDs from Σ using linear resolution by modifying the resolution rule to cope with the requirements of nested attributes:

$$\frac{X \to A \quad Y \to B}{\lceil X(Y \setminus A^{\downarrow})\rceil \to B} \qquad \text{(NA-Resolution)}$$

Similar to resolution in the relational model, we only need to apply the NA-Resolution rule when $Y \cap A^{\downarrow} \neq \emptyset$ and $B \notin X^{\downarrow}$.

Theorem 2.58. *Let Σ be singular. Then all maximal atomic FDs in Σ^{*a} can be derived from Σ using linear NA-Resolution.*

Proof. Let $X \to A$ be maximal atomic. We consider again the closure computation for X. Since A is maximal and unitary, the closure computation must use a FD $Y \to A \in \Sigma$. The remaining argument proceeds as in the proof for the relational case (Theorem 2.5). $\square$

Corollary 2.59. *The following axiom system is sound and complete for singular FDs:*

$$\frac{X \to A \quad Y \to B}{\lceil X(Y \setminus A^{\downarrow})\rceil \to B}, \quad \frac{}{XB \to A} A \leq B$$

Proof. As for the relational case (Corollary 2.9), but based on Theorem 2.58. $\square$

We denote the set of all maximal atomic FDs in Σ^{*a} as Σ^{*max}. Clearly Σ^{*max} forms a cover of Σ.

Corollary 2.60. *Σ^{*max} is $rb(lhs(\Sigma)) \to rb(rhs(\Sigma))$-representable.*

Proof. If Σ contains non-singular FDs we can replace them by equivalent singular ones without changing the representation basis, so we may assume that Σ is singular. Consider the NA-resolution rule: if $X \to A$ and $Y \to B$ are $rb(lhs(\Sigma)) \to rb(rhs(\Sigma))$-representable, so is $\lceil X(Y \setminus completionB)\rceil \to A$. Thus all FDs derivable from Σ using NA-resolution are $rb(lhs(\Sigma)) \to rb(rhs(\Sigma))$-representable, in particular all maximal atomic ones. $\square$

As before, we wish to speed up the computation by keeping only maximal atomic FDs for later resolution steps. Condition (i) automatically holds for any derived FD $X' \to B$.

If condition (iii) is violated, i.e., if there exists a unitary $B' > B$ with $X' \to B' \in \Sigma^*$, we can discard $X' \to B$. This is because for any FD $X'' \to B$ derivable from $X' \to B$ we can derive $X'' \to B'$ from $X' \to B'$, so $X'' \to B$ is not maximal atomic.

We test condition (ii) by removing attributes from X and checking if the new reduced LHS still determines B. However, unlike in the relational model where the removal of a single attribute was the smallest reduction possible, we now need to replace the removed attribute A by all the unitary true sub-attributes of A. The reduction of X now takes the form

$$X_{reduced} := (X \cup A^{\downarrow}) \setminus A$$

which requires us to add the completion $A^{\downarrow}$ of A to X. In this we can restrict ourselves to the completion w.r.t. $rb(lhs(\Sigma))$, since all maximal atomic FDs are $rb(lhs(\Sigma)) \to rb(rhs(\Sigma))$-representable.

If the reduced LHS still determines B, we compute its basis and replace X by it, then iterate the process. This gives us the following algorithm:

Algorithm "linear NA-resolution"

```
INPUT: set of FDs $\Sigma$
OUTPUT: maximal atomic closure $\Sigma^{*max}$

compute a maximal atomic cover $\Sigma'$ of $\Sigma$
$\Sigma^{*max} := \Sigma'$
for all $Y \to B \in \Sigma^{*max}$ do
    for all $X \to A \in \Sigma'$ with $A \in Y, B \notin X$ do
        // derive $\lceil X(Y \setminus A^{\downarrow}) \rceil \to B$ by rule (NA-Resolution)
        $X' := \lceil X(Y \setminus A^{\downarrow}) \rceil$
        if $X' \to B$ is maximal then
            $\Sigma^{*max} := \Sigma^{*max} \cup \{\text{lhs-minimize}(X' \to B, \Sigma)\}$
    end
end

function lhs-minimize($X \to B, \Sigma$)
$U := X$
for all $A \in U$ do
    $U' := \lceil (U \cup A^{\downarrow rb(lhs(\Sigma))}) \setminus A \rceil$
    if $U' \to B \in \Sigma^*$ then
        $U := U'$
end
return $U \to B$
```

We have just described an algorithm for constructing the set Σ^{*max} of all maximal atomic FDs in Σ^{*a}. However, as the following example shows, we cannot always find an uncritical cover of maximal atomic FDs whenever an uncritical cover exists.

Example 2.18. Consider the nested attribute $A\{BC\}$ with the set of FDs

$$\Sigma = \{A \to \{BC\}, \{B\} \to \{BC\}\}$$

Σ already contains all maximal atomic FDs in Σ^{*a}, but $A \to \{BC\}$ is critical. Σ does however have the uncritical atomic cover:

$$G = \{A \to \{B\}, \{B\} \to \{BC\}\}$$

□

This means that we cannot restrict ourselves to maximal atomic FDs only when trying to find an uncritical cover. It is simple to derive Σ^{*a} from Σ^{*max}: for every $X \to A \in \Sigma^{*max}$ and every unitary $A' < A$ check whether $X \to A'$ is atomic (i.e., whether X is minimal) and if so add it to Σ^{*a}. The problem with this approach however, is that the number of unitary subattributes A' of A can be exponential in the size of A:

Example 2.19. Consider the nested attribute $A\{B_1 \dots B_n\}$ with

$$\Sigma = \Sigma^{*max} = \{A \to \{B_1 \dots B_n\}\}$$

Then $\{B_1 \dots B_n\}$ has 2^n unitary subattributes, and Σ^{*a} contains 2^n atomic FDs. □

Again we solve this problem by restricting ourselves to subattributes which appear in the representation basis of Σ. This is sufficient, since we will show later that whenever Σ has an uncritical cover, it has an uncritical canonical cover representable in $rb(\Sigma)$.

Lemma 2.61. *Let Σ be a set of FDs and A a unitary attribute with $A \notin rb(lhs(\Sigma))$. Then for every set X of unitary attributes we have*

$$XA^* = \left(XA^{\downarrow} \setminus A\right)^* \cup A$$

Proof. Let $Y \to Z \in \Sigma$ with

$$Y \leq \left(XA^{\downarrow} \setminus A\right)^* \cup A$$

Since A does not appear in the LHS of any FD in Σ, we have

$$Y \leq \left(XA^{\downarrow} \setminus A\right)^*$$

As $\left(XA^{\downarrow} \setminus A\right)^*$ is closed under Σ we also get

$$Z \leq \left(XA^{\downarrow} \setminus A\right)^*$$

Since this holds for all $Y \to Z \in \Sigma$, the set $\left(XA^{\downarrow} \setminus A\right)^* \cup A$ is closed under Σ, which gives us

$$XA^* = \left(\left(XA^{\downarrow} \setminus A\right)^* \cup A\right)^* = \left(XA^{\downarrow} \setminus A\right)^* \cup A$$

$\square$

Lemma 2.62. *Let Σ be a set of FDs and $X \to A \in \Sigma^{*a}$ be atomic. Then X is $rb(lhs(\Sigma))$-representable.*

Proof. Let $B \in X$ be any unitary attribute in X. If B does not lie in $rb(lhs(\Sigma))$ then

$$X^* = \left((XB^{\downarrow}) \setminus B\right)^* \cup B$$

by Lemma 2.61. Since $X \to A$ is atomic and thus non-trivial, $A \nleq B$, we have

$$A \in \left((XB^{\downarrow}) \setminus B\right)^*$$

which contradicts the minimality of X. Thus all $B \in X$ are $rb(lhs(\Sigma))$-representable, and therefore X. $\square$

It follows in particular that for every canonical cover G of Σ the LHS of G is $rb(lhs(\Sigma))$-representable. One might hope that the RHS of G is $rb(rhs(\Sigma))$-representable, or at least $rb(\Sigma)$-representable. This, however does not hold for all canonical covers.

Example 2.20. Consider the set of FDs

$$\Sigma = \{A \to \{BCD\}, \{B\} \to \{BCD\}\}$$

and its canonical cover

$$G = \{A \to \{BC\}, \{B\} \to \{BCD\}\}$$

The unitary attribute $\{BC\} \in rb(rhs(G))$ does not lie in $rb(\Sigma)$. $\square$

The problem in the example above comes from the fact that the RHS of $A \to \{BC\}$ was larger than it had to be. We therefore will restrict ourselves to sets of FDs with minimal RHSs.

Definition 2.63. We call a set of FDs Σ *RHS-reduced* if for every FD $X \to A \in \Sigma$ the set

$$\Sigma \backslash \{X \to A\} \cup \{X \to A^{\downarrow} \setminus A\}$$

is not a cover of Σ.

Theorem 2.64. *Let Σ be a set of FDs and G a RHS-reduced canonical cover of Σ. Then G is $rb(lhs(\Sigma)) \to rb(\Sigma)$-representable.*

Proof. By Lemma 2.62 $lhs(G)$ is $rb(lhs(\Sigma))$-representable, thus we only need to show that $rhs(G)$ is $rb(\Sigma)$-representable. Assume the contrary and let $X \to A \in G$ with $A \notin rb(\Sigma)$. Let further $H := G \setminus \{X \to A\}$ and consider the closures X^{*G} of X w.r.t. G and XA^{*H} of XA w.r.t H. Clearly

$$X^{*G} = XA^{*G} = XA^{*H}$$

But since A does not lie in $rb(lhs(H)) \subseteq rb(\Sigma)$ we have by Lemma 2.61:

$$XA^{*H} = \left(XA^{\downarrow} \setminus A\right)^{*H} \cup A$$

Again using the fact that $A \notin rb(\Sigma)$ we find that $X \to A$ is not maximal atomic by Corollary 2.60, and thus $A < A'$ for some unitary A' with $X \to A' \in \Sigma^*$. From

$$A' \in X^{*G} = \left(XA^{\downarrow} \setminus A\right)^{*H} \cup A$$

we conclude that $A' \in \left(XA^{\downarrow} \setminus A\right)^{*H}$ and thus

$$X^{*G} = \left(XA^{\downarrow} \setminus A\right)^{*H}$$

But this means that $H \cup \{X \to A^{\downarrow} \setminus A\}$ is a cover of G, which contradicts G being RHS-reduced. □

Note that in particular, all RHS-reduced canonical covers of Σ have the same representation basis, as do their LHSs. This is not true for their RHSs though, as the following example shows.

Example 2.21. The sets of FDs on the attribute $A\{BC\}$

$$\begin{aligned} \Sigma &= \{A \to \{B\}, \{B\} \to \{BC\}, \{C\} \to \{BC\}\} \\ G &= \{A \to \{C\}, \{B\} \to \{BC\}, \{C\} \to \{BC\}\} \end{aligned}$$

are RHS-reduced canonical covers of each other, but the RBs of their RHSs differ. □

2.3.4 Faithful BCNF-Decomposition

The definition for Boyce-Codd Normal Form is the same for nested attributes as in the relational model:

Definition 2.65. A subattribute $S \leq N$ (or extended subattribute, see Definition 2.66) is in Boyce-Codd Normal Form w.r.t. a set Σ of FDs on S if every non-trivial FD $X \to Y \in \Sigma$ is a key dependency, i.e., $\Sigma \vDash X \to S$.

Having computed the set of all $rb(\Sigma)$-representable atomic FDs of Σ, we now use it to decompose the nested attribute N into normal form while preserving dependencies. When decomposing a nested attribute, we must ask ourselves what to decompose into. A first candidate might be subattributes, but as the following example shows, this can be an undesirable restriction.

Example 2.22. Let $N = \{AB\}CD$ be a nested attribute with FDs $\Sigma = \{\{A\}\{B\} \to C\}$. Then N has no faithful, lossless BCNF decomposition into subattributes: the smallest subattribute on which $\{A\}\{B\} \to C$ is defined is $\{AB\}C$, which is not in BCNF. It is however possible to decompose N into

$$\{A\}\{B\}C \text{ and } \{AB\}D$$

which is lossless, faithful and in BCNF. □

It appears that what we need are not subattributes, but rather tuples of subattributes such as $(\{A\}, \{B\}, C)$. Since the order of the attributes in a tuple does not matter for design purposes, we may as well talk about sets of subattributes instead.

Definition 2.66. 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. 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. Join and meet between extended subattributes are defined according to this extended subattribute order.

The requirement that subattributes in S be unitary identifies intuitively equivalent sets such as $\{\{A\}, \{B\}, C\}$, $\{\{A\}C, \{B\}\}$, $\{\{A\}, \{B\}C\}$ and $\{\{A\}C, \{B\}C\}$. Prohibiting subattributes of another attribute in a tuple prevents obvious redundancy.

From now on, when talking about decomposition of nested attributes, we will mean decomposition into extended subattributes.

Lemma 2.67. *Every faithful decomposition of R containing an extended subattribute which forms a key of R is lossless.*

Proof. As in the relational case [10]. □

In the relational case we can use this lemma to make a decomposition lossless "for free", since every minimal key is automatically in BCNF. This does not hold when we move to nested attributes however.

Example 2.23. Let $N = \{AB\}CD$ with FDs

$$\Sigma = \{\{A\}C \to \{B\}, \{B\}C \to D, \{B\}D \to C\}$$

Then N has the minimal keys $\{AB\}C$ and $\{AB\}D$. The first key violates BCNF, the second does not. □

Definition 2.68. A set S of attributes is called *critical* (w.r.t. Σ) if $\lceil S \rceil$ is not in BCNF (w.r.t. $\Sigma^*[\lceil S \rceil]$). A FD $X \to Y \in \Sigma^*$ is called *critical* if XY is critical. A cover G of Σ is called *critical* if it contains a critical FD. An attribute set, FD or cover that is not critical is called *uncritical.*

In the relational case it has been shown (e.g. [27]) that every lossless decomposition contains a key. However, this result does not translate directly to complex-valued databases. For the moment we shall just state the result, and postpone the proof and discussion of it to the next subsection. The term *LHS-restricted* will be defined there as well.

Lemma 2.69. *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 is a key of N.*

Theorem 2.70. *Let Σ not be LHS-restricted. Then the following are equivalent:*

(i) A schema (R, Σ) has a faithful, lossless decomposition into BCNF

(ii) Σ has an uncritical cover and an uncritical key

(iii) Σ has an uncritical RHS-reduced canonical cover and an uncritical minimal key

Proof. $(ii) \to (i)$: If Σ has an uncritical cover G, then we can synthesize a faithful BCNF decomposition $\mathcal{D}$ by creating a schema $\lceil XY \rceil$ for every $X \to Y \in G$, and if necessary add an uncritical key to $\mathcal{D}$ to make the decomposition lossless (Lemma 2.67).

$(i) \to (ii)$ Now let $\mathcal{D} = \{R_1, \ldots, R_n\}$ be a faithful BCNF decomposition of (R, Σ), and Σ_i be the FD set associated with R_i. Since $\mathcal{D}$ is faithful, the union $G := \bigcup \Sigma_i$ of all FDs on the extended subattributes forms a cover of Σ. For each FD $X \to Y \in G$ the extended subattribute $\lceil XY \rceil$ is an extended subattribute of some R_i. Since R_i is in BCNF, so are all of its extended subattributes (in particular $\lceil XY \rceil$), which means $X \to Y$ is uncritical. This makes G an uncritical cover of Σ as required. By Lemma 2.69 $\mathcal{D}$ contains an uncritical key.

$(ii) \to (iii)$ We can construct an uncritical RHS-reduced canonical cover G' of G, which is then a cover of Σ as well, as follows: For each FD $X \to Y \in G$ and each $A_i \in Y$ there exists a minimal $U_i \leq X$ such that $U_i \to A_i \in \Sigma^*$. Each such FD $U_i \to A_i$ is atomic, and since $U_i A_i \leq XY$, it is uncritical. Furthermore, $G'_{X\to Y} := \{U_i \to A_i \mid A_i \in Y\}$ implies $X \to Y$, therefore $G' := \bigcup_{X \to Y \in G} G'_{X \to Y}$ is a cover of G, and thus an uncritical atomic cover of Σ. By removing redundant FDs from G' we obtain an uncritical canonical cover. Reducing the RHS of the FDs remaining in G' keeps them uncritical and preserves the cover property. Finally, every uncritical key can be reduced to a minimal key which is still uncritical. □

Thus, in order to find a faithful lossless BCNF decomposition, we now need to do two things: find an uncritical cover, and an uncritical key. To find an uncritical key, we can use linear NA-resolution to compute all minimal keys of N.

Both of these computations are necessary. From the relational case it is already clear that the existence of an uncritical key does not imply the existence of an uncritical cover. The following example shows that the existence of an uncritical cover does not imply that every or any minimal key is uncritical, and that we can have critical and uncritical canonical covers as well as critical and uncritical minimal keys.

Example 2.24. Let $N = \{AB\}CDE$ with FDs

$$\Sigma = \{\{A\}C \to \{B\}, \{B\}C \to D, \{B\}D \to C, \{B\}C \to E, E \to C\}$$

As in example 2.23, N has the minimal keys $\{AB\}C$ and $\{AB\}D$, where the first key violates BCNF, the second does not. While the FD $\{B\}C \to E$ is critical, it can be replaced by the uncritical FD $\{B\}D \to E$, which give us the uncritical cover

$$G = \{\{A\}C \to \{B\}, \{B\}C \to D, \{B\}D \to C, \{B\}D \to E, E \to C\}$$

If we modify Σ to $\Sigma' = \Sigma \cup \{\{A\}D \to \{B\}\}$ we still find an uncritical cover $G' = G \cup \{\{A\}D \to B\}$, but now both minimal keys violate BCNF. □

2.3.5 Lossless Decomposition

We will prove Lemma 2.69 and demonstrate why the requirement of having Σ not LHS-restricted is necessary. For that, we first look into the corresponding proof in the relational case (adapting the correctness proof for the chase procedure [27]).

Lemma 2.71. *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 $\mathcal{D}$ 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 R_i^*, and contains unique values for attributes outside R_i^*. Then for every FD $X \to Y \in \Sigma$, two tuples t_i, t_j are identical on X iff $X \subseteq R_i^* \cap R_j^*$. Since $R_i^* \cap R_j^*$ is closed under Σ, Y also lies in it, so $X \to Y$ holds on r. Since $\mathcal{D}$ contains no key, the tuple t does not lie in r. It does however lie in

$$r[R_1] \bowtie \ldots \bowtie r[R_n]$$

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. This assumption is indeed necessary, since the lemma does not hold if we work with finite domains. For the remainder of this subsection we will allow domains (of base attributes) to be finite, although we still require them to contain at least two elements.

Example 2.25. Let $R = ABCD$ with FDs $\Sigma = \{A \to BC, BC \to A\}$ 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 2.25). 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:

$r' =$

A	B	C	D
0	0	0	d
a_1	0	c	0
a_2	b	0	0

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:

$r' =$

A	B	C	D
0	0	0	d
1	0	0	0

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 2.71 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, they arise naturally in complex-valued databases for subattributes such as $\{\lambda\}$.

In the example above, the attribute A occurred in the LHS of a FD in Σ. It turns out that requiring that these attributes have infinite domains is sufficient.

Lemma 2.72. *Let R be a relational schema and Σ a set of FDs on R. If all attributes which occur in the LHS of a FD in Σ have infinite domains, then every lossless decomposition $\mathcal{D}$ of R contains a key of R.*

Proof. As for Lemma 2.71, except that for attributes not occurring in the LHS of a FD in Σ we do not require the t_i to have unique values, but only values different to t. □

We now extend these results to complex-valued databases.

Definition 2.73. Let N be a nested attribute and S a unitary subattribute of N. We call S *restricted* if $dom(S)$ 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 2.69, 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 [29].

Lemma 2.74. *[29, Lemma 5.10] Let $N = L\{P\} \in \mathcal{NA}$, and $\emptyset \neq \mathcal{X} \subseteq Sub(N)$ an ideal with respect to $\leq$. Then there are $t_N, t'_N \in dom(N)$ with $\pi^N_W(t_N) = \pi^N_W(t'_N)$ if and only if $W \in \mathcal{X}$.*

Lemma 2.75. *[29, Lemma 5.13] Let $N = L\langle P\rangle \in \mathcal{NA}$, and $\emptyset \neq \mathcal{X} \subseteq Sub(N)$ an ideal with respect to $\leq$. Then there are $t_N, t'_N \in dom(N)$ with $\pi^N_W(t_N) = \pi^N_W(t'_N)$ if and only if $W \in \mathcal{X}$.*

Lemma 2.76. *[29, Lemma 5.14] Let $N \in \mathcal{N}A$, and $\emptyset \neq \mathcal{X} \subseteq Sub(N)$ 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 $t_N, t'_N \in dom(N)$ with $\pi^N_W(t_N) = \pi^N_W(t'_N)$ 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. What will take that view when formulating our next lemma.

Lemma 2.77. *Let $N \in \mathcal{N}A$, and $\emptyset \neq \mathcal{X}_i \subseteq N^{\downarrow}$ for $i = 1, \dots, k$ be ideals with respect to $\leq$. Then there are $t, t_1, \dots, t_n \in dom(N)$ with the following properties (for all $W \in N^{\downarrow}$):*

(i) $\pi^N_W(t_i) = \pi^N_W(t)$ *if* $W \in \mathcal{X}_i$

(ii) $\pi^N_W(t_i) \neq \pi^N_W(t)$ *if* $W \notin \mathcal{X}_i$ *and* W *unrestricted or a unit of* N

(iii) $\pi^N_W(t_i) \neq \pi^N_W(t_j)$ *if* $W \notin \mathcal{X}_i \cap \mathcal{X}_j$ *and* W *unrestricted*

Proof. We distinguish several cases, depending on the form of N.

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 $dom(A)$ is infinite. This obviously meets conditions (i)-(iii).

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. $\pi^N_{M_j}(\mathcal{X}_i)$[1]. 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.

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 $\lambda \in \mathcal{Y}_i$ then by Lemma 2.76 there exist tuples $t_{M,i}, t'_{M,i} \in dom(M)$ 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:

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

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

$$\pi^N_W(t_i) = \pi^N_W(t) \quad \text{iff} \quad \pi^N_W(t'_{M,i}) = \pi^N_W(t_{M,i})$$

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

$$\pi^N_W(t_i) = \pi^N_W(t_j) \quad \text{iff} \quad \pi^N_W(t'_{M,i}) = \pi^N_W(t_{M,i}) \wedge \pi^N_W(t'_{M,j}) = \pi^N_W(t_{M,j})$$

from which (iii) follows.

[1]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.

For $N = \langle M \rangle$ let $t_{N,i}, t'_{N,i} \in dom(N)$ be tuple pairs with $\pi^N_W(t_{N,i}) = \pi^N_W(t'_{N,i})$ if and only if $W \in \mathcal{X}_i$, which exist by Lemma 2.75. We then define

$$\begin{aligned} t &= c \cdot t_{N,1} \cup \ldots \cup c^n \cdot t_{N,n} \\ t_i &= c \cdot t_{N,1} \cup \ldots \cup c^i \cdot t'_{N,i} \ldots \cup c^n \cdot t_{N,n} \end{aligned}$$

with $c \in \mathbf{N}$ sufficiently large, i.e., larger than the multiplicity of any element in the multisets $t_{N,1}, \ldots, t_{N,n}, t'_{N,1}, \ldots, t'_{N,n}$. 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 determin the "origin" of an element e of t or t_i from its multiplicity $mult_e(t/t_i)$: For every tuple $t, t_1, \ldots, t_n$ we have

$$\begin{aligned} mult_e(t/t_i) &= mult_e\left(c \cdot M_1 \cup \ldots \cup c^n \cdot M_n\right) \\ &= c \cdot mult_e(M_1) + \ldots + c^n \cdot mult_e(M_n) \\ &= c \cdot a_1 + \ldots + c^n \cdot a_n \end{aligned} \tag{2.5}$$

for some multisets $M_1, \ldots, M_n$ and some values $a_1, \ldots, a_n \in \{0, \ldots, c-1\}$. Given a multiplicity $mult_e(t/t_i)$ there exists only one set of values $a_1, \ldots, a_n \in \{0, \ldots, c-1\}$ such that (2.5) holds. Consequently, we can uniquely determine the multisets $M_1, \ldots, M_n$ given t or some t_i. Conditions (i)-(iii) can then be shown as in the case of lists.

What remains is the case of sets. For $N = \{M\}$ with N restricted, we use that by lemma 5.10 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 5.10 in [29]. For every index $i = 1, \ldots, n$ we define different identifying terms:

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

Note that with this definition the identifying terms $\tau^i_N(M)$ of a subattribute M with infinite domain are pairwise different.

We then create pairs of tuples $t_{N,i}, t'_{N,i} \in dom(N)$ with $\pi^N_W(t_{N,i}) = \pi^N_W(t'_{N,i})$ if and only if $W \in \mathcal{X}_i$ as in [29, Lemma 5.10], but using the corresponding identifying terms $\tau^i_N(\ldots)$. That is, we have $\mathcal{X} = \{L\{X\} : X \in \mathcal{Y}\} \cup \{\lambda\}$ for some $\mathcal{Y} \subseteq Sub(M)$, and define

$$\begin{aligned} t_{N,i} &= \{\tau^i_M(X) : X \leq M\} \\ t'_{N,i} &= \{\tau^i_M(X) : X \in \mathcal{Y}\} \end{aligned}$$

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

$$\begin{aligned} t &= t_{N,1} \cup \ldots \cup t_{N,n} \\ t_i &= t_{N,1} \cup \ldots \cup t_{N,i-1} \cup t'_{N,i} \cup t_{N,i+1} \cup \ldots \cup t_{N,n} \end{aligned}$$

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

$$\tau^i_M(V) \in \pi^N_W(t_{N,i}) \setminus \pi^N_W(t'_{N,i})$$

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, $\tau^i_M(V)$ does not lie in any set $\pi^N_W(t_{N,j})$ for $j \neq i$ either. Thus $\pi^N_W(t_i) \neq \pi^N_W(t)$, which shows (ii). Condition (iii) is proven analogous to (ii). $\square$

The following example illustrates the construction for multisets and sets.

Example 2.26. [**Multiset**] Let $N = \langle\{\{A\}\}\rangle$ and

$$\begin{aligned} \mathcal{X}_1 &= \{\lambda, \langle\lambda\rangle, \langle\{\lambda\}\rangle, \langle\{\{\lambda\}\}\rangle\} \\ \mathcal{X}_2 &= \{\lambda, \langle\lambda\rangle, \langle\{\lambda\}\rangle\} \\ \mathcal{X}_3 &= \{\lambda, \langle\lambda\rangle\} \\ \mathcal{X}_4 &= \{\lambda\} \end{aligned}$$

Lemma 2.75 might give us the following tuples:

$$\begin{array}{ll} t_{N,1} = \langle\{\{a_1\}\}\rangle, & t'_{N,1} = \langle\{\{a_2\}\}\rangle \\ t_{N,2} = \langle\{\{a_1\}\}\rangle, & t'_{N,2} = \langle\{\emptyset\}\rangle \\ t_{N,3} = \langle\{\emptyset\}\rangle, & t'_{N,3} = \langle\emptyset\rangle \\ t_{N,4} = \langle\rangle, & t'_{N,4} = \langle\emptyset\rangle \end{array}$$

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

$$\begin{aligned} t &= 2 \cdot t_{N,1} \cup 4 \cdot t_{N,2} \cup 8 \cdot t_{N,3} \cup 16 \cdot t_{N,4} \\ &= 2 \cdot \langle\{\{a_1\}\}\rangle \cup 4 \cdot \langle\{\{a_1\}\}\rangle \cup 8 \cdot \langle\{\emptyset\}\rangle \cup 16 \cdot \langle\rangle \\ &= \langle\{\{a_1\}\}^6, \{\emptyset\}^8\rangle \\ t_1 &= \langle\{\{a_2\}\}^2, \{\{a_1\}\}^4, \{\emptyset\}^8\rangle \\ t_2 &= \langle\{\{a_1\}\}^2, \{\emptyset\}^{12}\rangle \\ t_3 &= \langle\{\{a_1\}\}^6, \emptyset^8\rangle \\ t_4 &= \langle\{\{a_1\}\}^6, \{\emptyset\}^8, \emptyset^{16}\rangle \end{aligned}$$

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

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

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

In both cases (multiset and set) it is easy to check that the t, t_i constructed meet the conditions (i)-(iii) of Lemma 2.77. □

Comparing the last lemma to Lemma 2.76, 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 2.27. 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 2.77. If $t = \emptyset$ then by condition (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

$$\pi^N_{\{\lambda\}}(t_1) = \{ok\} = \pi^N_{\{\lambda\}}(t)$$

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

We are now ready to prove Lemma 2.69.

Lemma 2.78 (2.69)**.** *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 2.71. Let $\mathcal{D} = \{N_1, \ldots, N_n\}$ 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 $\mathcal{X}_i := (N_i^*)^{\downarrow}$. Then there exist tuples $t, t_1, \ldots, t_n$ on N which meet the conditions (i)-(iii) of Lemma 2.77. We define $r = \{t_1, \ldots, t_n\}$, and start by showing that Σ holds on r.

So let $X \to Y \in \Sigma$ and $t_i, t_j \in r$ be two tuples with $\pi^N_X(t_i) = \pi^N_X(t_j)$. 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 $\pi^N_Y(t_i) = \pi^N_Y(t_j)$ by condition (i). It follows that $X \to Y$ 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. □

2.3.6 Testing for BCNF

We still require an efficient way to test whether an extended subattribute S is in BCNF. In the relational case we projected Σ^{*a} onto the subschema to be tested. However, as example 2.19 demonstrated, computing Σ^{*a} is often inefficient. Instead we will show that it is sufficient to project Σ^{*max} onto S, although we need to adjust our notion of projection.

Definition 2.79. Let Σ be a set of FDs on the nested attribute N, and $S \leq N$ be an extended subattribute of N. The *projection* of Σ onto S is

$$\Sigma[S] := \{X \to Y \sqcap S \mid X \to Y \in \Sigma \wedge X \leq S\}$$

The meet $Y \sqcap S$ on extended subattributes is induced by the extended subattribute order on them (Definition 2.66), and can be computed as

$$Y \sqcap S = \lceil \{y \sqcap s \mid y \in Y \wedge s \in S\} \rceil$$

Note that with this new notion of projection we do not lose any FDs, and all FDs gained through projection are still implied by Σ. In the relational case the two notions of projection are (almost) identical for singular FDs: If $X \to Y \in \Sigma$ is singular then either $Y \subseteq S$ or $Y \cap S = \emptyset$, and in the latter case $X \to Y \cap S$ is trivial. This is not true for nested attributes:

Example 2.28. Consider the set of singular FDs

$$\Sigma = \{A \to \{BC\}, \{BC\} \to A\}$$

on the nested attribute $N = A\{BC\}$. The projection of Σ onto $S = A\{B\}$ is

$$\Sigma[S] = \{A \to \{B\}\}$$

rather than the empty set. □

We can use this form of projection to construct a cover on an extended subattribute.

Lemma 2.80. *Let Σ be a set of FDs on N, and $S \leq N$. Then $\Sigma^{*max}[S]$ is a cover for $\Sigma^{*}[S]$.*

Proof. Clearly $\Sigma^{*}[S] \models \Sigma^{*max}[S]$, so we only need to show implication in the opposite direction. For that we use that the maximal atomic FDs

$$(\Sigma^{*}[S])^{*max} \subseteq \Sigma^{*}[S]$$

form a cover of $\Sigma^{*}[S]$, and show that each of them is implied by a FD in $\Sigma^{*max}[S]$.

So let $X \to A \in \Sigma^{*}[S]$ be maximal atomic on S. Then $X \to A$ is atomic w.r.t. Σ, and there exists a maximal atomic FD $X \to A' \in \Sigma^{*max}$ with $A \leq A'$. Thus

$$X \to A' \sqcap S \in \Sigma^{*max}[S]$$

and $X \to A' \sqcap S$ implies $X \to A$. □

The last lemma allows us to construct a cover of $\Sigma^{*}[S]$ efficiently, as Σ^{*max} can be computed using linear NA-resolution. This cover can then be used to test for BCNF as in the relational case.

Chapter 3

Canonical Covers

When given a set Σ of functional dependencies for schema decomposition, instance validation or similar tasks, we may choose to use a cover Σ' of functional dependencies equivalent to Σ instead. The choice of Σ' is important, as it determines the result of the decomposition (as we have seen in chapter 2), the speed of updates, and generally can have a huge impact on database performance. Optimal results are usually achieved by covers which are in some standard form, typically canonical or LR-reduced, but finding these optimal covers is often NP-hard.

We approach the problem by providing an algorithm for computing the set of all canonical covers, using an efficient form of representation. Once computed, finding the best canonical (or LR-reduced) cover w.r.t. some criteria is then a simple matter of comparing covers.

Example 3.1. Consider again the schema $CLRT$ from example 2.1 with

$$\Sigma = \{C \to L, CT \to R, LT \to C, RT \to C\}$$

Instead of using Σ, a designer could instead use any of its canonical covers:

$$\begin{array}{l}\{C \to L, CT \to R, LT \to C, RT \to C\},\\ \{C \to L, CT \to R, LT \to C, RT \to L\},\\ \{C \to L, LT \to R, RT \to C\}\end{array}$$

The last cover is the best choice, since it is smaller than the others and induced a BCNF decomposition when using it for the synthesis approach.

Note though that the number of canonical covers can be exponential in the number of attributes and FDs. It may therefore be more efficient to try computing only the cover which is actually needed. However, for some problems finding the desired cover directly (or testing whether one exists) can be difficult - we will see an example for this in section 4.5.1. But even if more efficient methods can be found, which compute the desired cover without computing all canonical covers first, the tools we develop for computing all canonical covers may prove useful in finding specific covers as well.

In order to make computing the set of all canonical covers feasible when the number of such covers is huge, we need to decompose it into smaller "partial" covers, to represent it efficiently. The decomposition we have in mind is the following:

Example 3.2. Let Σ consist of the following FDs:

$$\Sigma = \{AB \to C, C \to A, A \to D, D \to E, E \to A\}$$

Then Σ has the following canonical covers:

$$CC(\Sigma) = \left\{ \begin{array}{l}
\{AB \to C, C \to A, A \to D, D \to E, E \to A\}, \\
\{AB \to C, C \to A, A \to E, E \to D, D \to A\}, \\
\{AB \to C, C \to A, A \to D, D \to A, A \to E, E \to A\}, \\
\{AB \to C, C \to A, A \to D, D \to A, D \to E, E \to D\}, \\
\{AB \to C, C \to A, A \to E, E \to A, D \to E, E \to D\}, \\
\{DB \to C, C \to A, A \to D, D \to E, E \to A\}, \\
\{DB \to C, C \to A, A \to E, E \to D, D \to A\}, \\
\{DB \to C, C \to A, A \to D, D \to A, A \to E, E \to A\}, \\
\{DB \to C, C \to A, A \to D, D \to A, D \to E, E \to D\}, \\
\{DB \to C, C \to A, A \to E, E \to A, D \to E, E \to D\}, \\
\{EB \to C, C \to A, A \to D, D \to E, E \to A\}, \\
\{EB \to C, C \to A, A \to E, E \to D, D \to A\}, \\
\{EB \to C, C \to A, A \to D, D \to A, A \to E, E \to A\}, \\
\{EB \to C, C \to A, A \to D, D \to A, D \to E, E \to D\}, \\
\{EB \to C, C \to A, A \to E, E \to A, D \to E, E \to D\}, \\
\{AB \to C, C \to D, A \to D, D \to E, E \to A\}, \\
\{AB \to C, C \to D, A \to E, E \to D, D \to A\}, \\
\{AB \to C, C \to D, A \to D, D \to A, A \to E, E \to A\}, \\
\{AB \to C, C \to D, A \to D, D \to A, D \to E, E \to D\}, \\
\{AB \to C, C \to D, A \to E, E \to A, D \to E, E \to D\}, \\
\{DB \to C, C \to D, A \to D, D \to E, E \to A\}, \\
\{DB \to C, C \to D, A \to E, E \to D, D \to A\}, \\
\{DB \to C, C \to D, A \to D, D \to A, A \to E, E \to A\}, \\
\{DB \to C, C \to D, A \to D, D \to A, D \to E, E \to D\}, \\
\{DB \to C, C \to D, A \to E, E \to A, D \to E, E \to D\}, \\
\{EB \to C, C \to D, A \to D, D \to E, E \to A\}, \\
\{EB \to C, C \to D, A \to E, E \to D, D \to A\}, \\
\{EB \to C, C \to D, A \to D, D \to A, A \to E, E \to A\}, \\
\{EB \to C, C \to D, A \to D, D \to A, D \to E, E \to D\}, \\
\{EB \to C, C \to D, A \to E, E \to A, D \to E, E \to D\}, \\
\{AB \to C, C \to E, A \to D, D \to E, E \to A\}, \\
\{AB \to C, C \to E, A \to E, E \to D, D \to A\}, \\
\{AB \to C, C \to E, A \to D, D \to A, A \to E, E \to A\}, \\
\{AB \to C, C \to E, A \to D, D \to A, D \to E, E \to D\}, \\
\{AB \to C, C \to E, A \to E, E \to A, D \to E, E \to D\}, \\
\{DB \to C, C \to E, A \to D, D \to E, E \to A\}, \\
\{DB \to C, C \to E, A \to E, E \to D, D \to A\}, \\
\{DB \to C, C \to E, A \to D, D \to A, A \to E, E \to A\}, \\
\{DB \to C, C \to E, A \to D, D \to A, D \to E, E \to D\}, \\
\{DB \to C, C \to E, A \to E, E \to A, D \to E, E \to D\}, \\
\{EB \to C, C \to E, A \to D, D \to E, E \to A\}, \\
\{EB \to C, C \to E, A \to E, E \to D, D \to A\}, \\
\{EB \to C, C \to E, A \to D, D \to A, A \to E, E \to A\}, \\
\{EB \to C, C \to E, A \to D, D \to A, D \to E, E \to D\}, \\
\{EB \to C, C \to E, A \to E, E \to A, D \to E, E \to D\}
\end{array} \right\}$$

Instead of using this bulky "direct" representation, we decompose $CC(\Sigma)$ into sets of partial covers. We then obtain a full cover by selecting one partial cover from each set and forming their union:

$$CC(\Sigma) = \left\{ \begin{array}{l} \{AB \to C\}, \\ \{DB \to C\}, \\ \{EB \to C\} \end{array} \right\} \vee \left\{ \begin{array}{l} \{C \to A\}, \\ \{C \to D\}, \\ \{C \to E\} \end{array} \right\} \vee \left\{ \begin{array}{l} \{A \to D, D \to E, E \to A\}, \\ \{A \to E, E \to D, D \to A\}, \\ \{A \to D, D \to A, A \to E, E \to A\}, \\ \{A \to D, D \to A, D \to E, E \to D\}, \\ \{A \to E, E \to A, D \to E, E \to D\} \end{array} \right\}$$

Using this decomposed representation, dealing with $CC(\Sigma)$ becomes much easier.

3.1 Hypergraph Decomposition

The set of all canonical covers of a given set of FDs forms a simple hypergraph. We shall therefore establish some results on representing hypergraphs in general, which will be useful in representing (and even computing) the set of all canonical covers in particular.

Definition 3.1. A *hypergraph* H on a vertex set V is a set of subsets of V, i.e., $H \subseteq \mathcal{P}(V)$. The elements of H are called *edges*. A hypergraph is called *simple* if none of its edges is included in another.

Definition 3.2. The set ϑ_H of vertices actually appearing in edges of a hypergraph H is called the *support* of H:

$$\vartheta_H := \bigcup_{e \in H} e$$

Note that we *do* allow the empty edge in a hypergraph, and that we *do not* require that $V = \vartheta_H$. The latter point is not important but simplifies some arguments.

Definition 3.3. Let H, G be hypergraphs. We define the *cross-union* $H \vee G$ of H and G as

$$H \vee G := \{e_H \cup e_G \mid e_H \in H, e_G \in G\}$$

If V_H, V_G are the vertex sets of H and G then $H \vee G$ is a hypergraph on $V_H \cup V_G$.

Definition 3.4. Let H be a hypergraph on vertex set V and S some vertex set. The *projection* $H[S]$ of H onto S is

$$H[S] := \{e \cap S \mid e \in H\},$$

which is a hypergraph on $V \cap S$.

3.1.1 Autonomous Sets

We shall introduce the concept of an autonomous vertex set. Note that our definition is not meant to extend any use of the term "autonomous set" in the context of graphs, where it is better known as "module", and characterizes vertex sets M in which each vertex $v \in M$ has the same neighbors outside M. Using our terminology, autonomous sets are only interesting for hypergraphs but not for graphs. Essentially the only graphs with non-trivial autonomous sets are complete bipartite graphs, with the possibility to add isolated vertices and/or loops to all vertices of one side of the bipartition.

Definition 3.5. Let H be a hypergraph on the vertex set V. We call a vertex subset $S \subseteq V$ *autonomous* if $H = H[S] \vee H[\overline{S}]$ where $\overline{S} := V \setminus S$ denotes the complement of S.

Clearly the complement of an autonomous set is itself autonomous, and

$$H \subseteq H[S] \vee H[\overline{S}]$$

for any $S \subseteq V$. The sets $\emptyset, V$ are autonomous for any hypergraph H on V, as are all subsets of $V \setminus \vartheta_H$ and their complements.

Example 3.3. Consider the vertex set $V = ABCDE$, and on it the hypergraph

$$H = \{AC, AD, BC, BD\}$$

H is simple, and its support is $\vartheta_H = ABCD$. The set $S = AB$ is autonomous for H, as is its complement $\overline{S} = CDE$, since

$$H[AB] \vee H[CDE] = \{A, B\} \vee \{C, D\} = \{AC, AD, BC, BD\} = H$$

□

Lemma 3.6. *Let $S, T \subseteq V$ be autonomous. Then $S \cap T$ is autonomous as well.*

Proof. We need to show that for every pair of edges $e_1, e_2 \in H$ the edge $e'_1 \cup e'_2$ with $e'_1 := e_1 \cap (S \cap T)$ and $e'_2 := e_2 \cap \overline{S \cap T}$ lies in H as well. Since S is autonomous, the edge $e' := (e_1 \cap S) \cup (e_2 \cap \overline{S})$ lies in H. Thus, since T is autonomous, the edge $e'' := (e' \cap T) \cup (e_2 \cap \overline{T})$ lies in H. Clearly

$$e'' \cap (S \cap T) = e' \cap (S \cap T) = e_1 \cap (S \cap T) = e'_1$$

and similarly

$$\begin{aligned} e'' \cap \overline{S \cap T} &= (e' \cap (T \setminus S)) \cup (e_2 \cap \overline{T}) \\ &= (e_2 \cap (T \setminus S)) \cup (e_2 \cap \overline{T}) \\ &= e'_2 \end{aligned}$$

which shows $e'_1 \cup e'_2 = e'' \in H$. □

Corollary 3.7. *Let $S, T \subseteq V$ be autonomous. Then $S \cup T$ is autonomous.*

Proof. The complements and intersections of autonomous sets are autonomous by definition and by Lemma 3.6, respectively, and we have

$$S \cup T = \overline{\overline{S} \cap \overline{T}}$$

□

Proposition 3.8. *Let H, G be hypergraphs and S_1, S_2 vertex sets. Then we have*

(i) $H[S_1][S_2] = H[S_1 \cap S_2]$

(ii) $(H \vee G)[S_1] = H[S_1] \vee G[S_1]$

Lemma 3.9. *Let H be a hypergraph on V and $S \subseteq V$ autonomous for H. Then for any $T \subseteq V$ the set $S \cap T$ is autonomous for $H[T]$.*

Proof. Since S is autonomous for H we have $H = H[S] \vee H[\overline{S}]$. Thus

$$\begin{array}{rcl} H[T] & = & (H[S] \vee H[\overline{S}])[T] \\ & = & H[S][T] \vee H[\overline{S}][T] \\ & = & H[T][S \cap T] \vee H[T][\overline{S \cap T}] \end{array}$$

□

Theorem 3.10. *Let H be a hypergraph on V and $\{S_1, \dots, S_n\}$ a partition of V into autonomous sets. Then*

$$H = H[S_1] \vee \dots \vee H[S_n]$$

Proof. By induction on n. The equation hold trivially for $n = 1$. Assume now the theorem holds for a fixed value of n. To show the theorem for $n+1$ we use that $S_n \cup S_{n+1}$ is autonomous by Corollary 3.7, so that by assumption we have

$$H = H[S_1] \vee \dots \vee H[S_{n-1}] \vee H[S_n \cup S_{n+1}]$$

By Lemma 3.9 S_n is autonomous for $H[S_n \cup S_{n+1}]$, i.e., we have

$$H[S_n \cup S_{n+1}] = H[S_n] \vee H[S_{n+1}]$$

which shows the theorem for $n + 1$. □

When talking about *minimal autonomous sets*, we will always mean minimal w.r.t. inclusion among all non-empty autonomous sets, even though the empty set is always autonomous by definition. While it would be more precise to call them minimal *non-empty* autonomous sets, this quickly becomes tedious.

Theorem 3.11. *Every hypergraph H has a finest partition $\{S_1, \dots, S_n\}$ into minimal autonomous sets. The autonomous sets of H are just the unions of these sets.*

Proof. Let $S_1, \dots, S_n$ be the minimal autonomous sets of H. By Lemma 3.6 they are pairwise disjoint. The union of autonomous sets is itself autonomous by Corollary 3.7, in particular

$$S := \bigcup_{i=1}^{n} S_i$$

Furthermore the complement of S is autonomous, and since $\overline{S}$ does not include any minimal autonomous set it is empty, i.e., $S = V$. Thus the sets $S_1, \dots, S_n$ form a partition of V.

Whenever an autonomous set T intersects with some S_i it must include it completely, since otherwise $T \cap S_i$ would be a smaller non-empty autonomous set by Lemma 3.6. Thus each autonomous set is the union of those S_i it intersects with. □

Example 3.4. Consider again $H = \{AC, AD, BC, BD\}$ on $V = ABCDE$. Then

$$H = H[AB] \vee H[CD] \vee H[E] = \{A, B\} \vee \{C, D\} \vee \{\emptyset\}$$

so the minimal autonomous sets of H are AB, CD, E. Thus H has a total of 2^3 autonomous sets:

$$\emptyset, AB, CD, E, ABCD, ABE, CDE, ABCDE$$

□

We now consider another type of decomposition, which will help us in characterizing the autonomous sets of simple hypergraphs.

Definition 3.12. Let H be a hypergraph on vertex set V. The *subhypergraph* $H\langle S\rangle$ of H *induced* by $S \subseteq V$ is

$$H\langle S\rangle := \{e \in H \mid e \subseteq S\}$$

Definition 3.13. Let H be a hypergraph on the vertex set V. We call a vertex subset $S \subseteq V$ *isolated* if $H = H\langle S\rangle \cup H\langle \overline{S}\rangle$.

Clearly S is isolated if and only if every edge it intersects with lies completely in S. As with minimal autonomous sets, we will mean by *minimal isolated set*s the minimal sets (w.r.t. inclusion) among all non-empty isolated sets.

Definition 3.14. As with graphs, we say that two vertices v_1, v_n in a hypergraph H are *connected* if there exists a sequence $v_1, v_2, \ldots, v_n$ such v_i, v_{i+1} always lie in some common hyperedge of H. H is connected if all its vertices are connected. The *connected components* of H are its connected subhypergraphs.

It follows immediately that the minimal isolated sets of H are the vertex sets of its maximal connected components, and that the isolated sets of H are the unions of them.

Definition 3.15. Let H be a hypergraph on V. A set $t \subseteq V$ is a *transversal* of H if t intersects with every edge of H. We denote the set of all minimal transversals (w.r.t. inclusion) by $Tr(H)$, and call $Tr(H)$ the transversal hypergraph of H.

Clearly $Tr(H)$ is a simple hypergraph on V, even if H is not simple.

Theorem 3.16. *Let H, G be hypergraphs on disjoint vertex sets V_H and V_G. Then*

$$\begin{aligned} Tr(H \vee G) &= Tr(H) \cup Tr(G) \\ Tr(H \cup G) &= Tr(H) \vee Tr(G) \end{aligned}$$

Proof. (1) We first show that a set $t \subseteq V := V_H \cup V_G$ is a transversal of $H \vee G$ iff it intersects with every edge of H or with every edge of G. For the "if" part, assume w.l.o.g. that t intersects with every edge of H. Since every edge $e \in H \vee G$ is of the form $e = e_H \cup e_G$ with $e_H \in H, e_G \in G$, t intersects with e because it intersects with e_H. We show the "only if" part by contraposition and assume that there be edges $e_H \in H, e_G \in G$ such that t intersects with neither of them. But then t does not intersect $e_H \cup e_G \in H \vee G$ either, i.e., t is not a transversal of $H \vee G$.

We thus have that the transversals of $H \vee G$ are the transversals of H plus the transversals of G. Thus the minimal transversals of $H \vee G$ are the minimal elements of $Tr(H) \cup Tr(G)$. Since V_H and V_G are disjoint, all elements of $Tr(H) \cup Tr(G)$ are minimal. Thus

$$Tr(H \vee G) = Tr(H) \cup Tr(G)$$

(2) By definition a set $t \subseteq V$ is a transversal of $H \cup G$ iff it is a transversal of both H and G. Thus the transversals of $H \cup G$ are the unions of transversals of H with transversals of G. The minimal transversals of $H \cup G$ are therefore the minimal elements of $Tr(H) \vee Tr(G)$. Since V_H and V_G are disjoint, all elements of $Tr(H) \vee Tr(G)$ are minimal. Thus

$$Tr(H \cup G) = Tr(H) \vee Tr(G)$$

□

Lemma 3.17. *[7] Let H be a simple hypergraph. Then $Tr(Tr(H)) = H$.*

We are now able to characterize the autonomous sets of a simple hypergraph.

Theorem 3.18. *Let H be a simple hypergraph. Then the autonomous sets of H are the isolated sets of its transversal hypergraph $Tr(H)$.*

Proof. Let $S \subseteq V$ be autonomous for H, i.e., $H = H[S] \vee H[\overline{S}]$. Then

$$Tr(H) = Tr(H[S]) \cup Tr(H[\overline{S}])$$

by Theorem 3.16, so S is isolated for $Tr(H)$.

Conversely let S be any isolated set of $Tr(H)$. Then

$$Tr(H) = Tr(H)\left\langle S \right\rangle \cup Tr(H)\left\langle \overline{S} \right\rangle$$

and by Theorem 3.16 we have

$$Tr(Tr(H)) = Tr(Tr(H)\left\langle S \right\rangle) \vee Tr(Tr(H)\left\langle \overline{S} \right\rangle)$$

Thus S is autonomous for $Tr(Tr(H)) = H$. □

Example 3.5. The requirement that H be simple in Theorem 3.18 is necessary: Consider the hypergraphs

$$H = \{AC, AD, BC, BD\}$$
and
$$H' = H \cup \{ABC\}$$

Both H and H' have the same minimal transversals

$$Tr(H) = Tr(H') = \{AB, CD\}$$

Clearly AB, CD are isolated sets of $Tr(H')$, but AB and CD are not autonomous for H':

$$\begin{aligned} H'[AB] \vee H'[CD] &= \{A, B, AB\} \vee \{C, D\} \\ &= \{AC, AD, BC, BD, ABC, ABD\} \\ &= H' \cup \{ABD\} \neq H' \end{aligned}$$

□

Since graphs are just special hypergraphs, our theory of autonomous sets applies to them as well. Clearly all complete bipartite graphs have a non-trivial partition into two autonomous sets, but one may wonder whether there are others.

Lemma 3.19. *A simple graph G without isolated vertices has a non-trivial partition into autonomous sets iff it is complete bipartite.*

Proof. Let $S \notin \{\emptyset, \vartheta_G\}$ be autonomous. Since G contains no isolated vertices, $G[S]$ and $G[\overline{S}]$ contain non-empty edges. As all edges in a simple graph contain exactly two vertices, the edges of $G[S]$ and $G[\overline{S}]$ contain exactly one vertex each. Thus $G = G[S] \vee G[\overline{S}]$ is complete bipartite. □

We note that non-simple graphs with non-trivial autonomous partition may also have loops on all vertices of one side of the bipartition, as well as isolated vertices.

3.1.2 Superedges and Partial Superedges

While canonical covers will form the edges in our hypergraph, we will have to argue about atomic covers, i.e., sets of atomic FDs which form a cover, but may contain more FDs than needed. We call such supersets of edges "superedges".

Definition 3.20. Let H be a hypergraph on V. A set $E \subseteq V$ is called a *superedge* of H if it includes some edge $e \in H$, i.e. $e \subseteq E$. We call $E \subseteq S \subseteq V$ a *partial (super)edge* on S if E is a (super)edge of $H[S]$.

Lemma 3.21. *Let H be a hypergraph on V and $S \subseteq V$. A set $S' \subseteq S$ is a partial superedge on S iff $S' \cup \overline{S}$ is a superedge.*

Proof. By definition S' is a partial superedge on S iff it includes a partial edge $e_S \in H[S]$, i.e. iff there exists an edge $e \in H$ with $e \cap S = e_S \subseteq S'$. Since

$$e = e_S \cup (e \cap \overline{S}) \subseteq S' \cup \overline{S}$$

this implies that $S' \cup \overline{S}$ is a superedge. Conversely, if $S' \cup \overline{S}$ is a superedge, it includes an edge $e \in H$, which gives us

$$e \cap S \subseteq (S' \cup \overline{S}) \cap S = S'$$

□

Lemma 3.22. *Let H be a hypergraph on V and $P = \{S_1, \ldots, S_n\}$ a partition of V into autonomous sets. A set $E \subseteq V$ is a superedge iff $E_i := E \cap S_i$ is a partial superedge on S_i for $i = 1, \ldots, n$.*

Proof. If E is a superedge then it includes an edge $e \in H$. Thus E_i includes $e \cap S_i \in H[S_i]$, which makes E_i a partial superedge on S_i.

Now for each $i = 1, \ldots, n$ let E_i be a partial superedge on S_i, including the partial edge $e_i \in H[S_i]$. Thus E includes

$$e := e_1 \cup \ldots \cup e_n \in H[S_1] \vee \ldots \vee H[S_n] \overset{\text{(thm 3.10)}}{=} H$$

which makes E a superedge of H. □

We can therefore strengthen Lemma 3.21 when S is autonomous:

Lemma 3.23. *Let H be a hypergraph on V, $E \subseteq V$ a superedge and $S \subseteq V$ autonomous. A set $S' \subseteq S$ is a partial superedge on S iff $S' \cup (E \setminus S)$ is a superedge.*

Proof. By Lemma 3.22, the set $S' \cup (E \setminus S)$ is a superedge iff S' is a partial superedge on S and $E \setminus S$ is a partial superedge on $\overline{S}$. Since E is a superedge, Lemma 3.22 assures that $E \setminus S = E \cap \overline{S}$ is a partial superedge on $\overline{S}$. □

3.1.3 Computing Autonomous Sets

To complete this section, we now address the question of computing the minimal autonomous sets of H. While Theorem 3.18 suggests an approach (at least for simple hypergraphs), computing the transversal hypergraph can lead to exponential runtime. Instead, we shall utilize the following observation.

Lemma 3.24. *Let H be a hypergraph on V and $P = \{S_1, \ldots, S_n\}$ the partition of V into minimal autonomous sets. Let further $H'_1 \subseteq H[S_1]$ be non-empty, and $H' \subseteq H$ be the hypergraph*

$$H' := H'_1 \vee H[S_2] \vee \ldots \vee H[S_n].$$

Then $S_2, \ldots, S_n$ are minimal autonomous sets of H'.

Proof. By definition $S_2, \ldots, S_n$ are autonomous for H'. If one of those S_i were not minimal for H', i.e., could be partitioned into smaller autonomous sets $T_1, \ldots, T_k$, then

$$H[S_i] = H[T_1] \vee \ldots \vee H[T_k]$$

and thus the sets T_i would be autonomous for H as well, contradicting the minimality of S_i. $\square$

We use this to compute the partition of V into minimal autonomous sets as follows. We pick some vertex $v \in V$ and split H into two hypergraphs, one containing all the edges which contain v, the other one containing all those edges which do not contain v. We will need only one of them, so let H_v be the smaller one of the two, i.e., the one with fewer edges (if both contain exactly the same number of edges we may choose either one):

$$H_v := \text{smaller of } \left\{ \begin{array}{l} \{e \in H \mid v \in e\} \\ \{e \in H \mid v \notin e\} \end{array} \right.$$

If H_v is empty, then v lies in all or no edges of H, and in both cases the set $\{v\}$ is autonomous for H. This reduces the problem of finding the minimal autonomous sets of H to finding the minimal autonomous sets of $H[\overline{v}]$, where $\overline{v} := \vartheta_H \setminus \{v\}$, as they are also minimal autonomous sets of H.

Consider now the case where H_v is not empty. Let S_1 be the minimal autonomous set of H containing v. Then H_v has the same form as H' in Lemma 3.24, where H'_1 contains either the edges of $H[S_1]$ which do or those which do not contain v. We now compute the minimal autonomous sets of H_v, and check for each set whether it is autonomous for H. By Lemma 3.24 the sets autonomous for H are exactly the $S_2, \ldots, S_n$, while the sets not autonomous for H partition S_1. Taking the union of those non-autonomous sets and keeping the autonomous ones thus gives us the minimal autonomous sets of H. Note that the set $\{v\}$ is always autonomous for H_v, as v is contained in either all or no edges of H_v. Thus it suffices to compute the minimal autonomous sets of $H_v[\overline{v}]$.

In either case we have reduced the problem of finding the minimal autonomous set of H to that of finding the minimal autonomous sets of a hypergraph with fewer vertices. This gives us the following recursive algorithm.

Algorithm "Recursive Autonomous Partitioning"

```
INPUT: hypergraph $H$
OUTPUT: partition of $\vartheta_H$ into minimal autonomous sets

function RAP($H$)
select vertex $v \in \vartheta_H$
$H_v :=$ smaller of $\begin{cases} \{e \in H \mid v \in e\} \\ \{e \in H \mid v \notin e\} \end{cases}$
if $H_v = \emptyset$ then
  return $\{\{v\}\} \cup RAP(H[\overline{v}])$
else
  $Aut := \emptyset, S_1 := \{v\}$
  $Aut_v := RAP(H_v[\overline{v}])$
  for all $S \in Aut_v$ do
      if $S$ autonomous for $H$ then
        $Aut := Aut \cup \{S\}$
      else
        $S_1 := S_1 \cup S$
  end
  return $\{S_1\} \cup Aut$
```

While the test whether a set S is autonomous for H can be performed by computing $H' := H[S] \vee H[\overline{S}]$ and comparing it to H, the resulting set can easily contain up to $|H|^2$ edges if S is not autonomous. We observe that always $H \subseteq H'$ and thus $H = H'$ iff $|H| = |H'|$. Since $|H'| = |H[S]| \cdot |H[\overline{S}]|$, the later condition can be checked faster without actually computing H'.

Theorem 3.25. *Let H be a hypergraph with k vertices and n edges. Then the "Recursive Autonomous Partitioning" algorithm computes the partition of ϑ_H into minimal autonomous sets of H in time $O(nk^2)$.*

Proof. We have already argued that the algorithm computes the minimal autonomous sets of H correctly, so we only need to show the time bound.

The depth of recursion is at most k. In each call we compute H_v, which can be done in $O(n)$. If $H_v = \emptyset$ we only need to compute $H[\overline{v}]$, which is possible in $O(nk)$. Thus this part of the algorithm can be performed in $O(nk^2)$.

If $H_v \neq \emptyset$ we need to test each set found to be autonomous for H_v whether it is autonomous for H. The number of such tests is at most k, and each test can be performed in $O(nk)$, by computing $H[S]$ and $H[\overline{S}]$ and testing whether $|H| = |H[S]| \cdot |H[\overline{S}]|$. This leads to a complexity of $O(nk^2)$. Since the number of edges of H_v is at most half of the number of edges of H, the number of steps required for performing the tests on H_v (or the next subgraph in the recursion for which tests are required) is at most half as many. This leads to a total complexity of

$$O((n + \frac{n}{2} + \frac{n}{4} + \ldots)k^2) = O(nk^2)$$

□

3.2 Computing all Canonical Covers

Recall that we wish to compute the set of all canonical covers as a general method for finding covers which are best in some sense. Of cause, this approach only works if at least some of the optimal solutions are canonical, and comparing two given covers is easy. We will see later that these conditions are met for a number of hard problems, for which no efficient algorithms are known.

In the following we will develop an algorithm for computing the set of all canonical covers. As far as we know, no such algorithm has appeared in the literature so far.

Definition 3.26. Let Σ be a set of FDs. We denote the set of all canonical covers of Σ by

$$CC(\Sigma) := \{G \subseteq \Sigma^{*a} \mid G \text{ is a canonical cover of } \Sigma\}$$

Note that the problem of finding canonical covers is similar to that of finding minimal keys. Instead of seeking all minimal sets of attributes that determine all other attributes, we seek all minimal sets of FDs which imply all other FDs. For this we will use the linear resolution algorithm introduced in chapter 2.

3.2.1 Partial Covers

The set of all canonical covers of Σ forms a simple hypergraph on the FDs in Σ^{*a}. We may thus use the terms defined for hypergraphs for canonical covers as well. In particular, we shall talk about autonomous sets of FDs, and (partial) superedges. Note that in this context the superedges are the atomic covers, while the edges are the canonical covers.

Definition 3.27. We call a set of FDs in Σ^{*a} autonomous if it is autonomous for the hypergraph $CC(\Sigma)$. When talking about transversals, we always mean transversals of $CC(\Sigma)$.

Lemma 3.28. *A set $G \subseteq \Sigma^{*a}$ is a cover of Σ iff it intersects with all minimal transversals of $CC(\Sigma)$.*

Proof. G is a cover iff it is a superedge of $CC(\Sigma)$. Furthermore, $CC(\Sigma)$ is simple, and by Lemma 3.17 the edges of a simple hypergraph are the minimal sets which intersect with all minimal transversals. Thus superedges are simply sets (not necessarily minimal) which intersect with all minimal transversals. □

As superedges become (atomic) covers for the hypergraph $CC(\Sigma)$, partial superedges become partial covers.

Definition 3.29. Let Σ be a set of FDs and $G \subseteq S \subseteq \Sigma^{*a}$. We call G a *partial cover* of Σ on S if G is a partial superedge of $CC(\Sigma)$ on S.

When S is autonomous, testing whether a set of FDs is a partial cover on S is easy:

Lemma 3.30. *Let $S \subseteq \Sigma^{*a}$ be autonomous, and let $\Sigma' \subseteq \Sigma^{*a}$ be an atomic cover of Σ. Then a set $G \subseteq S$ is a partial cover on S iff $G \cup (\Sigma' \setminus S)$ is a cover of Σ.*

Proof. Follows directly from Lemma 3.23. □

Clearly $G \cup (\Sigma' \setminus S)$ is a cover of Σ iff $G \cup (\Sigma' \setminus S) \models \Sigma' \cap S$, which allows us to perform this test quickly.

We will identify some autonomous (but not necessarily minimal) sets of $CC(\Sigma)$. Theorem 3.18 relates autonomous sets to the minimal transversals of $CC(\Sigma)$. The following lemmas establish some results about the form of these minimal transversals.

Lemma 3.31. *Let $S \subseteq \Sigma^{*a}$ be a minimal transversal of $CC(\Sigma)$ and $X \to A \in S$. Then $\overline{S} = \Sigma^{*a} \setminus S$ is not a cover of Σ, but $\overline{S} \cup \{X \to A\}$ is.*

Proof. By Lemma 3.28, $\overline{S}$ is not a cover of Σ since it does not intersect with S. If $\overline{S} \cup \{X \to A\}$ were not a cover, then every cover would contain a FD in

$$\overline{\overline{S} \cup \{X \to A\}} = S \setminus \{X \to A\}$$

Thus $S \setminus \{X \to A\}$ would be a transversal, which contradicts the minimality of S. □

Definition 3.32. The sets of attributes X and Y are equivalent under a set of FDs Σ, written $X \leftrightarrow Y$, if $X \to Y$ and $Y \to X$ lie in Σ^*.

Lemma 3.33. *Let $X \to A, Y \to B$ be contained in a common minimal transversal $S \subseteq \Sigma^{*a}$ of $CC(\Sigma)$. Then X and Y are equivalent under $\overline{S} = \Sigma^{*a} \setminus S$.*

Proof. By Lemma 3.31 we have

$$\begin{aligned} \overline{S} &\not\models Y \to B \\ \overline{S} \cup \{X \to A\} &\models Y \to B \end{aligned} \tag{3.1}$$

Let us denote the closure of Y under $\overline{S}$ by $Y^{*\overline{S}}$. If $X \not\subseteq Y^{*\overline{S}}$ then

$$Y^{*\overline{S}} = Y^{*\overline{S} \cup \{X \to A\}}$$

which contradicts (3.1). Thus $\overline{S} \models Y \to X$, and by symmetry $\overline{S} \models X \to Y$. □

Definition 3.34. Let Σ be a set of FDs on R. We denote the set of FDs in Σ^{*a} with LHS equivalent to $X \subseteq R$ as

$$EQ_X := \{Y \to Z \in \Sigma^{*a} \mid Y \leftrightarrow X\}$$

The partition of Σ^{*a} into non-empty equivalence sets is denoted as

$$EQ := \{EQ_X \mid \exists Y.X \to Y \in \Sigma^{*a}\}$$

Theorem 3.35. *Let Σ be a set of FDs on R. Then every set $EQ_X \in EQ$ is autonomous.*

Proof. By Lemma 3.33 all FDs in a (maximal) connected component of $Tr(CC(\Sigma))$ have equivalent LHSs under Σ. Thus EQ_X is the union of vertex sets of maximal connected components of $Tr(CC(\Sigma))$, and therefore an isolated set of $Tr(CC(\Sigma))$. By Theorem 3.18 isolated sets of $Tr(CC(\Sigma))$ are autonomous for $CC(\Sigma)$. □

We are now ready to prove our main theorem for this section.

Theorem 3.36. *Let Σ be a set of FDs on R. A set $G \subseteq \Sigma^{*a}$ is a cover of Σ iff $G \cap EQ_X$ is a partial cover of Σ on EQ_X for every $EQ_X \in EQ$.*

Proof. By Theorem 3.35 the sets EQ_X form a partition of Σ^{*a} into autonomous sets, so the theorem is a special case of Lemma 3.22. □

Theorem 3.36 allows us to split the task of finding and representing all canonical covers of Σ into several smaller tasks. For every $EQ_X \in EQ$ we find the set C_X of all non-redundant partial covers on EQ_X. By Theorem 3.10 these describe $CC(\Sigma)$ in decomposed form:

$$CC(\Sigma) = C_{X_1} \vee \ldots \vee C_{X_n}$$

While we could easily compute $CC(\Sigma)$ by taking their cross-union, this decomposed description of $CC(\Sigma)$ is usually much smaller, and thus better suited for most tasks.

Note that the equivalence classes EQ_X need not be minimal autonomous sets of $CC(\Sigma)$. If we could find smaller autonomous sets we could speed up the computation of $CC(\Sigma)$ even more. However, we will show later (Theorem 3.81) that finding the minimal autonomous sets of $CC(\Sigma)$ is hard. In section 3.4 we will give an algorithm to find finer, but not necessarily minimal autonomous sets.

3.2.2 Relative Covers

Partial covers on a set S of atomic FDs are obtained by taking an atomic cover and intersecting it with S. A different concept which will become useful is that of relative covers, which can be obtained through 'relativation' of covers onto sets of attributes.

Definition 3.37. Let Σ be a set of FDs on R, and Σ_H be a set of FDs on $H \subseteq R, \overline{H} := R \setminus H$. We call Σ_H a *relative cover* of Σ on H if for all $X, Y \subseteq H$ we have

$$\Sigma_H \models X \to Y \Leftrightarrow \Sigma \models X \cup \overline{H} \to Y$$

Relative covers have been used previously by Saiedian and Spencer in [39] under the name contraction.

Definition 3.38. The *relativation* of a FD $X \to Y$ onto an attribute set H is

$$X \to Y\,]H[:= X \cap H \to Y \cap H$$

The *relativation* of a set Σ of FDs onto H is

$$\Sigma\,]H[:= \{X \to Y\,]H[\mid X \to Y \in \Sigma\}$$

Note that we do allow FDs with empty LHS. They arise naturally when relativating sets of FDs with non-empty LHSs.

We will show that relative covers can be constructed through relativation.

Lemma 3.39. *Let Σ_H be a relative cover of Σ on H. Then*

$$\Sigma_H^* = \Sigma^*\,]H[$$

Proof. By definition we have

$$\Sigma^*\,]H[= \{X \cap H \to Y \cap H \mid X \to Y \in \Sigma^*\}$$

Thus for any $X, Y \subseteq H$ we get

$$\begin{aligned} X \to Y \in \Sigma^*]H[&\Leftrightarrow \exists X' \subseteq \overline{H} \text{ with } X \cup X' \to Y \in \Sigma^* \\ &\Leftrightarrow X \cup \overline{H} \to Y \in \Sigma^* \\ &\Leftrightarrow X \to Y \in \Sigma_H^* \end{aligned}$$

The last correspondence holds since Σ_H is a relative cover. □

Lemma 3.40. *Let Σ be a set of FDs on R and $H \subseteq R$.*

(a) If $\Sigma \vDash X \to Y$ then $\Sigma]H[\vDash X \cap H \to Y \cap H$

(b) If $\Sigma]H[\vDash X \to Y$ then $\Sigma \vDash X \cup \overline{H} \to Y$

Proof. (a) If $X \to Y \in \Sigma$ then $X \cap H \to Y \cap H \in \Sigma]H[$. Otherwise $X \to Y$ can be derived from Σ using the Armstrong Axioms (1.1). We show that $X \cap H \to Y \cap H$ can be derived from $\Sigma]H[$ by induction on the length of the derivation tree used to derive $X \to Y$. This is straight forward:

$$\begin{array}{ccc} \text{derivation from } \Sigma & & \text{derivation from } \Sigma]H[\\[1ex] \dfrac{}{X \to Y} Y \subseteq X & \curvearrowright & \dfrac{}{X \cap H \to Y \cap H} Y \cap H \subseteq X \cap H \\[2ex] \dfrac{X \to Y}{XW \to YW} & \curvearrowright & \dfrac{X \cap H \to Y \cap H}{XW \cap H \to YW \cap H} \\[2ex] \dfrac{X \to Y \quad Y \to Z}{X \to Z} & \curvearrowright & \dfrac{X \cap H \to Y \cap H \quad Y \cap H \to Z \cap H}{X \cap H \to Z \cap H} \end{array}$$

(b) If $X \to Y \in \Sigma]H[$ then Σ contains a FD $X \cup X' \to Y \cup Y'$ with $X', Y' \subseteq \overline{H}$, which implies $X \cup \overline{H} \to Y$. The remaining argument proceeds as for (a). □

Lemma 3.41. *Let Σ be a set of FDs on R and $H \subseteq R$. Then*

$$\Sigma^*]H[= (\Sigma]H[)^*$$

Proof. We can show $\Sigma^*]H[\subseteq (\Sigma]H[)^*$ as follows:

$$\begin{array}{rcl} X' \to Y' \in \Sigma^*]H[& \Leftrightarrow & \exists X \to Y \in \Sigma^* \text{ with } X' = X \cap H, Y' = Y \cap H \\ & \Leftrightarrow & \Sigma \vDash X \to Y \\ \text{(Lemma 3.40a)} & \Rightarrow & \Sigma]H[\vDash X \cap H \to Y \cap H \\ & \Leftrightarrow & X' \to Y' \in (\Sigma]H[)^* \end{array}$$

The proof for $(\Sigma]H[)^* \subseteq \Sigma^*]H[$ is similar:

$$\begin{array}{rcl} X \to Y \in (\Sigma]H[)^* & \Leftrightarrow & \Sigma]H[\vDash X \to Y \\ \text{(Lemma 3.40b)} & \Rightarrow & \Sigma \vDash X \cup \overline{H} \to Y \\ & \Leftrightarrow & X \cup \overline{H} \to Y \in \Sigma^* \\ & \Rightarrow & X \to Y \in \Sigma^*]H[\end{array}$$

□

Using lemmas 3.39 and 3.41 we get:

Theorem 3.42. *Let Σ be a set of FDs on R and $H \subseteq R$. Then the relativation $\Sigma\,]H[$ of Σ is a relative cover of Σ on H.*

Proof. Let Σ_H be a relative cover of Σ on H. We need to show that $\Sigma_H^* = (\Sigma\,]H[)^*$. This is clear by lemmas 3.39 and 3.41:

$$\Sigma_H^* = \Sigma^*\,]H[= (\Sigma\,]H[)^*$$

□

Note that a variant of Theorem 3.42 has been shown previously: It corresponds to lemma 6 in [39]. Note further that the converse of Theorem 3.42 does not hold: not every relative cover is the relativation of a cover.

Example 3.6. Consider the set of FDs

$$\Sigma = \{AB \to C, C \to D, B \to D\}.$$

Its relativation onto $H = BCD$ is a relative cover of Σ on H:

$$\Sigma\,]H[= \{B \to C, C \to D, B \to D\}$$

The FD $B \to D$ is redundant in $\Sigma\,]H[$ so that the set

$$\Sigma_H := \{B \to C, C \to D\}$$

is also a relative cover of Σ on H. However, every cover of Σ contains $B \to D$ or $B \to BD$, so Σ_H cannot be the relativation of a cover of Σ. □

3.2.3 Implication Dependencies

In section 2.1.5 we have seen how minimal keys can be computed efficiently, and we noted that the problem of finding canonical covers is similar. However, in the case of the key finding problem, a set of FDs was used to describe determination between attribute sets, whereas implication of FDs is given implicitly. To utilize our linear resolution algorithm, we need to make implications explicit. This is done as follows.

Definition 3.43. Let Σ be a set of FDs. We call an expression of the form $S \Rightarrow T$ where $S, T \subseteq \Sigma$ an *implication dependency* (ID).

An ID $S \Rightarrow T$ is the equivalent of a FD $S \to T$ over Σ, where Σ is regarded as attribute set (i.e., we regard the FDs in Σ as independent attributes without any connection). We thus use the terminology defined for FDs for IDs as well, assuming an equivalent definition. In particular, we say that a set Π of IDs *implies* an ID $S \Rightarrow T$ iff $S \Rightarrow T$ can be derived from Π using the equivalent of the Armstrong axioms (with $\to$ replaced by $\Rightarrow$).

Definition 3.44. Let Σ be a set of FDs. We call a set Π of IDs on Σ an *implication cover* of Σ if for all sets $S, T \subseteq \Sigma$ we have

$$S \Rightarrow T \in \Pi^* \text{ iff } S \vDash T$$

We call Π *sound* if $S \Rightarrow T \in \Pi^*$ implies $S \vDash T$, and *complete* for Σ if $S \vDash T$ implies $S \Rightarrow T \in \Pi^*$. Furthermore, we call Π_H a *relative implication cover* on $H \subseteq \Sigma$ if for all $S, T \subseteq H$ we have

$$S \Rightarrow T \in \Pi_H^* \text{ iff } S \cup (\Sigma \setminus H) \vDash T$$

Note that the relationship of implication covers and relative implication covers is the same as for covers and relative covers: For any implication cover Π the condition

$$S \Rightarrow T \in \Pi_H^* \text{ iff } S \cup (\Sigma \setminus H) \vDash T$$

is equivalent to

$$\Pi_H \vDash S \Rightarrow T \text{ iff } \Pi \vDash S \cup (\Sigma \setminus H) \Rightarrow T$$

which resembles precisely Definition 3.37. In particular this gives us Theorem 3.42 for implication covers:

Corollary 3.45. *Let Σ be a set of FDs with implication cover Π and $H \subseteq \Sigma$. Then $\Pi]H[$ is a relative implication cover of Σ on H.*

Let us now recall Lemma 3.21. For the hypergraph $CC(\Sigma)$ it states the following:

Corollary 3.46. *Let Σ be a set of FDs and $H \subseteq \Sigma^{*a}$. A set $S \subseteq H$ is a partial cover on H iff $S \cup (\Sigma^{*a} \setminus H)$ is a cover of Σ.*

The last condition of Corollary 3.46 is equivalent to $S \cup (\Sigma^{*a} \setminus H) \vDash \Sigma^{*a}$, and thus to

$$S \cup (\Sigma^{*a} \setminus H) \vDash H$$

Comparing this to Definition 3.44, we can rewrite the condition as

$$S \Rightarrow H \in \Pi_H^*$$

for any relative implication cover Π_H of Σ^{*a} on H. This characterizes S as a key of H w.r.t. the set of IDs Π_H, giving us the following lemma.

Lemma 3.47. *Let $S \subseteq H \subseteq \Sigma^{*a}$, and Π_H be a relative implication cover of Σ^{*a} on H. Then S is a partial cover of Σ^{*a} on H iff $S \Rightarrow H \in \Pi_H^*$.*

Proof. See above. □

The last lemma allows us to find partial canonical covers as follows: We first find a relative implication cover Π_H, then use linear resolution to find all minimal keys w.r.t. Π_H, which are the partial canonical covers needed.

The next theorem allows us to compute a relative implication cover. To make the soundness proof for a (relative) implication cover easier, we first show a simple lemma.

Lemma 3.48. *A set Π of IDs on Σ is sound iff $S \vDash T$ for all $S \Rightarrow T \in \Pi$. A set Π_H of IDs on $H \subseteq \Sigma$ is sound iff $S \cup (\Sigma \setminus H) \vDash T$ for all $S \Rightarrow T \in \Pi_H$.*

Proof. We only need to show that $S \vDash T$ for all derived IDs $S \Rightarrow T \in \Pi^*$, and $S \cup (\Sigma \setminus H) \vDash T$ for IDs $S \Rightarrow T \in \Pi_H^*$. For each application of the armstrong axioms for IDs, it is easy to see that soundness of the premises (i.e., $S \vDash T$ or $S \cup (\Sigma \setminus H) \vDash T$, respectively) implies soundness of the derived IDs. □

Theorem 3.49. *Let Σ be a set of FDs, EQ the partition of Σ^{*a} into equivalence classes, and $H = EQ_X$ for some set $EQ_X \in EQ$. Construct Π_X as follows: For every pair of (different) FDs $Y \to A \in H, Z \to A \in \Sigma^{*a}$ with $\Sigma \models Y \to Z$ let Π_X contain the ID*

$$\{Y \to Z_i \in H \mid Z_i \in Z\} \cup \{Z \to A\} \Rightarrow Y \to A \tag{3.2}$$

provided $Z \to A \in H$, or

$$\{Y \to Z_i \in H \mid Z_i \in Z\} \Rightarrow Y \to A \tag{3.3}$$

otherwise. Then Π_X is a relative implication cover of Σ^{*a} on H.

Proof. We first show that Π_X is sound. By Lemma 3.48 it suffices to show that for every ID in Π_X of the form (3.2) or (3.3) we have

$$\{Y \to Z_i \in H \mid Z_i \in Z\} \cup \{Z \to A\} \cup (\Sigma^{*a} \setminus H) \models Y \to A \tag{3.4}$$

For every $Z_i \in Z \setminus Y$ there is a minimal $Y_i \subset Y$ such that $Y_i \to Z_i \in \Sigma^{*a}$. If $Y_i \neq Y$, then Y_i and Y are not equivalent, since Y is a minimal LHS. Thus $Y_i \to Z_i \in \Sigma^{*a} \setminus H$, so that all $Y_i \to Z_i$ are contained in the LHS of (3.4). Clearly $\{Y_i \to Z_i \mid Z_i \in Z \setminus Y\} \cup \{Z \to A\}$ implies $Y \to A$.

To prove that Π_X is complete, let $S, T \subseteq H$ with $S \cup (\Sigma^{*a} \setminus H) \models T$. We need to show that $S \Rightarrow T \in \Pi_X^*$. Assume the contrary, so that for $U' := S \cup (\Sigma^{*a} \setminus H)$ there exists a FD $Y' \to A' \in T$ (and thus $Y' \to A' \in H$) with (note that $U' \cap H = S$):

$$U' \models Y' \to A' \text{ and } U' \cap H \Rightarrow Y' \to A' \notin \Pi_X^*$$

Now let $U \subseteq U'$ be minimal such that there exists a FD $Y \to A \in H$ for which

$$U \models Y \to A \text{ and } U \cap H \Rightarrow Y \to A \notin \Pi_X^*$$

Consider closure computation for Y under U: Since we have $U \models Y \to A$ there must be a FD $Z \to A \in U$ such that $U_A := U \setminus \{Z \to A\}$ implies $Y \to Z$. Equivalently $U_A \models Y \to Z_i$ for all $Z_i \in Z$. Since $U_A \subsetneq U$ and U was chosen minimal, we get

$$U_A \cap H \Rightarrow Y \to Z_i \in \Pi_X^*$$

for all $Y \to Z_i \in H, Z_i \in Z$. Since $U_A \cap H \subseteq U' \cap H = S$ this gives us

$$S \Rightarrow \{Y \to Z_i \in H \mid Z_i \in Z\} \in \Pi_X^*$$

If $Z \to A \in H$ then $Z \to A \in U \cap H \subseteq S$, and since Π_X contains the ID (3.2) it follows that

$$S \Rightarrow Y \to A \in \Pi_X^*$$

which contradict our assumption. For $Z \to A \notin H$ the same follows with the ID (3.3). □

The size of the relative implication cover of EQ_X constructed is clearly polynomial in the size of Σ^{*a}. We note that using Theorem 3.36 to split up the problem of finding canonical covers into finding partial canonical covers for equivalence classes of FDs is helpful in two ways. First it allow us to represent the set of all canonical covers in an efficient manner. At the same time it simplifies the problem of finding implication covers by allowing for small relative implication covers. As example 3.7 demonstrates, it can happen that every implication cover of Σ^{*a} is exponential in the size of Σ^{*a}.

Example 3.7. Let $X_1 \dots X_{2n}, Y_1 \dots Y_n, A$ be attributes and

$$X = X_1 \dots X_{2n}, X^i = X \setminus X_i, Y = Y_1 \dots Y_n, Y^i = Y \setminus Y_i$$

be attribute sets. Let further

$$\Sigma = \left\{ \begin{array}{cc} X^1 \to Y_1, & X^2 \to Y_1, \\ X^3 \to Y_2, & X^4 \to Y_2, \\ \dots & \dots \\ X^{2n-1} \to Y_n, & X^{2n} \to Y_n, \\ Y \to A & \end{array} \right\}$$

and thus (note that $X^i \cup X^j = X$ for $i \neq j$)

$$\Sigma^{*a} = \Sigma \cup \left\{ \begin{array}{cc} X^1Y^1 \to A, & X^2Y^1 \to A, \\ X^3Y^2 \to A, & X^4Y^2 \to A, \\ \dots & \dots \\ X^{2n-1}Y^n \to A, & X^{2n}Y^n \to A, \\ X \to A & \end{array} \right\}$$

Then every implication cover of Σ^{*a} contains (at least) the 2^n atomic IDs

$$\{Z^1Y^1 \to A, Z^2 \to Y_2, Z^3 \to Y_3, \dots, Z^n \to Y_n\} \Rightarrow X \to A$$

where each Z^i is replaced by X^{2i-1} or X^{2i}. This is because the FDs in the LHSs do not imply any other FD in $\Sigma^{*a} \setminus \{X \to A\}$. □

3.2.4 The Algorithm

We summarize the algorithm developed below.

Algorithm "divide and resolve"

```
INPUT: set of FDs $\Sigma$
OUTPUT: set of all partial canonical covers $CC_X$ for every equivalence class $EQ_X$
of $\Sigma^{*a}$

compute $\Sigma^{*a}$
partition $\Sigma^{*a}$ into equivalence classes $EQ$
for each $EQ_X \in EQ$ do
    construct relative implication cover $\Pi_X$ of $EQ_X$
    $CC_X := \{$minimal keys of $EQ_X$ w.r.t. $\Pi_X\}$
end
```

The sets CC_X are the partial canonical covers of Σ on EQ_X by Lemma 3.47, and together they represent $CC(\Sigma)$ as described in Theorem 3.36.

Note that the partition of Σ^{*a} into autonomous sets EQ might not be minimal. We will see later in section 3.5 that deciding whether a set of FDs is autonomous for $CC(\Sigma)$ is co-NP-complete, even when given Σ^{*a}. However, given all minimal autonomous sets, testing whether a set is autonomous can be done in polynomial time by Theorem 3.11. Thus, unless P=NP, finding the minimal autonomous sets will not be possible in polynomial time. We contented ourselves with the non-minimal partition EQ since it was easy to identify and fast to compute. In section 3.4 we will discuss an efficient method for computing a finer partition into autonomous sets, although these sets may not be minimal either.

If we want to find the minimal autonomous partition *after* the sets CC_X have been computed, e.g., to store $CC(\Sigma)$ more efficiently, we can partition the hypergraphs CC_X further using the "Recursive Autonomous Partitioning" algorithm from section 3.1.

The following example calculation illustrates the algorithm "divide and resolve".

Example 3.8. Our goal is to compute all canonical covers for the set of FDs

$$\Sigma = \{AB \to C, AC \to B, AD \to C, AE \to C, BE \to A\}$$

We start by computing the atomic closure Σ^{*a} of Σ

$$\Sigma^{*a} = \Sigma \cup \{AE \to B, AD \to B, BE \to C\}$$

and partitioning Σ^{*a} into equivalence classes $EQ = \{EQ_{AB}, EQ_{AD}, EQ_{AE}\}$ with

$$\begin{array}{rcl} EQ_{AB} & = & \{AB \to C, AC \to B\} \\ EQ_{AD} & = & \{AD \to C, AD \to B\} \\ EQ_{AE} & = & \{AE \to B, AE \to C, BE \to A, BE \to C\} \end{array}$$

We then construct the relative implication cover for each equivalence class:

$$\begin{array}{rcl} \Pi_{AB} & = & \emptyset \\ \Pi_{AD} & = & \left\{ \begin{array}{ccc} \{AD \to B\} & \Rightarrow & \{AD \to C\}, \\ \{AD \to C\} & \Rightarrow & \{AD \to B\} \end{array} \right\} \\ \Pi_{AE} & = & \left\{ \begin{array}{ccc} \{AE \to B\} & \Rightarrow & \{AE \to C\}, \\ \{AE \to B, BE \to C\} & \Rightarrow & \{AE \to C\}, \\ \{AE \to C\} & \Rightarrow & \{AE \to B\}, \\ \{BE \to A, AE \to C\} & \Rightarrow & \{BE \to C\}, \\ \{BE \to A\} & \Rightarrow & \{BE \to C\} \end{array} \right\} \end{array}$$

The partial canonical covers can now be computed as minimal keys w.r.t. the relative implication covers:

$$\begin{array}{rcl} CC_{AB} & = & \{\{AB \to C, AC \to B\}\} \\ CC_{AD} & = & \{\{AD \to B\}, \{AD \to C\}\} \\ CC_{AE} & = & \{\{AE \to B, BE \to A\}, \{AE \to C, BE \to A\}\} \end{array}$$

Together this gives us all four canonical covers in

$$\begin{array}{rcl} CC(\Sigma) & = & CC_{AB} \vee CC_{AD} \vee CC_{AE} \\ & = & \left\{ \begin{array}{c} \{AB \to C, AC \to B, AD \to B, AE \to B, BE \to A\}, \\ \{AB \to C, AC \to B, AD \to B, AE \to C, BE \to A\}, \\ \{AB \to C, AC \to B, AD \to C, AE \to B, BE \to A\}, \\ \{AB \to C, AC \to B, AD \to C, AE \to C, BE \to A\} \end{array} \right\} \end{array}$$

□

3.2.5 Improvements and Complexity Analysis

When using linear resolution to find CC_X, the most time consuming step of eliminating extraneous attributes from the LHS of an ID is actually that of eliminating redundant FDs from a partial atomic cover on EQ_X. When minimizing the LHS of a FD, we remove one attribute A at a time. We then check whether A is still determined by the remaining attributes by computing their closure, using the FDs in Σ. The corresponding approach for an ID $\mathcal{L} \Rightarrow EQ_X$ would be to remove one FD $Y \rightarrow A$ from its LHS $\mathcal{L}$, and compute the closure of the remaining FDs $\mathcal{L} \setminus \{Y \rightarrow A\}$ w.r.t. the relative implication cover Π_X.

However, since Π_X can be large compared to Σ, it is usually more efficient to check whether $\mathcal{L} \setminus \{Y \rightarrow A\}$ is still a partial cover using Lemma 3.30. Since we know that $\mathcal{L}$ is a partial cover, we only need to check whether $Y \rightarrow A$ is implied by

$$(\mathcal{L} \setminus \{Y \rightarrow A\}) \cup (\Sigma \setminus EQ_X)$$

This gives us the following simple procedure:

```
**proc** minimize-partial-cover($\mathcal{L}, EQ_X, \Sigma$)
for all $Y \rightarrow A \in \mathcal{L}$ do
    $\mathcal{L}' := (\mathcal{L} \setminus \{Y \rightarrow A\}) \cup (\Sigma \setminus EQ_X)$
    $Y^{*\mathcal{L}'}$ := closure of $Y$ w.r.t. $\mathcal{L}'$
    if $A \in Y^{*\mathcal{L}'}$ then
        $\mathcal{L} := \mathcal{L} \setminus \{Y \rightarrow A\}$
end
```

To establish an upper bound for the complexity of the “divide and resolve” approach, we use the variables

$$f = |\Sigma^{*a}|, n = |\Sigma|, k = |R|$$

as defined earlier, as well as

$$c = \max \{|CC_X| \mid EQ_X \in EQ\}$$

i.e., c is the maximum number of partial canonical covers over all EQ_X.

Computing Σ^{*a} can be done in $O(f \cdot k^2 n^2)$. Partitioning Σ^{*a} can be done in $O(f \cdot kn)$ by computing the closure of each LHS. Constructing all Π_X takes $O(f^2 \cdot k^2)$ using the closures computed before. Computing the sets CC_X by linear resolution without the optimization described above would lead to a complexity of $O(c \cdot f^6)$, using that $|\Pi_X|$ is bounded by f^2. Using the partial cover test instead leads to a worst-time complexity of $O(c \cdot f^4 \cdot k)$, which can be argued as follows: The number of LHS minimizations is bounded by $c \cdot \sum |\Pi_X| \leq c \cdot f^2$, and each minimization requires at most f redundancy tests. Each redundancy test requires one closure computation relative to a subset of Σ^{*a}, which can be performed in $O(f \cdot k)$. The overall computation time is therefore bounded by the term

$$O(c \cdot f^4 \cdot k + f \cdot k^2 n^2 + f^2 \cdot k^2)$$

If we assume that n, k are bounded by f, which holds in all but some “lucky” cases (”lucky” because this means f is very small indeed), this can be simplified to

$$O(c \cdot f^4 \cdot k)$$

3.2.6 LR-reduced Covers

The general purpose of finding all canonical covers is to find covers which are best in some sense. This approach only works if we can be sure that the best cover (or at least one of the best if there are multiple optimal solutions) for a given problem is canonical. This is often the case, but not always. Another common type of cover are *LR-reduced* covers.

Definition 3.50. A set Σ of FDs is called *LR-reduced* if no attribute can be removed from any FD in Σ while maintaining the property of being a cover.

While we only computed the set $CC(\Sigma)$ of all canonical covers, the set of all LR-reduced covers can be constructed from $CC(\Sigma)$ easily. By "splitting" a FD $X \to Y$ into singular FDs, we mean to replace it by $\{X \to A \mid A \in Y\}$.

Lemma 3.51. *[32] Splitting FDs into singular FDs turns an LR-reduced cover into a canonical cover. Combining FDs with identical LHSs turns a canonical cover into an LR-reduced cover.*

As an example for the usefulness of LR-reduced covers, one may consider the problem of finding a cover with minimal number of attributes, which is known to be NP-hard [32].

Definition 3.52. The *area* of a FD $X \to Y$ is $|X| + |Y|$, i.e., the number of attributes appearing in $X \to Y$. The *area* of a set of FDs Σ is the sum of the areas of all FDs in Σ.

Definition 3.53. A set Σ of FDs is called *area optimal* if there exists no cover G of Σ with smaller area.

Area optimal covers can help in reducing the workload for checking whether dependencies hold on a relation, as well as speed up various algorithms [32].

Area optimal covers are rarely canonical, since combining FDs with equal LHS reduced the area. It is easy to see though that they are always LR-reduced.

Lemma 3.54. *[32] An area optimal set of FDs is LR-reduced.*

Thus, we may combine FDs with identical LHSs to obtain all LR-reduced covers in which no FDs have identical LHSs. Clearly those include all area optimal covers. We may determine all area optimal partial covers for each EQ_X separately to get a concise representation for all area optimal covers.

In [4] Ausiello, D'Atri and Sacca introduce several similar minimality criteria for covers, and show that some of the corresponding decision problems are NP-hard. While these minimal covers are not always canonical or LR-reduced, it is easy to see that some of them always are. Thus, while our approach may not find all minimal covers (w.r.t. the minimality criteria in [4]), we can always find some. In particular, optimal covers are always minimal w.r.t. the criteria from [4].

3.2.7 Related Work

Maier already noted in [32] that there is a correspondence between the equivalence classes of non-redundant covers. Our work generalizes these results by investigating the projections of (canonical) covers onto arbitrary autonomous sets, and placing them into a more general theoretic framework.

A unique representation for a set of unitary FDs is given in [26]. This representation is obtained by factoring the attribute determination graph via the equivalence relation on attributes induced by Σ. From this, partial canonical covers for the equivalence classes could be constructed as minimal strongly connected directed graphs.

In [28] the authors describe a unique representation for a set of functional dependencies, which is independent of the choice of cover.

Ausiello, D'Atri and Sacca introduce a graph representation to describe sets of FDs in [3], and use hypergraphs for describing them in [4]. However, their approach is completely different from ours, as their vertices are attributes or attribute sets, while each FD is modeled as a (hyper)edge.

3.3 Size and Number of Non-redundant Covers

In this section we establish some results concerning how much the number of FDs in two non-redundant covers can differ, and how many non-redundant covers a set of FDs can have.

Theorem 3.55. *Let Σ, G be equivalent non-redundant sets of FDs over R. Then $|G| \leq |\Sigma| \cdot |R|$.*

Proof. Since Σ and G are equivalent, every FD $X \to A$ in Σ is implied by G. When computing the closure of X using G, we only use FDs which contribute at least one new attribute. Thus G includes a subset $G_X \subseteq G$ of cardinality at most $|R \setminus X|$ which implies $X \to A$. Since their union $\bigcup\limits_{X \to A \in F} G_X \subseteq G$ is already a cover of Σ and G is non-redundant, $\bigcup\limits_{X \to A \in \Sigma} G_X = G$. Clearly the cardinality of $\bigcup\limits_{X \to A \in \Sigma} G_X$ is bounded by $|\Sigma| \cdot |R|$. □

In [19] Gottlob shows the bound $|G| \leq |\Sigma| \cdot (|R| - 1)$ for FDs with non-empty LHSs, and gives the following example to show that the bound is tight.

Example 3.9. Let $R = \{A, B_1, \ldots, B_n\}$ and $\Sigma = \{A \to B_1 \ldots B_n\}$. Then

$$G = \{A \to B_1, \ldots, A \to B_n\}$$

is a non-redundant cover of Σ, and $|G| = n = |\Sigma| \cdot (|R| - 1)$. □

While the example above was based on splitting non-singular FDs into singular ones, the next example shows that the bound of Theorem 3.55 cannot be improved significantly even if we restrict ourselves to canonical covers. Note though that $|\Sigma| = |G|$ if we restrict ourselves to non-redundant covers with FDs of the form $X \to X^*$, since these are actually minimal covers [40].

Example 3.10. Consider the relation schema $R = \{A_1, \ldots, A_n, B_1, \ldots, B_n, C\}$ with $2n+1$ attributes. Associate with R the canonical set of FDs

$$\Sigma = \left\{ \begin{array}{l} A_1 \to C, \ldots, A_n \to C, \\ C \to B_1, \ldots, C \to B_n, \\ B_1 \ldots B_n \to C \end{array} \right\}$$

of size $2n+1$. The set

$$G = \left\{ \begin{array}{l} A_1 \to B_1, \dots, A_n \to B_1, \\ \vdots \\ A_1 \to B_n, \dots, A_n \to B_n, \\ C \to B_1, \dots, C \to B_n, \\ B_1 \dots B_n \to C \end{array} \right\}$$

is a canonical cover of Σ and contains n^2+n+1 FDs. Thus $|G| > \frac{1}{4} \cdot |\Sigma| \cdot |R|$. □

Using the bound established in Theorem 3.55, we will argue why it can make sense to try and compute all canonical covers.

We first note that the number of arbitrary covers for a set Σ of FDs over a schema R can be (and usually is) hyper-exponential in the number of attributes. This is the case since the number of FDs in Σ^* can be exponential in the number of attributes, and the number of covers can be exponential in the number of FDs in Σ^*. Even by restricting ourselves to the "most powerful" FDs of the form $X \to X^*$ (with minimal LHS X) and requiring covers to be non-redundant (which together with the restriction on the form of FDs makes them minimal), we cannot avoid this. For brevity, we shall call non-redundant sets of LHS-reduced FDs of the form $X \to X^*$ *full*.

Example 3.11. Consider $R = \{A_1, \dots, A_{2n}\}$ and let Σ consist of all FDs $X \to R, X \subseteq R$ with $|X| = n$. Clearly Σ is non-redundant and contains $\binom{2n}{n}$ FDs. However, Σ has no full covers (other than itself). To create a large number of full covers, we add two extra attributes A and B to R, which gives us $R' = R \cup \{A, B\}$, and change Σ to

$$\Sigma' = \{AX \to R' \mid X \to R \in \Sigma\} \cup \{A \to B, B \to A\}.$$

It is easy to check that Σ' is full. However, each FD $AX \to R'$ can be replaced by $BX \to R'$, and these replacements can be done independently from one another. Thus Σ has at least $2^{\binom{2n}{n}}$ full covers. □

We are thus trying to compute a potentially hyper-exponential number of covers, which at first glance seems rather infeasible even for small cases. However, when constructing the example above, we used a set of FDs Σ whose size is exponential in the number of attributes. This is rather unusual, and for small sets Σ we can establish better bounds for the number of non-redundant covers.

Theorem 3.56. *Let Σ be a set of FDs over R, with cardinalities $|\Sigma| = n|, |R| = k$. The number of non-redundant covers of Σ is at most $\binom{2^{2k}}{nk} < 2^{2nk^2}$.*

Proof. The number of FDs on R is 2^{2k}, and any non-redundant cover of Σ contains at most nk FDs by Theorem 3.55. □

While the bound given can be improved, the relevant fact is that the number of covers is "only" exponential in the size of the input, rather than hyper-exponential. When considering arbitrary non-redundant covers, we can obtain different non-redundant covers by changing FDs slightly, e.g. by adding LHS attributes to the RHS. As such changes can be done independently from one another, the number of non-redundant covers *is* practically always exponential. Many of those variations are avoided by restricting ourselves to canonical covers. Using partial covers to represent them efficiently, we can hope to reduce the size of our representation to a reasonably small number. Experimental results can be found in the appendix.

3.4 Partial Implication Cycles

We wish to find a partition of Σ^{*a} into autonomous sets which is finer than EQ. This is motivated by the observation that schemas with multiple minimal keys can easily have a large number of canonical covers, which cannot be represented efficiently using the partition EQ. Autonomous sets of a finer partitions have a smaller number of partial covers on them.

Example 3.12. Consider the set

$$\Sigma = \{A \to BC_1 \dots C_n, B \to A\}$$

with the atomic closure

$$\Sigma^{*a} = \left\{ \begin{array}{l} A \to B, A \to C_1, \dots, A \to C_n, \\ B \to A, B \to C_1, \dots, B \to C_n \end{array} \right\}.$$

Each canonical cover of Σ contains $A \to B, B \to A$ and for each $i = 1 \dots n$ either $A \to C_i$ or $B \to C_i$, for a total of 2^n canonical covers. The LHSs A and B are equivalent, so the partition $EQ = \{\Sigma^{*a}\}$ is trivial. However, $CC(\Sigma)$ can be represented efficiently using a finer decomposition:

$$\begin{array}{rl} CC(\Sigma) = & \{\{A \to B, B \to A\}\} \vee \\ & \{\{A \to C_1\}, \{B \to C_1\}\} \vee \\ & \dots \\ & \{\{A \to C_n\}, \{B \to C_n\}\} \end{array}$$

□

The example above is not a rare case - one can expect to frequently find schemas with multiple minimal keys in practice. It is thus vital to find a good partition for cases similar to the last example. For this, we use the same idea as in section 2.2.3. We define adjacency of FDs in Σ^{*a} w.r.t. the hypergraph $Tr(CC(\Sigma))$.

Definition 3.57. We call FDs *adjacent* iff they lie in a common minimal transversal.

Note that by Lemma 3.31 adjacent FDs can be used to derive each other, i.e., each lies in a minimal subset of Σ^{*a} which implies the other FD. If we represent the relation between FDs "partially implies" by a directed graph, we find that adjacent FDs lie on a directed cycle.

Definition 3.58. Let $X \to A, Y \to B \in \Sigma^{*a}$. We say that $X \to A$ *partially implies* $Y \to B$, written $X \to A \overset{p}{\Rightarrow} Y \to B$ if $X \to A$ lies in a minimal $S \subseteq \Sigma^{*a}$ with $S \models Y \to B$. We denote the relation "partially implies" by $PI_\Sigma \subseteq \Sigma^{*a} \times \Sigma^{*a}$.

Lemma 3.59. *Let $X \to A, Y \to B \in \Sigma^{*a}$ be adjacent. Then $X \to A$ and $Y \to B$ partially imply each other.*

Proof. Follows from Lemma 3.31. □

Since relations can be regarded as directed graphs, we use terminology from graph theory. The last lemma then tells us that adjacent FDs lie on a directed cycle of PI_Σ. This gives us the following.

Theorem 3.60. *The maximal strongly connected components (MSCs) of PI_Σ are autonomous sets of $CC(\Sigma)$.*

Proof. Let $S \subseteq \Sigma^{*a}$ be (the vertex set of) a MSC of PI_Σ. By Lemma 3.59 every minimal transversal that intersects with S lies fully in S. Thus S is the union of maximal connected components of $Tr(CC(\Sigma))$, which are minimal autonomous sets of $CC(\Sigma)$ by Theorem 3.18. This makes S autonomous by Theorem 3.11. □

Theorem 3.60 provides us with an alternative proof for Theorem 3.35: By Lemma 2.35 the partial implication graph PI_Σ contains only arcs where the LHS of the target FD determines the LHS of the source FD. Thus there are no cycles between FDs in different equivalence classes, which means that every equivalence class is the union of MSCs of PI_Σ.

The partition of Σ^{*a} formed by the MSCs of PI_Σ is at least as fine as the one by equivalence classes and often finer (example 3.12), but it need not be minimal. This is because PI_Σ contains "extra" arcs between non-adjacent FDs, which can lead to larger MSCs. Extra arcs can make the partition found less fine, but it is still a partition into autonomous sets.

Example 3.13. Consider the set of FDs

$$\Sigma^{*a} = \left\{ \begin{array}{l} A \to B, C \to D, BD \to A, BD \to C, \\ E \to A, E \to B, E \to C, E \to D \end{array} \right\}$$

The subgraph of PI_Σ induced by the equivalence class

$$EQ_E = \{E \to A, E \to B, E \to C, E \to D\}$$

can be drawn as

$$\begin{tikzcd} (E \to A) \arrow[d, shift left] & (E \to D) \arrow[l] \arrow[d, shift left] \\ (E \to B) \arrow[u, shift left] \arrow[r] & (E \to C) \arrow[u, shift left] \end{tikzcd}$$

so EQ_E is a MSC of PI_Σ. However, the sets $\{E \to A, E \to B\}$ and $\{E \to C, E \to D\}$ are smaller autonomous sets:

$$\begin{array}{rl} CC(\Sigma) = & \{\{A \to B, C \to D, BD \to A, BD \to C\}\} \vee \\ & \{\{E \to A\}, \{E \to B\}\} \vee \\ & \{\{E \to C\}, \{E \to D\}\} \end{array}$$

□

We still have the problem of constructing PI_Σ. We will solve it with the help of implication covers.

Definition 3.61. Let Σ be a set of FDs. The *atomic implication closure* $AIC(\Sigma)$ of Σ is the atomic closure of any implication cover of Σ:

$$AIC(\Sigma) := \left\{ S \Rightarrow \{Y \to B\} \;\middle|\; \begin{array}{l} S \subseteq \Sigma^{*a}, Y \to B \in \Sigma^{*a} \setminus S, \\ S \text{ minimal with } S \models Y \to B \end{array} \right\}$$

This allows us to describe partial implication w.r.t. $AIC(\Sigma)$.

Definition 3.62. We say that $X \to A$ partially implies $Y \to B$ w.r.t. some set of implication dependencies Π iff Π contains an ID with $X \to A$ in its left-, and $Y \to B$ in its right hand side.

With this definition we get:

Proposition 3.63. *Let Σ be a set of FDs, and $X \to A, Y \to B \in \Sigma^{*a}$. Then $X \to A$ partially implies $Y \to B$ iff $X \to A$ partially implies $Y \to B$ w.r.t. $AIC(\Sigma)$.*

This means that partial implication is the same as partial determination, as defined in section 2.2.3, but for $AIC(\Sigma)$. That is, we have

$$PI_\Sigma = \mathcal{R}_{AIC(\Sigma)}$$

Since we are only interested in the MSCs of PI_Σ, we may as well use its transitive closure PI_Σ^+ instead, or any other relation with PI_Σ^+ as its transitive closure. By Theorem 2.30 such a relation can be constructed using any atomic implication cover of Σ.

We have seen earlier that implication covers of Σ can be large, even if Σ^{*a} is small. On the other hand, the size of relative implication covers of equivalence classes EQ_X of Σ^{*a} is at most quadratic in Σ^{*a}. We thus want to use relative implication covers instead, which we can construct easily using Theorem 3.49.

Lemma 3.64. *Let $H \subseteq \Sigma^{*a}$. Then the partial implication graph w.r.t. the relativation of the atomic implication closure $AIC(\Sigma)\,]H[$ is the subgraph of PI_Σ induced by H.*

Proof. PI_Σ is the partial implication graph w.r.t. $AIC(\Sigma)$ by Lemma 3.63. Relativating $AIC(\Sigma)$ onto H removes the FDs not in H, and thus removes from PI_Σ the arcs adjacent to FDs not in H. Clearly the result is the subgraph of PI_Σ induced by H. □

Example 3.14. Consider the set of FDs

$$\Sigma^{*a} = \{A \to B, A \to C, B \to C, C \to B\}$$

The atomic implication closure of Σ is

$$AIC(\Sigma) = \left\{ \begin{array}{c} \{A \to B, B \to C\} \Rightarrow \{A \to C\}, \\ \{A \to C, C \to B\} \Rightarrow \{A \to B\} \end{array} \right\}$$

This gives us the partial implication graph PI_Σ:

$$\begin{tikzcd} (A \to B) \arrow[d, shift left] & (C \to B) \arrow[l] \\ (A \to C) \arrow[u, shift left] & (B \to C) \arrow[l] \end{tikzcd}$$

Relativating $AIC(\Sigma)$ onto the set $H = \{A \to B, A \to C\}$ gives us

$$AIC(\Sigma)\,]H[= \left\{ \begin{array}{c} \{A \to B\} \Rightarrow \{A \to C\}, \\ \{A \to C\} \Rightarrow \{A \to B\} \end{array} \right\}$$

which induces the partial implication graph w.r.t. $AIC(\Sigma)\,]H[$:

$$\begin{tikzcd} (A \to B) \arrow[d, shift left] \\ (A \to C) \arrow[u, shift left] \end{tikzcd}$$

which is also the subgraph of PI_Σ induced by H. □

Theorem 3.65. *Let $H \subseteq \Sigma^{*a}$ and Π_H an atomic relative implication cover of Σ^{*a} on H. Let further PI_H be the partial implication graph of H w.r.t. Π_H. Then the MSCs of PI_H are subsets of the MSCs of PI_Σ.*

Proof. Let $PI_\Sigma[H]$ be the subgraph of PI_Σ induced by H. The MSCs of $PI_\Sigma[H]$ are clearly subsets of the MSCs of PI_Σ, and we will show that they include the MSCs of PI_H.

By Lemma 3.64 $PI_\Sigma[H]$ is the partial implication graph w.r.t. $AIC(\Sigma)\,]H[$. Since $AIC(\Sigma)$ is an implication cover of Σ^{*a}, $AIC(\Sigma)\,]H[$ is a relative implication cover of Σ^{*a} on H by Corollary 3.45. Note that $AIC(\Sigma)\,]H[$ need not be atomic, as the relativation of an atomic ID need not be atomic anymore (w.r.t. the relativated implication cover). It could however be transformed into an *atomic* relative implication cover Π'_H by removing IDs with empty right hand side, and LHS-minimizing the remaining IDs. For the corresponding partial implication graph PI'_H w.r.t. Π'_H this means that some arcs got removed, but no new arcs are added, so the MSCs of PI'_H are subsets of the MSCs of $PI_\Sigma[H]$.

PI'_H and PI_H are both atomic relative (implication) covers on the same set H, so by Lemma 3.39 they are atomic covers of each other. Thus their MSCs are identical by Theorem 2.30. □

Theorem 3.65 allows us to use atomic relative implication covers to find a partition which is at least as fine and possibly finer than the partition into autonomous sets induced by PI_Σ. We will show next that the sets in this finer partition are still autonomous, so that the use of relative implication covers not only makes the construction of an autonomous partition faster, but also leads to a better (i.e., finer) partition.

Theorem 3.66. *Let $H \subseteq \Sigma^{*a}$ be an autonomous set, and Π_H a relative implication cover on H. Let further PI_H be the partial implication graph w.r.t. Π_H. Then the MSCs of PI_H are autonomous sets.*

Proof. We may assume Π_H to be atomic, as making it atomic does not add arcs to PI_H and thus may only lead to a finer partition. By Theorem 2.30 we may then assume Π_H to be its atomic closure, since this does not affect the MSCs of PI_H.

We only show Lemma 3.59 for partial implication w.r.t. PI_H, the argument then continues as in Theorem 3.60. So let $X \to A, Y \to B \in H$ be adjacent. Then there exists some minimal transversal S with $X \to A, Y \to B \in S$, and by Lemma 3.31 we have

$$(\Sigma^{*a} \setminus S) \cup \{X \to A\} \models Y \to B$$

Thus $U \Rightarrow \{Y \to B\} \in \Pi_H$ for some minimal $U \subseteq (\Sigma^{*a} \setminus S) \cup \{X \to A\}$, and since $X \to A \in U$ by Lemma 3.31, $X \to A$ partially determines $Y \to B$ w.r.t. Π_H. □

We summarize the steps needed to compute a partition of Σ^{*a} into autonomous sets below.

Algorithm "partial implication partitioning"

INPUT: set of FDs Σ
OUTPUT: partition of Σ^{*a} into autonomous sets

```
compute $\Sigma^{*a}$ and partition it into equivalence classes $EQ$
for each $EQ_X \in EQ$ do
    construct relative implication cover $\Pi_X$ of $EQ_X$
    make $\Pi_X$ atomic
    construct $PI_X$ from $\Pi_X$
    compute MSCs of $PI_X$
end
```

Note again that the MSCs of a directed graph can be computed in linear time [41].

If PI_X is not strongly connected, we can try to partition its MSCs $msc_1, \ldots, msc_n$ again, using a relative implication cover on each msc_i, which by Theorem 3.42 can easily be obtained by relativating Π_X onto msc_i. This can be repeated until no finer partition is found.

Instead of using partial implication cycles to find autonomous sets of $CC(\Sigma)$, we can use partial determination cycles to find autonomous sets of the hypergraph of all minimal keys:

$$H = \{X \subseteq R \mid X \text{ is a minimal key of } R\}$$

For this we start we the trivial autonomous set R, and use Σ instead of an implication cover.

Saiedian and Spencer describe a similar approach in [39]. Using their algorithm for computing all partial canonical covers as the minimal keys w.r.t. Π_X is equivalent to determining smaller autonomous sets through partial implication cycles first, and then using some other algorithm (e.g. [31]) to find the minimal keys on them.

In [20] Gottlob, Pichler and Wei also use partial determination cycles to identify autonomous sets of H. This simplifies the test whether an attribute $A \in R$ is prime w.r.t Σ, since one only needs to check whether it is prime w.r.t. $\Sigma]S[$, where S is the autonomous set containing A.

3.5 Essential FDs

We are now interested in FDs that appear in some (or all) canonical covers.

Definition 3.67. We call an atomic FD $X \to A \in \Sigma^{*a}$ $\left\{\begin{array}{l} \textit{forced} \\ \textit{essential} \\ \textit{inessential} \end{array}\right\}$ iff it appears in $\left\{\begin{array}{l} \text{every} \\ \text{some} \\ \text{no} \end{array}\right\}$ canonical cover of Σ.

Note that essential FDs correspond to prime attributes, and that a FD is essential iff it lies in some minimal transversal.

It would be desirable if we could compute all essential FDs without computing inessential ones as well. This would allow us to speed up computations of all or specific (e.g. uncritical) canonical covers if the number of inessential FDs is large, as e.g. in example 2.12, where we had

$$\Sigma = \{A_1 \to B_1, \ldots, A_n \to B_n, B_1 \ldots B_n \to C\}$$

This leads to two different problems: “How can we decide whether a FD is essential?” and “How can we avoid computing inessential ones?”. In the following we will develop some answers to these questions.

3.5.1 Deriving essential FDs

As with adjacency, we define the neighborhood of FDs in Σ^{*a} w.r.t. the hypergraph $Tr(CC(\Sigma))$.

Definition 3.68. For $G \subseteq \Sigma^{*a}$ we define the *neighborhood* $N(G)$ of G as

$$N(G) := \{X \to A \in \Sigma^{*a} \mid \exists Y \to B \in G.X \to A \text{ adjacent to } Y \to B\}$$

In section 3.2.1 we already established some results about the minimal transversals of $CC(\Sigma)$. We will now show that they are cartesian products of LHSs and RHSs. In this, we use the following definition.

Definition 3.69. We call a set Σ of FDs *cartesian* if

$$X \to A, Y \to B \in \Sigma \quad \Rightarrow \quad X \to B \in \Sigma$$

In the following, transversal will always mean transversal of $CC(\Sigma)$.

Theorem 3.70. *All minimal transversals are cartesian.*

Proof. Let $X \to A, Y \to B$ be contained in a common minimal transversal $S \subset \Sigma^{*a}$. Since $X \leftrightarrow Y$ under $\overline{S} = \Sigma^{*a} \setminus S$ by Lemma 3.33 we have $X \to B \in \Sigma^{*}$. Furthermore $X \to B$ is non-trivial since $\overline{S} \not\models Y \to B$, which means there exists $U_X \subset X$ with $U_X \to B \in \Sigma^{*a}$. Assume $U_X \to B \notin S$. Then $\overline{S} \models Y \to X, U_X \to B$ and therefore $\overline{S} \models Y \to B$, which contradicts that S is a minimal transversal. Thus $U_X \to B \in S$. Again by Lemma 3.33 we have $U_X \leftrightarrow X$. Since X is a minimal LHS, this is only possible if $U_X = X$. □

Definition 3.71. We denote the *essential closure* of Σ by

$$\Sigma^{*e} := \{X \to A \in \Sigma^{*a} \mid X \to A \text{ is essential}\}$$

We call a set $G \subseteq \Sigma^{*a}$ *essentially cartesian* (w.r.t. Σ) iff

$$X \to A, Y \to B \in G, X \to B \in \Sigma^{*e} \quad \Rightarrow \quad X \to B \in G$$

The *essentially cartesian closure* G^{*ec} of $G \subseteq \Sigma^{*a}$ is the smallest essentially cartesian superset of G:

$$G^{*ec} := G \cup \{X \to B \in \Sigma^{*e} \mid \exists X \to A, Y \to B \in G\}$$

Definition 3.72. Let $G, H \subseteq \Sigma^{*a}$. We say that G *dominates* H if every minimal transversal which intersects with H intersects with G as well[1]. An atomic FD $X \to A$ dominates (is dominated) iff $\{X \to A\}$ dominates (is dominated). If two FDs dominate each other, we call them *equally dominating.*

[1]The property defined is more closely related to the vertex covering property than to the dominating set property. However, as the term ”covers” already has a distinct meaning for sets of FDs, we use ”dominates” to avoid confusion.

Lemma 3.73. *Let $G, H \subseteq \Sigma^{*a}$. If $G \models H$ then $G \cap N(H)$ dominates H.*

Proof. As only FDs in $N(H)$ can contribute to dominating H, it suffices to show that G dominates H. Assume the contrary, and let S be a minimal transversal containing a FD $X \to A \in H$ which does not intersect with G. As $G \models X \to A$ this contradicts Lemma 3.31. □

Lemma 3.74. *Let $X \to A, Y \to A \in \Sigma^{*e}$ be essential. Then $X \to A$ dominates $Y \to A$ iff $\Sigma^{*a} \setminus N(Y \to A) \models Y \to X$.*

Proof. If $X \to A$ dominates $Y \to A$ then $\Sigma^{*a} \setminus N(Y \to A) \cup \{X \to A\}$ intersects with every minimal transversal and thus is a cover of Σ. Argue as in Lemma 3.33.

On the other hand, let $\Sigma^{*a} \setminus N(Y \to A) \models Y \to X$. Then $\Sigma^{*a} \setminus N(Y \to A) \cup \{X \to A\}$ implies $Y \to A$ and $X \to A$ dominates $Y \to A$ by Lemma 3.73. □

When using linear resolution to compute the atomic closure, we would like to avoid intermediate inessential FDs, just as we avoided intermediate non-atomic ones by minimizing LHSs. This is possible, provided we can recognize inessential FDs.

Theorem 3.75. *Let Σ be essentially cartesian. Then every (essential) FD $X \to A \in \Sigma^{*a}$ can be obtained from Σ by linear resolution (with LHS-minimization of derived FDs) using only base FDs which dominate $X \to A$. The choices for LHS-minimization[2] are irrelevant.*

Proof. Trivial if $X \to A \in \Sigma$ or $X \to A$ inessential. Otherwise let $M \subseteq \Sigma$ be of minimal cardinality with $M \models X \to A$, and let $X_1 \to A_1, \ldots, X_n \to A \in M$ be the FDs in M in the order as they are used in computing the closure of X. Then there exists a linear resolution tree with LHS-minimized intermediate FDs - note that we make no assumption about which LHS-minimization is chosen - which uses only FDs in $M \setminus \{X_n \to A\}$ as substituting FDs in (1). Let $Y \to A \in \Sigma^{*a}$ be any (intermediate) base FD used in the resolution. We need to show that $Y \to A$ dominates $X \to A$. We have

$$T := \{Y \to A\} \cup M \setminus \{X_n \to A\} \models X \to A$$

and thus $T \cap N(X \to A)$ dominates $X \to A$ by Lemma 3.73.

Assume $X \to A$ were adjacent to $X_i \to A_i \in M \setminus \{X_n \to A\}$. Then $X_i \to A \in \Sigma^{*e}$ by Theorem 3.70, and thus $X_i \to A \in \Sigma$ since Σ is essentially cartesian. Since $X_1 \to A_1, \ldots, X_{i-1} \to A_{i-1}$ imply $X \to X_i$, the set $M' := \{X_1 \to A_1, \ldots, X_{i-1} \to A_{i-1}, X_i \to A\} \subseteq \Sigma$ implies $X \to A$ and is of smaller cardinality then M. Thus $X \to A$ is not adjacent to any FD in $M \setminus \{X_n \to A\}$, which means $T \cap N(X \to A) = \{Y \to A\}$. □

Definition 3.76. We call an essential FD *essentially derivable* from Σ, iff it can be derived from Σ using linear resolution (with LHS-minimization) without intermediate inessential FDs.

Corollary 3.77. *Let Σ be essentially cartesian. Then all essential FDs in Σ^{*e} are essentially derivable from Σ.*

Proof. Every FD that dominates an essential FD is essential itself. □

[2]E.g. it might be possible to LHS-minimize $AB \to C$ to $A \to C$ or $B \to C$.

We may thus discard any inessential FD we obtain during the linear resolution process. The prerequisite that Σ be essentially cartesian can easily be met by computing the essential cartesian closure of a canonical cover of Σ (or, if testing essential is too hard, adding all atomic FDs $X \to B$ with $X \to A, Y \to B \in \Sigma$).

Furthermore, Theorem 3.75 assures us that the base FDs used in the linear resolution process all dominate the final FD $X \to A$. Thus they are all pairwise adjacent, and we only need to consider derivations in which base and derived FD are adjacent (after LHS-minimization).

It is of interest to know whether the requirement in Theorem 3.75 that Σ be essentially cartesian is really necessary. The following example show that it is.

Example 3.15. Consider the canonical cover

$$\Sigma = \{B \to A, CD \to B, DE \to C, AE \to C\}$$

It is easily checked that $DE \to A \in \Sigma^{*a}$ is essential:

$$\Sigma' = \{B \to A, CD \to B, DE \to A, AE \to C\}$$

forms a canonical cover of Σ. The only linear derivation tree which derives $DE \to A$ from Σ is

$$\frac{DE \to C \quad \frac{CD \to B \quad B \to A}{CD \to A}}{DE \to A}$$

The FDs $CD \to B, B \to A$ are forced, as they are not implied by the remaining FDs in

$$\Sigma^{*a} = \left\{ \begin{array}{c} B \to A, CD \to B, DE \to C, AE \to C, \\ CD \to A, DE \to B, BE \to C, DE \to A \end{array} \right\}$$

This shows that $CD \to A$ is inessential, as it is implied by a set of forced FDs[3]. Thus we cannot derive $DE \to A$ without intermediate inessential FDs. □

In the example above, the essential FD $DE \to A$ was contained in the essentially cartesian closure of Σ. To show that this is not always the case, we could simply add the FDs $F \to E, E \to F$ to Σ in example 3.15, and try to derive the essential FD $DF \to A$. However, the essential FD $DF \to C$ could still be derived directly, without intermediate inessential FDs:

$$\frac{F \to E \quad DE \to C}{DF \to C}$$

so that we still could get $DF \to A$ by computing the essentially cartesian closure of the set of all essentially derivable FDs. This is not by accident.

Theorem 3.78. *Let Σ be atomic and E be the set of all FDs essentially derivable from Σ. Then the essentially cartesian closure E^{*ec} of E includes Σ^{*e}, i.e. all essential FDs.*

Proof. Same as proof for Theorem 3.75, except that we change the RHS attribute of the final derived FD, not of the substituting FD. □

The last theorem provides an alternative for computing Σ^{*e} by computing the essentially cartesian closure for the final set of FDs rather than the initial one. While computing the essentially cartesian closure can be more time-consuming, due to the larger size of the final set, we start with a smaller set of base FDs which may speed up our linear resolution computation.

[3]Note that this is a sufficient but not necessary criteria.

3.5.2 Testing essentiality

Theorems 3.75 and 3.78 show that all essential FDs can be derived while avoiding inessential ones. However, to do this, we require a test to recognize inessential FDs.

We will develop such a test next, and start by providing an efficient criteria for testing whether a FD is forced.

Theorem 3.79. *An atomic FD $X \to A \in \Sigma^{*a}$ is not forced iff there exists a $B \in X$ with $X^* \setminus AB \to A \in \Sigma^*$.*

Proof. Let $X \to A$ be not forced, i.e., $\Sigma^{*a} \setminus \{X \to A\} \vDash X \to A$. By looking at the closure algorithm we see that there exists some $Y \to A \in \Sigma^{*a} \setminus \{X \to A\}$ with $Y \subseteq X^*$. As Y is a minimal LHS for A it cannot include X (else it were equal to X), so there must be some $B \in X \setminus Y$. Thus $Y \subseteq X^* \setminus AB$ which shows $X^* \setminus AB \to A \in \Sigma^*$.

Now let $X^* \setminus AB \to A \in \Sigma^*$ for some $B \in X$. For every $A_i \in X^* \setminus XA$ there exists a minimal set $U_i \subseteq X$ with $U_i \to A_i \in \Sigma^{*a} \setminus \{X \to A\}$. As well, there exists a minimal $U \subseteq X^* \setminus AB$ with $U \to A \in \Sigma^{*a} \setminus \{X \to A\}$. Together they imply $X \to A$, thus $X \to A$ is not forced. □

As all forced FDs appear in every canonical cover of Σ, computing the set of all forced FDs in Σ^{*a} is easy. What we really need though is a criterion for testing whether a FD is essential. This, however, is an NP-complete problem, which we will show by reducing the following problem to it, which is known to be NP-complete [31]:

Problem "prime attribute"

> Given a set Σ of FDs on schema R and an attribute $A \in R$, is A a prime attribute, i.e., does A lie in a minimal key of R?

Theorem 3.80. *Given a set Σ of FDs, the problem of deciding whether a FD is essential is NP-complete.*

Proof. To verify that a FD is essential, we only need to guess the canonical cover containing it. By Theorem 3.55 the size of any canonical cover is polynomial in Σ, so the problem lies in NP.

We prove completeness by reducing the "prime attribute" problem to it. Let G be a set of FDs on R, and $A \notin R$ an additional attribute. We construct Σ as

$$\Sigma := G \cup \{A \to A_i \mid A_i \in R\}$$

We claim that A_i is a prime attribute w.r.t. G iff $A \to A_i$ is essential w.r.t. Σ. Since A does not appear in any FD in G we have

$$\Sigma^{*a} = G^{*a} \cup \{A \to A_i \mid A_i \in R\}$$

Now let Σ' be any canonical cover of Σ, and let $\Sigma'_A := \{A \to A_i \in \Sigma'\}$. Clearly the FD $A \to R \in \Sigma^*$ is implied by Σ' iff $A \to K$ is implied by Σ'_A for some minimal key K of R (w.r.t. G). This is the case iff Σ'_A consists of exactly (since Σ' is non-redundant) those FDs $A \to A_i$ for which $A_i \in K$. Thus $A \to A_i$ is essential iff A_i is prime. □

From this, we can deduce that identifying the (minimal) autonomous sets of $CC(\Sigma)$ is difficult as well.

Theorem 3.81. *Given a set Σ of FDs and an atomic FD $X \to A \in \Sigma^{*a}$, the problem of deciding whether the set $\{X \to A\}$ is autonomous for $CC(\Sigma)$ is co-NP-complete.*

Proof. $\{X \to A\}$ is autonomous iff $X \to A$ appears in all or no canonical covers of Σ. Thus, if $\{X \to A\}$ is not autonomous, we only need to guess one canonical cover which contains $X \to A$, and one which does not. By Theorem 3.55 the size of these canonical covers is polynomial in Σ. This shows that the problem lies in co-NP.

To show co-NP-hardness, we use that it is NP-hard to decide whether a FD $X \to A$ is essential. Let Σ' be any canonical cover of Σ. Such a canonical cover Σ' can be computed in polynomial time. By definition, $X \to A$ is essential iff it appears in some, but not necessarily all canonical covers of Σ. We distinguish two cases.

(1) If $X \to A \in \Sigma'$ then $X \to A$ is essential.

(2) If $X \to A \notin \Sigma'$ then $X \to A$ is essential iff it appears in some but not all canonical covers of Σ, i.e., iff $\{X \to A\}$ is not autonomous.

We thus reduced the NP-hard problem of deciding whether $X \to A$ is essential to the problem of deciding whether $\{X \to A\}$ is not autonomous. □

The last theorem shows that the autonomous set problem is hard when given Σ. However, when we try to find autonomous sets of $CC(\Sigma)$, we first compute Σ^{*a}, which can be exponential in the size of Σ. Thus it might be possible to decide whether a given set is autonomous in time polynomial in the size of Σ^{*a}. We show next that this is not the case.

Theorem 3.82. *Given a set Σ of FDs, its atomic closure Σ^{*a} and an atomic FD $X \to A \in \Sigma^{*a}$, the problem of deciding whether the set $\{X \to A\}$ is autonomous for $CC(\Sigma)$ is co-NP-complete.*

Proof (Sketch). In [31] the NP-hardness of the "prime attribute" problem is shown by first reducing the "vertex cover" problem to the "key of cardinality m" problem, and then in turn to "prime attribute". We will describe a slightly modified version of this reduction.

Let G be the graph for the vertex cover, Σ_{card} the set of FDs for the "key of cardinality m" problem (denoted $D[0]'$ in [31]), and Σ_{prime} the set of FDs for the "prime attribute" problem (denoted $D[0]$ in [31]). For the first reduction, Σ_{card} is constructed as follows:

$$\Sigma_{card} := \{N(v) \to v \mid v \text{ is vertex in } G\}$$

where $N(v)$ is the neighborhood of v in G.

The second reduction is more complicated. We give a modified version next (the modification occurs in condition (ii) below), which can be shown to be correct as in [31]. Let A' be the set of attributes occurring in Σ_{card}, A'' of cardinality $m < |A'|$ and b a new attribute. The attribute set for the "prime attribute" problem is then $A = A' \cup [A'' \times A']$, and Σ_{prime} is constructed as follows:

(i) for $E \to F \in \Sigma_{card}$ add $\{b\} \cup E \to F$ to Σ_{prime}

(ii) for $e \in A'$ add $\{e\} \to A'' \times \{e\}$ to Σ_{prime}

(iii) for $i \in A''$ and $e \in A'$ add $\{b, (i, e)\} \to \{e\}$ to Σ_{prime}

(iv) for $i \in A''$ and $e, f \in A'$ distinct add $\{(i,e),(i,f)\} \to \{b\}$ to Σ_{prime}

(v) for $e \in A'$ add $\{e\} \to \{b\}$ to Σ_{prime}

Using these constructions, we want to establish some bounds on the size of the LHSs of FDs. This can be done by verifying the following statements.

- The size of the LHS of a FD in Σ_{card} is the degree of the corresponding vertex in G
- $\Sigma^{*a}_{card} = \Sigma_{card}$
- The LHS of a FD in Σ^{*a}_{prime} is at most one larger that the LHS of a FD in Σ^{*a}_{card}

The reductions from the "prime attribute" problem to the problems "essential FD" and then "autonomous set" do not increase the size of the LHSs of atomic FDs. Thus the maximal size of the LHSs of FDs in Σ^{*a} is at most one larger that the maximal degree of vertices in G. It has been shown in [17] that the vertex cover problem is still NP-hard for graphs of maximal degree 3. These cases reduce to instances of the "autonomous set" problem for which all FDs in Σ^{*a} contain at most 4 attributes in their LHS. But there exist less than k^5 such FDs, where k is the number of different attributes occuring in Σ, so the size of Σ^{*a} is polynomial in the size of Σ, and can therefore be computed in polynomial time by Lemma 2.13. This shows the theorem. $\square$

In the following we will construct some sufficient (but not necessary) criteria that a FD is inessential. Recall that $N(X \to A)$ denotes the neighborhood of $X \to A$ in $Tr(CC(\Sigma))$.

Lemma 3.83. *$X \to A \in \Sigma^{*a}$ is essential iff $\Sigma^{*a} \setminus N(X \to A) \not\models X \to A$.*

Proof. If $X \to A$ is essential then $\Sigma^{*a} \setminus N(X \to A) \cup \{X \to A\}$ intersects with every minimal transversal and thus is a cover of Σ. As $\Sigma^{*a} \setminus N(X \to A)$ is not a cover of Σ by Lemma 3.31, $\Sigma^{*a} \setminus N(X \to A)$ cannot imply $X \to A$.

If $X \to A$ is inessential then $N(X \to A) = \{X \to A\}$, and $\Sigma^{*a} \setminus \{X \to A\}$ implies $X \to A$. $\square$

Thus, if we can find a set $S \subseteq \Sigma^{*a} \setminus N(X \to A)$ with $S \models X \to A$ then $X \to A$ must be inessential. Instead of Σ^{*a} we use only Σ (making Σ canonical if needed). Determining which FDs in Σ are adjacent to $X \to A$ is no easier than testing essentiality (since every essential FD has an adjacent FD in Σ). However, we do have some criteria that assure that a FD $Y \to B$ is not adjacent to $X \to A$: Clearly all forced FDs are adjacent only to themselves, and adjacent FDs have equivalent LHSs by Lemma 3.33. Thus we construct S from all forced FDs and all FDs in Σ with LHS smaller[4] than X, as FDs with LHS not implied by X cannot be used in deriving $X \to A$. Note that we do not lose anything by using Σ instead of Σ^{*a}: The FDs in Σ with LHS smaller than X form a cover for the set of all FDs in Σ^{*a} with LHS smaller than X by Lemma 2.35.

We can adapt the basic "linear resolution" algorithm to compute all essential FDs (and possibly more due to the inaccuracy of our essentiality test) as follows:

[4]w.r.t. the determination pre-order induced by Σ

Algorithm "essential linear resolution"

```
INPUT: set of FDs $\Sigma$
OUTPUT: superset of $\Sigma^{*e}$


compute a canonical cover $\Sigma'$ of $\Sigma$
$F_\Sigma := \emptyset$
for all $X \rightarrow A \in \Sigma'$ do
    if is_forced($X \rightarrow A$) then
        add $X \rightarrow A$ to $F_\Sigma$
end
$\Sigma^{*e} := \Sigma'$
for all $Y \rightarrow B \in \Sigma^{*e}$ do
    for all $X \rightarrow A \in \Sigma'$ with $A \in Y, B \notin X$ do
        // derive $(XY \setminus A) \rightarrow B$ by rule (2.1)
        find $U \subseteq (XY \setminus A)$ with $U \rightarrow B$ atomic
        if $U \rightarrow B \notin \Sigma^{*e}$ and is_essential($U \rightarrow B$) then
            add $U \rightarrow B$ to $\Sigma^{*e}$
    end
end


// compute essentially cartesian closure
for all $X \in LHS(\Sigma^{*e})$ do
    $RHS_X := X^* \setminus X$
    for all $A \in X$ do
        remove $(X \setminus A)^*$ from $RHS_X$
    end
    for all $B \in RHS_X$ do
        if $X \rightarrow B \notin \Sigma^{*e}$ and is_essential($X \rightarrow B$) then
            add $X \rightarrow B$ to $\Sigma^{*e}$
    end
end

**function is_essential($X \rightarrow A$)**
$smaller(X \rightarrow A) := \{Y \rightarrow B \in \Sigma \mid Y \subseteq X^* \wedge X \not\subseteq Y^*\}$
if $F_\Sigma \cup smaller(X \rightarrow A)$ implies $X \rightarrow A$ then
  return false
else
  return true

**function is_forced($X \rightarrow A$)**
for all $B \in X$ do
    if $\Sigma \models X^* \setminus AB \rightarrow A$ then
        return false
end
return true
```

Other optimizations such as those used in the revised linear resolution algorithm can be used in computing the essential closure (or a superset thereof) as well. Also, the test

$$\text{is\_essential}(X \to B)$$

for computing the essentially cartesian closure can be performed for all values B at once, by computing the closure of X under $F_\Sigma \cup \mathit{smaller}(X \to B)$.

Chapter 4

Domination Normal Form

A common approach in designing relational databases is to start with a relation schema, which is then decomposed into multiple subschemas [10, 34, 38, 42]. A good choice of subschemas can be determined using integrity constraints defined on the schema, such as functional, multivalued or join dependencies.

Many normal forms proposed so far, such as BCNF, 4NF or KCNF, characterize the absence of redundancy [1, 44]. This is desirable for several reasons, foremost the avoidance of update anomalies [33] and minimization of storage space [9]. However, these normal forms have significant drawbacks. First, they only consider a single relation schema, instead of considering all schemas in a decomposition together. While this is not strictly true for the normal form proposed by Topor and Wang in [45], their approach only considers pairs of schemas at a time, and thus cannot capture redundancy across larger schema sets.

The common generalization to multiple schemas is that the whole schema collection is in that normal form, if every schema taken individually is. But this means that those normal forms cannot capture redundancy which exists across multiple relations. As a trivial example, we can duplicate a schema. Clearly the extra schema is then superfluous in the whole schema collection, but each schema taken individually may still be redundancy free. But even if no schemas or attributes in a schema collection are superfluous, the design may not be desirable.

Example 4.1. Let $R = ABCD$ and $\Sigma = \{AB \to CD, CD \to B\}$. Then R is not in BCNF, but has a dependency preserving BCNF decomposition into the subschemas ABC, ABD, BCD.

For any instance r of R, the projections of r onto the schemas ABC and ABD together already take up more space than the original relation r: no tuples are lost in the projection since AB is a key, and the attributes A and B are stored twice. While we have not defined what redundancy means for multiple schemas, it seems intuitively clear that this decomposition should not be called "redundancy free". From a storage space point-of-view, it is clearly less desirable than the original schema R. $\square$

The second big problem is that dependency preserving decompositions into these normal forms do not always exist [5]. Thus, when faced with such a case, a designer must either accept the loss of some dependencies, or cannot achieve the normal form in question.

We believe that what a normal form should do, is the following:

> Characterize "good" representations (i.e., decompositions) of a schema, in such a way that a "good" representation does always exist.

In the following we will propose a normal form which meets this criterion.

4.1 Minimization as Normal Form

The approach we suggest is the following: among a set of suitable decompositions (e.g. the set of all lossless, or lossless and dependency preserving decompositions), characterize the "best" ones. We do so by defining an order on the decompositions, such that the "best" decompositions are the minimal ones with respect to that order.

This leaves the question of when to call one decomposition better than another one. The motivation for many normal forms proposed so far has been the elimination of redundancy (and with it, the absence of update-anomalies). This may suggest to define a quantitative measure of redundancy over multiple schemas, as has been done by Arenas and Libkin in [1].

We take a different approach here: instead of trying to minimize redundancy, we try to minimize the size of instances. Intuitively this should lead to similar results, an assumption which is supported by the findings of Biskup in [9], but measures for size appear easier to construct than measures for redundancy (cf. [1]). In the following we will define and motivate three different orders on decompositions, all of which intuitively compare decompositions by the size of instances for them. By proving them to be equivalent, we will establish a syntactic characterization for a semantically motivated definition.

4.1.1 Ordering by Size of Instances

Our first approach measures the space required to store an instance. For that we need to know for each element of a domain how much storage space it requires. We represent this knowledge by associating with each domain Dom a size function

$$size : Dom \to \mathbb{N}$$

Consider e.g. the following domains:

- $STRING$ containing strings of arbitrary length
- $STRING[40]$ containing strings of length up to 40
- INT containing arbitrarily large integers
- $INT(64)$ containing all 64-bit integers
- $BOOLEAN$ containing the values $TRUE$ and $FALSE$

A realistic measure for the size of a string might be its length, the size of an integer i might be defined as $log(i)$, and the size of $TRUE$ and $FALSE$ might be one.

In this work we shall assume that all domains are infinite, and that the size functions on them are positive and unbounded, i.e., can grow arbitrarily large. This can be justified as follows: For any infinite domain a bounded size function is not realistic, since we can store only a finite number of elements when restricted to a fixed amount of space. While not all domains are truly infinite, they often contain far more elements than the number of subschemas in a typical decomposition (e.g. 256^{40} for $STRING[40]$ or 2^{64} for $INT(64)$).

Treating these domains as infinite will allow us to draw a sharp boundary between small (bounded) increases in size from duplicated attributes on one hand, and potentially large (unbounded) increases in size from instances with large numbers of tuples on the other.

We note that this argument fails for domains such as $BOOLEAN$, and that a constant size function may be more realistic for domains such as $STRING[40]$ or $INT(64)$. However, in this work we will not concern ourselves with such considerations.

As it will turn out, the assumptions about infinite domains and unbounded size functions, which are common assumptions in database theory [27], are all we need to characterize our new normal form, i.e., we do not require detailed knowledge about the actual size functions.

Definition 4.1. For a relation r over R and a decomposition $\mathcal{D} = \{R_1, \dots, R_n\}$ of $R = \bigcup R_j$ we denote the decomposition of r by $\mathcal{D}$ as

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

where $r[R_j]$ is the projection of r onto the attributes in R_j. When talking about the tuples in $r[\mathcal{D}]$ containing an attribute A, we will mean the tuples from relations $R_j \in \mathcal{D}$ with $A \in R_j$.

Definition 4.2. Let $R = \{A_1, \dots, A_k\}$ be a schema. For a finite relation r over R we define the size of r as

$$size(r) := \sum_{t \in r} \sum_{i=1}^{k} size(\pi_{A_i}(t))$$

where $\pi_{A_i}(t)$ denotes the projection of tuple t onto the attribute A_i. We then define the size of the decomposition of r by $\mathcal{D} = \{R_1, \dots, R_n\}$ of R as

$$size(r[\mathcal{D}]) := \sum_{j=1}^{n} size(r[R_j])$$

While this gives us a suitable definition of size for any instance, we wish to compare decompositions w.r.t. the size of all valid instances. If for every valid instance r on R a decomposition $\mathcal{D}_1$ requires no more storage space than a decomposition $\mathcal{D}_2$, then $\mathcal{D}_1 \leq \mathcal{D}_2$ should certainly hold, indicating that $\mathcal{D}_1$ is "at most as big" and thus "at least as good" as $\mathcal{D}_2$. Recall that we only consider suitable decompositions, in particular lossless or lossless and dependency preserving ones.

This alone, however, is not sufficient to characterize good decompositions: for an instance r containing only a single element, the trivial decomposition $\{R\}$ requires less storage space than any other lossless decomposition, as those typically need to duplicate some attributes. It would be hard to argue though that decomposition is never necessary. So how can we distinguish decompositions finer, based on the size of instances?

Example 4.2. Let $R = ABC$ and $\Sigma = \{B \to C\}$. R can be faithfully decomposed into $\mathcal{D} = \{AB, BC\}$. Clearly every relation r decomposed by $\mathcal{D}$ (which is a set of relations) is at most twice as large as r. On the other hand, for every natural number k we can construct a relation

$r =$

A	B	C
1	1	"a very long string"
2	1	"a very long string"
$\vdots$	$\vdots$	$\vdots$
$k+1$	1	"a very long string"

which is more than k times larger than in decomposed form:

$r[AB] =$

A	B
1	1
2	1
$\vdots$	$\vdots$
$k+1$	1

$r[BC] =$

B	C
1	”a very long string”

$\square$

This observation motivates the following definitions, which compare decompositions similarly to the “big-O” comparison (e.g. $3x^2 + x \in O(x^2)$) from complexity theory.

Definition 4.3. Let R be a schema with constraints Σ and $\mathcal{D}_1, \mathcal{D}_2$ be decompositions of R. We say that $\mathcal{D}_1$ *c-dominates* $\mathcal{D}_2$ (where ”c” stands for complexity) if there exists a constant k such that for all finite relations r over R that satisfy Σ we have

$$size(r[\mathcal{D}_1]) \leq k \cdot size(r[\mathcal{D}_2])$$

We further say that $\mathcal{D}_1$ *dominates*[1]$\mathcal{D}_2$ if the above relationship holds for $k = 1$. We abbreviate c-domination and domination as $\mathcal{D}_1 \leq_c \mathcal{D}_2$ and $\mathcal{D}_1 \leq \mathcal{D}_2$, respectively. We say that $\mathcal{D}_1$ *strictly (c-)dominates* $\mathcal{D}_2$, written $\mathcal{D}_1 <_{(c)} \mathcal{D}_2$, if $\mathcal{D}_1$ (c-)dominates $\mathcal{D}_2$ but not vice-versa.

It is easy to see that both domination and c-domination are reflexive and transitive, and thus are pre-orders. Clearly domination implies c-domination.

Proposition 4.4. *Let $\mathcal{D}_1, \mathcal{D}_2$ be decompositions of R. If $\mathcal{D}_1$ dominates $\mathcal{D}_2$ then $\mathcal{D}_1$ c-dominates $\mathcal{D}_2$.*

Note however that strict domination does *not* imply strict c-domination. In example 4.1 the original schema $ABCD$ strictly dominates the decomposition $\{ABC, ABD, BCD\}$, but both decompositions are equivalent w.r.t. c-domination. Sometimes both criteria, domination and c-domination, are used to characterize the best decomposition for a schema, as the following example shows.

Example 4.3. Let $R = ABCDE$ and $\Sigma = \{AB \to CD, B \to E\}$. Then the decomposition $\mathcal{D}_1 = \{ABC, ABD, BE\}$ is minimal w.r.t. c-domination but strictly dominated by $\mathcal{D}_2 = \{ABCD, BE\}$. The trivial decomposition $\{R\}$ is minimal w.r.t. domination but strictly c-dominated by both $\mathcal{D}_1$ and $\mathcal{D}_2$. $\square$

We are now ready to define our normal form based on the idea of minimizing storage space.

Definition 4.5. Let R be a schema with constraints Σ and $\mathcal{D}$ be a decomposition of R. We say that $\mathcal{D}$ is in *domination normal form* (DNF) if

(i) $\mathcal{D}$ is minimal w.r.t. domination, and

(ii) $\mathcal{D}$ is minimal w.r.t. c-domination

[1]This is *not* related to domination as defined in section 3.5

with minimal meaning that no strictly smaller decomposition exists among a given set of 'suitable' decompositions.

Note that this definition depends on the choice which decompositions we consider 'suitable'. We will investigate two different cases (though other choices might be of interest as well): the set of all lossless, and the set of all lossless and dependency preserving decompositions. In each case, we effectively obtain a different DNF.

As the number of suitable decompositions of a given schema is finite, there must exist a decomposition among them which is minimal w.r.t. domination, as well as a (possibly different) schema which is minimal w.r.t. c-domination. It is however not clear yet wether a decomposition into DNF always exists, i.e., one which is minimal w.r.t. both criteria at once. We will show this next.

Theorem 4.6. *Every schema has a decomposition into DNF.*

Proof. We use the fact that domination implies c-domination. The c-domination pre-order induces a partition of all the (suitable) decompositions of R into equivalence classes and defines a partial order on these equivalence classes. Let EQ be a minimal equivalence class w.r.t. that order. Choose $\mathcal{D}$ to be minimal w.r.t. domination among the decompositions in EQ. We claim that $\mathcal{D}$ is minimal w.r.t. domination among *all* decompositions of R, and thus in DNF. Let $\mathcal{D}'$ be any decomposition with $\mathcal{D}' \leq \mathcal{D}$. Then $\mathcal{D}' \leq_c \mathcal{D}$, and since $\mathcal{D}$ is minimal w.r.t. c-domination, $\mathcal{D}' \in EQ$. But $\mathcal{D}$ is also minimal w.r.t. domination in EQ, and thus $\mathcal{D} \leq \mathcal{D}'$. Thus no decomposition $\mathcal{D}'$ strictly dominates $\mathcal{D}$. □

4.1.2 Ordering by Attribute Count of Instances

We now introduce an order which does not rely on a size function, but is otherwise similar to the order we used in the last section. Instead of measuring the total size of instances, we count the number of tuples an attribute appears in. This gives us domination and c-domination pre-orders for each attribute, and we can then combine these to get another pair of orderings for decompositions.

Definition 4.7. Let $R = \{A_1, \ldots, A_k\}$ be a schema. For a finite relation r over R and an attribute A we define the *count* of A on r as

$$count_A(r) := \begin{cases} |r| & \text{if } A \in R \\ 0 & \text{if } A \notin R \end{cases}$$

where $|r|$ denotes the number of tuples in r. We then define the count of A on r decomposed by a decomposition $\mathcal{D} = \{R_1, \ldots, R_n\}$ of R as

$$count_A(r[\mathcal{D}]) := \sum_{j=1}^{n} count_A(r[R_j])$$

Definition 4.8. Let R be a schema with FDs Σ, A an attribute and $\mathcal{D}_1, \mathcal{D}_2$ decompositions of R. We say that $\mathcal{D}_1$ *c-dominates* $\mathcal{D}_2$ w.r.t. A if there exists a constant k such that for all finite relations r over R that satisfy Σ we have

$$count_A(r[\mathcal{D}_1]) \leq k \cdot count_A(r[\mathcal{D}_2])$$

We further say that $\mathcal{D}_1$ *dominates* $\mathcal{D}_2$ w.r.t. A if the above relation holds for $k = 1$.

Thus, for each attribute A, we get a c-domination and domination pre-orders. We combine those pre-orders by intersection.

Definition 4.9. Let R be a schema and $\mathcal{D}_1, \mathcal{D}_2$ decomposition of R. We say that

$$\mathcal{D}_1 \left\{ \begin{array}{c} \text{dominates} \\ \text{c-dominates} \end{array} \right\} \mathcal{D}_2 \text{ w.r.t. attribute count}$$

if for every attribute $A \in R$ we have $\mathcal{D}_1 \left\{ \begin{array}{c} \text{dominates} \\ \text{c-dominates} \end{array} \right\} \mathcal{D}_2$ w.r.t. A.

We will prove later on that domination and c-domination w.r.t. attribute count and w.r.t. size are exactly the same orders.

4.1.3 Ordering by Containing Schema Closures

The previous two order pairs introduced are defined by considering all valid instances. This is not very practical if we wish to actually decide for two given decompositions whether one (c-) dominates the other. We therefore present a third pair of orders, which is defined by considering only the decompositions, rather than instances on them.

The approach we use is similar to attribute counting. The count of an attribute depends on the set of schemas it lies in. While it also depends on the instance r, it is easy to show that the number of tuples in $r[R_j]$ is determined by the closure R_j^* of R_j (and by r). Recall that the closure R_j^* of R_j under Σ is

$$R_j^* := \{A \in R \mid \Sigma \vDash R_j \rightarrow A\}$$

where Σ is a given set of constraints on R. In this work, we shall mainly be interested in functional, multi-valued and join dependencies.

Lemma 4.10. *Let R be a schema with constraints Σ, and $X \subseteq R$. Then for all relations r on R we have $|r[X]| = |r[X^*]|$.*

Proof. We can obtain $r[X]$ by projecting from $r[X^*]$. The only way for the number of tuples to decrease, is for $r[X^*]$ to contain different tuples which are identical on X. But this cannot happen since X functionally determines X^*. □

This motivates the following definitions.

Definition 4.11. Let R be a schema with constraints Σ, and $\mathcal{D}$ a decomposition of R. Then for any attribute $A \in R$ we define the *containing schema closures* (CSC) of A in $\mathcal{D}$ as the *multiset*

$$CSC_A(\mathcal{D}) = \{R_j^* \mid A \in R_j \in \mathcal{D}\}$$

The idea behind this definition is that the containing schema closure of an attribute A represents the attribute count for A in a way which does not rely on particular instances, but still allows comparison of different decompositions. It is necessary to use multisets rather than sets to represent the correct attribute count.

Example 4.4. Consider again the schema $R = ABCDE$ with constraints $\Sigma = \{AB \to CD, B \to E\}$ from example 4.3, and the decompositions

$$\mathcal{D}_1 = \{ABC, ABD, BE\}$$
$$\mathcal{D}_2 = \{ABCD, BE\}$$

They produce the multisets

$$CSC_A(\mathcal{D}_1) = \{ABCDE, ABCDE\}$$
$$CSC_A(\mathcal{D}_2) = \{ABCDE\}$$

which indicate that, for any relation r on R, the attribute A appears in twice as many tuples in $r[\mathcal{D}_1]$ as in $r[\mathcal{D}_2]$. Using sets would hide this difference. □

We will now compare decompositions using the respective CSCs of all attributes. For that we need mappings between multi-sets. We allow different instances of the same value in the source domain to map to different values, and call a mapping injective if a value in the target domain is mapped to at most as often as it occurs in the target domain.

Definition 4.12. Let M_1, M_2 be two multisets[2] of (attribute) sets. We say that

(i) M_1 *weakly inclusion-dominates* M_2 if there exists a mapping $f : M_1 \to M_2$ with $e \subseteq f(e)$ for all $e \in M_1$.

(ii) M_1 *strongly inclusion-dominates* M_2 if there exists an injective mapping $f : M_1 \to M_2$ with $e \subseteq f(e)$ for all $e \in M_1$.

Definition 4.13. Let $\mathcal{D}_1, \mathcal{D}_2$ be two decompositions of R. We say that $\mathcal{D}_1$ *weakly/strongly csc-dominates* $\mathcal{D}_2$ if for all attributes $A \in R$ we have that $CSC_A(\mathcal{D}_1)$ weakly/strongly inclusion-dominates $CSC_A(\mathcal{D}_2)$.

We will prove in the next section that weak csc-domination implies c-domination, and that strong csc-domination implies domination. While the opposite does not hold for arbitrary types of constraints, we will be able to show that it holds for sets of functional dependencies, and in the case of weak csc-domination/c-domination also for multi-valued and join dependencies. Thus we obtain a syntactic characterization for c-domination and, at least in the case of functional dependencies, for domination. For multi-valued dependencies, domination does not imply strong csc-domination, as will become evident in example 4.10. Finding a syntactic characterization for domination in this case is an open problem.

4.2 Equivalence of Orderings

We will show that, if the only integrity constraints on R are functional dependencies, then all three order pairs defined in section 4.1 are identical. If multi-valued and join dependencies are also allowed, then domination w.r.t. size or attribute count need not imply weak csc-domination, but we will show that the other equivalences between the orders in question still hold.

As multi-valued dependencies are just a special case of join dependencies, it suffices to consider only functional and join dependencies. We will assume throughout this section that functional and join dependencies are the only types of integrity constraints occurring.

[2]Note that for definition (i) we do not really need multi-sets.

4.2.1 Size vs. Attribute Count

We start by showing that the orders defined by size and attribute count are identical. Recall that we assume that all domains are infinite, and that the size functions associated with them are positive and unbounded.

Lemma 4.14. *Let R be a schema with functional and join dependencies Σ, and $\mathcal{D}_1, \mathcal{D}_2$ be decompositions of R. If $\mathcal{D}_1$ does not (c-)dominate $\mathcal{D}_2$ w.r.t. attribute count, then $\mathcal{D}_1$ does not (c-)dominate $\mathcal{D}_2$ w.r.t. size.*

Proof. If $\mathcal{D}_1$ does not c-dominate $\mathcal{D}_2$ w.r.t. attribute count, then for every integer k there exists a relation r and an attribute A with

$$count_A(r[\mathcal{D}_1]) > k \cdot count_A(r[\mathcal{D}_2])$$

For each k and associated r and A, we will construct a relation r' for which

$$size(r'[\mathcal{D}_1]) > k \cdot size(r'[\mathcal{D}_2])$$

holds. This shows that $\mathcal{D}_1$ does not c-dominate $\mathcal{D}_2$ w.r.t. size. For domination we only need to consider the case $k = 1$.

The construction works as follows. Since the relations in $r[\mathcal{D}_1]$ with attribute A contain more than k times as many tuples as those in $r[\mathcal{D}_2]$, there must be an attribute value v_A for A which appears more than k times as often in $r[\mathcal{D}_1]$ as in $r[\mathcal{D}_2]$.

We construct r' from r by substituting every occurrence of v_A by a new value v'_A which does not appear in r. As Σ contains only functional and join dependencies, these constraints still hold for r'. Let o_1, o_2 be the number of occurrences of v_A in $r[\mathcal{D}_1], r[\mathcal{D}_2]$. We choose v'_A sufficiently large, i.e., such that

$$size(v'_A) > \frac{k \cdot size(r[\mathcal{D}_2]) - size(r[\mathcal{D}_1])}{o_1 - k \cdot o_2} + size(v_A)$$

This gives us (note that $o_1 - k \cdot o_2 > 0$):

$$\begin{aligned} (o_1 - k \cdot o_2) \cdot (size(v'_A) - size(v_A)) &> k \cdot size(r[\mathcal{D}_2]) - size(r[\mathcal{D}_1]) \\ size(r[\mathcal{D}_1]) + o_1 \cdot (size(v'_A) - size(v_A)) &> k \cdot size(r[\mathcal{D}_2]) + k \cdot o_2 \cdot (size(v'_A) - size(v_A)) \\ size(r'[\mathcal{D}_1]) &> k \cdot size(r'[\mathcal{D}_2]) \end{aligned}$$

This concludes the proof. □

Lemma 4.15. *Let R be a schema with functional and join dependencies Σ, and $\mathcal{D}_1, \mathcal{D}_2$ be decompositions of R. If $\mathcal{D}_1$ does not (c-)dominate $\mathcal{D}_2$ w.r.t. size, then $\mathcal{D}_1$ does not (c-)dominate $\mathcal{D}_2$ w.r.t. attribute count.*

Proof. The proof is analogous to the last one. For every k, r ($k = 1$ for domination) with

$$size(r[\mathcal{D}_1]) > k \cdot size(r[\mathcal{D}_2])$$

we need to construct a relation r' such that for some attribute A we get

$$count_A(r'[\mathcal{D}_1]) > k \cdot count_A(r'[\mathcal{D}_2]).$$

Again we use that the attribute values occurring in $r[\mathcal{D}_1]$ and $r[\mathcal{D}_2]$ are the same. Since $size(r[\mathcal{D}_1]) > k \cdot size(r[\mathcal{D}_2])$, there must exist some attribute value v_A of an attribute A which occurs more than k times as often in $r[\mathcal{D}_1]$ than in $r[\mathcal{D}_2]$. We construct r' from r by selecting exactly those tuples which have the value v_A on attribute A. As Σ contains only functional and join dependencies, these constraints still hold for r'. And clearly we now have $count_A(r'[\mathcal{D}_1]) > k \cdot count_A(r'[\mathcal{D}_2])$. □

We can combine the last two lemmas.

Theorem 4.16. *Let R be a schema with functional and join dependencies Σ, and $\mathcal{D}_1, \mathcal{D}_2$ be decompositions of R. Then $\mathcal{D}_1$ (c-)dominates $\mathcal{D}_2$ w.r.t. size iff $\mathcal{D}_1$ (c-)dominates $\mathcal{D}_2$ w.r.t. attribute count.*

Proof. Follows immediately from the lemmas shown. □

4.2.2 Attribute Count vs. Containing Schema Closures - Part I

We will now show that the orders defined by attribute count and containing schema closures are actually the same. One direction of implication is easy to show.

Theorem 4.17. *Let R be a schema with constraints[3] Σ, and $\mathcal{D}_1, \mathcal{D}_2$ be decompositions of R. If $\mathcal{D}_1 \left\{ \begin{array}{c} \textit{weakly} \\ \textit{strongly} \end{array} \right\}$ csc-dominates $\mathcal{D}_2$ then $\mathcal{D}_1 \left\{ \begin{array}{c} \textit{c-dominates} \\ \textit{dominates} \end{array} \right\} \mathcal{D}_2$.*

Proof. Let r be any relation on R and A some attribute in R.

If $\mathcal{D}_1$ weakly csc-dominates $\mathcal{D}_2$, then for every schema $R_1 \in \mathcal{D}_1$ with $A \in R_1$ there exists a schema $R_2 \in \mathcal{D}_2$ with $A \in R_2$ and $R_1^* \subseteq R_2^*$. By Lemma 4.10 we have $|R_1| \leq |R_2|$, and each such schema R_2 is mapped to at most $|\mathcal{D}_1|$ times. Therefore the number of tuples containing attribute A in $r[\mathcal{D}_1]$ is at most $|\mathcal{D}_1|$ times larger than the number of tuples with A in $r[\mathcal{D}_2]$. Thus $\mathcal{D}_1$ c-dominates $\mathcal{D}_2$ with $k = |\mathcal{D}_1|$.

If $\mathcal{D}_1$ strongly csc-dominates $\mathcal{D}_2$, then by Lemma 4.10 and due to the injectivity of the mapping f in Definition 4.12, the number of tuples with attribute A in $r[\mathcal{D}_1]$ is no larger than the number of those in $r[\mathcal{D}_2]$. Thus $\mathcal{D}_1$ dominates $\mathcal{D}_2$. □

To show implication in the other direction, we will assume that $\mathcal{D}_1$ does not weakly or strongly csc-dominate $\mathcal{D}_2$, and construct example relations which show that $\mathcal{D}_1$ does not c-dominate or dominate $\mathcal{D}_2$. These constructions will require some work, and we devote the next subsection to them.

4.2.3 Subset Construction

Our goal is to construct relations over a schema $\mathcal{D}$ with functional and join dependencies Σ, for which the number of tuples in their projection onto non-key subschemas varies by an arbitrarily large factor.

Definition 4.18. Let R be a schema with constraints Σ and $\mathcal{D} = \{R_1, \ldots, R_n\}$ a decomposition of R. We say that a non-empty relation r over R is *k-reducing* w.r.t. $\mathcal{D}$ for an integer k if

[3]Theorem 4.17 holds for arbitrary constraints, not just functional and join dependencies.

(i) all constraints in Σ hold on r, and

(ii) for every subschema $R_j \in \mathcal{D}$ which is not a key of R, the projection of r onto R_j contains at most $\frac{1}{k}$ times as many tuples as r.

The following example illustrates the basic idea for constructing k-reducing relations.

Example 4.5. Consider the schema $R = ABCD$ with constraint set

$$\Sigma = \{AB \to C, C \to AB, A \to D, B \to D\}$$

and the (faithful and lossless) decomposition $\mathcal{D} = \{ABC, AD, BD\}$. We can construct a 2-reducing instance of R w.r.t. $\mathcal{D}$ as follows:

$r =$

A	B	C	D
1	1	(1, 1)	1
1	2	(1, 2)	1
2	1	(2, 1)	1
2	2	(2, 2)	1

The FDs in Σ clearly hold, and the projections of r onto AD and BD contain only 2 tuples, compared to 4 tuples in r. □

We constructed the example relation above by creating two variables v_A, v_B with domain $\{1, 2\}$, and for each value pair $(v_A = a, v_B = b)$ creating a tuple in r, making the value of A dependent on a, the value of B dependent on b, the value of C dependent on both and the value of D dependent on neither. We formalize this idea as follows.

Definition 4.19. Let Ω be a finite set and R be a set of attributes, where each $A_i \in R$ has a subset S_i of Ω associated with it. For a positive integer k we define the *k-mappings* of Ω as the total functions from Ω into $\{1, \ldots, k\}$. For each k-mapping f we define its *lifting* onto R as the tuple t_f on R, in which the attribute A_i has as value the partial function

$$f|_{S_i} : \Omega \to \{1, \ldots, k\} := \{x \to y \in f \mid x \in S_i\}$$

We get the *k-lifting* of R by taking all k-mappings of Ω and lifting them all onto R. Note that the k-lifting is a set of tuples on R, and thus a relation on R.

While the attribute values constructed by lifting k-mappings are partial functions rather than elements of the attribute's domain, it is easy to see that one could always substitute those values with values from the proper domains (recall that we assumed domains to be infinite) to get an isomorphic relation on R. Note that while this substitution may affect the size of relations, it has no affect on attribute count. As we are only trying to relate the containing schema closure measures to attribute count, rather than size measures directly, we will not worry about size or domains any further.

When giving examples, we will use attributes "$A, B, \ldots$" rather than "$A_1, A_2, \ldots$", and denote their associated subsets by "$S_A, S_B, \ldots$" instead of "$S_1, S_2, \ldots$".

Example 4.6. The given definitions can be related to example 4.5 as follows. We have

$$\Omega = \{v_A, v_B\}, S_A = \{v_A\}, S_B = \{v_B\}, S_C = \{v_A, v_B\}, S_D = \emptyset$$

Writing the total function $\{v_A \mapsto a, v_B \mapsto b\}$ short as (a, b), we obtain the set of all 2-mappings as

$$\{1,2\}^\Omega = \left\{ \begin{array}{c} (1,1), \\ (1,2), \\ (2,1), \\ (2,2) \end{array} \right\}.$$

These k-mapping get lifted onto R as follows, writing the partial function $\{v_A \mapsto a\}$ as $(a, -)$:

		A	B	C	D
$(1,1)$	$\curvearrowright$	$(1,-)$	$(-,1)$	$(1,1)$	$(-,-)$
$(1,2)$	$\curvearrowright$	$(1,-)$	$(-,2)$	$(1,2)$	$(-,-)$
$(2,1)$	$\curvearrowright$	$(2,-)$	$(-,1)$	$(2,1)$	$(-,-)$
$(2,2)$	$\curvearrowright$	$(2,-)$	$(-,2)$	$(2,2)$	$(-,-)$

Ignoring blanks and substituting $(-,-)$ by 1, we obtain relation r from example 4.5. □

The lifting of a k-mapping depends on the sets S_i associated with the attributes A_i, and these sets S_i are the only free choices we have in our construction. Note that elements of Ω which do not appear in any S_i do not affect the construction, thus we may as well assume that $\Omega = \bigcup S_i$. Given a schema R with constraints Σ and a decomposition $\mathcal{D}$ of R, we want to choose the sets S_i in such a way that the constructed relation is k-reducing. We will first consider the case where Σ contains only FDs.

Definition 4.20. For a subschema $X \subseteq R$ we call the set

$$S_X := \bigcup_{A_i \in X} S_i$$

the subset *associated* with X. For $v \in \Omega$ we call

$$R_v := \{A_i \in R \mid v \in S_i\}$$

the set of attributes *depending* on v.

Note that $\{S_i \mid A_i \in R\}$ and $\{R_v \mid v \in \Omega\}$ are dual hypergraphs.

Lemma 4.21. *Let Ω and R be as in Definition 4.19, X a subschema of R and S_X its associated subset of cardinality s. Let r_k be the k-lifting of R and x_k the k-lifting of X. Then*

$$x_k = r_k[X]$$

and x_k contains exactly k^s tuples.

Proof. Let f be a k-mapping of Ω, and $t_{f,R}$ and $t_{f,X}$ its liftings onto R and X, respectively. By definition $t_{f,X} = t_{f,R}[X]$, and since this holds for all f we get $x_k = r_k[X]$.

Clearly there are k^s k-mappings of S_X. We show that x_k contains k^s tuples by giving a one-to-one mapping between tuples of x_k and k-mappings of S_X.

The components of a tuple $t_{f,X} \in x_k$ are the partial functions $f|_{S_i}$. By taking their union we obtain $f|_{S_X}$, which is a k-mapping of S_X. Conversely, we can obtain $t_{f,X}$ from $f|_{S_X}$ by restricting $f|_{S_X}$ to the associated sets $S_i \subseteq S_X$ with $A_i \in X$. This gives us the one-to-one mapping we wanted. □

Lemma 4.22. *Let Ω and R be as in Definition 4.19, $k \geq 2$ and r_k the k-lifting of R. Let further $X, Y \subseteq R$, and S_X, S_Y be their associated subsets. Then the FD $X \to Y$ holds on r_k iff $S_X \supseteq S_Y$.*

Proof. (1) Let $S_X \supseteq S_Y$, and let $t_1, t_2 \in r_k$ be tuples in r_k. If $t_1[X] = t_2[X]$ then the k-mappings f_1, f_2 which were lifted onto R to obtain $t_1 = t_{f_1}$ and $t_2 = t_{f_2}$ have the same restriction to S_X, that is $f_1|_{S_X} = f_2|_{S_X}$. Since $S_X \supseteq S_Y$ this implies that $f_1|_{S_Y} = f_2|_{S_Y}$, and therefore $t_1[Y] = t_2[Y]$. Thus $X \to Y$ holds on r_k.

(2) Let $S_X \not\supseteq S_Y$, i.e., there exists some $v \in S_Y \setminus S_X$. Let f_1, f_2 be k-mappings of Ω which differ only on v. Then for their liftings $t_{f_1}, t_{f_2} \in r_k$ onto R we have $t_{f_1}[X] = t_{f_2}[X]$ but $t_{f_1}[Y] \neq t_{f_2}[Y]$. Thus $X \to Y$ does not hold on r_k. □

Definition 4.23. Let R be a schema with constraints Σ. We say that a subschema $X \subseteq R$ is *open* if its complement $R \setminus X$ is *closed* under Σ, that is,

$$(R \setminus X)^* = R \setminus X$$

Lemma 4.24. *Let Ω, R, Σ and S_i be as in Definition 4.19, $k \geq 2$ and r_k the k-lifting of R. Let Σ contain only FDs. Then Σ holds on r_k iff for every $v \in \Omega$ the subschema R_v of attributes depending on v (Definition 4.20) is open.*

Proof. (1) Let all R_v be open. Assume that some FD $X \to Y \in \Sigma$ does not hold on r_k. Then by Lemma 4.22 there exists some $v \in S_Y \setminus S_X$. This means that X contains no attributes which depend on v, i.e., no attributes in R_v, so $X \subseteq R \setminus R_v$. Since $R \setminus R_v$ is closed under Σ this implies $Y \subseteq R \setminus R_v$, and thus $v \notin S_Y$. This contradicts $v \in S_Y \setminus S_X$ and disproves our assumption.

(2) Let R_v not be open for some $v \in \Omega$. Then there exists an attribute $A \in R_v$ with $\Sigma \models R \setminus R_v \to A$. Due to $A \in R_v$ we have $v \in S_A$, and clearly $v \notin S_{R \setminus R_v}$ by Definition 4.20. Therefore the FD $R \setminus R_v \to A$ does not hold on r_k by Lemma 4.22. □

Lemmas 4.21 and 4.24 indicate how we should construct the S_i to obtain a relation on R which is k-reducing w.r.t. some decomposition $\mathcal{D}$. For every subschema $R_j \in \mathcal{D}$ which is not a key of R, there should be an element $v \in \Omega$ which does not lie in the subset S_{R_j} associated with R_j. At the same time, the set R_v should be open. This leads to the following construction.

Subset Construction for FDs: Let R be a schema with FDs Σ, and $\mathcal{D}$ a decomposition of R. For every subschema $R_j \in \mathcal{D}$ we form its closure R_j^*, and add a unique element v_j to every set S_i for which $A_i \notin R_j^*$. The set Ω from Definition 4.19 is then

$$\Omega := \bigcup S_i = \{v_j \mid R_j \in \mathcal{D} \text{ and } R_j^* \neq R\}$$

Note that if $\mathcal{D}$ contains multiple subschemas with the same closure, it would suffice to consider only one of them when constructing the sets S_i.

Example 4.7. Consider again the schema R and decomposition $\mathcal{D}$ from examples 4.5 and 4.6. Applying the subset construction we get

$$\begin{array}{lcll} ABC^* & = & ABCD & \curvearrowright \text{ do nothing} \\ BD^* & = & BD & \curvearrowright \text{ add } v_{BD} \text{ to } S_A, S_C \\ AD^* & = & AD & \curvearrowright \text{ add } v_{AD} \text{ to } S_B, S_C \end{array}$$

This gives us the associated subsets

$$S_A = \{v_{BD}\}, S_B = \{v_{AD}\}, S_C = \{v_{BD}, v_{AD}\}, S_D = \{\}$$

which (except for element names) are the same as in example 4.6. □

Theorem 4.25. *Let R be a schema with FDs Σ, and $\mathcal{D}$ a decomposition of R. Let the S_i be constructed using the subset construction for FDs. Then for every k the k-lifting r_k of R is k-reducing w.r.t. $\mathcal{D}$.*

Proof. (i) Let $v = v_j$ be added to Ω when considering R_j during the subset construction. By construction $R_v = R \setminus R_j^*$ is open, so Σ holds on r_k by Lemma 4.24.

(ii) For every subschema $R_j \in \mathcal{D}$ which is not a key of R, $S_R = \Omega$ contains at least one more element than S_{R_j}, namely the element v_j which was associated with all attributes in $R \setminus R_j^*$. Thus by Lemma 4.21, the projection of r_k onto R_j contains at most $\frac{1}{k}$ times as many tuples as r_k. □

We will now generalize the subset construction to work with functional and join dependencies. We first need to establish when a join dependency holds on a k-lifting.

Definition 4.26. Let Ω and R be as in Definition 4.19 and $\bowtie [R_1, \ldots, R_n]$ a join-dependency on R. For every $v \in \Omega$ the *synchronization hypergraph* of v w.r.t. $\bowtie [R_1, \ldots, R_n]$ is the projection of the hypergraph $\{R_1, \ldots, R_n\}$ onto R_v (the set of attributes depending on v), i.e.,

$$sync(v) = \{R_1 \cap R_v, \ldots, R_n \cap R_v\}$$

Lemma 4.27. *Let Ω and R be as in Definition 4.19, $k \geq 2$ and r_k the k-lifting of R. Let further $\bowtie [R_1, \ldots R_n]$ be a join-dependency on R. Then $\bowtie [R_1, \ldots, R_n]$ holds on r_k iff for all $v \in \Omega$ the synchronization hypergraph H_v of v w.r.t. $\bowtie [R_1, \ldots, R_n]$ is connected.*

Proof. Let S_R be the subset of Ω associated with R. By definition, the join dependency $\bowtie [R_1, \ldots R_n]$ holds on r_k iff

$$r_k[R_1] \bowtie \ldots \bowtie r_k[R_n] =: j_k \subseteq r_k$$

(1) Consider a single tuple $t \in j_k$, and recall that all its attribute values are partial functions $\Omega \to \{1, \ldots, k\}$. Let us denote the union of these partial functions by $\bigsqcup t$, which is a relation $f_t \subseteq \Omega \times \{1, \ldots, k\}$. If f_t is a function, then t is the lifting of f_t onto R, and thus $t \in r_k$. On the other hand, every $t' \in r_k$ is the lifting of some function $f : \Omega \to \{1, \ldots, k\}$, and thus $\bigsqcup t' = f$. Together this gives us that a tuple $t \in j_k$ lies in r_k iff $\bigsqcup t$ is a function.

(2) Let $t \in j_k, v \in \Omega$. For every subschema R_j from $\bowtie [R_1, \ldots, R_n]$ we have that $t[R_j] = t'[R_j]$ for some $t' \in r_k$, and therefore that $\bigsqcup t[R_j]$ is a partial function. Furthermore we have

$$\bigsqcup t[R_i \cup R_j] = \bigsqcup t[R_i] \cup \bigsqcup t[R_j]$$

If there exists an attribute $A \in R_i \cap R_j$, then every v associated with A has the same image[4] under $\bigsqcup t[R_i]$ as it has under $\bigsqcup t[R_j]$, so $\bigsqcup t[R_i \cup R_j]$ is a function for v, i.e., it

[4]One could say that the mapping of v is "synchronized", hence the name "synchronization hypergraph".

maps v to only a single value. For a given $v \in \Omega$ let R_v be the set of attributes depending on v. Then such an attribute A exists for v iff $R_i \cap R_v$ and $R_j \cap R_v$ are not disjoint, i.e., iff the partial synchronization hypergraph with edges $R_i \cap R_v$ and $R_j \cap R_v$ is connected.

(3) If H_v is connected, we can use the argument of (2) multiple times to show that for any $t \in j_k$ $\bigsqcup t$ is a function for v. If all H_v are connected, $\bigsqcup t$ is a function, which by (1) shows $t \in r_k$, and thus $j_k \subseteq r_k$. This proves the "if" direction of the lemma.

(4) If H_v is disconnected for some v (which implies $v \in S_R$), then we can partition $\{R_1, \ldots, R_n\}$ into two sets P_1, P_2 such that the attribute sets $\bigcup P_1 \cap R_v$ and $\bigcup P_2 \cap R_v$ are disjoint and non-empty. Since $k \geq 2$ we can find two functions $f_1, f_2 : \Omega \to \{1, \ldots, k\}$ which differ exactly for v. Let $t_1, t_2 \in r_k$ be their liftings onto R. Then by definition of j_k there exists a tuple $t \in j_k$ with

$$t[P_1] = t_1[P_1] \text{ and } t[P_2] = t_2[P_2]$$

But this gives us

$$\bigsqcup t = \bigsqcup t_1[P_1] \cup \bigsqcup t_2[P_2]$$

which is not a function since $\bigsqcup t_1[P_1]$ and $\bigsqcup t_2[P_2]$ map v to different values. As $\bigsqcup t$ is not a function we get $t \notin r_k$ by (1), and thus $j_k \nsubseteq r_k$. This shows the "only if" direction and completes the proof. □

Definition 4.28. Let $X \subseteq R$ be an attribute set and Σ a set of functional and join dependencies on R. The *dependency basis* of X w.r.t. Σ (and R) is the finest partition $DB_\Sigma(X)$ of $R \setminus X^*$ into non-empty sets, such that for every $Y \in DB_\Sigma(X)$ the multivalued dependency $X \twoheadrightarrow Y$ is implied by Σ. Where Σ is clear from the context we will just write $DB(X)$ for $DB_\Sigma(X)$.

It is well known that such a unique finest partition always exists [36]. Note that some texts define the dependency basis of X as the finest partition of R rather than $R \setminus X^*$. As Σ implies $X \twoheadrightarrow A$ for all $A \in X^*$, this partitions X^* into sets each consisting of only a single attribute. Thus knowing the partition of $R \setminus X^*$ immediately gives us the partition of R as well. We chose the given definition as it makes some formulations easier.

The following inference rules for functional and multivalued dependencies are well-known to be correct [33, 36], and will be useful in what follows:

$$\frac{X \twoheadrightarrow Y}{X \twoheadrightarrow XY} \quad \text{(Augmentation)}$$

$$\frac{X \twoheadrightarrow Y \quad Y \twoheadrightarrow Z}{X \twoheadrightarrow Z \setminus Y} \quad \text{(Pseudo-Transitivity)}$$

$$\frac{X \twoheadrightarrow Y \quad Y \to Z}{X \to Z \setminus Y} \quad \text{(Mixed Pseudo-Transitivity)}$$

Lemma 4.29. *Let Σ be a set of functional and join dependencies on R. Then for every $X \subseteq R$, every $Y \in DB(X)$ is open, i.e., $R \setminus Y$ is closed under Σ.*

Proof. Assume that $R \setminus Y$ were not closed under Σ. Then for some attribute $A \in Y$ we have $\Sigma \models R \setminus Y \to A$. Since $Y \in DB(X)$ we have $\Sigma \models X \twoheadrightarrow R \setminus Y$. Using these two facts together we can derive

$$\frac{X \twoheadrightarrow R \setminus Y \quad R \setminus Y \to A}{X \to A} \quad \text{(Mixed Pseudo-Transitivity)}$$

Thus $A \in X^* \cap Y$, which is a contradiction to $Y \in DB(X)$, since $DB(X)$ partitions $R \setminus X^*$. ☐

Our intermediate goal is to construct (for given R, Σ and $\mathcal{D}$) the sets S_i, such that for every k the resulting k-lifting r_k is k-reducing. We have found such a construction for the case where Σ contains only functional dependencies. When Σ contains join dependencies as well, we need to adapt our construction, since otherwise the join dependencies in Σ need not hold on r_k.

Example 4.8. Consider the schema $R = ABCD$ with constraints

$$\begin{aligned} \Sigma &= \{\bowtie [AB, AC, AD]\} \\ &\equiv \{A \twoheadrightarrow B|C|D\} \end{aligned}$$

and decomposition $\mathcal{D} = \{AB, AC, AD\}$. Using the subset construction for functional dependencies, we would get the sets

$$S_A = \{\}, S_B = \{v_{AC}, v_{AD}\}, S_C = \{v_{AB}, v_{AD}\}, S_D = \{v_{AB}, v_{AC}\}$$

which in turn lead to the 2-lifting (again writing partial functions as tuples):

$r_2 =$

A	B	C	D
$(-,-,-)$	$(-,1,1)$	$(1,-,1)$	$(1,1,-)$
$(-,-,-)$	$(-,1,2)$	$(1,-,2)$	$(1,1,-)$
$(-,-,-)$	$(-,2,1)$	$(1,-,1)$	$(1,2,-)$
$(-,-,-)$	$(-,2,2)$	$(1,-,2)$	$(1,2,-)$
$(-,-,-)$	$(-,1,1)$	$(2,-,1)$	$(2,1,-)$
$(-,-,-)$	$(-,1,2)$	$(2,-,2)$	$(2,1,-)$
$(-,-,-)$	$(-,2,1)$	$(2,-,1)$	$(2,2,-)$
$(-,-,-)$	$(-,2,2)$	$(2,-,2)$	$(2,2,-)$

It is easy to check that $\bowtie [AB, AC, AD]$ does not hold on r_2. ☐

We observe that the join dependency $\bowtie [AB, AC, AD]$ in the example above does not hold because the sets S_B, S_C, S_D share common elements, i.e., they are not pairwise disjoint. This establishes a connection between the values of B, C and D for every tuple in r_k. To avoid such connections, we could associate v_{AB}, v_{AC} and v_{AD} only with the attributes of one set Y in $DB(AB), DB(AC)$ and $DB(AD)$, respectively.

While in general it is critical that for every $v \in \Omega$ the set R_v of attributes associated with v is open, in order to ensure that the FDs in Σ hold on r_k (Lemma 4.24), Lemma 4.29 ensures us that every $Y \in DB(X)$ is open, for any $X \subseteq R$. This motivates the following construction:

Generalized Subset Construction: Let R be a schema with functional and join dependencies Σ, and $\mathcal{D}$ a decomposition of R. For every subschema $R_j \in \mathcal{D}$ with $R_j^* \neq R$ we form its dependency basis $DB(R_j)$. Then for some (arbitrary) set $Y \in DB(R_j)$, we add a unique element v_j to every set S_i for which $A_i \in Y$.

Note that when Σ contains only functional dependencies, we have

$$DB(R_j) = \{R \setminus R_j^*\}$$

Thus the generalized subset construction is identical to the subset construction for FDs in such cases, which justifies its name.

Example 4.9. Consider again the schema $R = ABCD$ with constraints

$$\Sigma = \{\bowtie [AB, AC, AD]\}$$

and decomposition $\mathcal{D} = \{AB, AC, AD\}$. Using the generalized subset construction we might get the sets (depending on the choices for Y)

$$S_A = \{\}, S_B = \{v_{AC}\}, S_C = \{v_{AD}\}, S_D = \{v_{AB}\}$$

which lead to the 2-lifting:

$r_2 =$

A	B	C	D
$(-,-,-)$	$(-,1,-)$	$(-,-,1)$	$(1,-,-)$
$(-,-,-)$	$(-,1,-)$	$(-,-,2)$	$(1,-,-)$
$(-,-,-)$	$(-,2,-)$	$(-,-,1)$	$(1,-,-)$
$(-,-,-)$	$(-,2,-)$	$(-,-,2)$	$(1,-,-)$
$(-,-,-)$	$(-,1,-)$	$(-,-,1)$	$(2,-,-)$
$(-,-,-)$	$(-,1,-)$	$(-,-,2)$	$(2,-,-)$
$(-,-,-)$	$(-,2,-)$	$(-,-,1)$	$(2,-,-)$
$(-,-,-)$	$(-,2,-)$	$(-,-,2)$	$(2,-,-)$

It is easy to verify that $\bowtie [AB, AC, AD]$ now holds on r_2. □

Proposition 4.30. *For every join dependency* $\bowtie [R_1, R_2, \ldots]$ *we have:*

$$\bowtie [R_1, R_2, \ldots] \models \bowtie [R_1 \cup R_2, \ldots]$$

i.e., if we replace any two (or more) subschemas in a join dependency by their union, we obtain an implied join dependency.

Proof. Clear by definition of join dependency. □

Theorem 4.31. *Let R be a schema with functional and join dependencies Σ, and $\mathcal{D}$ a decomposition of R. Let the S_i be constructed using the generalized subset construction. Then the k-lifting r_k of R is k-reducing, for every $k \in \mathbb{N}$.*

Proof. Part (ii) of the k-reducing property can be shown as in Theorem 4.25. The difficulty lies in showing part (i), namely that all constraints in Σ hold. This is clear for functional dependencies by lemmas 4.29 and 4.24.

Assume that some join dependency $\bowtie [R_1, \ldots, R_n] \in \Sigma$ does not hold for r_k, $k \geq 2$ ($k = 1$ is trivial). Then by Lemma 4.27 there must be some $v \in \Omega$ for which the synchronization hypergraph

$$H_v = \{R_1 \cap R_v, \ldots, R_n \cap R_v\}$$

of v w.r.t. $\bowtie [R_1, \ldots, R_n]$ is disconnected. Then we can partition R_v, the set of attributes depending on v, into non-empty sets

$$R_v = H_1 \cup H_2$$

such that every R_j from $\bowtie[R_1,\dots,R_n]$ is disjoint to H_1 or H_2. Let U_1 be the union of R_j disjoint to H_2, and U_2 of those disjoint to H_1. Then by Proposition 4.30 we have:

$$\Sigma \models \bowtie[R_1,\dots,R_n] \models \bowtie[U_1,U_2]$$

Now let X be the closure of the subschema in $\mathcal{D}$ for which v was added. By construction we have $R_v \in DB(X)$, so that we can partition R into X, R_v and the remaining attributes $Z := R \setminus (X \cup R_v)$, so that

$$\begin{aligned} R &= X \cup Z \cup R_v \\ &= X \cup Z \cup H_1 \cup H_2 \end{aligned}$$

Then $U_1 \subseteq X \cup Z \cup H_1$ and $U_2 \subseteq X \cup Z \cup H_2$, and thus

$$\begin{aligned} \Sigma &\models \bowtie[U_1,U_2] \\ &\models \bowtie[X \cup Z \cup H_1, X \cup Z \cup H_2] \\ &\equiv XZ \twoheadrightarrow H_1 \end{aligned}$$

Since Z is the union of elements of the dependency basis of X, we have $X \twoheadrightarrow Z$, and thus

$$\frac{X \twoheadrightarrow Z \quad XZ \twoheadrightarrow H_1}{X \twoheadrightarrow H_1} \quad \text{(Pseudo-Transitivity)}$$

This is a contradiction, since $\emptyset \neq H_1 \subsetneq R_v$, and $R_v \in DB(X)$. □

4.2.4 Attribute Count vs Containing Schema Closures - Part II

We are now ready to complete the equivalence proof for attribute count and csc-domination orders.

Lemma 4.32. *Let Σ be a set of functional and join dependencies on R, and $X, Y \subseteq R$. Then $\Sigma \cup \{Y \to R\} \models X \to Y$ iff $\Sigma \models X \to Y$.*

Proof. If Σ implies $X \to Y$ then clearly $\Sigma \cup \{Y \to R\}$ implies $X \to Y$ as well. Now let $\Sigma \not\models X \to Y$, so that there exists a relation r on R for which all dependencies in Σ hold, but not $X \to Y$. Then there exist at least two tuples $t_1, t_2 \in r$ with

$$t_1[X] = t_2[X], t_1[Y] \neq t_2[Y]$$

Among all such pairs of tuples, let (t_1, t_2) be one for which the set of attributes $D \subseteq R$ on which t_1 and t_2 differ is minimal. We claim that $r' := \{t_1, t_2\}$ is a relation for which the dependencies in $\Sigma \cup \{Y \to R\}$ hold but not $X \to Y$, thus showing $\Sigma \cup \{Y \to R\} \not\models X \to Y$.

Clearly $Y \to R$ holds for r', but not $X \to Y$. Furthermore all FDs in Σ hold for r' since $r' \subseteq r$. Now for any join dependency $\bowtie[R_1,\dots,R_n] \in \Sigma$ let

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

be the result of the corresponding project-join mapping of r'. By definition $\bowtie[R_1,\dots,R_n]$ holds for r' iff $r'' \subseteq r'$.

Let $t_3 \in r''$ be arbitrary. Since $t_1[Y] \neq t_2[Y]$ at least one of the inequalities $t_3[Y] \neq t_1[Y]$ and $t_3[Y] \neq t_2[Y]$ holds, say $t_3[Y] \neq t_1[Y]$. For every attribute $A \in R$ we have

$t_3[A] \in \{t_1[A], t_2[A]\}$ by construction of r''. Since $t_1[R \setminus D] = t_2[R \setminus D]$, t_3 differs from t_1 at most on the attributes in D. But t_1, t_2 were chosen to make D minimal, and due to

$$r'' \subseteq r[R_1] \bowtie \ldots \bowtie r[R_n] = r$$

we have $t_3 \in r$. Thus t_3 differs from t_1 (and therefore equals t_2) on all attributes in D. Consequently $t_3 = t_2 \in r'$, which shows $r'' \subseteq r'$ and completes the proof. □

Theorem 4.33. *Let R be a schema with functional and join dependencies Σ, and $\mathcal{D}_1, \mathcal{D}_2$ be decompositions of R. If $\mathcal{D}_1$ does not weakly csc-dominate $\mathcal{D}_2$, then $\mathcal{D}_1$ does not c-dominate $\mathcal{D}_2$.*

Proof. Recall that by definitions 4.12 and 4.13 $\mathcal{D}_1$ weakly csc-dominates $\mathcal{D}_2$ iff for every attribute A and every schema R_1 with $A \in R_1 \in \mathcal{D}_1$ there exists a schema $R_2 \in \mathcal{D}_2$ with $A \in R_2$ and $R_1^* \subseteq R_2^*$. As $\mathcal{D}_1$ does not weakly csc-dominate $\mathcal{D}_2$, there must exist A, R_1 with $A \in R_1 \in \mathcal{D}_1$, such that for every schema $R_2 \in \mathcal{D}_2$ with $A \in R_2$ we have $R_1^* \not\subseteq R_2^*$, i.e., $\Sigma \not\models R_2 \rightarrow R_1$. To show that $\mathcal{D}_1$ does not c-dominate $\mathcal{D}_2$, we construct a counterexample for any value k.

We construct the counterexample r on R by using the generalized subset construction for $CSC_A(\mathcal{D}_2)$, but for an extended set of constraints

$$\Sigma' := \Sigma \cup \{R_1 \rightarrow R\}$$

This makes R_1 a key of R w.r.t. Σ', and by Lemma 4.32 we still have $\Sigma' \not\models R_2 \rightarrow R_1$ for all R_2 in question. Then by Lemma 4.10 the number of tuples in $r[R_1]$ equals the number of tuples in r, which by Theorem 4.31 is at least k times larger than the number of tuples in $r[R_2]$, for any $R_2 \in \mathcal{D}_2$ with $A \in R_2$. Thus $\mathcal{D}_1$ does not c-dominate $\mathcal{D}_2$. □

The following theorem is well-known [21].

Theorem 4.34 (Hall's Theorem)**.** *Let M_1, M_2 be finite sets and $\pi : M_1 \rightarrow \mathcal{P}(M_2)$ associate a set of permitted values with each element in M_1. Then there exists an injective mapping $f : M_1 \rightarrow M_2$ with $f(e) \in \pi(e)$ for all $e \in M_1$ iff for all $m_1 \subseteq M_1$ we have $|m_1| \leq |\pi(m_1)|$, where*

$$\pi(m_1) := \bigcup\{\pi(e) \mid e \in m_1\}$$

Lemma 4.35. *Let Σ be a set of functional dependencies on R, and further $X, Y_1, \ldots, Y_n \subseteq R$. Then $\Sigma \cup \{Y_1 \rightarrow R, \ldots, Y_n \rightarrow R\} \models X \rightarrow R$ holds iff $\Sigma \models X \rightarrow Y_i$ for some $i \in \{1, \ldots, n\}$.*

Proof. (1) If $\Sigma \models X \rightarrow Y_i$ then $\Sigma \cup \{Y_i \rightarrow R\} \models X \rightarrow R$ by transitivity (rule 1.1).

(2) Otherwise let X^* denote the closure of X under Σ, and let $r = \{t_1, t_2\}$ be a relation on R containing two tuples with $t_1[X^*] = t_2[X^*]$ and $t_1[A] \neq t_2[A]$ for all $A \notin X^*$. Then Σ holds on r, and since for all Y_i we have $Y_i \not\subseteq X^*$, the functional dependencies $Y_i \rightarrow R$ hold as well. It is clear though that $X \rightarrow R$ does not hold on r, and thus is not implied by $\Sigma \cup \{Y_1 \rightarrow R, \ldots, Y_n \rightarrow R\}$. □

Theorem 4.36. *Let R be a schema with functional dependencies Σ, and $\mathcal{D}_1, \mathcal{D}_2$ be decompositions of R. If $\mathcal{D}_1$ does not strongly csc-dominate $\mathcal{D}_2$, then $\mathcal{D}_1$ does not dominate $\mathcal{D}_2$.*

Proof. Let $\mathcal{D}_1$ not strongly csc-dominate $\mathcal{D}_2$. This means that for some $A \in R$ there exists no injective mapping

$$f : M_1 := CSC_A(\mathcal{D}_1) \to M_2 := CSC_A(\mathcal{D}_2)$$

with $e \subseteq f(e)$ for all $e \in M_1$. The permitted values for $e \in M_1$ in such a mapping would be

$$\pi(e) := \{e' \in M_2 \mid e \subseteq e'\}$$

Note that M_1, M_2 are multisets, rather than sets. However, to make the formulation of the following arguments easier, we shall regard them as sets by treating multiple occurrences of elements in M_1, M_2 as different. This does not change whether an injective mapping f exists.

By Theorem 4.34 there exists a set $m_1 \subseteq CSC_A(\mathcal{D}_1)$ with $|m_1| > |m_2|$, where $m_2 := \pi(m_1)$. We now construct a counterexample r on R by using the subset construction for $M_2 \setminus m_2$, but again using an extended set of constraints

$$\Sigma' := \Sigma \cup \{Y \to R \mid Y \in m_1\}$$

This makes all elements in m_1 keys of R w.r.t. Σ', and by Lemma 4.35 we still have $\Sigma' \nvDash X \to R$ for all $X \in M_2 \setminus m_2$. Then by Lemma 4.10 we have $|r[m_1]| = |m_1| \cdot |r|$ and $|r[m_2]| = |m_2| \cdot |r|$. By Theorem 4.25 the number $|r|$ of tuples in r is at least k times larger than the number of tuples in $r[X]$, for any $X \in M_2 \setminus m_2$. By choosing k large enough we get

$$|r[M_2 \setminus m_2]| < |r|$$

This gives us

$$\begin{aligned}
|r[M_2]| &= |r[m_2]| + |r[M_2 \setminus m_2]| \\
&< |m_2| \cdot |r| + |r| \\
&\leq |m_1| \cdot |r| \\
&= |r[m_1]| \\
&\leq |r[M_1]|
\end{aligned}$$

which shows that $\mathcal{D}_1$ does not dominate $\mathcal{D}_2$. □

In the last theorem we restricted ourselves to functional dependencies. This is because it does not hold in the presence of multi-valued or join dependencies, as the following example shows.

Example 4.10. Let $R, \Sigma, \mathcal{D}_1, \mathcal{D}_2$ be as follows:

$$\begin{aligned}
&R = ABC, \Sigma = \{A \twoheadrightarrow B\}, \\
&\mathcal{D}_1 = \{AB, AC\}, \mathcal{D}_2 = \{ABC, A\}
\end{aligned}$$

It is easy to see that $\mathcal{D}_1$ does not strongly csc-dominate $\mathcal{D}_2$ w.r.t. the attribute A:

$$\begin{aligned}
CSC_A(\mathcal{D}_1) &= \{AB, AC\} \\
CSC_A(\mathcal{D}_2) &= \{ABC, A\}
\end{aligned}$$

It is clear that $\mathcal{D}_1$ dominates $\mathcal{D}_2$ w.r.t. B and C. To show domination w.r.t. attribute count, it thus suffices to prove that for every relation r on R we have

$$count_A(r[\mathcal{D}_1]) \leq count_A(r[\mathcal{D}_2])$$

To do so, we partition r into disjoint relations r_i with $|r_i[A]| = 1$ and $r_i[A] \neq r_j[A]$ for $i \neq j$. Then for each subschema R' of R containing A (i.e., all schemas in $\mathcal{D}_1, \mathcal{D}_2$), $r[R']$ is the disjoint union of the $r_i[R']$, and thus

$$count_A(r[\mathcal{D}_{1/2}]) = \sum count_A(r_i[\mathcal{D}_{1/2}])$$

Therefore we only need to show

$$count_A(r_i[\mathcal{D}_1]) \leq count_A(r_i[\mathcal{D}_2])$$

for all relations r_i. Note that $A \twoheadrightarrow B$ still holds for all r_i, and thus

$$r_i[BC] = r_i[B] \bowtie r_i[C]$$

Abbreviating the cardinalities of $r_i[B], r_i[C]$ with C_B, C_C we obtain

$$\begin{aligned} |r_i[AB]| &= C_B \\ |r_i[AC]| &= C_C \\ |r_i[ABC]| &= C_B \cdot C_C \\ |r_i[A]| &= 1 \end{aligned}$$

This gives us

$$\begin{aligned} count_A(r_i[\mathcal{D}_2]) - count_A(r_i[\mathcal{D}_1]) &= C_B \cdot C_C - (C_B + C_C) + 1 \\ &= (C_B - 1) \cdot (C_C - 1) \\ &\geq 0 \end{aligned}$$

which shows that $\mathcal{D}_1$ dominates $\mathcal{D}_2$ w.r.t. attribute count, even though $\mathcal{D}_1$ does not strongly csc-dominate $\mathcal{D}_2$. □

It is an open question how domination w.r.t. size or attribute count can be characterized syntactically in the presence of functional and join dependencies.

4.3 Relationship to other Normal Forms

A number of normal forms for characterizing well designed relational databases have been proposed, depending on the types of integrity constraints given. For functional dependencies, BCNF and 3NF are the most popular ones. 4NF [15] is an extension of BCNF for functional and multivalued dependencies. For functional and join dependencies 4NF has been extended to PJ/NF [16], 5NF [33, 43] and KCNF [44]. The last normal form, KCNF, will be of particular interest to us.

Definition 4.37. [44] Let R be a schema with functional and join dependencies Σ. Then R is in *Key-Complete Normal Form (KCNF)*, if for every join dependency $\bowtie [R_1, \ldots, R_n]$ implied by Σ, the keys among $R_1, \ldots, R_n$ contain all attributes in R. That is, we have

$$R = \bigcup \{R_i \in \bowtie [R_1, \ldots, R_n] \mid R_i^* = R\}$$

Note that in the case where Σ contains only functional and multivalued dependencies, KCNF is equivalent to 4NF, and when Σ contains only functional dependencies KCNF is equivalent to BCNF [44].

Given a single schema with constraints Σ, the absence of redundancy, as defined in [1, 44], is precisely characterized by BCNF, 4NF and KCNF. That is, a schema R is free of redundancy iff it is in BCNF, 4NF or KCNF [1, 44].

While our normal form has been designed to minimize size rather than redundancy, the intuition is that minimizing one minimizes the other as well. We will show that this intuition holds in so far, as that a single schema is in KCNF (and thus free of redundancy) iff it is in DNF among all lossless decompositions. Thus lossless DNF can be seen as an extension of BCNF, 4NF and KCNF.

Recall that a decomposition is in DNF if it is minimal *among a given set of 'suitable' decompositions*, and that for each such set we may obtain a different DNF.

Theorem 4.38. *Let R be a schema with functional and join dependencies Σ. Then R is in KCNF iff $\{R\}$ is in DNF w.r.t. all lossless decompositions of R.*

Proof. (1) Let R be in KCNF. Let $\mathcal{D} = \{R_1, \ldots, R_n\}$ be any lossless decomposition of R. Then Σ implies the join dependency $\bowtie [R_1, \ldots, R_n]$, and since R is in KCNF, every attribute $A \in R$ lies in some R_i which forms a key of R. Thus $R \in CSC_A(\mathcal{D})$ for all A, so $\{R\}$ strongly csc-dominates $\mathcal{D}$. Therefore $\{R\}$ dominates $\mathcal{D}$ by Theorem 4.17. As this holds for all lossless decompositions $\mathcal{D}$, $\{R\}$ is in DNF.

(2) Let R not be in KCNF. Then Σ implies a join dependency $\bowtie [R_1, \ldots, R_n]$ such that

$$\bigcup \{R_i \mid \Sigma \vDash R_i \to R\} \neq R$$

The decomposition $\mathcal{D} = \{R_1, \ldots, R_n\}$ is lossless, and there exists an attribute $A \in R$ which does not lie in any R_i which forms a key of R. Clearly $\mathcal{D}$ weakly csc-dominates $\{R\}$, and since $R \notin CSC_A(\mathcal{D})$ we have that $\{R\}$ does not weakly csc-dominate $\mathcal{D}$. Thus $\mathcal{D}$ strictly c-dominates $\{R\}$ by theorems 4.17 and 4.33, showing that $\{R\}$ is not in DNF. $\square$

Note that, while lossless DNF and BCNF are the same for a single schema, they differ significantly when applied to multiple schemas. We will demonstrate those differences using a detailed example.

4.3.1 A Detailed Example

A university has oral examinations at the end of each semester, and wants to manage related data using a relational database. The relevant attributes to be stored are

$$R = \{Student, Course, Chapter, Time, Room\}$$

Here Chapter denotes a chapter from the course textbook the student will be examined about. Every student can get examined about multiple chapters, and chapters may vary for each student. Multiple students can get examined at the same time in the same room, but the course must be the same. Further constraints are that a student gets examined for a course only once, and cannot be in multiple rooms at the same time. Those conditions can be expressed through functional dependencies as follows:

$$\Sigma = \left\{ \begin{array}{c} \{Student, Course\} \to Time, \\ \{Student, Time\} \to Room, \\ \{Time, Room\} \to Course \end{array} \right\}$$

We are now presented with the task of decomposing R. If it is deemed necessary to preserve dependencies, a reasonable Boyce-Codd Normal Form decomposition could be synthesized as follows:

$$\mathcal{D}_{DP-BCNF} = \left\{ \begin{array}{c} \{Student, Course, Time\}, \\ \{Student, Time, Room\}, \\ \{Course, Time, Room\}, \\ \{Student, Course, Chapter\} \end{array} \right\}$$

This decomposition, however, is not in dependency preserving Domination Normal Form - it is strictly dominated by the decomposition

$$\mathcal{D}_{DP-DNF} = \left\{ \begin{array}{c} \{Student, Course, Time, Room\}, \\ \{Student, Course, Chapter\} \end{array} \right\}$$

Note that the latter decomposition is in dependency preserving DNF (although we will not show this here), but not in BCNF.

If dependencies need not be preserved, we could use the well-known BCNF decomposition algorithm [27, 33, 36] to obtain the following BCNF decomposition (decomposing first by $\{Student, Course\} \rightarrow Time$, then by $\{Student, Course\} \rightarrow Room$):

$$\mathcal{D}_{BCNF} = \left\{ \begin{array}{c} \{Student, Course, Time\}, \\ \{Student, Course, Room\}, \\ \{Student, Course, Chapter\} \end{array} \right\}$$

Again, this is strictly dominated by the decomposition $\mathcal{D}_{DP-DNF}$, which is in DNF even among all lossless decompositions. At first look it might seem that $\mathcal{D}_{DP-DNF}$ is strictly c-dominated by

$$\mathcal{D}_{DNF} = \left\{ \begin{array}{c} \{Student, Time, Room\}, \\ \{Course, Time, Room\}, \\ \{Student, Course, Chapter\} \end{array} \right\}$$

A closer look reveals though that both c-dominate each other, since the attribute $Course$ already appears in the key schema $\{Student, Course, Chapter\}$ in both cases. The latter decomposition is both in DNF and BCNF. The decomposition

$$\mathcal{D}'_{DP-DNF} = \left\{ \begin{array}{c} \{Student, Course, Time, Room\}, \\ \{Student, Time, Chapter\} \end{array} \right\}$$

however, while in dependency preserving DNF, is not in DNF w.r.t. all lossless decomposition, since it is strictly c-dominated by

$$\mathcal{D}'_{DNF} = \left\{ \begin{array}{c} \{Student, Time, Room\}, \\ \{Course, Time, Room\}, \\ \{Student, Time, Chapter\} \end{array} \right\}$$

Which decomposition to choose is ultimately up to the designer, as storage space and redundancy are not the only design criteria to consider. The schema $\{Student, Course, Chapter\}$ appears to be a more intuitive choice than $\{Student, Time, Chapter\}$, although deciding this requires domain knowledge which is not encoded in the constraints.

We conclude this example by providing an instance of R and its projection onto the schemas appearing in the first four decompositions given. While DNF considers all valid instances rather than just a given one, we chose an instance which visualizes the relationship between a schema's closure and the size of relations projected onto it. It is easy to see that $\mathcal{D}_{DP-DNF}$ and $\mathcal{D}_{DNF}$ require less storage space than $\mathcal{D}_{DP-BCNF}$ and $\mathcal{D}_{BCNF}$.

Student	*Course*	*Ch.*	*Time*	*Ro.*
J.C. Denton	Networks	2	3/10, 1pm	101
J.C. Denton	Networks	6	3/10, 1pm	101
J.C. Denton	Security	1	4/10, 1pm	104
J.C. Denton	Security	5	4/10, 1pm	104
L. Nasher	Networks	3	3/10, 1pm	101
L. Nasher	Networks	4	3/10, 1pm	101
L. Nasher	Security	4	4/10, 1pm	104
L. Nasher	Security	7	4/10, 1pm	104
O. Shrek	Networks	2	3/10, 1pm	101
O. Shrek	Networks	8	3/10, 1pm	101
O. Shrek	Security	5	4/10, 1pm	104
O. Shrek	Security	2	4/10, 1pm	104
M. Smith	Security	4	4/10, 2pm	104
M. Smith	Security	6	4/10, 2pm	104
M. Anderson	Networks	3	3/10, 1pm	101
M. Anderson	Networks	5	3/10, 1pm	101
A. Cheng	Networks	2	3/10, 1pm	103
A. Cheng	Networks	4	3/10, 1pm	103
A. Cheng	Security	4	4/10, 2pm	104
A. Cheng	Security	5	4/10, 2pm	104
N. Cheng	Networks	1	3/10, 1pm	103
N. Cheng	Networks	7	3/10, 1pm	103
N. Cheng	Security	5	4/10, 2pm	104
N. Cheng	Security	6	4/10, 2pm	104
J.Zhao	Networks	2	3/10, 1pm	103
J.Zhao	Networks	5	3/10, 1pm	103

Student	*Course*	*Ch.*
J.C. Denton	Networks	2
J.C. Denton	Networks	6
J.C. Denton	Security	1
J.C. Denton	Security	5
L. Nasher	Networks	3
L. Nasher	Networks	4
L. Nasher	Security	4
L. Nasher	Security	7
O. Shrek	Networks	2
O. Shrek	Networks	8
O. Shrek	Security	5
O. Shrek	Security	2
M. Smith	Security	4
M. Smith	Security	6
M. Anderson	Networks	3
M. Anderson	Networks	5
A. Cheng	Networks	2
A. Cheng	Networks	4
A. Cheng	Security	4
A. Cheng	Security	5
N. Cheng	Networks	1
N. Cheng	Networks	7
N. Cheng	Security	5
N. Cheng	Security	6
J.Zhao	Networks	2
J.Zhao	Networks	5

Student	*Course*	*Time*	*Ro.*
J.C. Denton	Networks	3/10, 1pm	101
J.C. Denton	Security	4/10, 1pm	104
L. Nasher	Networks	3/10, 1pm	101
L. Nasher	Security	4/10, 1pm	104
O. Shrek	Networks	3/10, 1pm	101
O. Shrek	Security	4/10, 1pm	104
M. Smith	Security	4/10, 2pm	104
M. Anderson	Networks	3/10, 1pm	101
A. Cheng	Networks	3/10, 1pm	103
A. Cheng	Security	4/10, 2pm	104
N. Cheng	Networks	3/10, 1pm	103
N. Cheng	Security	4/10, 2pm	104
J.Zhao	Networks	3/10, 1pm	103

Student	*Course*	*Time*
J.C. Denton	Networks	3/10, 1pm
J.C. Denton	Security	4/10, 1pm
L. Nasher	Networks	3/10, 1pm
L. Nasher	Security	4/10, 1pm
O. Shrek	Networks	3/10, 1pm
O. Shrek	Security	4/10, 1pm
M. Smith	Security	4/10, 2pm
M. Anderson	Networks	3/10, 1pm
A. Cheng	Networks	3/10, 1pm
A. Cheng	Security	4/10, 2pm
N. Cheng	Networks	3/10, 1pm
N. Cheng	Security	4/10, 2pm
J.Zhao	Networks	3/10, 1pm

Student	*Time*	*Ro.*
J.C. Denton	3/10, 1pm	101
J.C. Denton	4/10, 1pm	104
L. Nasher	3/10, 1pm	101
L. Nasher	4/10, 1pm	104
O. Shrek	3/10, 1pm	101
O. Shrek	4/10, 1pm	104
M. Smith	4/10, 2pm	104
M. Anderson	3/10, 1pm	101
A. Cheng	3/10, 1pm	103
A. Cheng	4/10, 2pm	104
N. Cheng	3/10, 1pm	103
N. Cheng	4/10, 2pm	104
J.Zhao	3/10, 1pm	103

Student	*Course*	*Ro.*
J.C. Denton	Networks	101
J.C. Denton	Security	104
L. Nasher	Networks	101
L. Nasher	Security	104
O. Shrek	Networks	101
O. Shrek	Security	104
M. Smith	Security	104
M. Anderson	Networks	101
A. Cheng	Networks	103
A. Cheng	Security	104
N. Cheng	Networks	103
N. Cheng	Security	104
J.Zhao	Networks	103

Course	*Time*	*Ro.*
Networks	3/10, 1pm	101
Security	4/10, 1pm	104
Security	4/10, 2pm	104
Networks	3/10, 1pm	103

4.4 Computing Domination Normal Form

Having defined when schemas are in DNF, we are now looking for an algorithm which, given a schema R with constraints Σ, computes a decomposition of R which is in DNF. For this we will restrict ourselves to the case where the only constraints given are functional dependencies.

Lemma 4.39. *Let R be a schema with constraints Σ, and $\mathcal{D}$ be a decomposition of R. If $\mathcal{D}$ contains two different $R_1, R_2 \in \mathcal{D}$ with $R_1^* = R_2^*$, then*

$$\mathcal{D}' := \mathcal{D} \setminus \{R_1, R_2\} \cup \{R_1 \cup R_2\}$$

dominates $\mathcal{D}$. If $R_1 \cap R_2 \neq \emptyset$, then $\mathcal{D}'$ strictly dominates $\mathcal{D}$.

Proof. For every relation r on R the projections $r[R_1], r[R_2]$ can be obtained by projecting from $r[R_1 \cup R_2]$. As R_1 an R_2 are keys of

$$R_1 \cup R_2 \subseteq R_1^* = R_2^*$$

no tuples are lost in this projection. Thus the size of $\mathcal{D}$ and $\mathcal{D}'$ varies by the size for the values of attributes in $R_1 \cap R_2$, as those are stored twice in $\mathcal{D}$. $\square$

In chapter 3 we defined equivalence classes for functional dependencies, or equivalently for schemas. We shall use them again here, and recall the definition.

Definition 4.40. Let R be a schema with constraints Σ and $X, Y \subseteq R$. We call X and Y *equivalent* if $X^* = Y^*$. We say that X is of *higher order* than Y if $X^* \supseteq Y^*$.

We call two functional dependencies $X_1 \to Y_1, X_2 \to Y_2$ implied by Σ equivalent if their left hand sides X_1 and X_2 are equivalent (and similarly for higher order).

When partitioning a set Σ of FDs into equivalence classes, we denote them by

$$EQ_X := \{Y \to Z \in \Sigma \mid Y^* = X^*\}$$

and call X^* the closure of EQ_X.

This groups schemas and functional dependencies into equivalence classes. As there is an obvious correspondence between equivalence classes of schemas and those of functional dependencies, we will not always distinguish between them, and will compare equivalence classes w.r.t. higher order a well.

When searching for a DNF decomposition, Lemma 4.39 tells us that it suffices to only consider decompositions which contain at most one schema for each equivalence class of schemas.

Definition 4.41. Let $\mathcal{D}$ be a decomposition of R and $X \subseteq R$. We define the *higher order schemas* and *higher order attributes* of X in $\mathcal{D}$ as

$$HOS_X(\mathcal{D}) := \{R_j \in \mathcal{D} \mid X^* \subseteq R_j^*\}$$
$$HOA_X(\mathcal{D}) := \bigcup HOS_X(\mathcal{D})$$

Similarly, the *strictly higher order schemas* and *strictly higher order attributes* of X in $\mathcal{D}$ are

$$SHOS_X(\mathcal{D}) := \{R_j \in \mathcal{D} \mid X^* \subsetneq R_j^*\}$$
$$SHOA_X(\mathcal{D}) := \bigcup SHOS_X(\mathcal{D})$$

Lemma 4.42. *Let $\mathcal{D}_1, \mathcal{D}_2$ be two decompositions of R. Then $\mathcal{D}_1$ weakly csc-dominates $\mathcal{D}_2$ iff for every schema $X \in \mathcal{D}_1$ we have*

$$X \subseteq HOA_X(\mathcal{D}_2)$$

Proof. (1) By definition $\mathcal{D}_1$ weakly csc-dominates $\mathcal{D}_2$ iff for every attribute $A \in R$ and every $R_1^* \in CSC_A(\mathcal{D}_1)$ there exists $R_2^* \in CSC_A(\mathcal{D}_2)$ with $R_1^* \subseteq R_2^*$. In other words, for every pair (A, R_1) with $A \in R_1 \in \mathcal{D}_1$ there exists R_2 with $A \in R_2 \in \mathcal{D}_2$ and $R_1^* \subseteq R_2^*$.

(2) Furthermore we have $X \subseteq HOA_X(\mathcal{D}_2)$ for all $X \in \mathcal{D}_1$ iff for every pair (A, X) with $A \in X \in \mathcal{D}_1$ there exists $Y \in HOS_X(\mathcal{D}_2)$ with $A \in Y$, that is Y with $A \in Y \in \mathcal{D}_2$ and $X^* \subseteq Y^*$.

Using (1) and (2) together (with $X = R_1$ and $Y = R_2$) we obtain the claim. □

Definition 4.43. Let R be a schema with FDs Σ, and $\mathcal{D}$ a set of subschemas of R. We denote the set of all FDs in Σ which lie in schemas in $\mathcal{D}$ by

$$\Sigma[\mathcal{D}] := \bigcup_{R_j \in \mathcal{D}} \Sigma[R_j]$$

4.4.1 Dependency Preserving DNF

We are now able to construct an algorithm to compute a DNF decomposition for the case where Σ contains only functional dependencies, and where we consider only lossless and dependency preserving decompositions. For this, we recall the definition and properties of partial covers from sections 3.1 and 3.2.1, rephrased slightly to suit our needs.

Definition 4.44. Let Σ be a set of FDs, and $E_X \subseteq \Sigma$ be an equivalence class of Σ. A set $C \subseteq \Sigma^*$ is a *partial cover* of E_X (w.r.t. Σ) if

$$C[X^*] \cup (\Sigma \setminus E_X)$$

is a cover of Σ.

Lemma 4.45. *Let Σ be a set of FDs, and let EQ be the partition of Σ into equivalence classes. Then a set C of FDs is a cover of Σ iff C is a partial cover (w.r.t. Σ) for all equivalence classes $EQ_j \in EQ$.*

Lemma 4.46. *Let R be a schema with FDs Σ, and $\mathcal{D}$ a dependency preserving decomposition of R. Let further EQ_X be an equivalence class of Σ. Then*

$$\Sigma^{*a}[HOS_X(\mathcal{D})] \subseteq \Sigma^{*a}[HOA_X(\mathcal{D})]$$

each form a partial cover of EQ_X.

Proof. Since $\mathcal{D}$ is dependency preserving, $\Sigma^{*a}[\mathcal{D}]$ must form a cover of Σ, and thus a partial cover of EQ_X. By Definition 4.44 the only FDs of interest for forming a partial cover of EQ_X are those LHS-equivalent to the FDs in EQ_X. All of them lie in schemas R_j with $X^* \subseteq R_j^*$, so

$$\Sigma^{*a}[\{R_j \in \mathcal{D} \mid X^* \subseteq R_j^*\}]$$

already forms a partial cover of EQ_X. □

We use this to synthesize a dependency preserving decomposition as follows.

Algorithm "dependency preserving DNF decomposition"

```
INPUT: schema $R$, canonical cover $\Sigma$
OUTPUT: decomposition $\mathcal{D}$ in DNF

$\mathcal{D} := \emptyset$
if $\Sigma$ contains no key dependencies then
  add minimal key $R_{key}$ of $R$ to $\mathcal{D}$
partition $\Sigma$ into equivalence classes $EQ$
while $EQ \neq \emptyset$ do
      pick maximal $EQ_j \in EQ$ and remove it from $EQ$
      $R_j :=$ closure of FDs in $EQ_j$
      $A_{\mathcal{D}} := SHOA_{R_j}(\mathcal{D})$
      first for all $A \in R_j \setminus A_{\mathcal{D}}$ and then for all $A \in R_j \cap A_{\mathcal{D}}$ do
            if $\Sigma^{*a}[\mathcal{D} \cup \{R_j \setminus A\}]$ is a partial cover of $EQ_j$ then
              $R_j := R_j \setminus A$
      end
      if $R_j \neq \emptyset$ then
        $\mathcal{D} := \mathcal{D} \cup \{R_j\}$
end
```

Theorem 4.47. *Algorithm "dependency preserving DNF decomposition" returns a lossless, dependency preserving DNF decomposition of R.*

Proof. (1) We first show that $\mathcal{D}$ is dependency preserving and lossless. This is the case iff $\Sigma^{*a}[\mathcal{D}]$ is a cover of Σ, and $\mathcal{D}$ contains a key of R. If Σ contains no key dependency, then a minimal key R_{key} of R is added to $\mathcal{D}$. Otherwise it suffices to show that $\Sigma^*[\mathcal{D}]$ is a cover of Σ, since this implies that $\mathcal{D}$ contains a key of R. In each iteration of the while loop, it is ensured that for the decomposition $\mathcal{D}$ computed so far, $\Sigma^*[\mathcal{D}]$ forms a partial cover for the equivalence classes removed from EQ. After processing all equivalence classes, we therefore obtain a decomposition for which $\Sigma^*[\mathcal{D}]$ covers Σ.

It remains to show that $\mathcal{D}$ is not strictly dominated or c-dominated by any other lossless and dependency preserving decomposition $\mathcal{D}'$ of R.

(2) We start by proving that $\mathcal{D}'$ does not strictly dominate $\mathcal{D}$. Consider the schemas $R_1, \ldots, R_n \in \mathcal{D}$ (including R_{key} if it was added) in the order as they were added to $\mathcal{D}$ (with indices describing this order). Let R_k be the first such schema which is not contained in $\mathcal{D}'$ (if all R_j are contained in $\mathcal{D}'$ then clearly $\mathcal{D}$ dominates $\mathcal{D}'$), and $C := R_k^*$ its closure. By Lemma 4.46 the set

$$\Sigma^*[HOS_C(\mathcal{D}')]$$

forms a partial cover of EQ_k. Let further

$$H'_k := \bigcup (HOS_C(\mathcal{D}') \setminus \{R_1, \ldots, R_{k-1}\})$$

so that $\Sigma^*[\{R_1, \ldots, R_{k-1}\} \cup H'_k]$ forms a partial cover of EQ_k. Since R_k was constructed as minimal such that $\Sigma^*[R_1, \ldots, R_k]$ forms a partial cover of EQ_k, $H'_k \subsetneq R_k$ cannot hold. If $H_k = R_k$ then all schemas in $HOS_C(\mathcal{D}') \setminus \mathcal{D}$ must be of order EQ_k. By Lemma 4.39

we may assume that this does not happen, so $H_k \nsubseteq R_k$. Thus there exists at least one attribute A which lies in some schema

$$R'_A \in HOS_C(\mathcal{D}') \setminus \{R_1, \dots, R_{k-1}\}$$

but not in R_k.

Consider the containing schema closures $CSC_A(\mathcal{D})$ and $CSC_A(\mathcal{D}')$. If $\mathcal{D}'$ were to dominate $\mathcal{D}$, then $CSC_A(\mathcal{D}')$ would strongly inclusion dominate $CSC_A(\mathcal{D})$. Equivalently, $CSC_A(\mathcal{D}' \setminus \{R_1, \dots, R_{k-1}\})$ would have to strongly inclusion dominate $CSC_A(\mathcal{D} \setminus \{R_1, \dots, R_{k-1}\})$. However, we have

$$R'^*_A \in CSC_A(\mathcal{D}' \setminus \{R_1, \dots, R_{k-1}\})$$

but no schema in $CSC_A(\mathcal{D} \setminus \{R_1, \dots, R_{k-1}\})$ includes R'^*_A. This is a contradiction, which shows that $\mathcal{D}'$ does not dominate $\mathcal{D}$.

(3) Finally, we need to show that $\mathcal{D}'$ does not strictly c-dominate $\mathcal{D}$. Assume the contrary, so that by Lemma 4.42 we have for all $R' \in \mathcal{D}'$ that

$$R' \subseteq HOA_{R'}(\mathcal{D}),$$

and there exist a schema $R_w \in \mathcal{D}$ for which

$$R_w \nsubseteq HOA_{R_w}(\mathcal{D}')$$

Let R_w be the first such schema in the sequence of schemas $R_1, \dots, R_n \in \mathcal{D}$, and let $C := R^*_w$ be its closure. Then for every $R_j \in SHOS_C(\mathcal{D})$ we get

$$R_j \subseteq HOA_{R_j}(\mathcal{D}') \subseteq SHOA_C(\mathcal{D}')$$

and thus

$$SHOA_C(\mathcal{D}) \subseteq SHOA_C(\mathcal{D}')$$

Inclusion in the opposite direction holds by similar argument, showing equivalence:

$$SHOA_C(\mathcal{D}) = SHOA_C(\mathcal{D}')$$

This attribute set is computed as the set $A_\mathcal{D}$ during the construction of R_w: we have

$$A_\mathcal{D} = SHOA_C(\{R_1, \dots, R_{w-1}\}) = SHOA_C(\mathcal{D}) = SHOA_C(\mathcal{D}')$$

Since $R_w \nsubseteq HOA_{R_w}(\mathcal{D}') \supseteq A_\mathcal{D}$ we have $R_w \nsubseteq A_\mathcal{D}$. As we tried removing attributes outside $A_\mathcal{D}$ first when constructing R_w, the set

$$\Sigma^*[A_\mathcal{D}] = \Sigma^*[SHOA_C(\mathcal{D}')]$$

cannot form a partial cover of EQ_w. By Lemma 4.46 the set

$$\Sigma^*[HOA_C(\mathcal{D}')]$$

does form a partial cover of EQ_k, so $\mathcal{D}'$ must contain at least one schema R'_w with $R'^*_w = C$. By Lemma 4.39 we may assume that R'_w is the only such schema in $\mathcal{D}'$. We have by assumption that

$$R'_w \subseteq HOA_C(\mathcal{D}) = SHOA_C(\mathcal{D}) \cup R_w = A_\mathcal{D} \cup R_w$$

and thus

$$R'_w \setminus A_{\mathcal{D}} \subseteq R_w \setminus A_{\mathcal{D}}$$

Furthermore, we can split $HOS_C(\mathcal{D}')$ into $SHOS_C(\mathcal{D}')$ and R'_w, and get (using Lemma 4.46 once more) that

$$\Sigma^*[A_{\mathcal{D}} \cup R'_w] \supseteq \Sigma^*[SHOS_C(\mathcal{D}') \cup \{R'_w\}] = \Sigma^*[HOS_C(\mathcal{D}')]$$

must be a partial cover of EQ_w. However, by trying to remove attributes not in $A_{\mathcal{D}}$ first, we constructed R_w such that $R_w \setminus A_{\mathcal{D}}$ is minimal with $\Sigma^*[A_{\mathcal{D}} \cup R_w]$ being a partial cover for EQ_w (note that $\Sigma^*[\{R_1, \ldots, R_{w-1}\}] \subseteq \Sigma^*[A_{\mathcal{D}}]$) . Thus $R'_w \setminus A_{\mathcal{D}}$ cannot be a true subset of $R_w \setminus A_{\mathcal{D}}$, which gives us

$$R'_w \setminus A_{\mathcal{D}} = R_w \setminus A_{\mathcal{D}}$$

But this means that

$$R_w \subseteq R'_w \cup A_{\mathcal{D}} = R'_w \cup SHOA_C(\mathcal{D}') = HOA_C(\mathcal{D}')$$

which contradicts our initial assumption for R_w. □

4.5 Combining Normal Forms

While a decomposition into DNF minimizes storage space, other classical normal forms such as BCNF (for multiple schemas) offer other benefits, such as e.g. fast checking whether the given FDs hold [27]. We are therefore interested in the possibility of achieving both normal forms at the same time, and thus benefitting from both.

4.5.1 DNF and BCNF

Dependency preserving DNF and BCNF cannot always be combined - dependency preserving BCNF decompositions are known to not always exist, and deciding whether one exists is NP-hard [5]. We are now looking for an algorithm to decide whether a schema R with set of FDs Σ has a dependency preserving DNF decomposition $\mathcal{D}$, for which every schema $R_j \in \mathcal{D}$ is in BCNF. Furthermore, the algorithm should find such a decomposition $\mathcal{D}$ if it exists.

Note that algorithm "dependency preserving DNF decomposition" only produces decompositions which contain at most one schema for each equivalence class. This was motivated by Lemma 4.39. However, when looking for BCNF as well, we must consider decompositions which contain multiple schemas per equivalence class (i.e., multiple schemas with the same closure).

Example 4.11. Let $R = ABCDEFG$ with constraints

$$\Sigma = \left\{ \begin{array}{l} AB \to C, DE \to F, \\ DF \to AG, AG \to DF, \\ EF \to BG, BG \to EF, \\ AC \to DG, DG \to AC, \\ BC \to EG, EG \to BC \end{array} \right\}$$

Then R has a dependency preserving DNF and BCNF decomposition:

$$\mathcal{D} = \{ABC, DEF, ADFG, BEFG, ACDG, BCEG\}$$

Note though that $\mathcal{D}$ has two key schemas: ABC and DEF. Joining these two schemas leads to a decomposition

$$\mathcal{D}' = \{ABCDEF, ADFG, BEFG, ACDG, BCEG\}$$

which dominates $\mathcal{D}$ (and vice versa) and thus is also in dependency preserving DNF. However, $\mathcal{D}'$ is no longer in BCNF. □

The general idea for finding a dependency preserving DNF decomposition which is also in BCNF, is based on the results of chapter 3. We compute all canonical covers, and for each canonical cover C we generate all minimal (w.r.t. domination/c-domination) decompositions which preserve C. By "preserve" we mean that each FD $X \to A \in C$ lies in some schema $R_j \supseteq XA$. If a dependency preserving DNF and BCNF decomposition exists, it will be among them. While this general idea is not very efficient, we will optimize it in several ways.

First, we will show that for a given canonical cover C, it suffices to consider only one particular decomposition which preserves C.

Definition 4.48. For a FD $X \to Y$ we call XY the schema *induced* by $X \to Y$. For a set of FDs Σ we call

$$\mathcal{H} = \{XY \mid X \to Y \in \Sigma\}$$

the hypergraph induced by Σ. The maximal connected components of $\mathcal{H}$ partition its support (cf. section 3.1)

$$\vartheta_{\mathcal{H}} = \bigcup \mathcal{H}$$

We call $\vartheta_{\mathcal{H}}$ the *schema induced* by Σ, and the maximal connected components of $\mathcal{H}$ (which we regard as a set of schemas) the *schema partition induced* by Σ.

We will generate a decomposition which preserves a particular canonical cover Σ as follows. We first partition Σ into equivalence classes.We then preserve each equivalence class Σ_X by adding the schema partition induced by Σ_X to the decomposition.

Example 4.12. Consider again $R = ABCDEFG$ from example 4.11 with constraints

$$\Sigma = \left\{ \begin{array}{l} AB \to C, DE \to F, \\ DF \to AG, AG \to DF, \\ EF \to BG, BG \to EF, \\ AC \to DG, DG \to AC, \\ BC \to EG, EG \to BC \end{array} \right\}$$

To preserve Σ we need to preserve (in particular) the partial cover of EQ_R

$$\Sigma_R = \{AB \to C, DE \to F\}$$

The hypergraph induced by Σ_{AB} is

$$\mathcal{H} = \{ABC, DEF\}$$

which is also the schema *partition* induced by Σ_R, while

$$\vartheta_{\mathcal{H}} = ABCDEF$$

is the schema induced by Σ_R. We add the schemas ABC, DEF to our decomposition, rather than the schema $ABCDEF$, because ABC and DEF are in BCNF. □

Definition 4.49. Let R be a schema with FDs Σ, and EQ the partition of Σ into equivalence classes. For each $eq \in EQ$ let ϑ_{eq} be the schema induced by eq. Then we call

$$\mathcal{D}_1 := \{\vartheta_{eq} \mid eq \in EQ\}$$

the *principal decomposition* induced by Σ, and

$$\mathcal{D}_2 := \bigcup_{eq \in EQ} \{\text{schema partition induced by } eq\}$$

the *connected component decomposition* induced by Σ.

We will now show that it suffices to consider only the decompositions described above. For the following lemmas it is important to note the subtle difference between a dependency preserving decomposition of (R, Σ), and a decomposition of R which preserves Σ: A dependency preserving decomposition only needs to preserve a *cover* of Σ, not Σ itself.

Lemma 4.50. *Let R be a schema with FDs Σ. Then every BCNF decomposition $\mathcal{D}'$ which preserves Σ is dominated by the principal decomposition $\mathcal{D}$ induced by Σ.*

Proof. Let $\mathcal{D}'$ be any BCNF decomposition which preserves Σ, and A an attribute in R. By Theorem 4.17 strong csc-domination implies domination. Thus it suffices to show that there exists an injective function f, which maps every schema $\vartheta \in \mathcal{D}$ with $A \in \vartheta$ to a schema $\vartheta' \in \mathcal{D}'$ with $A \in \vartheta'$ and $\vartheta^* \subseteq \vartheta'^*$.

Now let $\vartheta \in \mathcal{D}$ with $A \in \vartheta$. By construction of $\mathcal{D}$ there must exists a FD $X \to Y \in \Sigma$ with $A \in XY$ and $XY^* = \vartheta^*$. Since $\mathcal{D}'$ preserves Σ, there exists a schema $\vartheta' \in \mathcal{D}'$ with $XY \subseteq \vartheta'$. Thus we have $A \in \vartheta'$ and $\vartheta^* = XY^* \subseteq \vartheta'^*$, so we let f map ϑ to ϑ'.

It remains to show that the function f constructed above is injective. As $\mathcal{D}$ is in BCNF, and $X \to Y$ is contained in ϑ', we have $XY^* = \vartheta'^*$, and thus $\vartheta^* = \vartheta'^*$. Since $\mathcal{D}$ contains only one schema per equivalence class, f is injective. □

Note that the principal and connected component decompositions are not lossless if Σ contains no key FD. In order to make them lossless, we need to add a minimal key. We consider this situation next.

Corollary 4.51. *Let R be a schema with FDs Σ, such that Σ contains no key FDs. Let $\mathcal{D}$ be the principal decomposition induced by Σ. Then for every lossless BCNF decomposition $\mathcal{D}'$ which preserves Σ, there exists a minimal key schema R_{key} such that $\mathcal{D}'$ is dominated by $\mathcal{D} \cup \{R_{key}\}$.*

Proof. Since $\mathcal{D}'$ is lossless, it contains a key schema R'_{key} by Lemma 2.1. Since $\mathcal{D}'$ is in BCNF, R'_{key} preserves no FDs in Σ, so $\mathcal{D}' \setminus \{R'_{key}\}$ still preserves Σ. Thus $\mathcal{D}$ dominates $\mathcal{D}' \setminus \{R'_{key}\}$ by Lemma 4.50. It follows that for every minimal key schema $R_{key} \subseteq R'_{key}$ the decomposition $\mathcal{D} \cup \{R_{key}\}$ dominates $\mathcal{D}'$. □

We are now ready to show that for finding a dependency preserving DNF and BCNF decomposition, is suffices to consider only connected component decompositions, plus minimal keys, if needed.

Theorem 4.52. *Let R be a schema with FDs Σ, such that R has a decomposition in dependency preserving DNF and BCNF. Then there exists a canonical cover Σ' of Σ and a minimal key R_{key} of R, such that the following statements hold:*

(i) If Σ contains a non-trivial key FD then $\mathcal{D}_{\Sigma'}$ is a dependency preserving DNF and BCNF decomposition of R.

(ii) If Σ contains no non-trivial key FD then $\mathcal{D}_{\Sigma'} \cup \{R_{key}\}$ is a dependency preserving DNF and BCNF decomposition of R.

where $\mathcal{D}_{\Sigma'}$ is the connected component decomposition induced by Σ'.

Proof. Let $\mathcal{D}$ be a decomposition of R in dependency preserving DNF and BCNF, which exists by assumption. Since $\mathcal{D}$ is dependency preserving there exists a cover Σ' of Σ such that $\mathcal{D}$ preserves Σ'. Let $\mathcal{D}^p_{\Sigma'}$ be the principal decomposition induced by Σ'.

(i) If Σ contains a non-trivial key FD, then so does Σ', which makes $\mathcal{D}^p_{\Sigma'}$ lossless (and dependency preserving). By Lemma 4.50 $\mathcal{D}^p_{\Sigma'}$ dominates $\mathcal{D}$. Since $\mathcal{D}$ is in dependency preserving DNF, the opposite must hold as well, i.e., $\mathcal{D}$ dominates $\mathcal{D}^p_{\Sigma'}$.

(ii) If Σ contains no non-trivial key FD, then Σ' contains no key FD. Then by Corollary 4.51 there exists a minimal key R_{key} of R such that $\mathcal{D}^p_{\Sigma'} \cup \{R_{key}\}$ dominates $\mathcal{D}$. Now $\mathcal{D}^p_{\Sigma'} \cup \{R_{key}\}$ is lossless by Lemma 2.1, and dependency preserving. Thus, since $\mathcal{D}$ is in dependency preserving DNF, $\mathcal{D}$ dominates $\mathcal{D}^p_{\Sigma'} \cup \{R_{key}\}$.

It is clear by definition that the principal decomposition $\mathcal{D}^p_{\Sigma'}$, and the connected component decomposition $\mathcal{D}_{\Sigma'}$ induced by Σ' dominate each other. Thus $\mathcal{D}$ dominates $\mathcal{D}_{\Sigma'}$ and vice versa, or $\mathcal{D}_{\Sigma'} \cup \{R_{key}\}$ and vice versa, for case (i) and case (ii) respectively.

Since $\mathcal{D}$ is in dependency preserving DNF, and $\mathcal{D}_{\Sigma'}$ (or $\mathcal{D}_{\Sigma'} \cup \{R_{key}\}$) is lossless and dependency preserving and dominates $\mathcal{D}$, the decomposition $\mathcal{D}_{\Sigma'}$ (or $\mathcal{D}_{\Sigma'} \cup \{R_{key}\}$) is also in dependency preserving DNF. It remains to show that $\mathcal{D}_{\Sigma'}$ (or $\mathcal{D}_{\Sigma'} \cup \{R_{key}\}$) is in BCNF. This is trivial for the minimal key R_{key}, so we only need to show it for $\mathcal{D}_{\Sigma'}$ in either case.

We have established that $\mathcal{D}, \mathcal{D}_{\Sigma'}$ and $\mathcal{D}^p_{\Sigma'}$ dominate each other. By Theorem 4.36 all attributes have the same containing schema closures w.r.t. $\mathcal{D}, \mathcal{D}_{\Sigma'}$ and $\mathcal{D}^p_{\Sigma'}$. Thus for every schema $R_j \in \mathcal{D}^p_{\Sigma'}$ the schemas in $\mathcal{D}$ with the same closure as R_j partition R_j, and similarly for the schemas in $\mathcal{D}_{\Sigma'}$. By construction $\mathcal{D}_{\Sigma'}$ uses the finest such partitions possible while preserving Σ', in particular as least as fine as those of $\mathcal{D}$. Thus every schema in $\mathcal{D}_{\Sigma'}$ is a subset of some schema in $\mathcal{D}$. Since $\mathcal{D}$ is in BCNF, $\mathcal{D}_{\Sigma'}$ must be in BCNF as well. $\square$

The last theorem allows us to restrict our search for dependency preserving DNF and BCNF decompositions to connected component decompositions induced by canonical covers, plus minimal key schemas if needed.

When generating the connected component decomposition $\mathcal{D}_\Sigma$ induced by some canonical cover Σ we apply another optimization. $\mathcal{D}_\Sigma$ is the disjoint union of several connected component decompositions, each induced by the partial canonical cover $\Sigma[eq]$ for some equivalence class eq of Σ. We thus compute the connected component decompositions induced by partial canonical covers for each equivalence class of Σ^{*a} individually. These can then be combined to obtain all connected component decompositions induced by canonical covers.

We now describe an algorithm which is based on our constructions so far.

Definition 4.53. Let ISP be a partial function mapping schemas to sets of schemas. Then for any set of schemas $\mathcal{D}$ we define

$$\begin{aligned} ISP(\mathcal{D}) := \{s \in \mathcal{D} \mid ISP(s) \text{ undefined}\} \cup \\ \bigcup\{ISP(s) \mid s \in \mathcal{D} \text{ and } ISP(s) \text{ defined}\} \end{aligned}$$

We will use this to replace all schemas by their partitions, except for the minimal key schema R_{key} which is added to make the decomposition lossless, if needed, and has no partition.

Algorithm "dependency preserving DNF and BCNF decomposition"

```
INPUT: schema $R$, set of FDs $\Sigma$
OUTPUT: decomposition of $R$ in DNF and BCNF

partition $\Sigma^{*a}$ into equivalence classes $EQ$
for each $EQ_j \in EQ$ do
    compute $CC_j$, the set of all partial canonical covers on $EQ_j$
    $IS_j := \emptyset$ (the induced schemas for $EQ_j$)
    $ISP := \emptyset$ (associated partitions in BCNF)
    for each partial canonical cover $pc \in CC_j$ do
        compute the schema partition $sp_{pc}$ induced by $pc$
        $R_{pc} := \bigcup sp_{pc}$ (the schema induced by $pc$)
        $IS_j := IS_j \cup \{R_{pc}\}$
        if (all schemas in $sp_{pc}$ are in BCNF and
           $ISP$ does not map $R_{pc}$ to anything) then
          $ISP := ISP \cup \{R_{pc} \mapsto sp_{pc}\}$
    end
    remove all non-minimal (w.r.t. inclusion) schemas from $IS_j$
    remove all schemas $R_{pc}$ from $IS_j$ with $ISP(R_{pc})$ undefined
end

if $EQ$ contains no key class then
  for all minimal keys $R_{key}$ of $R$ do
      decompose($EQ, \{R_{key}\}$)
  end
else
  decompose($EQ, \emptyset$)
```

```
**proc decompose**($EQ, \mathcal{D}$)
if $EQ = \emptyset$ then
  output $ISP(\mathcal{D})$
  exit
pick maximal $EQ_j \in EQ$
compute $A_{\mathcal{D}} := SHOA_{EQ_j}(\mathcal{D})$
for all $R_j \in IS_j$ do
    if ($R_j \setminus A_{\mathcal{D}}$ minimal with ($\curvearrowright$ c-domination optimal)
            $EQ_j[\mathcal{D} \cup \{(R_j \setminus A_{\mathcal{D}}) \cup A_{\mathcal{D}}\}]$ partial cover for $EQ_j$
        and $R_j$ minimal with ($\curvearrowright$ domination optimal)
            $EQ_j[\mathcal{D} \cup \{R_j\}]$ partial cover for $EQ_j$)
    then
        decompose($EQ \setminus \{EQ_j\}, \mathcal{D} \cup \{R_j\}$)
end
```

Theorem 4.54. *Algorithm "dependency preserving DNF and BCNF decomposition" outputs such a decomposition if one exists.*

Proof. (1) Only schema partitions with schemas in BCNF get added to the ISP_j, and only those get added to $ISP(\mathcal{D})$. Thus $ISP(\mathcal{D})$ is in BCNF. For each essential equivalence class EQ_j, a set of schemas sp gets added to $ISP(\mathcal{D})$ (first $\bigcup sp$ is added to $\mathcal{D}$ and then replaced), which holds a partial cover for EQ_j. These partial covers together form a cover for Σ by Lemma 4.45, thus $ISP(\mathcal{D})$ is dependency preserving. Furthermore, a key schema is added to $ISP(\mathcal{D})$, either initially or to hold a partial cover for EQ_{key}. Together with the fact that $ISP(\mathcal{D})$ preserves dependencies, we obtain that $ISP(\mathcal{D})$ is lossless.

It is easy to see that the algorithm's output $ISP(\mathcal{D})$ and $\mathcal{D}$ are equivalent w.r.t. domination. It thus suffices to show that $\mathcal{D}$ is in DNF, which then implies the same for $ISP(\mathcal{D})$ by definition of DNF. To do this, we will argue that $\mathcal{D}$ could have been produced using algorithm "dependency preserving DNF decomposition", for which we have already proven correctness. To keep things readable, we abbreviate algorithm "dependency preserving DNF and BCNF decomposition" by *A-DP-DNF-BCNF* and algorithm "dependency preserving DNF decomposition" by *A-DP-DNF*.

In A-DP-DNF-BCNF a minimal key R_{key} is added to $\mathcal{D}$ iff EQ contains no key class. Since EQ (at this time) contains all essential equivalence classes, this condition is equivalent to the corresponding condition in A-DP-DNF that the canonical cover Σ contains no key FDs.

After that, both algorithms add one schema R_j to $\mathcal{D}$ for each essential equivalence class $EQ_j \in EQ$, picking maximal classes first. In A-DP-DNF attributes are removed from R_j as long as $\Sigma^{*a}[\mathcal{D} \cup \{R_j\}]$ is partial cover for EQ_j, removing attributes in $A_{\mathcal{D}}$ first. Only FDs with LHSs equivalent to FDs in EQ_j contribute towards a partial cover of EQ_j, and since in A-DP-DNF-BCNF the equivalence classes EQ_j contain all such (atomic) FDs, we have for every decomposition $\mathcal{E}$:

$$\Sigma^{*a}[\mathcal{E}] \text{ is a partial cover of } EQ_j \text{ iff } EQ_j[\mathcal{E}] \text{ is}$$

As there are no other restrictions on the order in which attributes A are checked for removal, this can generate any schema R_j for which

$$\begin{array}{l} R_j \setminus A_{\mathcal{D}} \text{ minimal with} \\ \quad EQ_j[\mathcal{D} \cup \{(R_j \setminus A_{\mathcal{D}}) \cup A_{\mathcal{D}}\}] \text{ partial cover for } EQ_j \\ \text{and } R_j \text{ minimal with} \\ \quad EQ_j[\mathcal{D} \cup \{R_j\}] \text{ partial cover for } EQ_j, \end{array}$$

In particular this includes any schema R_j selected by A-DP-DNF-BCNF, as the conditions are checked in procedure "decompose".

(2) It remains to show that our algorithm always finds a dependency preserving DNF and BCNF decomposition if such a decomposition exists. Our argument is the following: Algorithm "dependency preserving DNF and BCNF decomposition" computes all partial canonical covers pc for each essential equivalence class and their induced schema partitions sp_{pc}. If we were to check all combinations of them, we would obtain all connected component decompositions induced by canonical covers of Σ. By Theorem 4.52 some DP-DNF-BCNF decomposition $\mathcal{E}$ would have to be amongst them (we name it $\mathcal{E}$ to avoid confusion with the variable $\mathcal{D}$ in the algorithm). Our algorithm does essentially that, although it applies some optimizations, and we need to argue that none of them prevents us from finding $\mathcal{E}$ (or at least some decomposition in DP-DNF-BCNF).

The first optimization happens when multiple partial canonical covers for an equivalence class EQ_j induce the same schema but different schema partitions. Here we check which of those schema partitions are in BCNF, and keep only one of them (if any BCNF partition exists at all). Neglecting schema partitions not in BCNF has no impact on the result, since they cannot be part of a BCNF decomposition $\mathcal{E}$. Different partitions of the same schema dominate each other, since all subschemas in the partition have the same closure. Thus, while we may not find the decomposition $\mathcal{E}$, we will find a decomposition $\mathcal{E}'$ which is equivalent to $\mathcal{E}$ w.r.t. domination and thus also in dependency preserving DNF and BCNF.

The second optimization comes with the line

remove all non-minimal (w.r.t. inclusion) schemas from IS

Let $s \in IS$ be non-minimal, i.e., $s' \subsetneq s$ for some $s' \in IS$. Any decomposition containing s is strictly dominated by a decomposition using s' instead of s, and thus not in DNF. Thus removing s from IS does not hinder us in finding $\mathcal{E}$.

The last optimization comes in the form of pruning the search tree with the check

```
if ($R_j \setminus A_{\mathcal{D}}$ minimal with ($\curvearrowright$ c-domination optimal)
      $EQ_j[\mathcal{D} \cup \{(R_j \setminus A_{\mathcal{D}}) \cup A_{\mathcal{D}}\}]$ partial cover for $EQ_j$
   and $R_j$ minimal with ($\curvearrowright$ domination optimal)
      $EQ_j[\mathcal{D} \cup \{R_j\}]$ partial cover for $EQ_j$)
```

Here we will show that any R_j which do not pass the "if-condition" cannot lead to a DP-DNF-BCNF decomposition. This can be argued as follows.

If $R_j \setminus A_{\mathcal{D}}$ is not minimal then we have

$$R'_j \setminus A_{\mathcal{D}} \subsetneq R_j \setminus A_{\mathcal{D}}$$

for some R'_j such that

$$EQ_j[\mathcal{D} \cup \{(R'_j \setminus A_{\mathcal{D}}) \cup A_{\mathcal{D}}\}]$$

forms a partial cover for EQ_j. But then $\mathcal{D} \cup \{R_j\}$ is strictly c-dominated by $\mathcal{D} \cup \{R'_j\}$ by Theorem 4.33, and this still holds after substituting schemas by their partitions using ISP or adding schemas (none of those being of higher order than R_j) to obtain a complete dependency preserving BCNF decomposition. Thus $ISP(\mathcal{D} \cup \{R_j\})$ cannot be completed to a DP-DNF-BCNF decomposition.

If R_j is not minimal we argue similarly with domination and Theorem 4.36. □

We finish by discussing further improvements, which can help to speed up the algorithm. First, we want to stress that the check

$$\begin{array}{c}\text{and } R_j \text{ minimal with } (\curvearrowright \text{ domination optimal}) \\ EQ_j[\mathcal{D} \cup \{R_j\}] \text{ partial cover for } EQ_j)\end{array}$$

in procedure "decompose" is not redundant, despite the removal of all non-minimal R_j in IS_j. The remaining R_j are minimal such that $EQ_j[R_j]$ forms a partial cover of EQ_j, but this is a different condition: While $ISP(\mathcal{D})$ is in BCNF, $\mathcal{D}$ need not be, so the FDs in $EQ_j[\mathcal{D}]$ could contribute towards a partial cover of EQ_j. However, the check can be skipped if $EQ_j[\mathcal{D}] = \emptyset$, which can be tested quickly.

The partial covers for a minimal equivalence class EQ_X of FDs all induce the same schema, namely the closure X^* for EQ_X. This can be argued as follows: Only the FDs in EQ_X can help in computing the closure of a minimal key of EQ_X, and every attribute in X^* must be part of some FD. Furthermore, X^* is in BCNF since no FDs $Y \to A \in \Sigma^{*a}$ with closure $Y^* \subsetneq X^*$ exist. Thus we do not need to compute the partial canonical covers on minimal equivalence classes.

Computing the set of all partial canonical covers on an equivalence class EQ_X directly can be inefficient. To improve this, we try to partition EQ_X into smaller autonomous sets $S_1, \ldots, S_n \subseteq EQ_X$ using partial implication cycles as described in section 3.4, and compute all partial canonical covers on them. We could now obtain the partial canonical covers on EQ_X by forming the cross-union of the partial canonical cover sets on $S_1, \ldots, S_n$, and then proceed with computing the induced schema partitions. Instead, we first compute the sets $ISP_1, \ldots, ISP_n$ of schema partitions induced by the partial canonical covers on $S_1, \ldots, S_n$. Note that different partial canonical covers can (and often do, as our experiments show) induce the same schema partitions. We then compute the cross-union

$$\mathcal{H}_X := ISP_1 \vee \ldots \vee ISP_n$$

which is a set of hypergraphs on R, with each hypergraph corresponding naturally to one or more partial canonical covers on EQ_X. It is easy to see that each hypergraph in $\mathcal{H}_X$ induces the same schema partitions as the partial canonical covers it corresponds to. Since the number of hypergraphs in $\mathcal{H}_X$ can be much smaller than the number of partial canonical covers on EQ_X, computing the induced schema partitions from $\mathcal{H}_X$ can be much faster.

Finally, we may abort immediately if some $ISP|_{IS_j}$ is empty, since then no partial dependency preserving BCNF decomposition without duplicate attributes exists for EQ_j.

4.5.2 DNF and EKNF

As dependency preserving BCNF decompositions do not always exist, combining DNF and BCNF is not always a feasible option. To avoid this problem, or as an alternative should a dependency preserving BCNF decomposition not exist, one could try to ensure other normal forms which always allow dependency preserving decompositions. The most well-known normal form with this property is 3NF. However, as was pointed out in [46], 3NF does not always enforce beneficial decomposition, even though they may not cause any loss of dependencies. The following example, taken from [46], illustrates this.

Example 4.13. Let $R = ABC$ and $\Sigma = \{A \rightarrow B, B \rightarrow A\}$. Then AC and BC are minimal keys of R, and thus all attributes are prime. Therefore R is already in 3NF, even though dependency preserving decompositions exist, such as $\{AB, BC\}$ or $\{AB, AC\}$. □

As an improvement, the authors suggest a new normal form which is stronger than 3NF but still allows dependency preserving decompositions. They strengthen 3NF by allowing as RHS of a non-key FD only those prime attributes, which appear in the LHS of an atomic key dependency. Note that atomic FDs are called *elementary* in [46].

Definition 4.55. [46] Let R be a schema with FDs Σ. A FD $X \rightarrow A$ is called *elementary* if Σ^* contains no FD $X' \rightarrow A$ with $X' \subsetneq X$. A key is elementary if it forms the LHS of an elementary FD. An attribute is an elementary key attribute if it lies in an elementary key of R.

Definition 4.56. [46] Let R be a schema with FDs Σ. Then R is in *elemental key normal form* (EKNF) if for every non-trivial FD $X \rightarrow A$ on R

(a) X is a key of R, or

(b) A is an elementary key attribute for R.

Note that the schema R from example 4.13 is not in EKNF, since neither A nor B are elementary key attributes. Thus EKNF may enforce useful decomposition which 3NF does not.

As it turns out, algorithm "dependency preserving DNF decomposition" already produces a decomposition into EKNF.

Lemma 4.57. *Let R be a single schema with FDs Σ. If R is in dependency preserving DNF, then it is also in EKNF.*

Proof. Assume that R is not in EKNF. Then there exists a FD $X \rightarrow A \in \Sigma^{*a}$ such that X is not a key of R, and A does not lie in the LHS of any key FD in Σ^{*a}. Furthermore, R is strictly c-dominated by the decomposition

$$\mathcal{D} := \{R \setminus A\} \cup \{S \subsetneq R \mid S \text{ is not a key of } R\}$$

since A does not lie in $R \setminus A$, which is the only key schema in $\mathcal{D}$.

It remains to show that $\mathcal{D}$ is dependency preserving. Clearly the only FDs in Σ^{*a} which do not lie in $\Sigma^*[\mathcal{D}]$ are key FDs containing A. They must be of the form $Y \rightarrow A$, since A does not lie in the LHS of any key FD. However, the FDs $Y \rightarrow X$ and $X \rightarrow A$ both lie in $\Sigma^*[\mathcal{D}]$, and together they imply $Y \rightarrow A$. □

Theorem 4.58. *Let $\mathcal{D}$ be a decomposition produced by algorithm "dependency preserving DNF decomposition". Then every schema $R_X \in \mathcal{D}$ is in dependency preserving DNF w.r.t. $\Sigma^*[R_X]$.*

Proof. Let $\mathcal{D}_X$ be any dependency preserving decomposition of R_X. Then $\mathcal{D}_X$ is dominated by the single schema

$$R'_X := \bigcup\{R_j \in \mathcal{D}_X \mid R_j \text{ is a key of } R_X\}$$

Clearly R'_X preserves all key FDs in $\Sigma^*[\mathcal{D}]$. Thus $EQ_X[\mathcal{D}_X] \subseteq EQ_X[R'_X]$, so $EQ_X[R'_X]$ implies $EQ_X[R_X]$. However, R_X has been constructed minimal such that $EQ_X[R_X]$ has some partial cover property. Thus $R'_X = R_X$, which shows that R_X dominates every dependency preserving decomposition $\mathcal{D}_X$. It follows that R_X is in dependency preserving DNF. □

Corollary 4.59. *Algorithm "dependency preserving DNF decomposition" produces a decomposition into EKNF.*

Proof. Follows immediately from the last lemma and theorem. □

We note that this result is only due to our construction method, i.e., dependency preserving DNF does not imply EKNF in general.

Example 4.14. Let $R = ABCD$ and $\Sigma = \{A \to B, B \to C, CD \to A\}$. Then the decomposition $\mathcal{D} = \{ABC, ACD\}$ is in dependency preserving DNF (note that it is *not* strictly c-dominated by $\{AB, BC, ACD\}$ since C already appears in the key schema ACD). However, $\mathcal{D}$ is not in EKNF, since ABC contains $B \to C$ which violates EKNF.

One could say that the benefit of EKNF is that it enforces "locally" well-designed schemas, something which may not be forced by DNF if this "local optimization" does not provide a significant benefit for the overall size of the decomposition. The same holds true for BCNF or other "local" normal forms, i.e., normal forms which consider only a single schema.

Chapter 5

Summary

We will briefly summarize the main results we obtained, and related problems which still remain open.

5.1 Main Results

In chapter 2 we developed algorithms for computing a dependency preserving BCNF decomposition. The main result was an "linear resolution" algorithm for computing the atomic closure Σ^{*a} for a set of functional dependencies Σ. While Σ^{*a} can be exponential in Σ, we identified polynomial cases and showed that for finding a dependency preserving BCNF decomposition, it suffices to compute a subset of Σ^{*a}. Finally, we demonstrated how the results can be extended to a complex-valued data model.

The "linear resolution" algorithm was then used in chapter 3 to compute the set of all canonical covers $CC(\Sigma)$. For that we showed how hypergraphs can be decomposed using autonomous sets, which led to an efficient representation of $CC(\Sigma)$. Perhaps more important than the actual algorithm, we obtained insights into how functional dependencies interact. In particular, Theorem 3.36 allows us to split the task of creating a canonical cover into smaller, independent tasks of creating partial covers. Our theory of autonomous sets may well have applications in other disciplines.

In chapter 4 we returned to database normalization by defining a new normal form "DNF" and providing algorithms for computing decompositions into DNF. This new normal form was characterized both semantically and syntactically, and one of the main difficulties was in showing that the characterizations match. We established that in some sense, DNF is the proper generalization of existing normal forms, in particular BCNF, onto multiple schemas. Finally, we showed how dependency preserving DNF decompositions can be computed, and how dependency preserving DNF and BCNF can be obtained simultaneously.

Overall, this work focused on computing good schema decompositions. We provided characterizations of such "good" decompositions and practical algorithms to obtain them. The results offer new insights, in particular into the interaction of functional dependencies, and are of immediate practical use in creating automated design tools.

5.2 Open Problems

We will point out some open problems related to our work, in the order as they appear.

The results we obtained are based on the relational data model. In section 2.3 we discussed how the main results of chapter 2 can be extended to complex-valued data models. We believe that most (if not all) of the work done here can be extended to complex-valued data models, including XML-databases, in a similar fashion, though we did not investigate this. Other extensions to the classic relational model may also be of interest for further research. In particular, FDs behave differently in the presence of null values, which may lead to quite different results for derivation and normalization problems.

In section 3.5 we discussed how we could restrict ourselves to deriving essential FDs when computing all canonical covers. This could be of use when trying to find dependency preserving decompositions, as we did in chapter 2. The problem here lies in testing whether a FD is critical, and it is not clear how this can be done efficiently, having computed only essential FDs. In some cases, the work done by Gottlob [18] might offer an efficient solution.

The connection between the semantic and syntactic characterizations of DNF has been established using theorems 4.33 and 4.36. However, Theorem 4.36 is restricted to functional dependencies, and does not hold in the presence of multivalued or join dependencies. It would be interesting to know what the correct syntactic characterization of DNF would be in such cases.

Also, our assumption that all domains are infinite and unbounded is not always realistic. It is not clear yet how DNF behaves for finite or bounded domains.

Our definition of DNF focuses on minimizing size, with the intuition that this could minimize redundancy and update anomalies as well. Alternatively one could attempt to minimize redundancy or update anomalies directly, and it would be interesting to see whether the results are the same. Additionally, one could try to achieve other desirable properties for a decomposition, such as e.g. acyclicity [6].

While we have an algorithm for computing a decomposition into dependency preserving DNF, we currently lack such an algorithm for lossless DNF.

Finally, it is unclear how to test whether a given decomposition is in DNF, and how difficult such a test is. We suspect that it is at least co-NP hard.

Index

3NF, 6

algorithm
- 3NF Synthesis, 7
- BCNF Decomposition, 7
- closure, 5
- d.p. DNF and BCNF decomposition, 120
- d.p. DNF decomposition, 114
- divide and resolve, 70
- essential linear resolution, 86
- least critical cover synthesis, 28
  - revised, 31
- linear NA-resolution, 43
- linear resolution, 16
  - revised, 18
- partial implication partitioning, 79
- recursive autonomous partitioning, 61
- substitute FD, 30
- update atomic closure, 23

armstrong axioms, 4
associated set, 98
atomic closure, 12
atomic implication closure, 77
attribute, 3
- cyclic, 33
- depending, 98
- extraneous, 15
- flat, 36
- nested, 36
  - basis, 41
  - completion, 41
  - representable, 41
  - unitary, 40
- prime, 6, 83

autonomous set, 55, 62

BCNF, 6

c-domination, 91–93
cartesian, 80
complete, 4
connected components, 57
containing schema closures, 93
cover, 4
- canonical, 12
- LR-reduced, 72
- partial, 63, 113
- relative, 64

csc-domination, 94
cycle, 32
cyclic closure, 33

decomposition
- connected component, 118
- lossless, 5
- principal, 118

dependency
- functional, 4
  - adjacent, 76
  - atomic, 12, 41
  - base, 14
  - critical, 26, 46
  - derived, 14
  - essential, 80
  - essentially derivable, 82
  - forced, 80
  - maximal atomic, 41
  - neighborhood, 80
  - redundant, 4
  - singular, 12, 41
  - standard form, 41
  - substituting, 14
  - trivial, 4
  - unitary, 20
- join, 5
- multivalued, 5

dependency basis, 101
derivation trees, 4
domain, 3
domination, 81, 91–93

domination normal form, 91

edge
  partial, 59
elemental key normal form, 124
elementary, 124
equivalence, 63, 64, 112
essential closure, 81
essentially cartesian closure, 81

faithful, 6
finite implication, 4
fixed parameter tractable, 22

hypergraph, 54
  simple, 54

implication cover, 67
  relative, 67
implication dependency, 67
inclusion-domination, 94
integrity constraints, 3
isolated set, 57

join, 5, 38
  lossless, 6

k-mapping, 97
k-reducing, 96
KCNF, 107

LHS-minimal, 7
LHS-reduced, 7
LHS-restricted, 49
lifting, 97
LM-resolution, 15

meet, 38
minimal autonomous sets, 56
minimal cover, 18
minimal isolated set, 57

partial determination, 32
partial implication, 76, 77
projection, 6, 53

relation, 3
relativation, 65
representation basis, 40
resolution:linear, 15
restricted, 49

schema, 3
  closed, 99
  induced, 117
  induced partition, 117
  open, 99
sound, 4
subset construction
  for functional dependencies, 99
  generalized, 102
superedge, 59
  partial, 59
support, 54
synchronization hypergraph, 100

transversal, 57, 62

width, 21

Bibliography

[1] M. Arenas and L. Libkin. An information-theoretic approach to normal forms for relational and XML data. In *PODS*, pages 15–26, 2003.

[2] W. W. Armstrong. Dependency structures of data base relationships. In *IFIP Congress*, pages 580–583, 1974.

[3] G. Ausiello, A. D'Atri, and D. Saccà. Graph algorithms for functional dependency manipulation. *J. ACM*, 30(4):752–766, 1983.

[4] G. Ausiello, A. D'Atri, and D. Saccà. Minimal representation of directed hypergraphs. *SIAM J. Comput.*, 15(2):418–431, 1986.

[5] C. Beeri and P. A. Bernstein. Computational problems related to the design of normal form relational schemas. *ACM Transactions on Database Systems*, 4(1):30–59, 1979.

[6] C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the desirability of acyclic database schemes. *J. ACM*, 30(3):479–513, 1983.

[7] C. Berge. *Hypergraphs: Combinatorics of Finite Sets.* Elsevier Science Pub. Co., 1989.

[8] J. Biskup. Inferences of multivalued dependencies in fixed and undetermined universes. *Theor. Comput. Sci.*, 10:93–105, 1980.

[9] J. Biskup. Achievements of relational database schema design theory revisited. In L. Libkina and B. Thalheim, editors, *Semantics in Databases*, volume 1358 of *Lecture Notes in Computer Science*, pages 29–54. Springer, 1995.

[10] J. Biskup, U. Dayal, and P. A. Bernstein. Synthesizing independent database schemas. In *SIGMOD Conference*, pages 143–151, 1979.

[11] M. A. Casanova, R. Fagin, and C. H. Papadimitriou. Inclusion dependencies and their interaction with functional dependencies. In *PODS '82*, pages 171–176, New York, NY, USA, 1982. ACM Press.

[12] E. F. Codd. Further normalization of the data base relational model. *IBM Research Report, San Jose, California*, RJ909, 1971.

[13] R. G. Downey and M. R. Fellows. *Parameterized Complexity.* Springer, 1999.

[14] R. Fadous and J. Forsyth. Finding candidate keys for relational data bases. In W. F. King, editor, *Proceedings of the 1975 ACM SIGMOD International Conference on Management of Data, San Jose, California, May 14-16, 1975*, pages 203–210. ACM, 1975.

[15] R. Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst.*, 2(3):262–278, 1977.

[16] R. Fagin. Normal forms and relational database operators. In P. A. Bernstein, editor, *SIGMOD Conference*, pages 153–160. ACM, 1979.

[17] M. R. Garey, D. S. Johnson, and L. J. Stockmeyer. Some simplified NP-complete graph problems. *Theor. Comput. Sci.*, 1(3):237–267, 1976.

[18] G. Gottlob. Computing covers for embedded functional dependencies. In *PODS*, pages 58–69, 1987.

[19] G. Gottlob. On the size of nonredundant FD-covers. *Inf. Process. Lett.*, 24(6):355–360, 1987.

[20] G. Gottlob, R. Pichler, and F. Wei. Tractable database design through bounded treewidth. In *PODS*, pages 124–133, 2006.

[21] F. Harary. *Graph Theory.* Addison-Wesley, 1995.

[22] J. Järvinen. Dense families and key functions of database relation instances. In R. Freivalds, editor, *FCT*, volume 2138 of *Lecture Notes in Computer Science*, pages 184–192. Springer, 2001.

[23] P. Kandzia and M. Mangelmann. On covering Boyce-Codd normal forms. *Inf. Process. Lett.*, 11(4/5):218–223, 1980.

[24] H. Koehler. Finding faithful Boyce-Codd normal form decompositions. In S.-W. Cheng and C. K. Poon, editors, *AAIM*, volume 4041 of *Lecture Notes in Computer Science*, pages 102–113. Springer, 2006.

[25] H. Koehler. Domination normal form: decomposing relational database schemas. In *ACSC '07: Proceedings of the thirtieth Australasian conference on Computer science*, pages 79–85, Ballarat, Australia, 2007. Australian Computer Society, Inc.

[26] J. Lechtenbörger. Computing unique canonical covers for simple FDs via transitive reduction. *Inf. Process. Lett.*, 92(4):169–174, 2004.

[27] M. Levene and G. Loizou. *A Guided Tour of Relational Databases and Beyond.* Springer, 1999.

[28] E. A. Lewis, L. C. Sekino, and P. D. Ting. A canonical representation for the relational schema and logical data independence. In *IEEE COMPSAC*, pages 276–280, 1977.

[29] S. Link. *Dependencies in Complex-valued Databases.* PhD Thesis, 2004.

[30] S. Link. On multivalued dependencies in fixed and undetermined universes. In *FoIKS*, pages 258–277, 2006.

[31] C. L. Lucchesi and S. L. Osborn. Candidate keys for relations. *Journal of Computer and System Sciences*, 17(2):270–279, 1978.

[32] D. Maier. Minimum covers in the relational database model. *Journal of the ACM*, 27(4):664–674, 1980.

[33] D. Maier. *The Theory of Relational Databases.* Computer Science Press, 1983.

[34] M. E. Majster-Cederbaum. Ensuring the existence of a BCNF-decomposition that preserves functional dependencies in O(n) time. *Information Processing Letters*, 43(2):95–100, 1992.

[35] H. Mannila and K.-J. Räihä. Design by example: An application of armstrong relations. *J. Comput. Syst. Sci.*, 33(2):126–141, 1986.

[36] H. Mannila and K.-J. Räihä. *The Design of Relational Databases.* Addison-Wesley, 1987.

[37] H. Mannila and K.-J. Räihä. Practical algorithms for finding prime attributes and testing normal forms. In *Proceedings of the Eighth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, March 29-31, 1989, Philadelphia, Pennsylvania*, pages 128–133. ACM Press, 1989.

[38] S. L. Osborn. Testing for existence of a covering Boyce-Codd normal form. *Information Processing Letters*, 8(1):11–14, 1979.

[39] H. Saiedian and T. Spencer. An efficient algorithm to compute the candidate keys of a relational database schema. *Comput. J.*, 39(2):124–132, 1996.

[40] R. C. Shock. Computing the minimum cover of functional dependencies. *Inf. Process. Lett.*, 22(3):157–159, 1986.

[41] R. E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[42] D.-M. Tsou and P. C. Fischer. Decomposition of a relation scheme into Boyce-Codd normal form. In *ACM annual conference*, pages 411–417, 1980.

[43] M. W. Vincent. A corrected 5NF definition for relational database design. *Theor. Comput. Sci.*, 185(2):379–391, 1997.

[44] M. W. Vincent. Redundancy elimination and a new normal form for relational database design. In *Semantics in Databases*, volume 1358 of *Lecture Notes in Computer Science*, pages 247–264. Springer, 1998.

[45] J. Wang and R. W. Topor. Removing XML data redundancies using functional and equality-generating dependencies. In H. E. Williams and G. Dobbie, editors, *ADC*, volume 39 of *CRPIT*, pages 65–74. Australian Computer Society, 2005.

[46] C. Zaniolo. A new normal form for the design of relational database schemata. *ACM Trans. Database Syst.*, 7(3):489–499, 1982.

Appendix

Some of the algorithms described in this work have been implemented, and we present experimental results using randomly generated test data.

Test Data

One issue that always arises in this context is that of using appropriate test data. In our case, we require relations R and sets of FDs Σ on them.

These were created as follows: For a fixed number k of attributes in R and n FDs in Σ, we generate n FDs on R independently at random. In that, a FD $X \to Y$ is constructed by independently choosing the attribute sets $X, Y \subseteq R$. We create an attribute subset by selecting m attributes uniformly at random. In this we allow duplicates, so that the resulting set could have cardinality less than m. The probability for m to have value v is $\frac{1}{2}^{v}$, i.e., there is a chance of $\frac{1}{2}$ that $m = 1$, a chance of $\frac{1}{4}$ that $m = 2$ a.s.o.

Note that this approach favors FDs with small left- and right hand sides, but every FD $X \to Y$ on R with non-empty sets X, Y can potentially be generated.

Algorithms

The following algorithms have been implemented and tested:

- Linear Resolution (for computing atomic closure)

  We implemented the revised version from section 2.1.2. This improved performance slightly, compared to the basic version of the algorithm.

- Least Critical Cover Synthesis (for finding faithful BCNF decomposition)

  We implemented the revised version from section 2.2.2. This had a huge impact on performance in all cases, again compared to the basic algorithm.

- Divide and Resolve (for computing all partial canonical covers)

  We used partial implication cycles to find smaller autonomous sets, as described in section 3.4. This had a huge impact on performance in many cases.

- dependency preserving DNF and BCNF Decomposition

  We implemented all the improvements described in section 4.5.1. Computing the induced schema partitions on the smaller autonomous sets before forming the cross-union had a huge impact on performance in some cases.

Results

The tests are organized by number of attributes k in the relation R and number of FDs n in the original cover Σ. For each case we ran our algorithms on 1000 test sets of FDs and recorded the following data:

- size of the atomic closure Σ^{*a}, measured in number of FDs
- number of canonical covers in $CC(\Sigma)$
- number of partial canonical covers per autonomous set (average)
- number of disjoint non-trivial[1] autonomous sets found
- time taken for computing:
  - atomic closure Σ^{*a}
  - canonical covers $CC(\Sigma)$ in decomposed form (after computing Σ^{*a})
  - dependency preserving BCNF decomposition (after computing Σ^{*a})
  - dependency preserving DNF and BCNF decomposition (after computing $CC(\Sigma)$)

  For purpose of judging runtime, we note that the implementation was in java, and tests were performed on a 1.8 GHz PC.

We summarize the test results by reporting the mean value for each parameter measured, i.e., the minimum value x such that at least 50% of the test results are less or equal to x.

We report the mean instead of e.g. the average, since the parameters measured can grow exponentially, and do so in some cases. As a result, only the maximal value has a significant impact on the average. Furthermore, we aborted tests in some cases to avoid excessive computation time and/or memory problems. Such invalid tests do no pose much of a problem for measuring the mean values, since we can safely count them as results larger than the mean (as long as less than 50% of all tests are invalid, which was always the case).

$\#Att$	$\#FDs$	$\#AFDs$	$\#CCs$	$\#PCCs$	$\#Aut$	t_{AC}	t_{CCs}	t_{BCNF}	t_{DNF}
5	5	6	1	1	4	$< 1ms$	$2ms$	$1ms$	$1ms$
10	5	10	1	1	6	$< 1ms$	$3ms$	$1ms$	$1ms$
10	10	26	12	1.4	9	$1ms$	$24ms$	$1ms$	$3ms$
15	10	37	2	1	13	$3ms$	$20ms$	$2ms$	$4ms$
15	15	71	284	1.9	17	$7ms$	$144ms$	$4ms$	$10ms$
20	15	88	18	1.2	21	$48ms$	$98ms$	$5ms$	$11ms$
20	20	156	10935	3.4	25	$62ms$	$450ms$	$7ms$	$36ms$
30	15	105	2	1	24	$59ms$	$32ms$	$6ms$	$12ms$
30	20	222	24	1.1	31	$100ms$	$177ms$	$9ms$	$21ms$
50	20	198	1	1	36	$113ms$	$88ms$	$10ms$	$68ms$
50	30	1026	24	1.1	52	$826ms$	$445ms$	$373ms$	$664ms$

[1]By trivial we mean autonomous sets for which the empty set is a partial cover.

Observations

We shall conclude by noting some observations we made when examining the test results.

- The algorithm “Least Critical Cover Synthesis” performs much faster than suggested by the complexity bound which we established in section 2.2.2. This appears to be due to the improvements we made over the basic algorithm. It is not clear whether the complexity bound established can be improved.

- Algorithm “dependency preserving DNF and BCNF Decomposition” also performed much faster than we expected. The main reason for this appears to be that backtracking after a recursive call of procedure “decompose” is hardly ever necessary.

- In cases where the ratio of FDs over attributes is large, we very often obtained huge numbers of canonical covers. The reason for this becomes clear when we observe the asymptotic case, where Σ contains all non-trivial FDs with non-empty LHS. Then every attribute in R is a key, and Σ^{*a} is small (quadratic in the number of attributes k). However, the number of canonical covers becomes hyper-exponential in the number of attributes. This can be seen as follows: Every canonical cover corresponds to a minimal strongly connected graph on k vertices. This includes all directed cycles (among others), and there are $(k-1)!$ such cycles.

  In all examples we observed, where the number of canonical covers was huge, we found a maximal equivalence class which had a structure very similar to the asymptotic case. It thus might be worth investigating how computing all partial canonical covers on such equivalence classes could be avoided.