Dependencies in Complex-value Databases

Sebastian Link

A dissertation presented in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Information Systems at Massey University

Supervisor:
Prof. Dr. Klaus-Dieter Schewe

Co-Supervisor:
Prof. Dr. Mike Hendy

Internal Examiner:
Associate Prof. Dr. Sven Hartmann

New Zealand Examiner:
Prof. Dr. Robert Goldblatt

Overseas Examiner:
Prof. Dr. Joachim Biskup

Date of Examination:
15.12.2004

Abstract

The relational data model has been the dominant model in database design for more than three decades. It considers data to be stored in matrices where rows correspond to individuals, columns correspond to attributes, and every cell constains a single atomic value. However, today's database technology trends, e.g. spatial, genetic or web-based data, require extended data models. Within the last decade new, complex-value data models such as the higher-order entity-relationship model, object-oriented data models, semi-structured data models, and XML have evolved which allow cells to contain lists, sets, multisets, trees, matrices or even more complex type constructors, references to other cells (which lead to infinite structures), and null values (indicating missing, unknown or vague data).

Matrices as such allow the storage of inconsistent data, invalid in the semantic sense. As this is not acceptable, additional requirements called dependencies have to be formulated when designing a database. The correct specification and use of dependencies needs a sound mathematical basis. For the relational data model more than 90 different classes of dependencies have been defined and studied intensively. The major problems in dependency theory are the axiomatisability of classes of dependencies, determination of the closure of a chosen set of dependencies (as certain dependencies can be implied by others) and the characterisation of semantically desirable properties for well-designed databases (such as absence of redundancies or abnormal update behavior) by syntactic properties on closed sets of dependencies.

With few exceptions research has only dealt with dependencies for the relational data model. Only recently, the emergence of XML as the standard format for web-based data and the rapidly increasing usage of persistent XML databases revealed the lack of a sound mathematical basis for complex-value data models. If they are expected to serve as first class data models they require a theoretical investigation of issues like integrity, consistency, data independence, recovery, redundancy, access rights, views and integration. The goal of this thesis is to develop a dependency theory for complex-value databases that is independent from any individual data model. Therefore, an abstract algebraic approach is taken that can be adapted to the presence of different combinations of type constructors such as records, lists, sets and multisets. Data models are classified according to the data types they support. In this framework the major objective is to initiate research on the following problems

- investigate the axiomatisation of important dependency classes, relevant to complex-value data models, by sound and complete sets of inference rules that permit the determination of all dependencies implied by some chosen set of dependencies.
- characterise semantically desirable properties by normal forms for complex-value data models and investigate whether these normal forms can always be achieved without violating other desirable properties.
- develop efficient algorithms for determining the closure of a chosen set of dependencies and for restructuring databases such that normal forms are satisfied and no information is lost.

In a single thesis it is impossible to consider all classes of relational dependencies in all different combinations of type constructors. Therefore the focus is put on extending two popular classes of relational dependencies: functional and multi-valued dependencies. The axiomatisation and implication of functional dependencies is investigated for all combinations of record, list, set and multiset type. Furthermore, a normal form with respect to functional dependencies in the presence of records and lists is proposed and semantically justified. It is also shown how to obtain databases which are in this normal form. Finally, axiomatisation and implication for the class of multi-valued dependencies and the class of functional and multi-valued dependencies are studied in the context of records and lists. The work of this thesis may lead to a unified dependency theory for complex-value data models.

Acknowledgement

I would like to thank *Klaus-Dieter Schewe* for giving me the opportunity to pursue an academic career under his supervision. Klaus-Dieter introduced me to theoretical computer science, and in particular to the topic of databases. His enthusiastic attitude towards research has motivated me since the beginning of my studies. Klaus-Dieter suggested to look at dependencies in the Higher-Order Entity-Relationship model, and it was later on that the investigations resulted in a more general treatment. The idea of classifying data models according to the data types they support is similar to the idea from [243] where object-oriented data models are classified according to different type systems. I also like to thank Klaus-Dieter for showing me how to work scientifically and for supporting decisions to reduce my teaching workload at critical times.

My thank also goes to *Mike Hendy* who kindly agreed to act as co-supervisor of this thesis.

Special thanks go to *Sven Hartmann* for having numerous fruitful discussions on the subject of this work. Sven made a lot of suggestions for improving the outcome, in particular on the mathematical rigorousness of various notions and proof arguments. Sven has always been prepared to talk about any issues, even in times he was very busy. I have learned a great deal from him.

I would further like to acknowledge *Bernhard Thalheim* and *Joachim Biskup* whose great experiences in the field of databases I have benefited from a lot.

Finally, I would like to thank my parents *Karla* and *Hans-Jürgen Link* who have always been there for me and supported me in every way possible.

Moreover, I am very thankful to my partner *Toni Floyd* who can always solve all my problems at once just by being who she is.

I dedicate this thesis to my parents,
Karla and *Hans-Jürgen Link*.

Table of Contents

List of Figures

Chapter 1

Introduction

The first goal of this chapter is to provide an informal overview of achievements in research on dependency theory in the context of the relational data model (RDM). The aim is not to give a complete overview, but to keep focus on those results which are relevant for this thesis. The second major objective is to motivate the need for an extension of these achievements to deal with complex objects that cannot be described by purely relational structures. The introduction has been inspired to great parts by [158, 181, 264].

1.1 Relational Dependency Theory

Commercial database management systems have been around for more than three decades now, at the beginning in the form of hierarchical and network models. It was in the early seventies when two opposing trends in database research started. The development of semantic data models was mostly influenced by semantic networks. These are generally object-oriented and provide at least four types of primitive relationships between objects: classification (instance of), aggregation (part of), generalisation (is-a), and association (member of). On the other hand, the RDM revolutionised the field by strictly separating data representation from the underlying implementation. Most significantly, the inherent simplicity of the model admitted the development of powerful, non-procedural query languages and a lot of useful theoretical results.

Generalised database management systems are considered as basic tools for programming languages, translators and operating systems. Much effort is devoted to establish a definite foundation of database technology in order to design more efficient and transparent systems and to enable optimisation methods. With such an improved understanding of the systems application will be improved as well. The philosophy behind database technology is sometimes not quite understood because many users are unaware of the goals of database management systems. Consequently, these systems are often used incorrectly. The first step towards a solid foundation of database theory is a precise definition of data models. Without a precise definition, a data model cannot be understood for purposes of design, analysis, and implementation of schemata, transaction and databases. A database model is a collection of mathematically sound concepts defining the intended structural

and behavioral properties of objects involved in a database application. In the axiomatic approach, a database model is defined by the properties of its structures and operators. By the axiomatic approach conventional mathematics and logic were used to define the structural and behavioral properties of objects within the database model. Properties of data structures are given by axioms which are formal statements simple enough to be self-evident. Behavioral or dynamic properties are the operations that together with the data structures form the data model. Behavioral properties are given by inference rules which permit the deduction of the resulting properties for each meaningful database operation. In terms of logic, the semantics of each database within the database model can be deduced precisely by the application of valid inference rules to the set of axioms. Alternatively, the semantics of a syntactically correct schema are given by the axioms which characterise the databases to be accepted.

One of the most important database models is the RDM. One of the major advantages of the RDM is its uniformity. All data are seen being stored in tables, with each row in the table having the same format. Each row in the table represents some object or relationship in the real world. The benefits and aims of the RDM are: to provide data schemata which are very simple and easy to use, to improve logical and physical independence without references to the means of access to data, to provide users with high level languages which could be used by non-specialists in computing, to optimise access to the database, to improve integrity and confidentiality, to take into account a wide variety of applications, to provide a methodological approach for schema design and database design.

These benefits are based on a powerful theory the core of which is the theory of dependencies. Database dependencies can be regarded as a language for specifying the semantics of databases. They specify which of the databases are meaningful for the application and which of them are meaningless. Thus, the syntactic specification is joined with semantic specification. Dependencies constitute an inherent property of database systems. They express the different ways by that data are associated with one another. Since many different associations of data exist, a lot of different classes of dependencies (more than 90) are considered in more than one thousand papers. For some classes the implication problem is solved. By studying their respective properties it can be shown how different types of dependencies interact with one another. These properties may be considered as inference rules which allow the deduction of new dependencies as well as the generation of the closure of all dependencies. Solving this problem we can test whether two given sets of dependencies are equivalent or whether a given set of dependencies is redundant. A solution for these problems seems to be a significant step towards automated database schema design, towards automated solution of the seven aims mentioned above and towards recognising computationally-feasible problems and the infeasible ones.

At least five fields of applying dependency theory are known:

- normalisation for a more efficient storage, search and modification,
- reduction of relations to subsets with the same information together with the semantic constraints,
- utilisation of dependencies for deriving new relations from basic relations in the view

concept or in so-called deductive databases,
- verification of dependencies for a more powerful and user-friendly, nearly natural language design of databases,
- transformation of queries into more efficient search strategies.

Other important applications of the relational database theory are in other branches of computer science, in discrete mathematics, in most of other database models, in optimisation, in pattern recognition and in algebra.

1.1.1 Relational Dependencies

Relational dependencies, in general, are semantically meaningful and syntactically restricted sentences of the predicate calculus that must be satisfied by any legal database. Their presence remedies some of the semantic poverty of relations, e.g., with pure relations one has trouble representing the fact that some relationships are one-to-one or one-to-many. Since the topic of this thesis is dependency theory, at least the definition of relational dependencies should be given. For details we refer to [264].

A relation schema $\mathcal{R}$ is given by a finite set R of so-called attributes, a set $\mathcal{D}$ of domains, and by a domain function $dom : R \to \mathcal{D}$ which associates a domain with every attribute. If $\mathcal{D}$ and dom are obvious or defined by the context or arbitray or not of importance for the topic under consideration then $\mathcal{D}$ and dom are omitted, and a relation schema is simply a finite set R of attributes.

A tuple on $\mathcal{R} = (R, \mathcal{D}, dom)$ is a function $t : R \to \bigcup_{D \in \mathcal{D}} D$ with $t(A) \in dom(A)$ for all $A \in R$. If an order is defined on R, say $R = \{A_1, \ldots, A_n\}$, the tuple t can be represented by $(t(A_1), \ldots, t(A_n))$. For $X \subseteq R$ let $t[X]$ denote the restriction of the function t to X. The set of all tuples on $\mathcal{R}$ is denoted by $T(\mathcal{R})$. Any subset r of $T(\mathcal{R})$ is called a relation on $\mathcal{R}$.

A given sequence $\mathcal{S} = \mathcal{R}_1, \ldots, \mathcal{R}_n$ of relation schemata is compatible if $dom_i(A) = dom_j(A)$ holds for all $A \in R_i \cap R_j$ where $\mathcal{R}_i = (R_i, \mathcal{D}_i, dom_i)$. For a compatible sequence of relation schemata, a common function dom with $dom(A) = dom_i(A)$ for $A \in R_i$ can be defined. For a given compatible sequence $\mathcal{S} = \mathcal{R}_1, \ldots, \mathcal{R}_n$ of relation schemata and a function $C : \mathcal{P}(T(\mathcal{R}_1) \times \cdots \times T(\mathcal{R}_n)) \to \{0, 1\}$ where $\mathcal{P}(S)$ denotes the powerset of S, the pair $(\mathcal{S}, C)$ is called database schema, and the function C is called integrity constraint. For a given database schema $(\mathcal{S} = \mathcal{R}_1, \ldots, \mathcal{R}_n, C)$ a relational database is given by the sequence $r_1, \ldots, r_n$ where the r_i are relations on $\mathcal{R}_i$ for $1 \leq i \leq n$ and $C(r_1, \ldots, r_n) = 1$.

The function C can be made more concrete, i.e., defined using a purely relational first-order language with equality over a compatible sequence $\mathcal{S} = \mathcal{R}_1, \ldots, \mathcal{R}_n$ of relation schemata [264, p. 10]. A database schema is then a pair $(\mathcal{S}, \Sigma)$ where Σ denotes a set of well-formed formulae over that language. If not stated otherwise we will assume from now on that Σ is always finite. Only such databases are considered for $(\mathcal{S}, \Sigma)$ in which all integrity constraints in Σ are valid, i.e., for a given database schema $(\mathcal{S} = \mathcal{R}_1, \ldots, \mathcal{R}_n, \Sigma)$ a database is given by the family $(r_1, \ldots, r_n)$ where the r_i are relations on $\mathcal{R}_i$ for $1 \leq i \leq n$ and the formulae from Σ are valid. Validity is defined in the usual way, see [264, pp. 11,12].

The class of dependencies is a class of integrity constraints that must be satisfied by the database of interest. Suppose two compatible sequences $\mathcal{S}_j = \mathcal{R}_{j_1}, \ldots, \mathcal{R}_{j_k}$ with $j \in \{1, 2\}$ of relation schemata $\mathcal{R}_{j_i} = (R_{j_i}, \mathcal{D}_{j_i}, dom_{j_i})$ with $R_{j_i} = \{A_{i_1}, \ldots, A_{i_n}\}$ for $i = 1, \ldots, k$ are given. A database $(r_1, \ldots, r_k)$ over $\mathcal{S}_1$ and a database $(s_1, \ldots, s_k)$ over $\mathcal{S}_2$ are said to be similar if they have exactly the same relations, i.e., $r_i = s_i$ for $1 \leq i \leq k$. A well-formed formula φ is said to be domain-independent, if for all similar databases $r = (r_1, \ldots, r_k)$ and $s = (s_1, \ldots, s_k)$ it is true that r satisfies φ if and only if s satisfies φ. The aim of this special class of formulae is to be able to determine the satisfiability of a formula in a database by merely taking into consideration the values that appear in one of the tuples given by the relations. One can say that domain-independent formulae guarantee that the elements of a response constitute elementary information actually contained in the relation. A database $(r_1, \ldots, r_n)$ is called trivial if $\mid r_i \mid \leq 1$, for $1 \leq i \leq n$.

Domain-independent formulae which hold in any trivial relational database are called relational dependencies.

The main property of dependencies, the domain-independence can be considered as the independence of formulae from the domains used by the database schema. If we consider only dependencies, the formulae can be considered for a class of languages which are using the same attribute sets and the same predicates, but which are independent from the underlying domains. The formula $\exists x_1, \ldots, \exists x_n P(x_1, \ldots, x_n, c)$ called existence constraint in [168] is not domain-independent and therefore not a dependency.

In what follows, we consider different classes of dependencies, but not from a logical point of view. For a classification and systematic study of relational dependencies see [109, 158, 264].

In the following, we will focus on two of the most important classes of relational dependencies: functional dependencies (FDs) and multi-valued dependencies (MVDs). According to a study in [87], FDs make up approximately two thirds of uni-relational dependencies (dependencies defined over a single relation schema) in use. Moreover and according to the same study, the class of FDs and MVDs together constitutes around 75 percent of uni-relational dependencies used in practical applications. It is therefore the goal of the next section to give an overview of some results on these two classes of relational dependencies. The overview will focus on those results which will be extended to complex-value databases in this thesis.

1.1.2 Functional Dependencies

Dependencies constitute an inherent property of database systems. They express the different ways that data are associated with each other and therefore, the semantics in relational database schemata. Functional dependence is an important property of a relation. In a relation which satisfies some FD, there is a functional connection between parts of tuples. FDs can be defined like functions $f : X \to Y$ which are mappings satisfying the conditions:

- for each element $x \in X$ there is an element $y \in Y$ with $f(x) = y$,

– for all $x, x' \in X$: if $x = x'$, then $f(x) = f(x')$.

The second property of functions is used for the definition of FDs.

A functional dependency (FD), defined on some relation schema R, is an expression $X \rightarrow Y$ where $X, Y \subseteq R$. A relation r over R satisfies the FD $X \rightarrow Y$, denoted by $\models_r X \rightarrow Y$, if and only if for every $t_1, t_2 \in r$ the following condition is satisfied: if $t_1[X] = t_2[X]$, then also $t_1[Y] = t_2[Y]$. That is, the values on X uniquely determine the values on Y.

Axiomatisation. Functional dependencies are not independent from one another. If a relation exhibits certain FDs, then there are usually other FDs which are satisfied by that relation as well. This applies to dependencies in general and leads ultimately to the notion of logical implication.

That is, a dependency σ is *implied* by a set Σ of dependencies, denoted by $\Sigma \models \sigma$, if σ is satisfied by every relation which already satisfies all dependencies in Σ. In general, this notion is different from finite implication where Σ *finitely implies* σ, denoted by $\Sigma \models_f \sigma$, if every finite relation satisfying all dependencies in Σ also satisfies σ.

However, in the case of FDs, finite and unrestricted implication problem coincide and are therefore decidable. Although finite implication is the relevant notion from a practical standpoint, implication is also important because it is closely related to unsatisfiability of logical sentences.

If a database designer chooses several FDs to be satisfied by every meaningful relation over the relation schema analysed, then *all* implied FDs have to be determined. This allows to gain complete knowledge about all consequences of the semantics defined, and may avoid inconsistencies and undesired behavior. In practice, however, it is not possible to study all relations and determine whether a dependency is implied by some given set of dependencies. Therefore, one is much more interested in syntactical inference rules which may allow to solve this implication problem. Such inference rules have the form $\frac{A_1, \ldots, A_n}{C}\beta$ with parameterised dependencies $A_1, \ldots, A_n$ called premises, a further parameterised dependency C called the conclusion, and a constraint β on $A_1, \ldots, A_n, C$ which needs to be satisfied in order to apply the rule. Let Σ be a set of dependencies and σ a further dependency. Then σ is *derivable* from Σ using a set $\mathfrak{R}$ of such inference rules, if there is a finite sequence $\sigma_1, \ldots, \sigma_n = \sigma$ of dependencies such that every σ_i is an element of Σ or an instantiation of a conclusion in any of the rules in $\mathfrak{R}$ where the instantiations of all the premises in that rule must be among $\{\sigma_j : 1 \leq j < i\}$ and the constraint β is satisfied. $\mathfrak{R}$ is called *sound* for the implication of dependencies, if every dependency which can be derived from Σ using only inference rules in $\mathfrak{R}$, is also implied by Σ. The set $\mathfrak{R}$ is called *complete* for the implication of dependencies if every dependency implied by Σ must also be derivable from Σ using only rules in $\mathfrak{R}$.

A sound and complete set of inference rules for the implication of FDs in the RDM was discovered by Armstrong in [15] (see also [16]). Note that the union $X \cup Y$ of two attribute sets X and Y is abbreviated by simply writing XY.

Theorem 1.1 (Armstrong, 1974). *The following inference rules*

$$\frac{}{X \to Y}\, Y \subseteq X, \qquad \frac{X \to Y}{X \to XY}, \qquad \frac{X \to Y, Y \to Z}{X \to Z}$$

(reflexivity axiom) *(extension rule)* *(transitivity rule)*

form a sound and complete set for the implication of FDs in the RDM. □

In the context of the RDM such inference rules are easily available, the reason being a well-founded algebraic, yet simple foundation. The set of all attribute sets for some relation schema forms a Boolean algebra with respect to set inclusion, set union, set intersection and set complement. This solid foundation is one of the key reasons for the success of the RDM.

Implication Problem. Consider the following example taken from [158, p.1093]. As was observed by Nicolas [212], FDs can be represented as sentences of first-order logic with equality. Let us demonstrate how this is done for a relation schema $R = \{A, B, C\}$ and the FD $\sigma = A \dashrightarrow B$. The vocabulary in the logic will be $\{R\}$, where the arity of R is 3 and A corresponds to the first argument of relation symbol R etc. The FD σ is expressed by the sentence

$$\varphi_\sigma = \forall x \forall y \forall z \forall y_1 \forall z_1 (R(x, y, z) \wedge R(x, y_1, z_1)) \Rightarrow (y = y_1) \quad .$$

It follows from the definition of FDs that the set of finite relational structures satisfying φ_σ and the set of relations satisfying σ are the same. This is also true for infinite relations. Similar arguments show that any set of FDs can be expressed by a set of sentences in first-order logic with equality. Note, however, that the database notation for FDs is often preferable, because it is less cumbersome to use and it intuitively captures the meaning of FDs.

The identification of a dependency σ with a sentence φ_σ is true for FDs and for many other dependencies. One of its consequences is the reduction of the (finite) implication problem to the (finite) unsatisfiability problem of first-order logic. For $\Sigma = \{\sigma_1, \ldots, \sigma_k\}$ let $\varphi_\Sigma = \varphi_{\sigma_1} \wedge \cdots \wedge \varphi_{\sigma_k}$. Then $\Sigma \models_{(f)} \sigma$ if and only if we have that the sentence $\varphi_\Sigma \wedge \neg\varphi_\sigma$ is (finitely) unsatisfiable.

Recall that a sentence is (finitely) unsatisfiable if it has no (finite) models. Also, unsatisfiability for first-order logic with equality is recursively enumerable (r.e.), by the Gödel Completeness Theorem [127], and finite unsatisfiability is co-r.e., by enumerating and testing all finite structures. From unsatisfiability we can infer finite unsatisfiability, from finite satisfiability we can infer satisfiability and from implication we can infer finite implication (but the converses do not always hold). From this discussion it follows that if a set of dependencies is identified with a set of sentences, for which satisfiability and finite satisfiability coincide, then dependency implication and finite implication coincide and are decidable.

Theorem 1.2. *For FDs implication and finite implication are the same and decidable.* □

The following argument for this theorem is somewhat of an overkill, since the theorem can be shown without an excursion into satisfiability. However, it is quite instructive since it may be used for many non-trivial extensions of FDs.

Let us look at the structure of the sentence $\varphi_{\sigma_1} \wedge \cdots \wedge \varphi_{\sigma_k} \wedge \neg \varphi_\sigma$ in the FD case. This can be written as a $\exists^* \forall^*$-sentence, that is, a sentence in prenex normal form whose quantifier prefix consists of a string of $\exists$s followed by a string of $\forall$s. This is known as a sentence of the initially extended Bernays-Schönfinkel class, for which satisfiability and finite satisfiability coincide [52].

The computational complexity of FD implication was considered by Beeri and Bernstein in [29, 40, 191], who demonstrated that implication can be performed optimally in linear time. This required a more detailed analysis than the decidability property, which follows from the equality of implication and finite implication.

Theorem 1.3. *(Finite) Implication of FDs is decidable in linear time.* □

Extensive use of this algorithm [29, 40] has been made in database schema design. Polynomial time algorithms for deciding the equivalence of two given sets of FDs [35] and deriving minimal covers for FDs [191] have been developed. A solution to these problems was a big step towards automated database schema design [35, 41] which some researchers see as the ultimate goal in dependency theory [30].

From the relationship of dependency (finite) implication and (finite) unsatisfiability it follows that $\Sigma \models_{(f)} \sigma$ if and only if $\varphi_\Sigma \models_{(f)} \varphi_\sigma$ where the second $\models_{(f)}$ is (finite) implication for sentences of first-order logic with equality. This is not the only relationship with mathematical logic, FDs have a number of elegant algebraic properties. See [158, pp. 1096-1098] for a detailed discussion and further references.

Boyce-Codd Normal Form. Relational database systems have evolved to the de-facto industry standard since their invention by E.F. Codd in 1970 [68]. This is a result of the simplicity and the sound theoretical basis of the RDM. One important issue associated with the use of relational databases is the correct structure or design of data to be used. Several criteria, referred to as *normal forms*, have been proposed as conditions for relation schemata that a database design should satisfy to ensure an absence of processing difficulties with the database. These normal forms give a database designer unambiguous guidelines in deciding which database schemata are good in the quest to avoid bad designs that have redundancy problems and update anomalies. Such normal forms have already been introduced in [70] by Codd. In general, they are dependent on the type of integrity constraints which apply to data items within the database. FDs cause difficulties such as redundancy in the representation of data and update anomalies. Codd proposed the Boyce-Codd normal form (BCNF) in [72] to overcome these difficulties. Recall that a dependency is called trivial if and only if it is satisfied by every relation over the schema it is defined on. An FD $X \rightarrow Y$ is trivial if and only if $Y \subseteq X$ holds. A subset $X \subseteq R$ is called a superkey for R with respect to a given set Σ of FDs on R if and only if $\Sigma \models X \rightarrow R$. The values on any superkey are therefore sufficient to uniquely identify any tuple in a relation.

A relation schema is in BCNF with respect to a given set Σ of FDs if and only if the left-hand side of each non-trivial FD implied by Σ is a superkey for the relation schema with respect to Σ. Codd conjectured that BCNF is an exact condition on a relation schema that avoids redundancies and update anomalies. Later on, after the notions of redundancy and update anomaly had been clarified and formalised (see for instance [280]), it was shown in [42, 105, 279, 280] that this is indeed the case.

Theorem 1.4. *Let R be a relation schema and Σ a set of FDs on R. R is in BCNF*

- *iff the left-hand side of every non-trivial FD in Σ is a superkey for R,*
- *iff every relation r over R that satisfies all key dependencies implied by Σ already satisfies all dependencies in Σ,*
- *iff R is non-redundant with respect to Σ,*
- *iff R does not have any insertion anomalies,*
- *iff R does not have any replacement anomalies of type 1,*
- *iff R does not have any replacement anomalies of type 2,*
- *only if R does not have any replacement anomalies of type 3.* □

The formal proofs of these statements make use of Armstrong's axioms. It follows that BCNF is a completely justified normal form in that sense. This is a big step towards automatically verifying whether a relation schema is well-designed. Since BCNF is a simple syntactic condition, the results above show that the relation schema is indeed well-designed in the sense that no redundancies and no update anomalies in terms of FDs can occur, and integrity checking reduces to the simple problem of verifying whether any two tuples deviate on a certain set of attributes.

A good database decomposition ought to have both the lossless join property and the dependency-preserving property. Informally, a decomposition $\{R_1, \ldots, R_n\}$ of a relation schema R is called *lossless* if every relation that satisfies all dependencies on R is the natural join of its projections on the subschemata R_i, i.e., $r = \pi_{R_1}(r) \bowtie \cdots \bowtie \pi_{R_n}(r)$ where $\pi_X(r) = \{t[X] \mid t \in r\}$ denotes the *projection* of $r \subseteq dom(R)$ to $X \subseteq R$, and $r_1 \bowtie r_2 = \{t \in dom(R_1 \cup R_2) \mid \exists t_1 \in r_1, t_2 \in r_2 . t[R_1] = t_1 \text{ and } t[R_2] = t_2\}$ denotes the *natural join* of $r_1 \subseteq dom(R_1)$ and $r_2 \subseteq dom(R_2)$. Note that $\pi_\emptyset(r)$ is the 0-ary relation $\{()\}$ which is also the left and right identity for the natural join operator. The process of decomposition does therefore neither add nor remove any information. A decomposition $\{R_1, \ldots, R_n\}$ of a relation schema R is dependency-preserving with respect to Σ if the closure of the union of all projected dependencies from Σ on the subschemata R_i coincides with the closure of Σ, and where closure refers to logical implication. That is, no semantic information is lost or added in terms of the dependencies that are given. For formal definitions see for example [181]. Furthermore and in terms of FDs, each relation schema in a well-designed database schema should be in BCNF.

Having answered the question what a good database schema constitutes, the next question is how to find such a good schema. The answer to this question is not as easy as the answer to the first question. In fact, it has been shown in [26, 29, 273] that there may be no decomposition of a relation schema into BCNF that is dependency-preserving. Thus,

obtaining a lossless join and dependency-preserving decomposition into BCNF is an unrealistic goal. However, it is always possible to achieve a lossless-join decomposition where each subschema is in BCNF. A semi-naive approach to this problem resulted first in an algorithm which runs in exponential time in the size of R and Σ, see for example [181]. The inefficiency results from the intractability of computing sets of projected dependencies. However, in [270] a polynomial time algorithm in the sizes of R and Σ that outputs a lossless join decomposition in BCNF with respect to Σ was given.

1.1.3 Multi-valued Dependencies

Multi-valued dependencies (MVDs) have been independently introduced by Fagin [103], Zaniolo [303] and Delobel [86]. They subsume the class of FDs as a special case. In particular, any relation r over R that satisfies the FD $X \to Y$ can be decomposed without loss of information, i.e., satisfies $r = \pi_{XY}(r) \bowtie \pi_{X(R-Y)}(r)$. Thus, satisfaction of an FD is a sufficient condition for a relation to be decomposed into its projections without losing information. The condition, however, is not necessary, i.e., there are relations r with $r = \pi_{XY}(r) \bowtie \pi_{X(R-Y)}(r)$, but $\not\models_r X \to Y$.

A multi-valued dependency, defined on some relation schema R, is an expression $X \twoheadrightarrow Y$ with $X, Y \subseteq R$. A relation r over R satisfies the MVD $X \twoheadrightarrow Y$ on R, denoted by $\models_r X \twoheadrightarrow Y$, if and only if for all $t_1, t_2 \in r$ with $t_1[X] = t_2[X]$ there is some $t \in r$ with $t[XY] = t_1[XY]$ and $t[X(R-Y)] = t_2[X(R-Y)]$. It turns out that the satisfaction of MVDs is an exact condition for a relation to be decomposable without loss of information. More precisely, $\models_r X \twoheadrightarrow Y$ if and only if $r = \pi_{XY}(r) \bowtie \pi_{X(R-Y)}(r)$. This gives an equivalent definition of MVDs and means that MVDs coincide with so-called binary join dependencies, see [103]. This fact is a key reason for the interest in the study of MVDs.

Axiomatisation. As it was the case with FDs, finite and unrestricted implication coincide for the class of MVDs. In fact, this is true for so-called full dependencies which subsumes the classes of FDs and MVDs [38]. In [32], Beeri, Fagin and Howard proposed sound and complete sets of inference rules for the implication of MVDs, and the implication of FDs and MVDs. The following sets are slightly different from the original proposal and are taken from [204] and [220].

Theorem 1.5. *The following set of inference rules*

$$\frac{}{X \twoheadrightarrow Y}\, Y \subseteq X \quad \textit{(reflexivity)} \qquad \frac{X \twoheadrightarrow Y}{XU \twoheadrightarrow YV}\, V \subseteq U \quad \textit{(augmentation)} \qquad \frac{X \twoheadrightarrow Y, Y \twoheadrightarrow Z}{X \twoheadrightarrow Z - Y} \quad \textit{(pseudo-transitivity)} \qquad \frac{X \twoheadrightarrow Y}{X \twoheadrightarrow R - Y} \quad \textit{(complementation)}$$

$$\frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow YZ} \quad \textit{(union)} \qquad \frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow Z - Y} \quad \textit{(difference)} \qquad \frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow Y \cap Z} \quad \textit{(intersection)}$$

is sound and complete for the implication of MVDs in the RDM. □

It should be stressed at this point that the inference rules for MVDs take advantage of the full expressive power of the Boolean algebra $(\mathcal{P}(R), \subseteq, \cup, \cap, (\cdot)^{\mathcal{C}}, \emptyset, R)$. The complementation rule makes use of the complement operation, and also union, intersection and difference are applied.

Theorem 1.6. *The following set of inference rules*

$$\frac{}{X \to Y}\ Y \subseteq X \quad \textit{(reflexivity)} \qquad \frac{X \to Y}{X \to XY} \quad \textit{(extension)} \qquad \frac{X \to Y, Y \to Z}{X \to Z} \quad \textit{(transitivity)}$$

$$\frac{X \twoheadrightarrow Y}{X \twoheadrightarrow R - Y} \quad \textit{(complementation)} \qquad \frac{X \twoheadrightarrow Y}{XU \twoheadrightarrow YV}\ V \subseteq U \quad \textit{(augmentation)} \qquad \frac{X \twoheadrightarrow Y, Y \to Z}{X \twoheadrightarrow Z - Y} \quad \textit{(pseudo-transitivity)}$$

$$\frac{X \to Y}{X \twoheadrightarrow Y} \quad \textit{(implication)} \qquad \frac{X \twoheadrightarrow Y, Y \to Z}{X \to Z - Y} \quad \textit{(mixed pseudo-transitivity)} \qquad \frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow YZ} \quad \textit{(union)}$$

$$\frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow Z - Y} \quad \textit{(difference)} \qquad \frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow Y \cap Z} \quad \textit{(intersection)}$$

is sound and complete set for the implication of FDs and MVDs in the RDM. □

Implication Problem. As was already pointed out before, FDs and MVDs are both subsets of so-called full dependencies, also called total dependencies in [38]. For full dependencies, however, implication and finite implication coincide and are decidable [5, 38]. Consider the following example from [212] where $R = \{A, B, C, D\}$ and $\sigma = A \twoheadrightarrow C$. The MVD σ is expressed by the first-order sentence

$$\forall w_1 \forall w_2 \forall w_3 \forall w_4 (\exists w_2' \exists w_4' R(w_1, w_2', w_3, w_4') \wedge \exists w_3' R(w_1, w_2, w_3', w_4)) \Rightarrow R(w_1, w_2, w_3, w_4) \quad .$$

Note that the existential quantifiers only appear in the antecedent of the implication. This means again that implication of MVDs is equivalent to the validity of sentences in the initially extended Bernays-Schönfinkel class (see above).

Theorem 1.7. *For MVDs implication and finite implication are the same and decidable.* □

Let Σ be a set of MVDs and $\sigma = X \twoheadrightarrow Y$. The implication problem $\Sigma \models \sigma$ was solved in $\mathcal{O}(n^4)$ in [27] where n denotes the total number of occurrences of attributes in Σ, and further analysed in [98, 118, 135, 152, 173, 223, 239, 277]. The computational behavior is due in large part to the algebraic properties of what is called in [32] a dependency basis. The best current bound for solving $\Sigma \models \sigma$ is $\mathcal{O}((1 + \min\{s, \log p\}) \cdot n)$ from [118] where s denotes the number of dependencies in Σ and p the number of sets in the dependency basis of X that have non-empty intersection with Y.

Theorem 1.8. *The implication problem for the class of MVDs and class of FDs and MVDs can be solved in almost linear time.* □

Minimality and Complementation Rule. Theorems 1.5 and 1.6 bring up questions about the independence of inference rules in the respective axiomatisations. The importance of FDs and MVDs for relational database design suggests that the behavior of these dependencies is worth investigating in full detail. It is therefore interesting to study which inference rules are implied by others and which are independent. Further motivation for this study is given by Mendelzon in [204] who determines all minimal complete subsets of inference rules from Theorem 1.5. Essentially, there is only one such minimal subset.

Theorem 1.9. *Reflexivity axiom, pseudo-transitivity rule and complementation rule from Theorem 1.5 form a minimal set of inference rules for the implication of MVDs in the RDM.*

Reflexivity axiom, extension rule, transitivity rule, pseudo-transitivity rule, complementation rule, implication rule and mixed pseudo-transitivity rule from Theorem 1.6 form a minimal set of inference rules for the implication of FDs and MVDs in the RDM. □

The complementation rule has a distinguished role among the rules of Theorem 1.9. There are a few papers [32, 46, 204] which point out the significance of this rule. The complementation rule is the only rule that does not have a direct analogue in the axiomatisation of functional dependencies, since it is the only rule that takes into account the context of the dependencies, that is, the underlying relation schema R, while all others apply independently of whatever relation schema the attributes are embedded in. It is therefore interesting to study whether one can obtain a (minimal) sound and complete set of inference rules that does not include the complementation rule. The R-axiom $\frac{}{\emptyset \twoheadrightarrow R}$, introduced in [46], is a very weak form of the complementation rule. The following result was shown in [46].

Theorem 1.10. *R-axiom, augmentation rule and pseudo-transitivity rule form a minimal set of inference rules for the implication of MVDs.* □

In this thesis an attempt is made to extend the majority of the previous results from relational databases to complex-value databases which support several type constructors.

1.1.4 Additional Remarks and Literature

As said before, it is not intended to give a complete summary of results on FDs and MVDs. Instead the previous section was more a reminder of those results which are to be extended to complex-value databases in this thesis. Therefore, there are many interesting topics in dependency theory which have not yet been mentioned at all.

First of all, the book [264] identifies more than 90 different classes of relational dependencies, and axiomatisations and remarks on the implication problem for many of these

classes are discussed. The papers [109, 158, 278] all provide excellent surveys on the motivations and history of research into relational dependencies.

Research on general integrity constraints considered from the perspective of first-order logic is presented in [119]. Other early work in this framework includes [212] which observes that FDs and MVDs have a natural representation in logic, and [213] which considers incremental maintenance of integrity constraints under updates to the underlying state.

FDs were introduced by Codd [71]. The axiomatisation is due to Armstrong [15, 16]. The implication problem was studied in [29, 191]. Several alternative formulations of FD implication, including formulation in terms of the propositional calculus perspective, are mentioned in [158]. They are due to [74, 75, 240].

Armstrong relations, i.e. relations which *precisely* satsify a given set of FDs and its implications, were introduced and studied in [31, 106, 107]. Interesting practical applications of Armstrong relations are proposed in [196, 251]. The idea is that, given a set Σ of FDs, an Armstrong relation for Σ with natural column entries is presented to a user, who can then determine whether Σ includes all of the desired restrictions.

The structure of Armstrong relations specified by a set of FDs is studied in [125, 147]. Interesting results on the combinatorial structures that arise from certain classes of dependencies can be found in [89, 90, 91, 92, 93, 94, 95, 96].

FDs on linear-ordered data domains are introduced and axiomatised in [211]. Order dependencies are studied in [124] which extend FDs to incorporate information involving partial order. A sound and complete set of inference rules is proposed and the implication problem of order dependencies is shown to be *coNP*-complete.

Multi-valued dependencies were discovered independently in [86, 103, 303]. They were generalised in [7, 212, 234]. The axiomatisation of FDs and MVDs is from [32]. The construction of Armstrong databases for FDs and MVDs can be found in [32, 288]. A probabilistic view of MVDs in terms of conditional independence is presented in [227, 228]. This provides an alternative motivation for the study of such dependencies.

The book [197] provides an in-depth coverage of relational schema design, including both the theoretical underpinnings and other, less formal factors that go into good design. Extensive treatments of the topic are also found in [81, 115, 181, 274, 292]. References [161, 162, 163] illustrate the many difficulties that arise in schema design, primarily with a host of intriguing examples that show how skilled the human mind is at organising diverse information and how woefully limiting data models are.

The area of normal forms and relational database design was studied intensively in the 1970s and early 1980s. Much more complete coverage of this topic than presented here may be found in [81, 181, 192, 274, 292]. Some of the most important papers in this area should be mentioned here. First normal form [68] is actually fundamental to the relational data model: a relation is in first normal form if each column contains only atomic values. This restriction shall be relaxed in this thesis. References [69, 70] raised the issue of update anomalies and initiated the search for normal forms that prevent them by introducing second and third normal forms. The mostly used definition for third normal form is from [304]. Boyce-Codd normal form was introduced in [72] to provide a normal form simpler than third normal form. Another improvement of third normal form is proposed in [186].

Fourth normal form was introduced in [103]. Even richer normal forms include project-join normal form [104] and domain-key normal form [105].

Relational database design is a good example of how theory can have an important and direct influence on practice. Recent work on providing semantic justification for various normal forms is of fundamental importance, since it can provide us with an explanation of what we actually achieve by the process of database design [280].

In addition to introducing second and third normal form, [70] initiated the search for normalisation algorithms by proposing the first decomposition algorithms. This spawned other research on decomposition [88, 221, 236] and synthesis [41, 43, 293]. The fact that these two criteria are not equivalent was stressed in [234] where it is proposed that both be attempted. Early surveys on these approaches to relational design include [30, 102, 235]. Algorithms for synthesis into third normal form include [41, 49], for decomposition into BCNF include [270], and for decomposition into fourth normal form include [103, 133]. Computational issues raised by decompositions are studied in [29, 112, 190, 270] and elsewhere. Reference [132] presents a good heuristic for finding covers of the projection of a set of FDs. The third normal form synthesis algorithm begins with a minimal cover of a set of FDs. Maier [191] shows that minimal covers can be found in polynomial time. Investigations on minimal covers of sets of MVDs can be found in [216].

The more formal study of decompositions and their properties was initiated by [234], which considered decompositions into two-element sets and proposed the notion of independent components; and [17], which studied decompositions that are lossless and dependency-preserving. This was extended independently to arbitrary decompositions over FDs by [37] and [193]. Lossless join was further investigated in [276].

The idea that not all integrity constraints specified in a schema should be considered for the design process was implicit in the works on semantic data models (e.g. [63, 184, 185]). It was stated explicitly in connection with relational schema design in [108, 247]. An extensive application of this approach towards schema design that incorporated both FDs and MVDs is in [34]. A very different form of decomposition, called horizontal decomposition, is introduced in [85]. This involves splitting a relation into pieces, each of which satisfies a given set of FDs.

Finally, the reader is referred to [47, 48] for a critique on the overall achievements and prospects of database design.

1.2 Challenges with Complex-value Databases

The second part of the introduction starts with a brief overview of data models that have been used for extending the RDM. For each class of these data models it is highlighted which types of complex objects are supported.

1.2.1 Extensions to the Relational Data Model

The relational data model has gained acceptance in the market place to such a degree that many database users expect their database systems to be relational by default. However,

users are demanding new facilities which are not directly supported by the model. Such facilities include support for deduction mechanisms, complex non-first normal form data, object-oriented features and production rules. The availability of database systems on a wide variety of computer platforms has meant that there is a growing demand for the use of databases in non-business applications, such as office automation, computer-aided design, multimedia, text retrieval, expert systems and scientific applications such as geographical and statistical analysis. This demand is a motivating factor for extending the relational model to provide such new facilities.

Semantic Data Models. Relational database management systems represent information in a simple record-based format. Semantic data models provide richer data structuring capabilities for database applications. Research in this area has articulated a number of constructs that provide mechanisms for representing structurally complex interrelations among data typically arising in commercial applications. Semantic models were developed to provide a higher level of abstraction for modelling data, allowing database designers to think of data in ways that correlate more directly to how data arises in the world [25, 64, 269]. The primary components of semantic models are the explicit representation of objects, attributes of and relationships among objects, *type constructors for building complex types*, ISA relationships, and derived schema components. Commonly, semantic data models support at least two constructed types: *aggregation* and *grouping*. An example for aggregation is for instance a type Address which is composed out of Street, City and Zip. It allows the user to focus on the abstract notion of Address while ignoring its component parts. Grouping is used to build sets of elements of an existing type, for example the atomic type *Language* can be used to construct the type *spoken Language* which represents the set of languages a particular person speaks. In general, three advantages of semantic data models over traditional, record-oriented systems are observed [149]:

- increased separation of conceptual and physical components,
- decreased semantic overloading of relationship types,
- availability of convenient abstraction mechanisms.

Historically, almost all semantic data models have focused almost exclusively on aggregation and grouping. Primary examples of such data models are the *Entity-Relationship model* [63], the *functional data model* [164, 250] and the *semantic data model* [136]. The Entity-Relationship model was the first semantic data model centered around relationships, not attributes. It views the world as consisting of entities and relationships among entities. The functional data model was the first of a number of semantic data models based on explicit representation of attributes. It is a simple, elegant model with easily understood visual representation. One of the major benefits of this model is the capacity to reference functions directly when manipulating properties of objects. Further examples of semantic data models include the semantic association model SAM^* [259], which is oriented in part to scientific and statistical applications and supports sets, vectors, ordered sets and matrices, *IFO* [4] and SHM^+ [54]. An excellent overview of semantic database modelling is [149]. Over the last years there have been several attempts to improve the

original Entity-Relationship model from [63]. A theoretically well-founded extension with proven applicability is the *Higher-Order Entity-Relationship model* [264, 265]. It introduces complex attributes, entity-, relationship- and cluster types of higher order which enhance a natural modelling process. Cluster types represent the *disjoint union* of relationship types.

The Nested Relational Data Model. If we interpret a domain as a data type, which can have potentially arbitrary complexity, then we should also allow relation-valued attributes as a special kind of data type. This argument from [82] becomes quite convincing realising that SQL supports domains such as character strings and dates, which can be viewed as aggregates of simpler data types, i.e., single characters and day, month and year, respectively.

The original proposal for generalising the relational model to allow entries in relations to be sets is often attributed to Makinouchi [195]. An extensive coverage of the field can be found in [148]. The nested relational data model is studied in [156, 179, 238, 267].

The nested relational data model distinguishes between atomic attributes such as PName, and relation-valued attributes such as (Hobby)* or (Child, Age)*. Values over PName are atomic, while (Hobby)*-values are relations over a relation schema with attribute Hobby and (Child, Age)*-values are relations over a relation schema with attributes Child and Age. Relation-valued attribute values can be empty, i.e., their value can be the empty set $\emptyset$. In the following nested relation r, Kane does not have any hobbies and Sebastian does not have any children.

PName	(Hobby)*	(Child	Age)*
Kane		Jill	8
		Jacob	10
		John	11
Sebastian	tennis		
	movies		
Jeff	photography	Maria	4
	reading		

Besides the common relational algebra operators from the relational data model, the nested relational algebra extends the relational algebra with NEST and UNNEST which restructure relations, and *empty* which creates a nested relation with a single relation-valued attribute which is empty. The NEST operator transforms a nested relation into a "more deeply" nested relation while the UNNEST operator transforms a nested relation into a "flatter" nested relation. As an example, the following table shows the unnesting of the previous nested relation r with respect to (Hobby)*.

PName	Hobby	(Child	Age)*
Sebastian	tennis		
Sebastian	movies		
Jeff	photography	Maria	4
Jeff	reading	Maria	4

Due to a result in [222], the power of the nested relational algebra is equivalent to that of the flat relational algebra. So the power of the nested relational algebra lies in its ability to represent and manipulate nonflat data rather than in its ability to pose additional queries that cannot be expressed in the flat relational algebra.

Nested relations can be viewed in the wider context of complex object types. It is assumed that a collection of atomic object types are available, where the values of each such atomic type are taken from an atomic domain. An object type can now be defined as a tree whose leaves represent atomic object types, and whose internal nodes represent the application of either the *tuple construct*, which aggregates its children object types into a tuple, or the *set construct*, which groups its single child object type into a set. Thus a nested relation type is a special case of a complex object type, where the root of the tree represents a tuple construct, each child node of a node representing a tuple construct either represents a set construct or an atomic object type, and the single child node of a node representing a set construct represents a tuple construct.

Normal form proposals within the Nested Relational Data Model have been based on the appropriate nesting of a flat relation schema with respect to a given set of functional and multi-valued dependencies defined on the flat schema. Examples for such normal form proposals include [187, 207, 215, 217, 237, 238, 262]. A comparison of the various normal forms proposed in [207, 215, 217, 237, 238] can be found in [206]. The work in [284] characterises data equivalence of nested relation schemata in partitioned normal form. One major result is that two schemata in partitioned normal form are data equivalent if and only if the sets of multi-valued dependencies induced by the corresponding schema trees are equivalent. A further reference for many topics on nested relations in databases is [3].

Object-Oriented and Object-Relational Data Models. The main characteristics of object-oriented database systems are described in the database manifesto [20]. It is proposed that the first mandatory feature of any object-oriented database shall be the support of complex objects with orthogonal behavior: "Thou shalt support complex objects". Complex objects mentioned are records, sets, bags (multisets) and lists. As a minimum set of supported complex objects they consider records, sets and lists. The work in [243] proposes to classify data models according to the underlying type system which may include records, sets, lists, bags, unions, and recursion. It is in fact this approach that will be followed in this thesis.

Collections of papers on object-oriented databases can be found in [22, 166, 306]. An influential discussion of some foundational issues around the object-oriented database paradigm is [28]. An important survey of subtyping and inheritance from the perspective of programming languages, including the notion of domain-inclusion semantics, is [62]. Further examples of object-oriented data models are [114, 121].

Object-oriented databases are, of course, closely related to object-oriented programming languages. The first of these is Smalltalk [128], and C++ [257] is fast becoming the most widely used object-oriented programming language. Several commercial object-oriented database systems are essentially persistent versions of C++. Several object-oriented ex-

tensions of Lisp have been proposed; the article [50] introduces a rich extension called CommonLoops and surveys several others.

There have been a number of approaches to provide a formal foundation [6, 28, 150, 165, 243] for object-oriented databases. We can also cite as precursors attempts to formalise semantic data models [4] and object-based models [151, 172]. Recent graph-oriented models, although they do not stress object orientation, are similar in spirit (e.g. [134]).

The paper [262] proposes an object normal form, but is mainly dealing with semantic issues as opposed to removing redundancy. Further papers that consider path functional dependencies in pointer-based data models are [51, 153].

The book [256] specifically explores the marriage of relational databases with object technology resulting in object-relational database systems. A further example for such a proposal is [244]. An object-relational database can be defined as one which supports SQL3 [83, 203]. Four fundamental characteristics of object-relational DBMSs are identified in [256]: (i) the ability to add to the database system user-defined data types and functions operating on these types (this can be viewed as an abstract data type facility), (ii) the ability to construct complex object types via general purpose type constructors, (iii) the ability to define supertypes and subtypes together with the support of inheritance from supertype to subtype, and (iv) support for active database rules or alternative triggers.

Hypertext Datamodels. An emerging field in the broad area of information systems is that of hypertext (or more generally hypermedia), whose aim is to provide database support for networks of "electronic documents" which are logically linked together. Hypertext is concerned with authoring, managing, designing and navigating through the electronic documents of such networks. The vision of virtual electronic libraries is becoming a reality and hence there is a strong need for a formal data model of hypertext. Although it would be naive to consider a data model of such an electronic library to be an extension of the relational model, relational database theory can provide inspiration for the development of such a data model. A hypertext database can be viewed as an instance of a semistructured database in the sense that such a database does not come with a separate schema, since it does not have a regular structure. Although the digraph representing a hypertext database is unstructured, individual pages may have some structure attached to them. For instance, pages which are HTML documents have some structure attached to them in the form of informational tags, but these are insufficient for the purpose of constructing a relation schema over the document space. Semistructured data is often self-describing in the sense that its internal structure, when it exists, can be inferred from the data itself.

The eXtensible Markup Language (XML,[53]) has emerged as the standard for information exchange between Web applications. It offers a convenient syntax for representing data from heterogeneous sources, but provides little semantic information. To specify the semantics of XML data, a variety of approaches have been proposed: type systems [36, 67, 84, 177, 268], description logics [61], meta-data descriptions [200] etc. As some of these proposals [84, 177, 268] point out, integrity constraints are important for semantic specifications of XML data. In addition, they are useful for query optimisation [97, 116], update anomaly prevention [5], and for information preservation in data integration [2, 67].

The emergence of XML has recently led to a revival of dependency theory and resulted in many research papers on that topic.

The work in [111] proposes an extension of XML DTDs (document type definitions) that specifies both syntactic structure and integrity constraints for XML data. The authors investigate keys, foreign keys, inverse constraints and ID constraints for capturing the semantics of object identities in the framework of XML. Some complexity and axiomatisation results for the (finite) implication problems for these constraints are established. In [110] the consistency problem whether there is an XML document that conforms to a DTD and satisfies a given set of keys and foreign keys is studied. In general, this problem turns out to be undecidable, but it is proven to be *NP*-complete for unary keys and foreign keys. Arenas and Libkin define functional dependencies in terms of paths in DTDs and propose a normal form for XML documents in [14]. They show, in particular, that the implication problem for their class of functional dependencies is not finitely axiomatisable. For so-called simple DTDs, however, the implication problem is shown to be solvable in quadratic time. For relational DTDs the implication problem is *NP*-complete, for disjunctive DTDs it is shown to be *coNP*-complete. The authors continue their work on normalisation in the context of XML in [13]. Here, they use techniques of information theory to define a measure of information content of elements in a database with respect to a set of constraints. The papers [11, 12] demonstrate the costly effect of slightly changing the semantics of keys, foreign keys and uniqueness constraints in XML schema design. In particular, known hardness results on consistency checking extend to XML schemata, but tractability results do not. It is shown that even without foreign keys, and with very simple DTD features, checking consistency of XML schemata is intractable. The papers [56, 57, 58, 59] investigate integrity constraints in XML and semistructured data as well, and particularly focus on different proposals for XML keys and foreign keys, path and inclusion constraints. The work in [285] extends the definition of functional dependencies in incomplete relations to XML documents. The authors argue that their approach overcomes difficulties with the approach in [14]. In particular, they give a precise syntactic definition of strong satisfaction of an XFD in XML documents and do not require the existence of a DTD as opposed to [14]. In [286], the same authors consider multi-valued dependencies in the context of XML. They justify their definition of an XMVD by showing that for a very general class of mappings from relations to XML, a relation satisfies an MVD if and only if the corresponding XML document satisfies the corresponding XMVD. In [287], a normal form based on XFDs and XMVDs is proposed and formally justified by showing the absence of redundancy in those XML documents which are in the normal form proposed.

Complex Objects in other Fields of Application. Complex values have been subject to studies in the deductive and temporal database community for some time. An overview of integrity constraints in *deductive databases* is provided in [19]. Unique key constraints for deductive databases are proposed in [305]. Dependency theory has also been extended to deal with future and past, i.e., to *temporal databases* in [123, 169, 296]. One approach in temporal databases is to express integrity constraints as essentially arbitrary sentences in some temporal logic [10, 65, 66, 188]. An alternative approach examines restricted classes

of temporal integrity constraints, which may be called temporal dependencies [294]. This approach is also used in [297] to explore temporal functional dependencies in the context of complex objects, i.e. temporal databases that include object identity.

The papers [208, 233] emphasise the importance of including a list constructor in deductive database models. Trees, lists, finite sets and multisets are considered in [80, 291] where the complexity of checking whether a query defines a nonempty set is studied for nonrecursive queries. Logic programming with sets is considered in [170, 171].

The need for lists arises from applications that store ordered relations, time-series data, meteorological and astronomical data streams, runs of experimental data, multidimensional arrays, textual information, voices, sound, images, video, etc. Recently, bioinformatics has become a very important field of research. Of course, lists and sets occur naturally in *genomic sequence databases* [55, 183, 248].

Set-valued attributes appear in several application domains, e.g. in retail databases they can represent the set of different products purchased by a customer, in multimedia databases they can be used to represent the set of objects contained in an image, in web server logs they correspond to web pages and links visited by a user. Finally, in data mining applications set-valued attributes are commonly used to store time-series and market basket data.

The multiset is a notion that has appeared again and again in many areas of mathematics and computer science, sometimes called a bag. As a data structure this notion stands "in-between" strings/lists, where a linear ordering of symbols/items is present, and sets, where no ordering and no multiplicity is considered; in a multiset only the multiplicity of elements matters, not their ordering. Actually, in between lists and multisets we also have pomsets, partially ordered multisets.

Confining ourselves to computer science, we may mention many areas where multisets are used: formal power series, Petri nets, databases, logics, formal language theory (in relation with Parikh mapping, commutative grammars, etc.), concurrency and so on. In the last few years, the notion has occurred in a rather natural way in the molecular computing area. An aqueous solution of chemical compounds, swimming together in a given space, without any given spatial relation between individual elements, is just a multiset. Actually, chemical metaphor was used several years before the occurrence of what is now called molecular computing, as the basic ingredient of the Gamma language [21] and the Chemical Abstract Machine [44]. Then, multisets were used in relation with DNA computing [8, 159, 224], especially in the context of computing by splicing (H systems): taking into account the number of DNA molecules proved to be a very powerful feature of H systems, leading to computational completeness.

In the prolongation of the chemical metaphor, the membrane computing area has recently emerged [99, 225, 226], as an abstraction of the living cell structure and biochemistry: in the compartments defined by a membrane structure, one processes multisets of chemical compounds, denoted by symbols or by strings over a given alphabet.

For a recent survey on the use of multisets in various areas of logic and computer science see [60], in which [174] specifically focuses on database systems. Multisets also appear in logic [23], linguistics [73, 131], artificial life [260, 261], etc. Several papers have considered

fuzzy variants of multisets ([182, 299]) while [126] and [24] deal with pomsets.

1.2.2 Real-World Examples for Complex Constraints

This section shall be used to give a few typical "real-world" examples in which constraints among complex data types appear. The descriptions of the various applications as well as the description of the constraints are all informal in this section. Later on, the examples will be formalised and used as illustrations for several notions and algorithms.

Bioinformatics. According to [209], GenBank is the NCBI genetic sequence database, an annotated collection of all publicly available DNA sequences. There are approximately 28,507,990,166 bases in 22,318,883 sequence records as of January 2003. GenBank is part of the International Nucleotide Sequence Database Collaboration, which comprises the DNA DataBank of Japan (DDBJ), the European Molecular Biology Laboratory (EMBL), and GenBank at NCBI. These three organisations exchange data on a daily basis.

A detailed description of every field in a GenBank record can be found in [210]. For the purposes of this thesis, a simplified description of such a GenBank record shall suffice.

The record starts with a nucleotide sequence, i.e., a sequence of A, C, G, T representing the four nucleotide bases adenine, cytosine, guanine and thymine, respectively. The total number of each of these bases is recorded as well. Regions of biological interest within the original nucleotide sequence are known as genes. The information of such a gene consists of the start and end position of the subsequence within the original sequence, the subsequence itself and a translation of this subsequence into a sequence of amino acids. An amino acid is represented by a letter and is encoded by a triplet of nucleotide bases.

The following nucleotide sequence belongs to the organism *Saccharomyces cerevisiae* (budding yeast) and is taken from [210].

```
   1 gatcctccat atacaacggt atctccacct caggtttaga tctcaacaac ggaaccattg
  61 ccgacatgag acagttaggt atcgtcgaga gttacaagct aaaacgagca gtagtcagct
 121 ctgcatctga agccgctgaa gttctactaa gggtggataa catcatccgt gcaagaccaa
 181 gaaccgccaa tagacaacat atgtaacata tttaggatat acctcgaaaa taataaaccg
 241 ccacactgtc attattataa ttagaaacag aacgcaaaaa ttatccacta tataattcaa
 301 agacgcgaaa aaaaaagaac aacgcgtcat agaacttttg gcaattcgcg tcacaaataa
 361 attttggcaa cttatgtttc ctcttcgagc agtactcgag ccctgtctca agaatgtaat
 421 aatacccatc gtaggtatgg ttaaagatag catctccaca acctcaaagc tccttgccga
 481 gagtcgccct cctttgtcga gtaattttca cttttcatat gagaacttat tttcttattc
 541 tttactctca catcctgtag tgattgacac tgcaacagcc accatcacta gaagaacaga
 601 acaattactt aatagaaaaa ttatatcttc ctcgaaacga tttcctgctt ccaacatcta
 661 cgtatatcaa gaagcattca cttaccAtga cacagcttca gatttcatta ttgctgacag
 721 ctactatatc actactccat ctagtagtgg ccacgcccta tgaggcatat cctatcggaa
 781 aacaataccc cccagtggca agagtcaatg aatcgtttac atttcaaatt tccaatgata
 841 cctataaatc gtctgtagac aagacagctc aaataacata caattgcttc gacttaccga
 901 gctggctttc gtttgactct agttctagaa cgttctcagg tgaaccttct tctgacttac
 961 tatctgatgc gaacaccacg ttgtatttca atgtaatact cgagggtacg gactctgccg
1021 acagcacgtc tttgaacaat acataccaat ttgttgttac aaaccgtcca tccatctcgc
```

1081 tatcgtcaga tttcaatcta ttggcgttgt taaaaaacta tggttatact aacggcaaaa
1141 acgctctgaa actagatcct aatgaagtct tcaacgtgac ttttgaccgt tcaatgttca
1201 ctaacgaaga atccattgtg tcgtattacg gacgttctca gttgtataat gcgccgttac
1261 ccaattggct gttcttcgat tctggcgagt tgaagtttac tgggacggca ccggtgataa
1321 actcggcgat tgctccagaa acaagctaca gttttgtcat catcgctaca gacattgaag
1381 gattttctgc cgttgaggta gaattcgaat tagtcatcgg ggctcaccag ttaactacct
1441 ctattcaaaa tagtttgata atcaacgtta ctgacacagg taacgtttca tatgacttac
1501 ctctaaacta tgtttatctc gatgacgatc ctatttcttc tgataaattg ggttctataa
1561 acttattgga tgctccagac tgggtggcat tagataatgc taccatttcc gggtctgtcc
1621 cagatgaatt actcggtaag aactccaatc ctgccaattt ttctgtgtcc atttatgata
1681 cttatggtga tgtgatttat ttcaacttcg aagttgtctc cacaacggat ttgtttgcca
1741 ttagttctct tcccaatatt aacgctacaa ggggtgaatg gttctcctac tatttttgc
1801 cttctcagtt tacagactac gtgaatacaa acgtttcatt agagtttact aattcaagcc
1861 aagaccatga ctgggtgaaa ttccaatcat ctaatttaac attagctgga gaagtgccca
1921 agaatttcga caagctttca ttaggtttga aagcgaacca aggttcacaa tctcaagagc
1981 tatattttaa catcattggc atggattcaa agataactca ctcaaaccac agtgcgaatg
2041 caacgtccac aagaagttct caccactcca cctcaacaag ttcttacaca tcttctactt
2101 acactgcaaa aatttcttct acctccgctg ctgctacttc ttctgctcca gcagcgctgc
2161 cagcagccaa taaaacttca tctcacaata aaaaagcagt agcaattgcg tgcggtgttg
2221 ctatcccatt aggcgttatc ctagtagctc tcatttgctt cctaatattc tggagacgca
2281 gaagggaaaa tccagacgat gaaaacttac cgcatgctat tagtggacct gatttgaata
2341 atcctgcaaa taaaccaaat caagaaaacg ctacaccttt gaacaacccc tttgatgatg
2401 atgcttcctc gtacgatgat acttcaatag caagaagatt ggctgctttg aacactttga
2461 aattggataa ccactctgcc actgaatctg atatttccag cgtggatgaa aagagagatt
2521 ctctatcagg tatgaataca tacaatgatc agttccaatc ccaaagtaaa gaagaattat
2581 tagcaaaacc cccagtacag cctccagaga gcccgttctt tgacccacag aataggtctt
2641 cttctgtgta tatggatagt gaaccagcag taaataaatc ctggcgatat actggcaacc
2701 tgtcaccagt ctctgatatt gtcagagaca gttacggatc acaaaaaact gttgatacag
2761 aaaaactttt cgatttagaa gcaccagaga aggaaaaacg tacgtcaagg gatgtcacta
2821 tgtcttcact ggacccttgg aacagcaata ttagcccttc tcccgtaaga aaatcagtaa
2881 caccatcacc atataacgta acgaagcatc gtaaccgcca cttacaaaat attcaagact
2941 ctcaaagcgg taaaaacgga atcactccca caacaatgtc aacttcatct tctgacgatt
3001 ttgttccggt taaagatggt gaaaattttt gctgggtcca tagcatggaa ccagacagaa
3061 gaccaagtaa gaaaaggtta gtagattttt caaataagag taatgtcaat gttggtcaag
3121 ttaaggacat tcacggacgc atcccagaaa tgctgtg**A**tt atacgcaacg atattttgct
3181 taattttatt ttcctgtttt atttttatt agtggtttac agataccta tattttattt
3241 agtttttata cttagagaca tttaatttta attccattct tcaaatttca tttttgcact
3301 taaaacaaag atccaaaaat gctctcgccc tcttcatatt gagaatacac tccattcaaa
3361 attttgtcgt caccgctgat taatttttca ctaaactgat gaataatcaa aggccccacg
3421 tcagaaccga ctaaagaagt gagttttatt ttaggaggtt gaaaaccatt attgtctggt
3481 aaattttcat cttcttgaca tttaacccag tttgaatccc tttcaatttc tgctttttcc
3541 tccaaactat cgaccctcct gtttctgtcc aacttatgtc ctagttccaa ttcgatcgca
3601 ttaataactg cttcaaatgt tattgtgtca tcgttgactt taggtaattt ctccaaatgc
3661 ataatcaaac tatttaagga agatcggaat tcgtcgaaca cttcagtttc cgtaatgatc
3721 tgatcgtctt tatccacatg ttgtaattca ctaaaatcta aaacgtattt ttcaatgcat
3781 aaatcgttct ttttattaat aatgcagatg gaaaatctgt aaacgtgcgt taatttagaa
3841 agaacatcca gtataagttc ttctatatag tcaattaaag caggatgcct attaatggga
3901 acgaactgcg gcaagttgaa tgactggtaa gtagtgtagt cgaatgactg aggtgggtat
3961 acatttctat aaaataaaat caaattaatg tagcatttta agtataccct cagccacttc
4021 tctacccatc tattcataaa gctgacgcaa cgattactat ttttttttc ttcttggatc
4081 tcagtcgtcg caaaaacgta taccttcttt ttccgacctt ttttttagct ttctggaaaa

4141 gtttatatta gttaaacagg gtctagtctt agtgtgaaag ctagtggttt cgattgactg
4201 atattaagaa agtggaaatt aaattagtag tgtagacgta tatgcatatg tatttctcgc
4261 ctgtttatgt ttctacgtac ttttgattta tagcaagggg aaaagaaata catactattt
4321 tttggtaaag gtgaaagcat aatgtaaaag ctagaataaa atggacgaaa taaagagagg
4381 cttagttcat ctttttcca aaaagcaccc aatgataata actaaaatga aaaggatttg
4441 ccatctgtca gcaacatcag ttgtgtgagc aataataaaa tcatcacctc cgttgccttt
4501 agcgcgtttg tcgtttgtat cttccgtaat tttagtctta tcaatgggaa tcataaattt
4561 tccaatgaat tagcaatttc gtccaattct ttttgagctt cttcatattt gctttggaat
4621 tcttcgcact tcttttccca ttcatctctt tcttcttcca aagcaacgat ccttctaccc
4681 atttgctcag agttcaaatc ggcctctttc agtttatcca ttgcttcctt cagtttggct
4741 tcactgtctt ctagctgttg ttctagatcc tggtttttct tggtgtagtt ctcattatta
4801 gatctcaagt tattggagtc ttcagccaat tgctttgtat cagacaattg actctctaac
4861 ttctccactt cactgtcgag ttgctcgttt ttagcggaca aagatttaat ctcgttttct
4921 ttttcagtgt tagattgctc taattctttg agctgttctc tcagctcctc atatttttct
4981 tgccatgact cagattctaa ttttaagcta ttcaatttct ctttgatc

All over, there are 1510 As, 1074 Cs, 835 Gs and 1609 Ts. The subsequence that starts with the first underlined bold upper-case *A* and finishes with the second underlined bold upper-case *A* encodes the so-called *plasma membrane glycoprotein*. It starts at position 687 and finishes at position 3158 within the original sequence. The subsequence contains 2472 bases. The encoded protein consists therefore of 824 amino acids:

MTQLQISLLLTATISLLHLVVATPYEAYPIGKQYPPVARVNESFTFQISNDTYKSSVD
KTAQITYNCFDLPSWLSFDSSSRTFSGEPSSDLLSDANTTLYFNVILEGTDSADSTSL
NNTYQFVVTNRPSISLSSDFNLLALLKNYGYTNGKNALKLDPNEVFNVTFDRSMFTNE
ESIVSYYGRSQLYNAPLPNWLFFDSGELKFTGTAPVINSAIAPETSYSFVIIATDIEG
FSAVEVEFELVIGAHQLTTSIQNSLIINVTDTGNVSYDLPLNYVYLDDDPISSDKLGS
INLLDAPDWVALDNATISGSVPDELLGKNSNPANFSVSIYDTYGDVIYFNFEVVSTTD
LFAISSLPNINATRGEWFSYYFLPSQFTDYVNTNVSLEFTNSSQDHDWVKFQSSNLTL
AGEVPKNFDKLSLGLKANQGSQSQELYFNIIGMDSKITHSNHSANATSTRSSHHSTST
SSYTSSTYTAKISSTSAAATSSAPAALPAANKTSSHNKKAVAIACGVAIPLGVILVAL
ICFLIFWRRRRENPDDENLPHAISGPDLNNPANKPNQENATPLNNPFDDDASSYDDTS
IARRLAALNTLKLDNHSATESDISSVDEKRDSLSGMNTYNDQFQSQSKEELLAKPPVQ
PPESPFFDPQNRSSSVYMDSEPAVNKSWRYTGNLSPVSDIVRDSYGSQKTVDTEKLFD
LEAPEKEKRTSRDVTMSSLDPWNSNISPSPVRKSVTPSPYNVTKHRNRHLQNIQDSQS
GKNGITPTTMSTSSSDDFVPVKDGENFCWVHSMEPDRRPSKKRLVDFSNKSNVNVGQV
KDIHGRIPEML.

We list a number of constraints that a database designer may choose to specify for this (simplified) application:

1. The original nucleotide sequence determines the total number of each of the four bases.
2. All four total numbers of bases determine the length of the original nucleotide sequence.
3. The length of the original nucleotide sequence together with the total numbers of any three bases determines the total number of the remaining base.
4. The original nucleotide sequence together with start and end position of the subsequence determines the subsequence itself.
5. The nucleotide subsequence determines the sequence of amino acids.
6. The length of the nucleotide subsequence determines the length of the amino acid sequence and vice versa.

7. The start position of the subsequence together with the length of the subsequence determine the end position of the subsequence.
8. The end position of the subsequence together with the length of the subsequence determine the start position of the subsequence.
9. Start and end position of the subsequence together determine the length of the subsequence.

Note that the length of the amino acid sequence is simply one third of the length of the nucleotide subsequence. The database schema for this example will be formalised in Example 2.2 on page 48, the constraints in Example 3.3 on page 53.

In order to explain a different kind of constraint we look at a further example of nucleotide sequences. Suppose one would like to compare two nucleotide sequences each of which has a certain characteristic, say a fixed starting sequence and a certain number of occurrences of a particular base. Apart from the two starting sequences and from the two numbers of occurrences of a certain base, the database stores a list of pairs of nucleotide bases. First and second component in the kth pair of that list are the kth element of the first and second nucleotide sequence, respectively. Such a database might be helpful to find good alignments of nucleotide sequences.

As an example we compare the candidates AACGA and AATGA for a sequence with starting sequence AA and three occurrences of an A with the candidates AATCT and AATTC for a sequence with starting sequence AAT and two occurrences of a T. Moreover, we compare the candidates CGGC and CGCG for a sequence with starting sequence CG and two occurrences of a G with the candidates CATT, CTAT and CCAC for a sequence with starting sequence C and one occurrence of an A. The first comparison results in the first four tuples of the following database, the second comparison in the last six tuples:

([A,A],[A,A,T],(3,A),(2,T),[(A,A),(A,A),(C,T),(G,C),(A,T)]),
([A,A],[A,A,T],(3,A),(2,T),[(A,A),(A,A),(C,T),(G,T),(A,C)]),
([A,A],[A,A,T],(3,A),(2,T),[(A,A),(A,A),(T,T),(G,C),(A,T)]),
([A,A],[A,A,T],(3,A),(2,T),[(A,A),(A,A),(T,T),(G,T),(A,C)]),
([C,G],[C],(2,G),(1,A),[(C,C),(G,A),(G,T),(C,T)]),
([C,G],[C],(2,G),(1,A),[(C,C),(G,T),(G,A),(C,T)]),
([C,G],[C],(2,G),(1,A),[(C,C),(G,C),(G,A),(C,C)]),
([C,G],[C],(2,G),(1,A),[(C,C),(G,A),(C,T),(G,T)]),
([C,G],[C],(2,G),(1,A),[(C,C),(G,T),(C,A),(G,T)]),
([C,G],[C],(2,G),(1,A),[(C,C),(G,C),(C,A),(G,C)]).

A constraint that a database designer of this database may want to specify is the following. Both starting sequences together with both numbers of occurrences of a nucleotide base determine the set of lists of the first component, independently from the set of lists of the second component. That is, if there are two elements of the database which have the same starting sequences and the same number of occurrences of the two particular nucleotide bases, then there is another element in that database which is coincident with the first element on the first nucleotide sequence (and the starting sequences and number

of occurrences of nucleotides) and coincident with the second element on the second nucleotide sequence. Take for instance the second and third element of the snapshot above. They have the same starting sequences and the same number of occurrences of the two nucleotide bases. The first element of the database is the witness for the satisfaction of the constraint above: its list of first components is the list of first components of the second element, and its list of second components is the list of second components of the third element. The database schema for this example will be formalised in Example 2.3 on page 48, the constraints in Example 4.2 on page 95.

Image Processing. Halftoning is one of the oldest applications of image processing, since it is essential for the printing process. With the evolution of computers and their gradual introduction to typesetting, printing, and publishing, the field of halftoning that was previously limited to the so-called halftoning screen [271] evolved into its successor: digital halftoning. Today, digital halftoning plays a keyrole in almost every discipline that involves printing and displaying. All newspapers, magazines, and books are printed with digital halftoning. It is used in image display devices capable of reproducing two-level outputs such as scientific workstations, laser printers, and digital typesetters. It is also important for facsimile transmission and compression.

There are many methods to perform digital halftoning. They can be grouped in three major categories:

1. dithering [205, 245, 271, 272],
2. error diffusion [117, 155, 160, 254, 258], and
3. direct binarisation [219, 246].

Error diffusion revolutionised the digital halftoning field and has given the spark for the development of a great number of new methods. Error diffusion is based on the simple principle that once a pixel has been quantised, thus introducing some error, this error should affect the quantisation of the pixels in the region of its neighbors.

Digital halftoning is an application of the matrix rounding problem [18]. The problem is to convert a continuous-tone image into a binary one that looks similar. The input matrix A represents a digital (gray) image, where a_{ij} represents the brightness level of the (i,j)-pixel in the $N \times N$ pixel grid. Typically, N is between 256 and 4096, and a_{ij} is an integral multiple of $\frac{1}{256}$: this means that we use 256 brightness levels. If we want to send an image using fax or print it out by a dot or ink-jet printer, brightness levels available are limited. Instead, we replace the input matrix A by an integral matrix B so that each pixel uses only two brightness levels, i.e., black or white. Here, it is important that B looks similar to A; in other words, B should be a good approximation of A. A typical family of regions is given by the set of all submatrices of a certain form. In this sense, a good approximation of input matrix A is a $\{0,1\}$-matrix B that minimises the distance $\left| \sum_{(i,j)\in R} a_{i,j} - \sum_{(i,j)\in R} b_{i,j} \right|$ for all $R \in \mathcal{R}$. Herein, $\mathcal{R}$ denotes the set of regions, for instance the set of all pairs of indices that denote $2 \times 1, 1 \times 2$ and 2×2 submatrices. A region has therefore one of the following

forms

and can be represented as a list of either two or four elements. Of course, the regions can have all different kinds of shapes in practice. In order to make the example more illustrative, we assume from now on that the input matrix has entries in $\{0, \frac{1}{2}, 1\}$, i.e., uses three brightness levels. Input regions can be best approximated by a number of different output regions. All inputs with overall brightness $\frac{1}{2}$ and length two, i.e. $[0, \frac{1}{2}]$ or $[\frac{1}{2}, 0]$, could be mapped to any of [0,1], [1,0] or [0,0], each of which has distance $\frac{1}{2}$. In this sense, the set of input regions ($\{[0, \frac{1}{2}], [\frac{1}{2}, 0]\}$) is determined by the overall brightness of the input region ($\frac{1}{2}$) and the length of the input region (2), independently of the set of output regions ($\{[0, 1], [1, 0], [0, 0]\}$). This is true for any inputs and outputs, e.g., all inputs with overall brightness $\frac{3}{2}$ and length four such as $[0, 0, 1, \frac{1}{2}]$ can be mapped to any of [0,0,0,1], [0,0,1,0], [0,1,0,0], [1,0,0,0], [0,0,1,1], [0,1,0,1], [1,0,0,1], [0,1,1,0], [1,0,1,0], [1,1,0,0].

Consider a database which stores input and output regions as lists together with the overall brightness of the input region. It is then desirable to find a {0,1}-matrix B that has for every of the possible regions of input matrix A a corresponding output region that is stored in the database. The input matrix $A = \begin{pmatrix} 0 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$ has for instance the approximation $B = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$. Every 2×2 matrix has five input regions and the mappings that produce B from A are as follows: $[0, 0] \mapsto [0, 0]$, $[\frac{1}{2}, \frac{1}{2}] \mapsto [0, 1]$, $[0, \frac{1}{2}] \mapsto [0, 0]$ (left column), $[0, \frac{1}{2}] \mapsto [0, 1]$ (right column) and $[0, 0, \frac{1}{2}, \frac{1}{2}] \mapsto [0, 0, 0, 1]$.

The matrix $\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, however, is not an approximation of A as the region $[\frac{1}{2}, \frac{1}{2}]$ should not be mapped to $[0, 0]$.

Constraints that a database designer may choose to specify for this application are the following:

1. The length of the input region determines the length of the output region, and vice versa.
2. The overall brightness and length of the input region together determine the set of all input regions independently from the set of the output regions.

In practice, the amount of information that needs to be stored depends on the set of regions considered and the brightness levels available. The database schema for this example will be formalised in Example 2.4 on page 48, the constraints in Example 4.3 on page 95.

Retailers. Consider a retailer which keeps track of its sales on a daily basis. For each day the sequence of incoming orders is stored. Every order consists of information about the customer who places the order, the collection of articles ordered, and the total value of the order. A customer is described by its name, address and payment details. Every article in

that order has a title, a description and a price. Besides the sequence of incoming orders, the retailer stores the different products which were sold that day. In fact, not only the title of the sold item is stored but also the name of the customer who bought it. Moreover, the company keeps information about the total value of sales, the total number of orders, the total number of products sold and the total number of shippings for each day. A few reasonable constraints that a database designer may specify for this application are the following.

1. As the information is stored on a daily basis, the day determines the rest of the information.
2. The list of multisets of article titles determines the set of those items that were sold.
3. The list of multisets of individual article prices determines the list of total values of each order.
4. The list of total values of each order determines the total value of sales.
5. The list of customer names that placed an order determines the set of customer names that bought an item.
6. The list of multisets of article titles together with the name of the customer placing that order determines the set of sold item/customer information.
7. The length of the list of orders determines the number of orders and vice versa. In fact, these values are equal.
8. The list of individual numbers of articles in each order determines the total number of products.
9. Moreover, the list of individual numbers of articles together with the address of the customer who placed that order determines the total number of shippings.

The database schema for this example will be formalised in Example 2.5 on page 48, the constraints in Example 5.2 on page 141.

1.3 Contributions

Apart from a few approaches, the support of complex object types by several data models has not led to investigations about the new expressive power for certain classes of dependencies. Research has rather focused on how to appropriately represent flat data using the types supported. The fact that the introduction of complex types results in constraints among the complex objects has more or less been neglected. Some approaches which do consider the new expressiveness will be discussed later on.

A further problem is given by the great amount of different data models. A key problem is to develop dependency theories (or preferably a unified theory) for most of these advanced data models. Biskup [47, 48] lists in particular two challenges for database design theory: finding a unifying framework and extending achievements to deal with advanced database features such as complex object types.

As in [243] we propose to classify data models according to the type constructors which are supported by the model. The RDM, for instance, is completely captured by the record

type, the nested relational data model and most of the semantic data models by the record and finite set type. The Higher-Order Entity-Relationship model [264, 265] is captured by record, set and disjoint union type. According to the object-oriented database manifesto [20], at least record, set, list and multiset type need to be supported by any object-oriented database model. Union and reference type are also important for object-oriented database models and hypertext data models such as XML.

This view of data models allows to study problems in dependency theory for various classes of dependencies in the presence of various combinations of types, and gives a clear outline of future research, as illustrated in Figure 1.1.

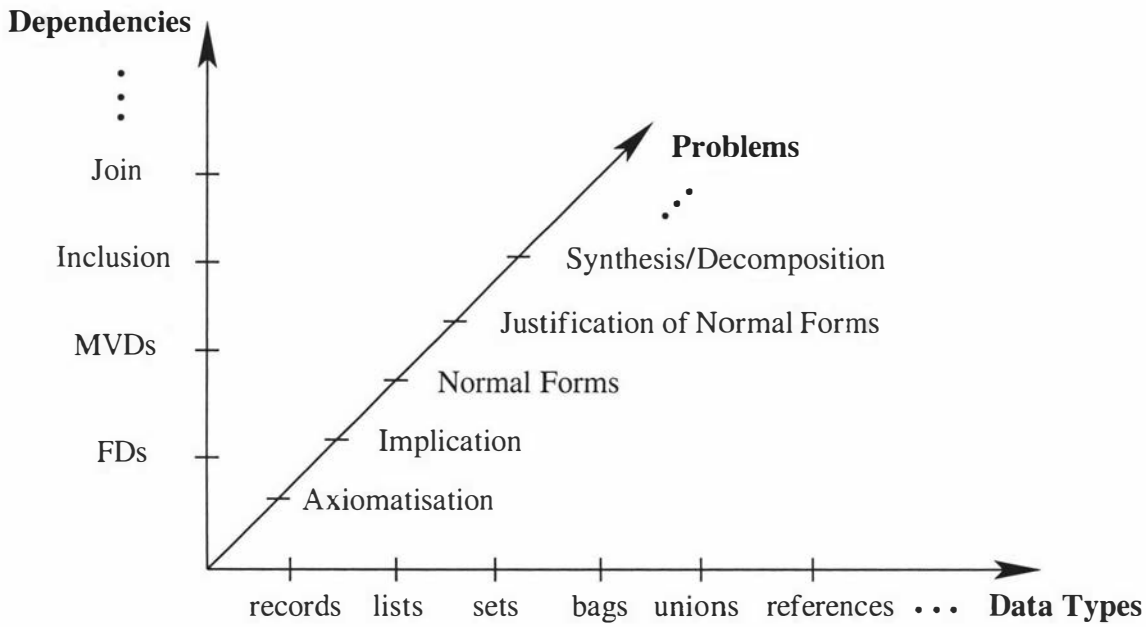

Fig. 1.1. Research on Dependency Theory

It is of course not proposed that this is the ultimate line of future research as it cannot be assumed that dependency theory will be exactly the same in the future as today except with more type constructors. However, whenever a new data model arises that supports a certain complex data type, then the need to deal with these problems in the presence of this type does become apparent for that data model.

The goal of this thesis is to examine some of these problems along the *data type dimension*, and to extend current knowledge to that direction. In doing so, we are much more interested in the constraints among complex objects than in the appropriate representation of the flat data using the complex objects. Typical examples of constraints we are interested in are those from the previous section. The contributions of this thesis are as follows:

– proposal of a mathematically sound framework for the study of various dependency

classes in complex-value databases that is independent from any specific data model but classifies data models according to the data types that are supported,
- definition of functional dependencies in the presence of all combinations of record, list, set and multiset type that include at least the record type,
- definition of multi-valued dependencies in the presence of record and list type,
- complementary expressiveness to those dependency classes that have previously been studied in other data models,
- illustration of the relevance by real-world examples for complex constraints (e.g. genetic databases and retailer databases),
- minimal axiomatisations of functional dependencies in the presence of all combinations of record, list, set and multiset type that include at least the record type,
- provably-correct and polynomial-time algorithms that decide the implication problem of functional dependencies in the presence of all combinations of record, list, set and multiset type that include at least the record type,
- minimal axiomatisations for the class of multi-valued dependencies and the class of functional and multi-valued dependencies in the presence of records and lists,
- provably-correct and polynomial-time algorithms that decide the implication problem for the class of multi-valued dependencies and the class of functional and multi-valued dependencies in the presence of records and lists,
- the applicability of efficiently solving the various implication problems is demonstrated by proposing efficient algorithms for computing non-redundant covers of sets of dependencies and deciding whether a (set of) nested attribute(s) is a superkey with respect to a given set of dependencies,
- differences to the relational data model are highlighted and explained, for instance that MVDs imply non-trivial FDs in the presence of records and lists,
- proposal of the Nested List normal form (NLNF) for nested attributes with respect to functional dependencies in the presence of records and lists, including several semantic justifications and a provably-correct lossless NLNF decomposition algorithm,
- several open problems for future research are identified.

1.4 Outline

The first goal of this thesis is to introduce a theoretically well-founded data model which is sufficiently flexible to support different complex object types. The data model that will be introduced in Chapter 2 is algebraic in nature. Flat attribute names can be nested in various ways using for instance a record, list, set or multiset constructor. These nested attributes can be partially ordered according to the amount of information they represent. Having fixed a nested attribute it becomes interesting to study the structure that the set of all its subattributes carries. It turns out that in the presence of list, set or multiset constructor, the full toolbox of a Boolean algebra cannot be applied. Instead, the set of all subattributes of a fixed nested attribute carries the structure of a Brouwerian algebra (co-Heyting algebra).

In Chapter 3 and 4 the list type and its impact on two important uni-relational dependency classes is investigated in detail. Chapter 3 studies various problems for FDs. As it turns out, FDs can still be captured by a generalisation of Armstrong's axioms in the presence of lists. Next the implication problem of FDs is studied, using a representation theorem for Brouwerian algebras which provides a different, topological view on this class of dependencies. This view allows to extend a provably-correct and linear-time algorithm for deciding implication of FDs in the presence of lists. Next, the problem of syntactically describing well-designed nested attributes with respect to a given set of FDs is addressed. The Nested List Normal Form is proposed and semantically justified in several ways. It is proven that a nested attribute is in Nested List Normal Form if and only if this nested attribute is free from redundancies with respect to the given set of FDs. Moreover, the equivalence of this normal form to the absence of various types of update anomalies is formally shown. Nested List Normal Form is strictly weaker than a simple generalisation of Boyce-Codd Normal Form due to a slight adaption of the notion of redundancy. Finally, the question how to obtain nested attributes in Nested List Normal Form is addressed.

Chapter 4 extends the notion of MVDs to the presence of lists. It is shown that MVDs are still equivalent to binary join dependencies, even in this extended context. This means that an instance satisfies an MVD exactly if this instance can be decomposed without loss of information. Next up, sound inference rules for the implication of FDs and MVDs are introduced. When attempting to show the completeness of these rules it becomes necessary to include a further rule which is trivial in the context of the RDM, but no longer trivial in the presence of lists. This rule allows to infer non-trivial FDs from MVDs, something that is impossible in the RDM. Given this rule, the completeness proof from the RDM can be generalised. A further difference to the RDM is revealed when the independence of the inference rules is studied. The relational counterpart of the join rule for MVDs is implied by other rules in the RDM, and is therefore not contained in the standard minimal set of inference rules. In the presence of lists, however, this join rule is independent from the counterparts of these rules in the minimal set. The Brouwerian complement rule plays a similar role as the complementation rule in the RDM. Finally, the implication problem for the class of FDs and MVDs is studied. A membership algorithm for computing the dependency basis of a nested attribute with respect to some set of FDs and MVDs is proposed for solving this problem, proven to work correctly and shown to work in polynomial time in the size of the input. In conclusion to this chapter the class of MVDs is studied. Minimal axiomatisations and a polynomial-time algorithm for solving the implication problem are provided, and a different, topological view on MVDs proposed.

In Chapter 5, the number of complex object types that are considered is increased. FDs are studied in all combinations of record, list, set and multiset type that contain at least the record type, see Figure 1.2.

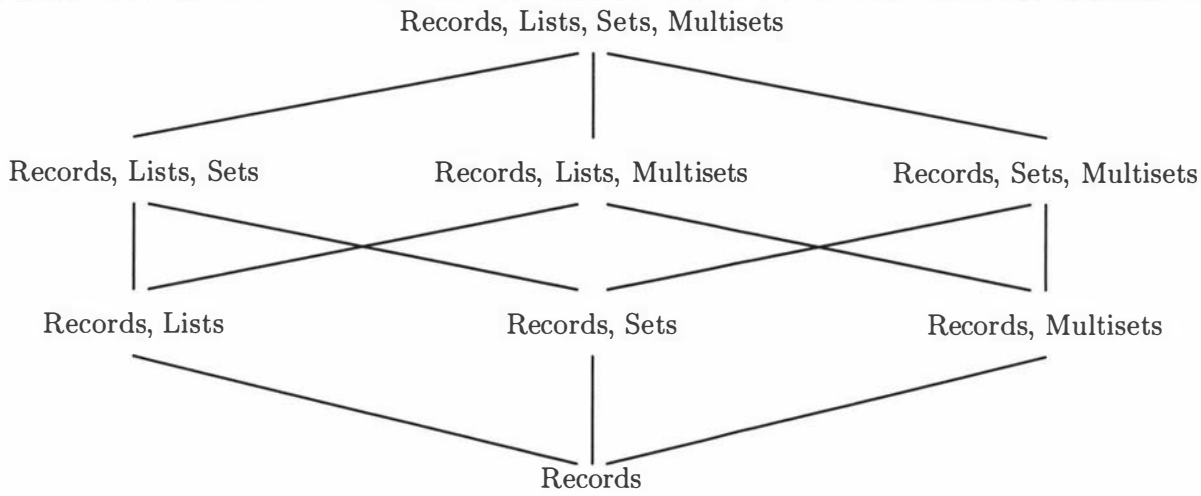

Fig. 1.2. The Boolean algebra of Type Constructors

First of all, the introduction of set or multiset type results in the failure of the extension rule for FDs. This means that the projection of a tuple on two subattributes does not determine the projection of that tuple on the join of these two subattributes. As a consequence, sets of subattributes need to be considered since they are semantically different from the join of these subattributes. This is a fundamental difficulty and results in a more sophisticated axiomatisation of FDs. There are situations, however, when the projection of a tuple on two subattributes still determines the projection of that tuple on the join of these two subattributes. A condition that implies the presence of such a situation is proposed. As a matter of fact, this condition turns out to characterise those situations precisely. Two new axioms are necessary to capture FDs in the presence of records, lists, sets and multisets. The completeness proof follows the lines of the traditional proof but requires deeper arguments. In fact, the proof remains still constructive, i.e., a two element instance is constructed where the elements are coincident exactly on the closure of a subattribute with respect to the set of FDs given. The construction of this two element instance is done inductively for flat, record and list type, but separate and direct arguments are given for set- and multiset case. The case of multisets involves some combinatorial arguments and takes advantage of some further facts on the structure of subattributes. The main result is a minimal axiomatisation for FDs in the presence of all combinations of record, list, set and multiset type that contain at least the record type, i.e. at least capture the RDM. Furthermore, the implication problem of FDs is studied in the same type contexts. An algorithm for deciding these problems is proposed, proven correct and to work in polynomial time in the number of subattributes and the number of FDs given.

Chapter 2

The Algebra of Nested Attributes

In this chapter the data model is introduced which all further studies in this thesis will be based on. The data model is abstract in the sense that it can be adapted to the particular data types of current interest, for instance lists, sets, multisets, unions, references etc. The record type by itself captures the RDM. Therefore, the record type will be present in each of the different type systems we are interested in.

The fundamental, yet simple, observation that led to this data model is based on the algebraic nature of the RDM. A relation schema R forms a Boolean algebra with respect to set inclusion, union, intersection and complement. The RDM is a flat data model in the sense that any value in a relational database is a single value from the domain of the corresponding attribute. Nested attributes result from flat attributes by recursively applying various type constructors. These nested attributes can be partially ordered according to the level of information they represent. Thus, a subattribute represents at most as much information as any of its superattributes does. Therefore, the notion of a subattribute generalises the notion of a subset from the RDM.

The main objective of this chapter is the study of the algebraic structure of the set of subattributes for some fixed nested attribute. It turns out that even in the presence of various type constructors a powerful algebraic toolbox can still be applied. Although the structure of a Boolean algebra can no longer be maintained in general, a slightly less powerful framework can be utilised.

Short versions of the contents of this chapter have appeared as introductory sections in [139, 140, 141, 142, 143, 145, 146].

2.1 Brouwerian Algebras

Familiarity is assumed with such notions as partially-ordered set $(P, \leq)$, lattice $(L, \leq, \sqcup, \sqcap)$ and Boolean algebra $(B, \leq, \sqcup, \sqcap, (\cdot)^{\mathcal{C}}, 0, 1)$. The fundamental algebraic objects in this thesis are so-called Brouwerian algebras. These algebras have been introduced and studied by McKinsey and Tarski in [202] to establish precise connections with closure algebras in topology. We repeat the basic notions and results that will be important for our further studies. At the end of this chapter, some areas are listed which Brouwerian algebras have

been applied to. We start off with the basic definition of a Brouwerian algebra as it was introduced in [202].

Definition 2.1. An algebra $(B, \leq, \sqcup, \sqcap, \dot{-}, 1)$ is called *Brouwerian algebra* if and only if

1. $(B, \leq, \sqcup, \sqcap)$ is a lattice with top element 1, and
2. B is closed under $\dot{-}$, and
3. for all $a, b, c \in B$ the formulae $a \dot{-} b \leq c$ and $a \leq b \sqcup c$ are equivalent.

The operation $\dot{-}$ is called *pseudo-difference.* □

According to Definition 2.1, the pseudo-difference $a \dot{-} b$ of b relative to a is the smallest c such that $a \leq b \sqcup c$. Of special interest is the pseudo-difference when its first argument is the top element 1. For this case, a special notion and notation is introduced.

Definition 2.2. If $(B, \leq, \sqcup, \sqcap, \dot{-}, 1)$ is a *Brouwerian algebra* and a is any element of B, the element $\neg a$ defined by $\neg a = 1 \dot{-} a$ is called the *Brouwerian complement* of a. □

It follows by the third property of Brouwerian algebras in Definition 2.1 that for all $b, c \in B$, the formulae $\neg b \leq c$ and $b \sqcup c = 1$ are equivalent. Consequently, the Brouwerian complement $\neg b$ of b is the smallest c with $b \sqcup c = 1$. Next, we list some important and fundamental properties of Brouwerian algebras. They will be used in what follows, and have already been proven in [202, Theorem 1.3.].

Theorem 2.3. *Let $\mathcal{B} = (B, \leq, \sqcup, \sqcap, \dot{-}, 1)$ be a Brouwerian algebra. Then*

1. *$\mathcal{B}$ has a bottom element 0 determined by the formula $0 = 1 \dot{-} 1$.*
2. *$\mathcal{B}$ is a distributive lattice, i.e., for all $a, b, c \in B$ we have*

$$a \sqcup (b \sqcap c) = (a \sqcup b) \sqcap (a \sqcup c) \quad \text{and} \quad a \sqcap (b \sqcup c) = (a \sqcap b) \sqcup (a \sqcap c).$$

3. *If $a \leq b$, then $a \dot{-} c \leq b \dot{-} c$, $c \dot{-} b \leq c \dot{-} a$, and $\neg b \leq \neg a$.*
4. *$a \leq b$ is equivalent to $a \dot{-} b = 0$.*
5. *$a \leq b \sqcup (a \dot{-} b)$.*
6. *$(a \sqcup b) \dot{-} b \leq a$.*
7. *$a \dot{-} c \leq (a \sqcup b) \dot{-} c$.*
8. *$c \sqcup (a \dot{-} b) = c \sqcup [(c \sqcup a) \dot{-} (c \sqcup b)]$.*
9. *$c \dot{-} (a \sqcap b) = (c \dot{-} a) \sqcup (c \dot{-} b)$.*
10. *$(a \sqcup b) \dot{-} c = (a \dot{-} c) \sqcup (b \dot{-} c)$.*
11. *$\neg\neg a \leq a$.*
12. *$\neg\neg\neg a = \neg a$.*
13. *$\neg 0 = 1$ and $\neg 1 = 0$.*
14. *$a \sqcup \neg a = 1$.* □

Next we are concerned with special constructions of Brouwerian algebras. Interesting for our purposes are finite direct products and augmentation of a new minimum. In fact, the next result shows that Brouwerian algebras are closed under finite direct products. The proof is immediate.

Theorem 2.4. *Let* $(B_i, \leq_i, \sqcup_i, \sqcap_i, \dot{-}_i, 1_i)$ *be a Brouwerian algebra for* $i = 1, \ldots, k$. *Let* $\mathcal{B} = (B, \leq, \sqcup, \sqcap, \dot{-}, 1)$ *be the algebra defined as follows:*

- $B = B_1 \times \cdots \times B_k = \{(b_1, \ldots, b_k) : b_i \in B_i\}$,
- $(b_1, \ldots, b_k) \leq (b'_1, \ldots, b'_k)$, *if* $b_i \leq b'_i$ *for* $i = 1, \ldots, k$, *and*
- $(b_1, \ldots, b_k) \circ (b'_1, \ldots, b'_k) = (b_1 \circ_1 b'_1, \ldots, b_k \circ_k b'_k)$ *for* $\circ \in \{\sqcup, \sqcap, \dot{-}\}$.

Then $\mathcal{B}$ *is a Brouwerian algebra.* □

The following definition is again due to [202, Definition 1.6.].

Definition 2.5. If $\mathcal{B} = (B, \leq, \sqcup, \sqcap, \dot{-}, 1)$ is a Brouwerian algebra and a is an element of B, then we put $\mathcal{B}_a = (B_a, \leq, \sqcup, \sqcap, \dot{-}_a, 1)$ where B_a is the set of all elements b of B such that $a \leq b$ and $b \dot{-}_a c = a \sqcup (b \dot{-} c)$ for arbitrary elements b and c of B_a. $\mathcal{B}_a$ is referred to as *the relativised subalgebra of* $\mathcal{B}$ *with respect to* a. □

If $\mathcal{B}$ is a Brouwerian algebra and a is any element of $\mathcal{B}$, then $\mathcal{B}_a$ is also a Brouwerian algebra, see [202, Theorem 1.7.]. The next result shows that Brouwerian algebras are closed under the augmentation of a new bottom element. This is proven as Theorem 1.9. in [202].

Theorem 2.6. *If* $\mathcal{B}$ *be a Brouwerian algebra, then there exists a Brouwerian algebra* $\mathcal{B}'$ *with the following properties:*

- *The bottom element* $0'$ *of* $\mathcal{B}'$ *is the only element of* $\mathcal{B}'$ *which is not in* $\mathcal{B}$,
- $\mathcal{B} = (\mathcal{B}')_a$ *where* $a = 0$ *is the bottom element of* $\mathcal{B}$. □

Not every Brouwerian algebra is necessarily a Boolean algebra. The following result [202, Theorem 1.12.] describes the connection more precisely.

Theorem 2.7. *Let* $(B, \leq, \sqcup, \sqcap, \dot{-}, 1)$ *be a Brouwerian algebra.* $(B, \leq, \sqcup, \sqcap, \neg, 0, 1)$ *is a Boolean algebra if and only if* $a \sqcap \neg a = 0$ *holds for every* $a \in B$. □

The condition $a \sqcap \neg a = 0$ of the last theorem can be replaced by the formula $a = \neg\neg a$ which holds in arbitrary Boolean algebras, but not in arbitrary Brouwerian algebras.

A Brouwerian algebra is also called a co-Heyting algebra or a dual Heyting algebra. While in a Heyting algebra the join of an element and its complement is not necessarily the top element, in a Brouwerian algebra the meet of an element and its Brouwerian complement is not necessarily the bottom element. The system of all closed subsets of a topological space is a well-known Brouwerian algebra. To illustrate this a bit further we define a topological space with respect to a closure operation as in [100, 201].

Definition 2.8. A *topological space* $\mathcal{T}$ is a structure $(S, \mathfrak{C})$ where S is a set and $\mathfrak{C}$ an operation that maps subsets of S to subsets of S satisfying, for all $A, B \subseteq S$:

- $A \subseteq \mathfrak{C}A$,
- $\mathfrak{C}A = \mathfrak{C}\mathfrak{C}A$,
- $\mathfrak{C}(A \cup B) = \mathfrak{C}A \cup \mathfrak{C}B$,

– $\mathfrak{C}\emptyset = \emptyset$.

A subset A of S is *closed* just in case $\mathfrak{C}A = A$. □

The closed elements of Definition 2.8 have the usual properties: $\emptyset, S$ are both closed, the set union of any pair of closed elements is closed, and the set intersection of arbitrarily many closed sets is closed as well.

As mentioned before, every family of closed subsets of a topological space carries the structure of a Brouwerian algebra [100, 201, 255, 263].

Theorem 2.9. *Let $(S, \mathfrak{C})$ be a topological space, and let $\mathcal{C}$ be the family of closed subsets of S. Then $(\mathcal{C}, \subseteq, \cup, \cap, \dot{-}, S)$ is a Brouwerian algebra, where $\subseteq$ denotes set-inclusion, $\cup$ set-union, $\cap$ set-intersection, and $\dot{-}$ is given by $A \dot{-} B = \mathfrak{C}\{x \mid x \in A \text{ and } x \notin B\}$.* □

There is a representation theorem for Brouwerian algebras due to Stone, McKinsey and Tarski [202, 255]. In fact, every Brouwerian algebra is isomorphic to a subalgebra of the algebra of closed sets of a topological space. We will state this theorem only for finite Brouwerian algebras.

Given a poset $(S, \leq)$, we define for $A \subseteq S$, $\mathfrak{C}A = \{b \in S \mid b \leq a \text{ for some } a \in A\}$. That means $\mathfrak{C}A$ closes A downwards with respect to $\leq$. The topological space $(S, \mathfrak{C})$ is called a *PO-space.*

In order to prove the representation theorem it can be shown that for any finite Brouwerian algebra $(B, \leq_B, \sqcup, \sqcap, \dot{-}, 1)$ there is some poset $(S, \leq_S)$ such that the Brouwerian algebra of closed sets of the corresponding PO-space is isomorphic to the original Brouwerian algebra. It is not possible to simply take S to be B and $\leq_S$ to be $\leq_B$, since this PO-space will in general have more closed sets than there are elements in B.

Definition 2.10. An element a of a lattice $(L, \leq, \sqcup, \sqcap)$ with bottom element 0 is called *join-irreducible* if and only if $a \neq 0$ and, for all $b, c \in L$, if $a = b \sqcup c$, then $a = b$ or $a = c$. □

We are now prepared to state the representation theorem. For a proof of the dual statement see [100]. The interested reader may also consult [45] for more details.

Theorem 2.11. *Let $\mathcal{B} = (B, \leq, \sqcup, \sqcap, \dot{-}, 1)$ be a finite Brouwerian algebra, and $(\mathcal{C}, \subseteq, \cup, \cap, \dot{-}_{\mathcal{C}}, J)$ the Brouwerian algebra of closed sets of the PO-space on the set J of join-irreducible elements of $\mathcal{B}$ under the restriction of the partial order $\leq$ to J. Then, $\vartheta(a) = \{d \in J \mid d \leq a\}$, defines an isomorphism between $\mathcal{B}$ and $(\mathcal{C}, \subseteq, \cup, \cap, \dot{-}_{\mathcal{C}}, J)$, and for all $a, b \in B$, $\vartheta(a \dot{-} b) = \vartheta(a) \dot{-}_{\mathcal{C}} \vartheta(b)$.* □

2.2 Nested Attributes

We will follow the same approach as the RDM by trying to capture the characteristics of objects in the target database by attribute names. Starting point is the definition of flat attributes and values for them.

Definition 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.* □

For the sake of convenience we will make the assumption that for every $A \in \mathcal{U}$ the domain $dom(A)$ contains at least two different elements. For the relational data model a universe was sufficient. That is, a relation schema is defined as a finite and non-empty subset $\mathcal{R} \subseteq \mathcal{U}$. For data models supporting complex object types, however, nested attributes are needed. In the following definition we use a set $\mathcal{L}$ of labels, and assume that the symbol λ is neither a flat attribute nor a label, i.e., $\lambda \notin \mathcal{U} \cup \mathcal{L}$. Moreover, flat attributes are not labels and vice versa, i.e., $\mathcal{U} \cap \mathcal{L} = \emptyset$.

Definition 2.13. Let $\mathcal{U}$ be a universe and $\mathcal{L}$ a set of labels. The set $\mathcal{NA}(\mathcal{U}, \mathcal{L})$ of *nested attributes over* $\mathcal{U}$ *and* $\mathcal{L}$ is the smallest set satisfying the following conditions:

1. $\lambda \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$,
2. $\mathcal{U} \subseteq \mathcal{NA}(\mathcal{U}, \mathcal{L})$,
3. for $L \in \mathcal{L}$ and $N_1, \ldots, N_k \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$ with $k \geq 1$ we have $L(N_1, \ldots, N_k) \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$,
4. for $L \in \mathcal{L}$ and $N \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$ we have $L[N] \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$,
5. for $L \in \mathcal{L}$ and $N \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$ we have $L\{N\} \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$,
6. for $L \in \mathcal{L}$ and $N \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$ we have $L\langle N \rangle \in \mathcal{NA}(\mathcal{U}, \mathcal{L})$.

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

From now on we assume that a universe $\mathcal{U}$ and a set $\mathcal{L}$ of labels have been fixed, and we usually drop the index simply writing $\mathcal{NA}$ instead of $\mathcal{NA}(\mathcal{U}, \mathcal{L})$.

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

Definition 2.14. For a nested attribute $N \in \mathcal{NA}$ we define the *domain* $dom(N)$ as follows:

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

Note that a relation schema $R = \{A_1, \ldots, A_n\}$ is captured by the record-valued attribute $R(A_1, \ldots, A_n)$ 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 $r \subseteq dom(N)$.

2.2.1 Subattributes

Dependency theory in the relational data model is based on the powerset $\mathcal{P}(R)$ for a relation schema R. In fact, $\mathcal{P}(R)$ is a powerset algebra with partial order $\subseteq$, set union $\cup$, set intersection $\cap$ and set difference $-$. We will generalise these operations for nested attributes starting with a partial order $\leq$.

Definition 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. $L(N_1, \ldots, N_k) \leq L(M_1, \ldots, M_k)$ whenever $N_i \leq M_i$ for all $i = 1, \ldots, k$,
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 $R(A, B, C)$.

The subattribute relation $\leq$ on nested attributes is reflexive, anti-symmetric and transitive.

Lemma 2.16. *The subattribute relation is a partial order on nested attributes.*

Proof. Reflexivity is given by the first rule of Definition 2.15, i.e., $N \leq N$ for every nested attribute N over $\mathcal{U}$ and $\mathcal{L}$.

Let $N, M \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ with $N \leq M$ and $M \leq N$. We show that $M = N$ by induction on the structure of N. If $N = \lambda$, then the only rule in Definition 2.15 that gives $M \leq \lambda$ is the first one, i.e., $M = \lambda$ and therefore $M = N$. If $M = \lambda$, then $N \leq \lambda$ is again implied by the first rule, and $N = \lambda$ follows. If $N \in \mathcal{U}$ is a flat attribute, then the only rule that gives $N \leq M$ is again the first one, and therefore $M = N$. If $N = L(N_1, \ldots, N_k)$ is a record-valued attribute, then the hypothesis tells us for $i = 1, \ldots, k$ that if $M_i \leq N_i$ and $N_i \leq M_i$, then $N_i = M_i$. From $M \leq N$ follows $M = L(M_1, \ldots, M_k)$ and we have $M_i \leq N_i$ for $i = 1, \ldots, k$. From $N \leq M$ follows $N_i \leq M_i$ for $i = 1, \ldots, k$, and consequently $N_i = M_i$ for $i = 1, \ldots, k$ by hypothesis. This shows again $N = M$. If $N = L[N']$ is a list-valued attribute, then the hypothesis is that $M' \leq N'$ and $N' \leq M'$ imply $N' = M'$. Since $M \leq N$ it remains to consider the case where $M = L[M']$, i.e. $M' \leq N'$. On the other hand, $N \leq M$ shows also $N' \leq M'$. Consequently, $N = M$ since $M' = N'$ by hypothesis. The arguments for the cases where N is a set-valued or multiset-valued attribute are the same as for list-valued attributes. This shows the antisymmetry of $\leq$.

Let $N, M, K \in \mathcal{N}\mathrm{A}(\mathcal{U}, \mathcal{L})$ with $N \leq M$ and $M \leq K$. We show that $N \leq K$ by induction on the structure of K. If $K = \lambda$, then the only rule in Definition 2.15 that gives $M \leq \lambda$ is

the first one, i.e., $M = \lambda$. As $N \leq M$ holds as well, we have $N = \lambda$ and therefore $N \leq K$ by the first rule. If $K \in \mathcal{U}$ is a flat attribute, then $M \leq K$ means $M = \lambda$ by the second rule or $M = K$ by the first rule. If $M = \lambda$, then $N = \lambda$ since $N \leq M$, and $N \leq K$ by the second rule. If $M = K$, then $N \leq K$ as $N \leq M$. Let $K = L(K_1, \ldots, K_l)$ be a record-valued attribute. The hypothesis says for every $i = 1, \ldots, l$ that if $N_i \leq M_i$ and $M_i \leq K_i$, then $N_i \leq K_i$. Since $N \leq M$ and $M \leq K$ we have $M = L(M_1, \ldots, M_l)$ and $N = L(N_1, \ldots, N_l)$ with $N_i \leq M_i$ and $M_i \leq K_i$ for $i = 1, \ldots, l$. We therefore conclude by hypothesis that $N_i \leq K_i$ for $i = 1, \ldots, l$. This shows $N \leq K$ by the fourth rule. Let $K = L[K']$ be a list-valued attribute. We know by hypothesis that if $N' \leq M'$ and $M' \leq K'$, then $N' \leq K'$. If $N = \lambda$, then $N \leq K$ by rule three. From $M \leq K$ follows $M = \lambda$, which implies $N = \lambda$ and $N \leq K$ by rule three, or $M = L[M']$ with $M' \leq K'$. From $N \leq M$ follows $N = \lambda$ and $N \leq K$ by rule three, or $N = L[N']$ with $N' \leq M'$. We apply hypothesis to the remaining case where $N' \leq M'$ and $M' \leq K'$, obtaining $N' \leq K'$. An application of rule five shows $N \leq K$. The arguments for the cases where N is a set-valued or multiset-valued attribute are the same as for list-valued attributes. This shows the transitivity of $\leq$. □

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

Definition 2.17. Let $N, M \in \mathcal{NA}$ with $M \leq N$. The *projection function* $\pi_M^N : dom(N) \to dom(M)$ is defined as follows:

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

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

2.2.2 The Brouwerian algebra of Subattributes

We will now fix a nested attribute and study the structure of the set of all its subattributes.

Definition 2.18. Let $N \in \mathcal{NA}$ be a nested attribute. The set $Sub(N)$ of *subattributes* of N is $Sub(N) = \{M \mid M \leq N\}$. □

Lemma 2.16 indicates that the restriction of $\leq$ to $Sub(N)$ is a partial order on $Sub(N)$. We investigate the algebraic structure of $(Sub(N), \leq)$. In the following we will define operations of join, meet and pseudo-difference on $(Sub(N), \leq)$. Obviously, the nested attribute N is the top element of $(Sub(N), \leq)$. What is the bottom element?

Definition 2.19. The *bottom element* λ_N of $Sub(N)$ is given by $\lambda_N = L(\lambda_{N_1}, \ldots, \lambda_{N_k})$ whenever $N = L(N_1, \ldots, N_k)$, and $\lambda_N = \lambda$ whenever N is not a record-valued attribute. □

According to Definition 2.15, λ_N is indeed the bottom element of $(Sub(N), \leq)$. Therefore, $(Sub(N), \leq, \lambda_N, N)$ is a bounded poset with bottom element λ_N and top element N.

Definition 2.20. Let $N \in \mathcal{NA}$ and $X, Y \in Sub(N)$. The *join* $X \sqcup_N Y$, *meet* $X \sqcap_N Y$ and *pseudo-difference* $X \dot{-}_N Y$ of X and Y in $Sub(N)$ are inductively defined as follows:

1. if $X \leq Y$, then $X \sqcup_N Y = Y, X \sqcap_N Y = X$ and $X \dot{-}_N Y = \lambda_N$,
2. $X \dot{-}_N \lambda_N = X$,
3. if $N = L(N_1, \ldots, N_k), X = L(X_1, \ldots, X_k)$ and $Y = L(Y_1, \ldots, Y_k)$, then $X \circ_N Y = L(X_1 \circ_{N_1} Y_1, \ldots, X_k \circ_{N_k} Y_k)$ for $\circ \in \{\sqcup, \sqcap, \dot{-}\}$,
4. if $N = L[M], X = L[X'], Y = L[Y']$, then $X \circ_N Y = L[X' \circ_M Y']$ for $\circ \in \{\sqcup, \sqcap\}$ and if $X \not\leq Y$, then $X \dot{-}_N Y = L[X' \dot{-}_M Y']$,
5. if $N = L\{M\}, X = L\{X'\}, Y = L\{Y'\}$, then $X \circ_N Y = L\{X' \circ_M Y'\}$ for $\circ \in \{\sqcup, \sqcap\}$ and if $X \not\leq Y$, then $X \dot{-}_N Y = L\{X' \dot{-}_M Y'\}$,
6. if $N = L\langle M \rangle, X = L\langle X' \rangle, Y = L\langle Y' \rangle$, then $X \circ_N Y = L\langle X' \circ_M Y' \rangle$ for $\circ \in \{\sqcup, \sqcap\}$ and if $X \not\leq Y$, then $X \dot{-}_N Y = L\langle X' \dot{-}_M Y' \rangle$. □

We are now going to show that the operations in Definition 2.20 are well-defined, i.e., $(Sub(N), \leq, \sqcup_N, \sqcap_N, \dot{-}_N, N)$ is a Brouwerian algebra. It is obvious that join, meet and pseudo-difference are closed. It therefore remains to show that $(Sub(N), \leq, \sqcup_N, \sqcap_N)$ is a lattice, and that $\dot{-}_N$ is indeed the pseudo-difference operation.

Lemma 2.21. *$(Sub(N), \leq, \sqcup_N, \sqcap_N)$ is a lattice.*

Proof. It remains to show that $\sqcup_N$ and $\sqcap_N$ indeed define join and meet, respectively. Let $X, Y \in Sub(N)$. If $X \leq Y$, then $X, Y \leq Y = X \sqcup_N Y$ and $X \sqcap_N Y = X \leq X, Y$. If $Z \in Sub(N)$ with $X, Y \leq Z$, then $X \sqcup_N Y = Y \leq Z$. If $Z \in Sub(N)$ with $Z \leq X, Y$, then $Z \leq X = X \sqcap_N Y$.

Let $N = L(N_1, \ldots, N_k)$. Consequently, $X = L(X_1, \ldots, X_k)$ and $Y = L(Y_1, \ldots, Y_k)$ with $X_i, Y_i \leq N_i$ for $i = 1, \ldots, k$. $X, Y \leq X \sqcup_N Y$ by rule four of Definition 2.15 as $X_i, Y_i \leq X_i \sqcup_{N_i} Y_i$ for $i = 1, \ldots, k$. Similarly, $X \sqcap_N Y \leq X, Y$ as $X_i \sqcap_{N_i} Y_i \leq X_i, Y_i$ for $i = 1, \ldots, k$. Let $Z \in Sub(N)$ with $X, Y \leq Z$. It follows that $Z = L(Z_1, \ldots, Z_k)$ and

$X_i, Y_i \leq Z_i$ for $i = 1, \ldots, k$. We conclude that $X_i \sqcup_{N_i} Y_i \leq Z_i$ holds for $i = 1, \ldots, k$, and therefore $X \sqcup_N Y \leq Z$ by rule four of Definition 2.15. Let $Z \in Sub(N)$ with $Z \leq X, Y$. It follows that $Z = L(Z_1, \ldots, Z_k)$ and $Z_i \leq X_i, Y_i$ for $i = 1, \ldots, k$. We conclude that $Z_i \leq X_i \sqcap_{N_i} Y_i$ holds for $i = 1, \ldots, k$, and therefore $Z \leq X \sqcap_N Y$.

Let $N = L[N']$. It remains to consider the case where $X = L[X']$ and $Y = L[Y']$ with $X', Y' \leq N'$. It follows that $X', Y' \leq X' \sqcup_{N'} Y'$ and $X' \sqcap_{N'} Y' \leq X', Y'$. This shows $X, Y \leq X \sqcup_N Y$ and $X \sqcap_N Y \leq X, Y$ by rule five of Definition 2.15. Let $Z \in Sub(N)$ with $X, Y \leq Z$. It follows that $Z = L[Z']$ with $X', Y' \leq Z'$. Consequently, $X' \sqcup_{N'} Y' \leq Z'$ and $X \sqcup_N Y \leq Z$ by rule five of Definition 2.15. Let $Z \in Sub(N)$ with $Z \leq X, Y$. If $Z = \lambda$, then $Z \leq X \sqcap_N Y$ by rule three of Definition 2.15. Otherwise, $Z = L[Z']$ with $Z' \leq X', Y'$. This shows $Z' \leq X' \sqcap_{N'} Y'$ and we conclude that $Z \leq X \sqcap_N Y$ by rule five of Definition 2.15.

The cases where N is a set-valued or multiset-valued attribute follow the same arguments that have been used for list-valued attributes. □

Lemma 2.22. *For all $X, Y, Z \in Sub(N)$ the formulae $X \dot{-}_N Y \leq Z$ and $X \leq Y \sqcup_N Z$ are equivalent.*

Proof. We proceed by induction on the structure of N. If $X \leq Y$, then $X \dot{-}_N Y = \lambda_N$ and both formulae are true. If $Y = \lambda_N$, then both formulae reduce to $X \leq Z$. In what follows, we can therefore assume that $X \not\leq Y$, and X and Y are both distinct from λ_N.

If $N = L(N_1, \ldots, N_k)$, $X = L(X_1, \ldots, X_k)$, $Y = L(Y_1, \ldots, Y_k)$ and $Z = L(Z_1, \ldots, Z_k)$, then the formulae $X_i \dot{-}_{N_i} Y_i \leq Z_i$ and $X_i \leq Y_i \sqcup Z_i$ are equivalent for all $i = 1, \ldots, k$. According to Definition 2.20 this shows the equivalence of $X \dot{-}_N Y \leq Z$ and $X \leq Y \sqcup_N Z$.

Let $N = L[N']$, $X = L[X']$ and $Y = L[Y']$. If $Z = \lambda$, then both formulae are equivalent to $X \leq Y$ according to Definition 2.20. Let $Z = L[Z']$. The formulae $X \dot{-}_N Y \leq Z$ is then equivalent to $X' \dot{-}_{N'} Y' \leq Z'$, which itself is equivalent to $X' \leq Y' \sqcup_{N'} Z'$. The last formula, however, is equivalent to $X \leq Y \sqcup_N Z$.

The cases of set-valued and multiset-valued attributes follow again the same line of reasoning as the case of list-valued attributes. □

Lemma 2.21 and Lemma 2.22 show the following result. It generalises the fact that $(\mathcal{P}(R), \subseteq, \cup, \cap, -, \emptyset, R)$ is a Boolean algebra for a relation schema R in the RDM.

Theorem 2.23. *$(Sub(N), \leq, \sqcup_N, \sqcap_N, \dot{-}_N, N)$ forms a Brouwerian algebra for every $N \in \mathcal{NA}$.* □

An alternative way of showing that $(Sub(N), \leq)$ carries the structure of a Brouwerian algebra is the following. $Sub(\lambda_N)$ is isomorphic to the Boolean algebra of order 0, and $Sub(A)$, A a flat attribute, is isomorphic to the Boolean algebra of order 1 (second rule). Furthermore, $Sub(L(N))$ is isomorphic to $Sub(N)$, $Sub(L(N_1, \ldots, N_k))$ isomorphic to the direct product of $Sub(N_1), \ldots, Sub(N_k)$ (fourth rule), and $Sub(L[N])$, $Sub(L\{N\})$, $Sub(L\langle N\rangle)$ are all isomorphic to $Sub(N)$ augmented by a new minimum (rule three and five, six, seven, correspondingly). Theorem 2.4 and Theorem 2.6 show that Brouwerian

algebras are closed under (finite) direct products and augmentations of a new bottom element, respectively. This shows that $(Sub(N), \leq)$ carries the structure of a Brouwerian algebra. Given some nested attribute $N \in \mathcal{N}\mathrm{A}$ and $Y, Z \in Sub(N)$, we use $Y_N^{\mathcal{C}} = N \dot{-} Y$ to denote the *Brouwerian complement* of Y in $Sub(N)$. Consequently, for all $X \in Sub(N)$ the formulae $Y^{\mathcal{C}} \leq X$ and $X \sqcup Y = N$ are equivalent. Note that $\lambda_N = N_N^{\mathcal{C}}$ holds in particular.

Corollary 2.24. *The algebra* $(Sub(N), \leq, \sqcup, \sqcap, (\cdot)^{\mathcal{C}}, \lambda_N, N)$ *is in general not Boolean.*

Proof. Take $N = L[A]$ and $Y = L[\lambda]$. Then $Y_N^{\mathcal{C}} = N$ and $Y \sqcap Y_N^{\mathcal{C}} = Y \neq \lambda$. Furthermore, $(Y_N^{\mathcal{C}})_N^{\mathcal{C}} = \lambda \neq Y$. The corollary follows from Theorem 2.7. □

It might be interesting to note that not every finite Brouwerian algebra is isomorphic to a Brouwerian algebra of nested attributes. In fact, Figure 2.1 shows the structure of a Brouwerian algebra which cannot be a Brouwerian algebra of any nested attribute. The structure in this figure is not the structure of a null or flat attribute. Neither does it result from a direct product of two substructures, and is therefore not the structure of a record-valued attribute. It cannot be the structure of a list-valued nor set-valued nor multiset-valued attribute since such a structure has a bottom element with a single superattribute.

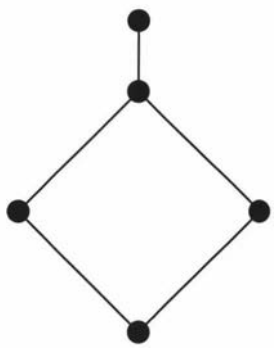

Fig. 2.1. A Brouwerian algebra that is not an algebra of any nested attribute.

2.2.3 Notation, Examples and Intuition

In this section we make some remarks on notation, in particular that of nested attributes. First of all, if the context allows we will omit the index N from the operations $\sqcup_N, \sqcap_N$, $\dot{-}_N$ and from λ_N as well as $Y_N^{\mathcal{C}}$. In order to simplify the notation for subattributes, occurrences of λ in a record-valued attribute are usually omitted if this does not cause any ambiguities. That is, the subattribute $L(M_1, \ldots, M_k) \in Sub(L(N_1, \ldots, N_k))$ is abbreviated by $L(M_{i_1}, \ldots, M_{i_l})$ where

$$\{M_{i_1}, \ldots, M_{i_l}\} = \{M_j : M_j \neq \lambda_{N_j} \text{ and } 1 \leq j \leq k\} \text{ and } i_1 < \cdots < i_l.$$

If $M_j = \lambda_{N_j}$ for all $j = 1, \ldots, k$, then we write λ instead of $L(\lambda_{N_1}, \ldots, \lambda_{N_k})$.

EXAMPLE 2.1. The subattribute $L_1(A, \lambda, L_2[L_3(\lambda, \lambda)])$ of $L_1(A, B, L_2[L_3(C, D)])$ is abbreviated by $L_1(A, L_2[\lambda])$. However, the subattribute $L(A, \lambda)$ of $L(A, A)$ will not be abbreviated by $L(A)$ since this may also refer to $L(\lambda, A)$. □

Next, we give some examples for Brouwerian algebras of nested attributes. The algebra $\mathcal{B}$ for $K\{L(A, M[N(B, C)])\}$ is illustrated in Figure 2.2.

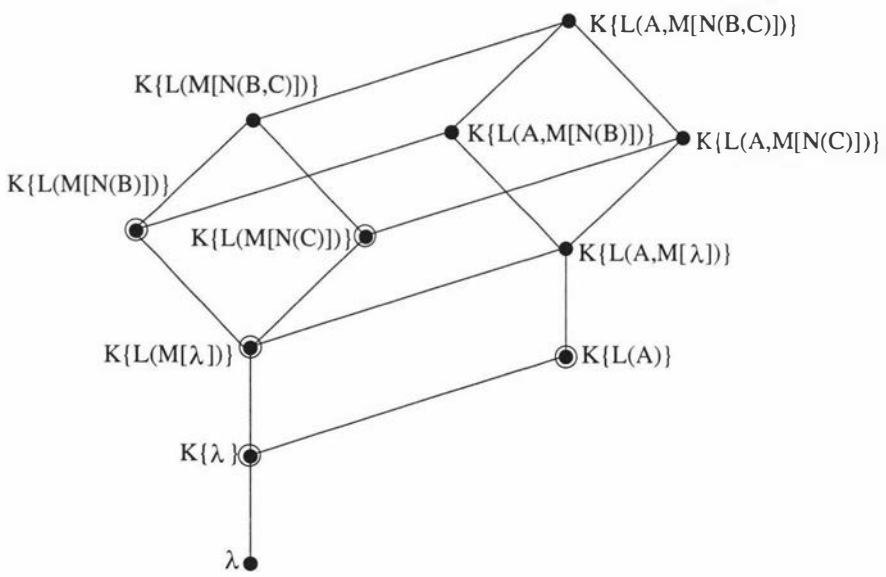

Fig. 2.2. The Brouwerian algebra $\mathcal{B}$ of $K\{L(A, M[N(B, C)])\}$

We will now give an example of Theorem 2.11. In Figure 2.2, the join-irreducible subattributes of $\mathcal{B}$ are circled. Let $V = K\{L(M[N(B)])\}$, $W = K\{L(M[N(C)])\}$, $X = K\{L(M[\lambda])\}$, $Y = K\{L(A)\}$ and $Z = K\{\lambda\}$. The set $\mathcal{C}$ of all closed subsets of the *PO*-space on the poset $(J, \leq)$ in Figure 2.3 consists of the following elements $\emptyset$, $\{Z\}$, $\{X, Z\}$, $\{Y, Z\}$, $\{V, X, Z\}$, $\{W, X, Z\}$, $\{X, Y, Z\}$, $\{V, W, X, Z\}$, $\{V, X, Y, Z\}$, $\{W, X, Y, Z\}$, $\{V, W, X, Y, Z\}$.

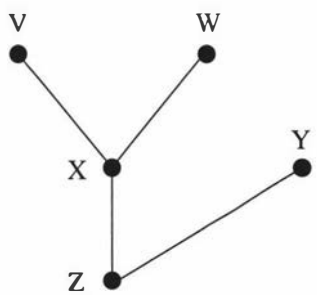

Fig. 2.3. The poset $(J, \leq)$ of the join-irreducible elements of $\mathcal{B}$.

The Brouwerian algebra of these closed subsets of the PO-space on $(J, \leq)$ is illustrated in Figure 2.4. Take for instance the subattributes $S = K\{L(M[N(B, C)])\}$ and

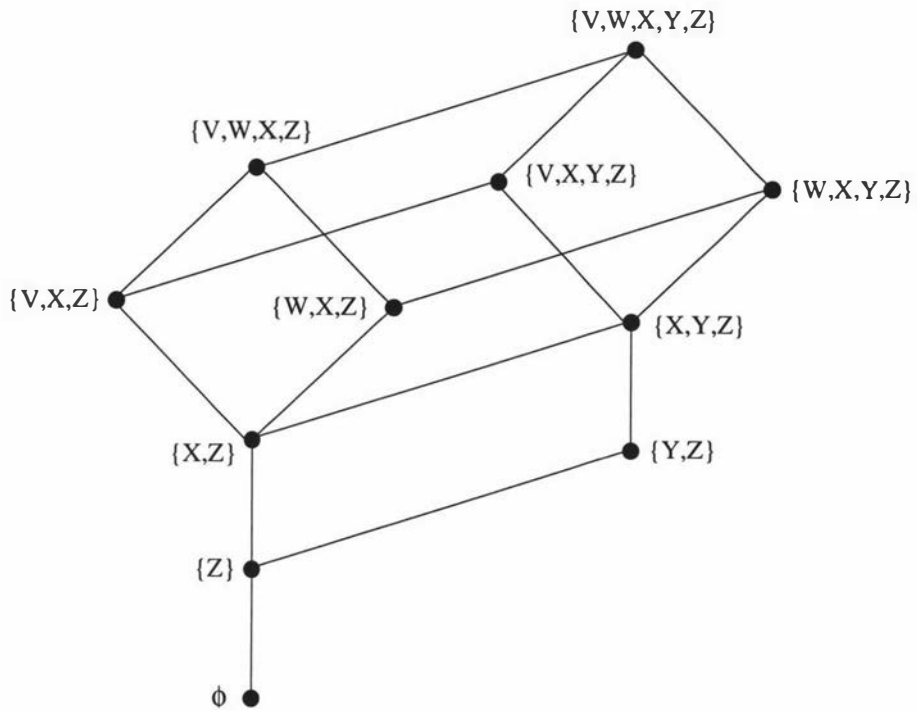

Fig. 2.4. Brouwerian algebra of closed subsets of PO-space on $(J, \leq)$.

$T = K\{L(M[N(C)])\}$. The pseudo-difference $S \dot{-} T$ equals $K\{L(M[N(B)])\}$. If ϑ denotes the isomorphism of Theorem 2.11, then $\vartheta(S) = \{V, W, X, Z\}$ and $\vartheta(T) = \{W, X, Y, Z\}$. Consequently, $\vartheta(S) \dot{-}_{\mathcal{C}} \vartheta(T) = \mathfrak{C}\{V\} = \{V, X, Z\}$, and $\vartheta^{-1}(\{V, X, Z\}) = K\{L(M[N(B)])\}$.

We conclude with a further example. The Brouwerian algebra for $K\{M(O\{A\}, P\{B\})\}$ is illustrated in Figure 2.5.

Finally, some intuitive information about the null attribute λ shall be given. Generally, it can be interpreted as: some information exists, but is currently not of interest. Take for example the attribute

Postcard(Person(First,Last),Address(Street,Town,Zip,Country)),

The subattribute Postcard(Person(λ,Last),Address(Street,Town,λ,Country)) neglects the information about the first name and about the ZIP code of the town. Elements of the corresponding domain contain the value *ok* on those attributes which indicates that some information exists but is not made explicit.

We will now look at the intuitive interpretations of the null attribute in presence of the various constructors. In case of the record type, the null attribute maintains the structural information of the fixed nested attribute. If we take again Postcard(Person(First,Last),Address(Street,Town,Zip,Country)), then the corresponding bot-

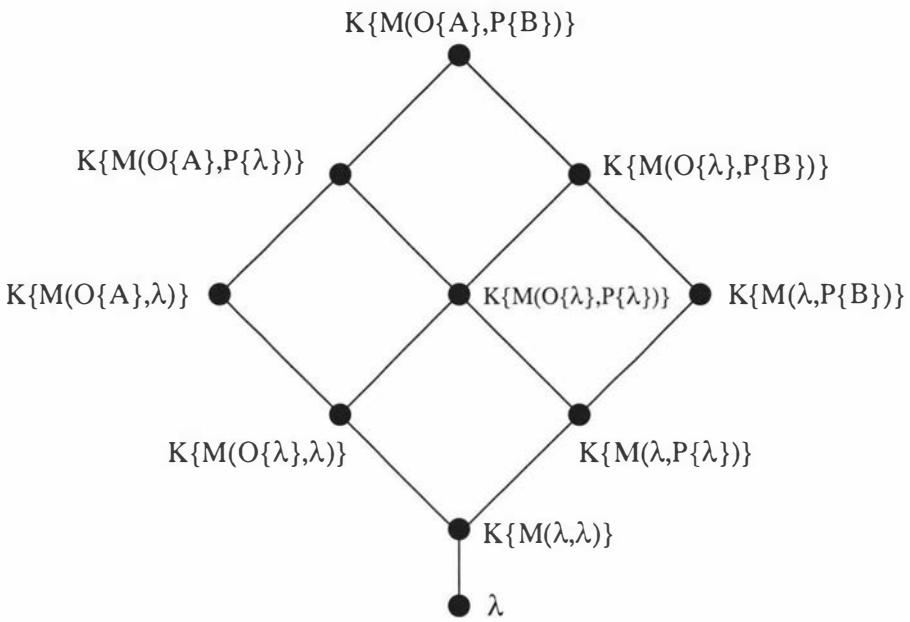

Fig. 2.5. The Brouwerian algebra of $K\{M(O\{A\}, P\{B\})\}$

tom element is Postcard(Person(λ,λ),Address(λ,λ,λ)) and the corresponding element of its domain is $((ok, ok), (ok, ok, ok))$.

In case of the set type, the subattribute $K\{\lambda\}$ of $K\{A\}$ indicates whether a set is inhabited or uninhabited, i.e. non-empty or empty. Suppose we take the nested attribute

Person(Name,Speaks{Foreign-Language})

to store the set of spoken languages with every person, then the subattribute Person(Name,Speaks{λ}) still indicates whether a person speaks a foreign language or not. Take for instance the elements (Bernhard,{English, Russian}) and (John,$\emptyset$). Their projections on Person(Name,Speaks{λ}) are (Bernhard,{ok}) and (John,$\emptyset$), respectively. One can conclude that Bernhard speaks at least one foreign language, but John does not speak any foreign language at all. This interpretation is due to the fact that sets do not have duplicates.

The interpretation of the null attribute changes in case of list and multiset type. Here, the null attribute actually counts the elements in the list or multiset. Take for instance the nested attribute

Person(Name,Speaks[Foreign-Language])

to store a list of foreign languages spoken by a person. The languages may be ordered according to the abilities of its speaker. The projections of (Bernhard,[Russian,English]) and (John,[ ]) on Person(Name,Speaks[λ]) are (Bernhard,[ok,ok]) and (John,[ ]), respectively. One can conclude that Bernhard speaks two foreign languages (assuming that no language had been listed twice before) and John does not speak any foreign language at all. This interpretation is due to the fact that lists and multisets allow the duplication of data.

2.3 Formalisation of Real-World Examples

In this section an illustration shall be given of how to formalise informal descriptions of real-world situations using nested attributes. For this purpose each of the examples from Section 1.2.2 is considered in turn.

EXAMPLE 2.2. Consider the simplified GenBank record format from Subsection 1.2.2. We use Origin[Base] to represent the original nucleotide sequence, Count(A,C,G,T) to represent the total numbers of adenine, cytosine, guanine, and thymine, respectively, Sub[Nucleo] to represent the subsequence of the original nucleotide sequence, Start and End to denote start and end position of the subsequence, and finally Translation[Amino] to represent the translated amino acid sequence. This results in the following nested attribute:

DNA(Origin[Base],Count(A,C,G,T),Gene(Start,End,Sub[Nucleo],Translation[Amino]))

which is generated from flat attributes using record and list constructor only. □

EXAMPLE 2.3. Consider the example of the comparison of two nucleotide sequences with specific characteristics from Subsection 1.2.2. We use Sti[Seq$i$] to capture the start of nucleotide sequence Seq$i$ and Num$i$(Occ$i$,Nuc$i$) gives us the number of occurrences Occ$i$ of nucleotide base Nuc$i$ in the nucleotide sequence Seq$i$ for $i = 1, 2$. Finally, Comp[Pair(N1,N2)] is the list in which the pair (N1,N2) is stored at position k, whenever Ni is the nucleotide in position k of nucleotide sequence Seq$i$. The nested attribute

Align(St1[Seq1],St2[Seq2],Num1(Occ1,Nuc1),Num2(Occ2,Nuc2),Comp[Pair(N1,N2)])

is again generated from flat attributes using record and list constructor only. □

EXAMPLE 2.4. Consider the example of halftoning from Subsection 1.2.2. We use Input[Level] to represent the list of the input region, Output[Bit] to denote the list of the output region and Brightness to describe the overall brightness of the input region. This results in the nested attribute

Halftoning(Brightness,Input[Level],Output[Bit])

which is again generated from flat attributes using record and list constructor only. □

EXAMPLE 2.5. Consider the example of a retailer from Subsection 1.2.2. In order to capture database instances of this retailer the following nested attribute might be used as a schema. An order can be described by

Order(Cart⟨Article(Title,Description,Price)⟩,Customer(Name,Address,Payment),SubTotal)

in which a cart is used to collect a multiset of articles, and SubTotal is used to denote the total value of the order. In what follows, we will use the label Order to abbreviate the nested attribute above. The final nested attribute itself may look as follows

Sales(Day,List[Order],Sold{Product(Item,CustName)},Total,NOrd,NProd,NShip).

Product(Item,CustName) denotes an item together with the name of the customer who bought it. Total denotes the total value of sales, NOrd the total number of orders, NProd the total number of products and NShip the total number of shippings. □

2.4 Brouwerian algebras in the Literature

Some references are given to other fields which Brouwerian algebras have been applied to. Brouwerian algebras have been studied in the context of topology by Stone, McKinsey and Tarski [201, 202, 255, 263]. McKinsey and Tarski use Brouwerian algebras to study closed elements in closure algebras [202] extending previously-established results from [201].

Lawvere in [175, 176] pointed out the role that the Brouwerian complement plays in grasping the geometrical notion of *boundary* as well as the physical concepts of *sub-body* and *essential core of a body*. In [175] it is claimed that a part a may be considered a sub-body if and only if $\neg\neg a = a$, i.e. a is a *regular element* according to [202]. Lawvere points out that the notion of boundary is definable by means of the Brouwerian complement $\neg$ in the following manner:

$$\delta(a) = a \sqcap \neg a.$$

Moreover, Lawvere notices that any element a in a Brouwerian algebra is the join of its core and its boundary: $a = \neg\neg a \sqcup \delta(a)$.

Figure 2.6 is used in [194] to comment on the mathematical nature of physics.

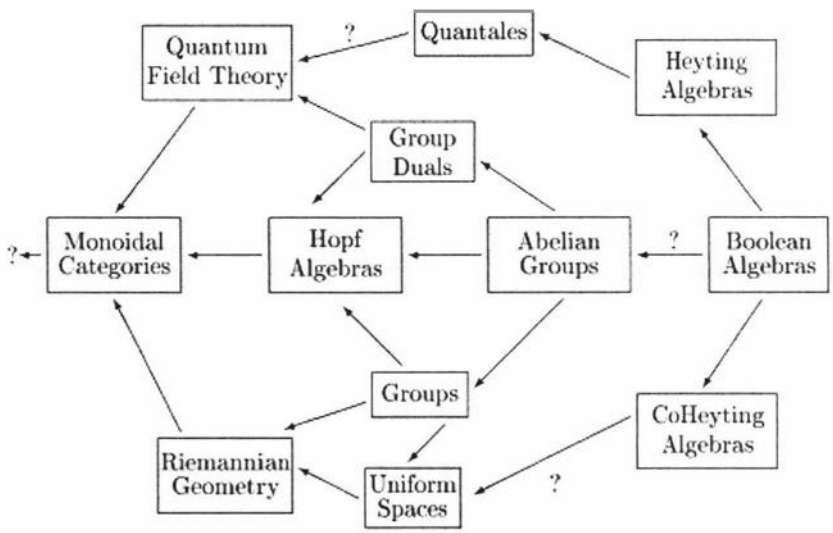

Fig. 2.6. Mathematical Concepts and Physics.

The following paragraph of [194] comments on the possible future roles of Brouwerian algebras in theoretical physics. "We take the view that the simplest theories of physics are based on classical logic or, roughly speaking, Boolean algebras. It appears that the relevant duality here may be provided by complementation with Boolean algebras considered self-dual according to De Morgan's theorem. The situation is summarised on the right in the figure. Going above the axis to Heyting algebras and beyond takes us into intuitionistic logic and ultimately into an axiomatic framework for quantum field theory. A Heyting

algebra describes logic in which one drops the familiar 'law of the excluded middle' that either a proposition or its negation is true. This generalisation is also the essential feature of the logical structure of quantum mechanics. Dual to this is the notion of co-Heyting algebra and co-intuitionistic logic in which one drops the axiom that the intersection of a proposition and its negation is empty. It has been argued by F.W. Lawvere and his school that this intersection is like the 'boundary' of the proposition, and, hence, that these co-Heyting algebras are the 'birth' of geometry. The long-term programme at this end of physics is to develop this geometrical interpretation of co-intuitionistic logic further into the notion of metric spaces and ultimately into Riemannian or Lorentzian geometry."

Various papers on dual-intuitionistic logic have been written. Besides the work of McKinsey and Tarski in [201, 202], Curry [78] presents what he called *absolute implicational lattices* and *absolute subtractive lattices*. Rauszer [230, 231] uses algebraic, Hilbert-style and relational methods to study *intuitionistic logic with dual operators* where the connective of pseudo-difference is the dual to intuitionistic implication. Czermak [79] investigates dual intuitionistic logic by restricting Gentzen's sequent calculus **LK** to *singletons on the left* which is the natural dual notion to Gentzen's *singletons on the right* restriction for the sequent calculus **LJ**. Goodman [129] uses Brouwerian algebras to investigate logic of contradictions. Goodman mentions that Kripke semantics also exist in which "any formula, once false, remains false", but he fails to give the crucial clause for satisfiability for his pseudo-difference connective. He also gives a sequent calculus for his logic but does not investigate cut-elimination. Urbas [275] highlights several deficiencies of Goodman's analysis and defines several Gentzen calculi with the *singletons on the left* restriction, but adds rules for incorporating both implication and its dual connective. The works in [77, 130, 298] deliver display calculi for dual intuitionistic logic and some of its extensions.

Pagliani [218] argues that rough set analysis adequately and elegantly grasps the notion of *boundary* in incomplete information analysis via the algebraic features provided by Brouwerian algebras.

The work in [253] shows that Heyting algebras, Brouwerian algebras and Bi-Heyting algebras have considerable potential for building the theoretical basis of ontologies for spatial entities in spatial information theory. Ganascia points out in [120] that Brouwerian algebras are useful for generalising and comparing many machine learning algorithms. The paper [232] studies algebraic properties in the context of program integration. Therefore, a program-integration algorithm is reformulated as an operation in a Brouwerian algebra constructed from sets of a program dependence graph. An application of Brouwerian logic to medical diagnosis can be found in [241]. Another application of closure algebras is proposed in [249] where a systematic approach to maintaining geometric representations is developed.

Chapter 3

Functional Dependencies in the Presence of Lists

In this chapter we will investigate the impact of the list constructor on the perhaps most common class of relational dependencies. Functional dependencies are introduced in the context of null, flat, record-valued and list-valued attributes. The goal of this chapter is to extend the solution to such problems as axiomatisation, implication, normalisation and its justification and decomposition algorithms for the class of FDs from the RDM to the presence of lists.

It turns out that Armstrong's original axioms, when generalised to the presence of lists, are still sound. We will prove that these generalised Armstrong's axioms are also complete.

Based on this axiomatisation we investigate the implication problem for the class of FDs in the presence of lists. Therefore, an alternative view of FDs is proposed first, which is based on the representation theorem for Brouwerian algebras. The main result is a linear time, provably-correct algorithm for deciding implication.

Finally, we turn to normalisation issues in the presence of lists. The Nested List normal form (NLNF) is proposed as a normal form for nested attributes and the class of FDs in the presence of lists. NLNF is strictly weaker than a simple extension of Boyce-Codd normal form. The key reason is an argument that non-maximal join-irreducible subattributes do not cause redundancies. NLNF is characterised and thereby justified in many ways, for instance by the absence of redundancies, simpler integrity checking and the absence of many forms of update anomalies. The results generalise well-known results from the relational data model, thus giving a formal semantic justification for the normal form proposal.

To conclude, a lossless decomposition of nested attributes into subattributes in NLNF is considered. A provably-correct algorithm is proposed that works, in general, in exponential time in the size of the underlying nested attribute and the number of FDs given.

Some results of this chapter can be found in the literature. The axiomatisation of FDs appears in [146], and the Nested List normal form proposal and its semantic justification are published in [143].

3.1 Axiomatisation

We define FDs, introduce a generalisation of Armstrong's axioms and prove that these rules are sound and complete for the implication of FDs in the presence of lists.

3.1.1 Definition of FDs

Given that a nested attribute N generalises the notion of a relation schema, the subattribute relationship extends the subset relationship and the projection function subsumes the restriction of tuples, we obtain the following definition.

Definition 3.1. Let $N \in \mathcal{N}\mathrm{A}$ be a nested attribute. A *functional dependency on* N is an expression of the form $X \to Y$ where $X, Y \in Sub(N)$. A set $r \subseteq dom(N)$ *satisfies* the functional dependency $X \to Y$ on N, denoted by $\models_r X \to Y$, if and only if $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ whenever $\pi^N_X(t_1) = \pi^N_X(t_2)$ for any $t_1, t_2 \in r$ holds. □

Given the relation schema $R = \{A, B, C\}$ the FD $A \to B$ on R can be read as the FD $R(A, \lambda, \lambda) \to R(\lambda, B, \lambda)$ on the record-valued attribute $R(A, B, C)$. We consider the following examples to become more acquainted with FDs defined on nested attributes.

EXAMPLE 3.1. Suppose the nested attribute Pubcrawl(Person,Visit[Drink(Beer,Pub)]) is used to store sequences of beers consumed by a person together with the pub in which the person had that beer. A snapshot r of such a database may look as follows:

{ (Sven, [(Lübzer, Deanos), (Kindl, Highflyers)]),
(Sven, [(Kindl, Deanos), (Lübzer, Highflyers)]),
(Klaus-Dieter, [(Guiness, Irish Pub), (Speights, 3Bar),(Guiness, Irish Pub)]),
(Klaus-Dieter, [(Kölsch, Irish Pub), (Bönnsch, 3Bar), (Guiness, Irish Pub)]),
(Sebastian, [ ]) } .

It is then obvious that $\models_r$ Pubcrawl(Person) → Pubcrawl(Visit[Drink(Pub)]) holds. This means that in this particular snapshot, the same person always visits the same pubs in the same order. □

EXAMPLE 3.2. Suppose we have a database for storing the prime factorisation of positive integers n. The nested attribute

Factor(Integer,Prime[Number],Exponent[Number])

would be suitable where Prime[Number] is the list of different prime factors of n in increasing order, and Exponent[Number] the list of exponents for each corresponding prime factor in Prime[Number]. A small snapshot of the database could be

{ (12,[2,3],[2,1]),
(35,[5,7],[1,1]),
(37,[37],[1]),
(936,[2,3,13],[3,2,1]) } .

The fundamental theorem of number theory states that every positive integer has a unique prime factorisation. This means that the nested attribute Factor(Integer,Prime[Number],Exponent[Number]) carries the following semantic information in terms of functional dependencies:

- Factor(Integer) $\rightarrow$ Factor(Prime[Number], Exponent[Number]),
- Factor(Prime[Number], Exponent[Number]) $\rightarrow$ Factor(Integer).

Further examples of FDs which every snapshot of this database satisfies are

- Factor(Prime[λ]) $\rightarrow$ Factor(Exponent[λ]) and
- Factor(Exponent[λ]) $\rightarrow$ Factor(Prime[λ]).

Informally, they state that the number of different prime factors determines the number of exponents, and vice versa. □

EXAMPLE 3.3. Consider the nested attribute N for the GenBank in Example 2.2. The set Σ of FDs that were informally described in Section 1.2.2 can now be specified formally as follows:

1. DNA(Origin[Base]) $\rightarrow$ DNA(Count(A,C,G,T)),
2. DNA(Count(A,C,G,T)) $\rightarrow$ DNA(Origin[λ]),
3. DNA(Origin[λ],Count(A,C,G)) $\rightarrow$ DNA(Count(T)),
   DNA(Origin[λ],Count(A,C,T)) $\rightarrow$ DNA(Count(G)),
   DNA(Origin[λ],Count(A,G,T)) $\rightarrow$ DNA(Count(C)),
   DNA(Origin[λ],Count(C,G,T)) $\rightarrow$ DNA(Count(A)),
4. DNA(Origin[Base],Gene(Start,End)) $\rightarrow$ DNA(Gene(Sub[Nucleo])),
5. DNA(Gene(Sub[Nucleo])) $\rightarrow$ DNA(Gene(Translation[Amino])),
6. DNA(Gene(Sub[λ])) $\rightarrow$ DNA(Gene(Translation[λ])),
   DNA(Gene(Translation[λ])) $\rightarrow$ DNA(Gene(Sub[λ]))
7. DNA(Gene(Start,Sub[λ])) $\rightarrow$ DNA(Gene(End)),
8. DNA(Gene(End,Sub[λ])) $\rightarrow$ DNA(Gene(Start)),
9. DNA(Gene(Start,End)) $\rightarrow$ DNA(Gene(Sub[λ])). □

3.1.2 Implication and Derivation

We will use this section to introduce the notions of semantic implication and syntactical derivation for classes of data dependencies (and with respect to a given set of inference rules). In what follows, $\mathcal{C}$ denotes a certain class of dependencies, for example functional dependencies in the presence of null, flat, record- and list-valued attributes.

Definition 3.2. Let N be some nested attribute, Σ be a finite set of dependencies in $\mathcal{C}$ whose elements are all defined on N, and τ a dependency of class $\mathcal{C}$ on N. We say that τ is implied by Σ (Σ implies τ, or τ follows from Σ), denoted by $\Sigma \models \tau$, if and only if every $r \subseteq dom(N)$ satisfying all $\sigma \in \Sigma$ also satisfies τ. We say that τ is implied by Σ (Σ implies τ, or τ follows from Σ) in the finite sense, denoted by $\Sigma \models_f \tau$, if and only if every finite $r \subseteq dom(N)$ satisfying all $\sigma \in \Sigma$ also satisfies τ. The *(finite) semantic hull* $\Sigma^*_{\mathcal{C}}$ ($\Sigma^*_{\mathrm{fin},\mathcal{C}}$) of

Σ in $\mathcal{C}$ on N is the set of all dependencies τ in $\mathcal{C}$ on N implied by Σ (in the finite sense), i.e. $\Sigma^*_{\mathcal{C}} = \{\tau \in \mathcal{C} \mid \Sigma \models \tau\}$ ($\Sigma^*_{\text{fin},\mathcal{C}} = \{\tau \in \mathcal{C} \mid \Sigma \models_f \tau\}$). □

In order to capture the semantic notion of (finite) implication syntactically, one is interested in the notion of inference using certain inference rules.

Definition 3.3. An *inference rule* consists of a finite set $\mathfrak{P} = \{\varphi_1, \ldots, \varphi_n\}$ of parameterised dependencies, another non-empty, finite set $\mathfrak{C} = \{\psi_1, \ldots, \psi_m\}$ of parameterised dependencies and a finite set $\mathcal{C}on = \{C_1, \ldots, C_k\}$ of constraints on the parameters in $\mathfrak{P}$ and $\mathfrak{C}$. The φ_i $(i = 1, \ldots, n)$ are called the *premises* of the rule; the ψ_i $(i = 1, \ldots, m)$ are called the *conclusions* of the rule. An inference rule with no premises ($\mathcal{P} = \emptyset$) is called an *axiom*. The notation

$$\frac{\varphi_1, \ldots, \varphi_n}{\psi_1, \ldots, \psi_m} C_1, \ldots, C_k$$

is used to denote inference rules. □

Given some inference rule $\frac{\varphi'_1, \ldots, \varphi'_n}{\psi'_1, \ldots, \psi'_m} C_1, \ldots, C_k$ the intention is to formalise derivation. Whenever we have dependencies $\varphi_1, \ldots \varphi_n$ arising from the premises $\varphi'_1, \ldots, \varphi'_n$ by substituting the parameters, then we can derive the dependencies $\psi_1, \ldots, \psi_m$ which result from the conclusions $\psi'_1, \ldots, \psi'_m$ by the same substitution provided all the conditions $C_1, \ldots, C_k$ are satisfied. In such a case we speak of an instantiation $\frac{\varphi_1, \ldots, \varphi_n}{\psi_1, \ldots, \psi_m}$ of this inference rule.

Definition 3.4. Let $\mathfrak{R}$ be a set of inference rules and let Σ be a set of dependencies in $\mathcal{C}$ whose elements are all defined on the nested attribute N. A *derivation tree* over $\mathfrak{R}$ and Σ is a directed tree satisfying the following conditions:

- each node in the tree has an attached dependency in $\mathcal{C}$ that is defined on N;
- whenever a node with attached dependency ψ has successor nodes with attached dependencies $\varphi_1, \ldots, \varphi_n$, then
  - either there exists an instantiation $\frac{\varphi_1, \ldots, \varphi_n}{\psi_1, \ldots, \psi_m}$ of a rule $\frac{\varphi'_1, \ldots, \varphi'_n}{\psi'_1, \ldots, \psi'_m} C_1, \ldots, C_k$ in $\mathfrak{R}$ such that $\psi_i = \psi$ holds for some i
  - or the node is a leaf and $\psi \in \Sigma$ holds. □

Examples of derivation trees can be found in the proof of Lemma 3.11 on page 58. Please note that these derivation trees really "grow" from bottom to top. Instead of drawing edges between the parent node and every of its successors, we will draw one single horizontal line between all the successor nodes and the parent node.

We are now prepared to define the derivation of dependencies from a given set Σ of dependencies using a particular underlying set of inference rules.

Definition 3.5. Let $\mathfrak{R}$ be a set of inference rules and let Σ be a set of dependencies in $\mathcal{C}$ whose elements are defined on the nested attribute N. A dependency τ is *derivable* from Σ using $\mathfrak{R}$ if and only if there exists a derivation tree over $\mathfrak{R}$ and Σ with root τ (notation: $\Sigma \vdash_{\mathfrak{R}} \tau$). The *syntactic hull* $\Sigma^+_{\mathfrak{R}}$ of Σ under $\mathfrak{R}$ is the set of all dependencies that are derivable from Σ using $\mathfrak{R}$, i.e., $\Sigma^+_{\mathfrak{R}} = \{\tau \mid \Sigma \vdash_{\mathfrak{R}} \tau\}$. □

One is interested in meaningful sets $\mathfrak{R}$ of inference rules for deriving dependencies. That is, every dependency that is derivable from Σ using $\mathfrak{R}$ should also be implied by Σ. In order to capture the semantic notion of implication by the syntactical notion of inference the set $\mathfrak{R}$ must have a further property. Every dependency implied by Σ must also be derivable from Σ by using only inference rules from $\mathfrak{R}$.

Definition 3.6. A set $\mathfrak{R}$ of inference rules is called *sound for the (finite) implication of dependencies in* $\mathcal{C}$ if and only if for every nested attribute N and for every set Σ of dependencies in $\mathcal{C}$ on N we have $\Sigma^+_{\mathfrak{R}} \subseteq \Sigma^*_{\mathcal{C}}$ ($\Sigma^+_{\mathfrak{R}} \subseteq \Sigma^*_{\mathrm{fin},\mathcal{C}}$). A set $\mathfrak{R}$ of inference rules is called *complete for the (finite) implication of dependencies in* $\mathcal{C}$ if and only if for every nested attribute N and for every set Σ of dependencies in $\mathcal{C}$ on N we have $\Sigma^*_{\mathcal{C}} \subseteq \Sigma^+_{\mathfrak{R}}$ ($\Sigma^*_{\mathrm{fin},\mathcal{C}} \subseteq \Sigma^+_{\mathfrak{R}}$). The class $\mathcal{C}$ is called *(finitely) axiomatisable* if and only if there is a (finite) sound and complete set of inference rules for the implication of dependencies in $\mathcal{C}$. □

Sound and complete sets of inference rules for the implication of dependencies in $\mathcal{C}$ are sometimes called $\mathcal{C}$-sound or $\mathcal{C}$-complete. A further interesting question deals with the independence of inference rules.

Definition 3.7. Let $\mathfrak{R}$ denote some set of inference rules. An inference rule R is *$\mathcal{C}$-independent from* $\mathfrak{R}$ if and only if there is a nested attribute N and a set Σ of dependencies in $\mathcal{C}$ on N as well as some dependency σ with $\sigma \notin \Sigma^+_{\mathfrak{R}}$ but $\sigma \in \Sigma^+_{\mathfrak{R}\cup\{R\}}$. A $\mathcal{C}$-sound and $\mathcal{C}$-complete set $\mathfrak{R}$ of inference rules is called *minimal for the implication of dependencies in* $\mathcal{C}$ if and only if every $R \in \mathfrak{R}$ is $\mathcal{C}$-independent from $\mathfrak{R} - \{R\}$, i.e., there is no $\mathfrak{R}' \subset \mathfrak{R}$ which is $\mathcal{C}$-complete as well. □

Strictly speaking, the notion of minimality should also take the set $\mathcal{C}on$ of constraints of every inference rule into consideration. It may well be that all the rules are independent from one another but some constraint in $\mathcal{C}on$ can still be weakened.

EXAMPLE 3.4. The following set of inference rules is sound and complete for the implication of FDs in the RDM.

$$\frac{}{X \to Y}\, Y \subseteq X, \qquad \frac{X \to Y}{XW \to YV}\, V \subseteq W, \qquad \frac{X \to Y, Y \to Z}{X \to Z}$$

None of the rules can be omitted without losing completeness. However, the reflexivity axiom $\frac{}{X \to Y}\, Y \subseteq X$ can be replaced by the weaker axiom $\frac{}{\emptyset \to \emptyset}$ as the second inference rule allows to derive the reflexivity axiom from the weaker axiom. □

However, in this thesis we will focus on the notion of minimality as introduced in Definition 3.7, and study the stricter version in the future.

Usually, once a class $\mathcal{C}$ has been fixed, the index $\mathcal{C}$ is dropped from $\Sigma_{\mathcal{C}}^{+}$, $\Sigma_{\mathcal{C}}^{*}$ and $\Sigma_{\text{fin},\mathcal{C}}^{*}$. In this chapter, we will consider the class $\mathcal{C}$ of functional dependencies in the presence of null, flat, record- and list-valued attributes. The index $\mathfrak{R}$ is dropped from $\Sigma_{\mathfrak{R}}^{+}$ once the set of inference rules has been fixed.

3.1.3 The generalised Armstrong Axioms

Let Σ be a set of FDs, and σ an FD, all defined on some nested attribute N. Real life databases are inherently finite. Therefore, our attention should be firstly directed towards the finite implication problem $\Sigma \models_f \sigma$. However, in the case of FDs the finite implication problem coincides with the unrestricted implication problem $\Sigma \models \sigma$. Interpreting $\models_{(f)}$ as relations, it is immediate that $\models \subseteq \models_f$ holds. If there is an infinite $r \subseteq dom(N)$ with $\models_r \Sigma$ and $\not\models_r \sigma$, i.e., $(\Sigma, \sigma) \notin \models$, then there are $t_1, t_2 \in r$ with $\not\models_{\{t_1,t_2\}} \sigma$. However, $\models_{\{t_1,t_2\}} \Sigma$ follows directly from $\models_r \Sigma$, and thus $(\Sigma, \sigma) \notin \models_f$. This shows that also $\models_f \subseteq \models$ holds, i.e., unrestricted and finite implication coincide for the class $\mathcal{C}$ of FDs. Consequently, $\Sigma^* = \Sigma_{\text{fin}}^*$ for any set Σ of FDs defined on any nested attribute. Next, we introduce extensions of Armstrong's axioms to the presence of records and lists.

Definition 3.8. The *generalised Armstrong axioms for functional dependencies* are

$$\frac{}{X \to Y}\, Y \leq X, \qquad \frac{X \to Y}{X \to X \sqcup_N Y}, \qquad \frac{X \to Y,\ Y \to Z}{X \to Z}.$$

These rules are called the *reflexivity axiom*, the *extension rule* and the *transitivity rule.* □

In what follows $\mathfrak{R}$ denotes the generalised Armstrong axioms for FDs. Consider the following example as an illustration for some inferences of FDs.

EXAMPLE 3.5. Consider Example 3.2 again. The reflexivity axiom tells us that

Factor(Integer,Prime[Number],Exponent[Number])

carries FDs such as

Factor(Prime[Number]) → Factor(Prime[λ]).

The extension rule allows to infer the FD

Factor(Integer) → Factor(Integer, Prime[Number], Exponent[Number])

from

Factor(Integer) → Factor(Prime[Number], Exponent[Number]).

From

Factor(Integer) → Factor(Prime[Number]) and
Factor(Prime[Number]) → Factor(Prime[λ])

we infer the FD

$$\text{Factor(Integer)} \rightarrow \text{Factor(Prime}[\lambda])$$

using the transitivity rule. □

We show the fundamental fact that, in the presence of lists, the projection of some $t \in dom(N)$ on two subattributes X and Y of N determines its projection on $X \sqcup Y$.

Lemma 3.9. *Let $N \in \mathcal{NA}$, $X, Y \in Sub(N)$ and $t_1, t_2 \in dom(N)$. If $\pi^N_X(t_1) = \pi^N_X(t_2)$ and $\pi^N_Y(t_1) = \pi^N_Y(t_2)$, then $\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$.*

Proof. We proceed by induction on the structure of N.

If $X \leq Y$, then

$$\pi^N_{X \sqcup Y}(t_1) = \pi^N_Y(t_1) = \pi^N_Y(t_2) = \pi^N_{X \sqcup Y}(t_2)$$

and similar in the case when $Y \leq X$ holds. This covers the lemma for the cases where N is the null attribute λ or a flat attribute.

Let $N = L(N_1, \ldots, N_k)$. The hypothesis says for every $i = 1, \ldots, k$ that for all $X_i, Y_i \in Sub(N_i)$ and all $t_1^i, t_2^i \in dom(N_i)$ such that $\pi^{N_i}_{X_i}(t_1^i) = \pi^{N_i}_{X_i}(t_2^i)$ and $\pi^{N_i}_{Y_i}(t_1^i) = \pi^{N_i}_{Y_i}(t_2^i)$, we also have $\pi^{N_i}_{X_i \sqcup Y_i}(t_1^i) = \pi^{N_i}_{X_i \sqcup Y_i}(t_2^i)$. From $N = L(N_1, \ldots, N_k)$ and $X, Y \in Sub(N)$ follows $X = L(X_1, \ldots, X_k)$ and $Y = L(Y_1, \ldots, Y_k)$. Since $t_1, t_2 \in dom(N)$ it follows that $t_1 = (t_1^1, \ldots, t_1^k)$ and $t_2 = (t_2^1, \ldots, t_2^k)$ with $t_1^i, t_2^i \in dom(N_i)$ for $i = 1, \ldots, k$. From $\pi^N_X(t_1) = \pi^N_X(t_2)$ and $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ follow $\pi^{N_i}_{X_i}(t_1^i) = \pi^{N_i}_{X_i}(t_2^i)$ and $\pi^{N_i}_{Y_i}(t_1^i) = \pi^{N_i}_{Y_i}(t_2^i)$ for $i = 1, \ldots, k$. We conclude by hypothesis that $\pi^{N_i}_{X_i \sqcup Y_i}(t_1^i) = \pi^{N_i}_{X_i \sqcup Y_i}(t_2^i)$ holds for $i = 1, \ldots, k$. Then we have

$$\begin{aligned}
\pi^N_{X \sqcup Y}(t_1) &= (\pi^{N_1}_{X_1 \sqcup Y_1}(t_1^1), \ldots, \pi^{N_k}_{X_k \sqcup Y_k}(t_1^k)) \\
&= (\pi^{N_1}_{X_1 \sqcup Y_1}(t_2^1), \ldots, \pi^{N_k}_{X_k \sqcup Y_k}(t_2^k)) \\
&= \pi^N_{X \sqcup Y}(t_2).
\end{aligned}$$

It remains to consider the case where $N = L[N']$. The hypothesis tells us that for all $X', Y' \in Sub(N')$ and all $a, a' \in dom(N')$ such that $\pi^{N'}_{X'}(a) = \pi^{N'}_{X'}(a')$ and $\pi^{N'}_{Y'}(a) = \pi^{N'}_{Y'}(a')$, we also have $\pi^{N'}_{X' \sqcup Y'}(a) = \pi^{N'}_{X' \sqcup Y'}(a')$. It remains to consider the case where $X = L[X']$ and $Y = L[Y']$. Since $t_1, t_2 \in dom(N)$ it follows that $t_1 = [a_1, \ldots, a_n]$ and $t_2 = [a'_1, \ldots, a'_l]$ with $a_i, a'_j \in dom(N')$ for $i = 1, \ldots, n$ and $j = 1, \ldots, l$. From $\pi^N_X(t_1) = \pi^N_X(t_2)$ and $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ follow $n = l$ and $\pi^{N'}_{X'}(a_i) = \pi^{N'}_{X'}(a'_i)$ and $\pi^{N'}_{Y'}(a_i) = \pi^{N'}_{Y'}(a'_i)$ for $i = 1, \ldots, n$. We conclude by hypothesis that also $\pi^{N'}_{X' \sqcup Y'}(a_i) = \pi^{N'}_{X' \sqcup Y'}(a'_i)$ hold for $i = 1, \ldots, n$. Then we have

$$\begin{aligned}
\pi^N_{X \sqcup Y}(t_1) &= [\pi^{N'}_{X' \sqcup Y'}(a_1), \ldots, \pi^{N'}_{X' \sqcup Y'}(a_n)] \\
&= [\pi^{N'}_{X' \sqcup Y'}(a'_1), \ldots, \pi^{N'}_{X' \sqcup Y'}(a'_n)] \\
&= \pi^N_{X \sqcup Y}(t_2).
\end{aligned}$$

This concludes the proof of this lemma. □

We will now argue that the Armstrong axioms are sound, i.e., everything that is derivable from Σ by using any of these rules is also implied by Σ.

Proposition 3.10. *The generalised Armstrong axioms are sound for the implication of functional dependencies in the presence of records and lists.*

Proof. For the soundness of the *reflexivity rule* take $X, Y \in Sub(N)$ with $Y \leq X$ and let $r \subseteq dom(N)$. Let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. Recall that for $Y \leq X$ follows $\pi^N_Y = \pi^X_Y \circ \pi^N_X$ where $\circ$ denotes the composition of functions. We conclude that $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ holds as well, and therefore $\models_r X \to Y$.

For the soundness of the *extension rule* assume $X, Y \in Sub(N)$ and let $r \subseteq dom(N)$ with $\models_r X \to Y$. Let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. Since $\models_r X \to Y$ we also know that $\pi^N_Y(t_1) = \pi^N_Y(t_2)$. Lemma 3.9 shows that $\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$. This means $\models_r X \to X \sqcup Y$ and concludes the proof for the soundness of the extension rule.

For the soundness of the *transitivity rule* take $X, Y, Z \in Sub(N)$ and let $r \subseteq dom(N)$ with $\models_r X \to Y$ and $\models_r Y \to Z$. Let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. From $\models_r X \to Y$ follows $\pi^N_Y(t_1) = \pi^N_Y(t_2)$, and applying $\models_r Y \to Z$ shows $\pi^N_Z(t_1) = \pi^N_Z(t_2)$. Consequently, $\models_r X \to Z$. □

Next, we derive some more sound inference rules for FDs. They will enable faster inferences.

Lemma 3.11. *The following inference rules are derivable from the generalised Armstrong axioms, and hence are sound:*

$$\frac{X \to Y, X \to Z}{X \to Y \sqcup Z} \quad \textit{(join rule)} \qquad \frac{X \to Y, X \to Z}{X \to Y \sqcap Z} \quad \textit{(meet rule)}$$

$$\frac{X \to Y}{X \to Y \dot{-} X} \quad \textit{(reduction rule)} \qquad \frac{X \to Y}{X \to Z}\ Z \leq Y \quad \textit{(subattribute rule)}$$

$$\frac{X \to Y}{U \to V}\ \substack{X \leq U,\\ V \leq X \sqcup Y} \quad \textit{(general extension rule)} \qquad \frac{X \to Y, U \to V}{W \to Z}\ \substack{U \leq X \sqcup Y, X \leq W,\\ Z \leq V \sqcup W} \quad \textit{(general transitivity rule)}$$

Proof. Every application of an inference rule above can be replaced by an application of one of the following inference schemata which make use of the generalised Armstrong axioms only. This shows the soundness of each inference rule of this lemma.

join rule:

$$\dfrac{\dfrac{X \to Y}{X \to X \sqcup Y} \quad \dfrac{\dfrac{\dfrac{}{X \sqcup Y \to X}\, X \leq X \sqcup Y \quad X \to Z}{X \sqcup Y \to Z}}{X \sqcup Y \to X \sqcup Y \sqcup Z} \quad \dfrac{}{X \sqcup Y \sqcup Z \to Y \sqcup Z}}{X \sqcup Y \to Y \sqcup Z}}{X \to Y \sqcup Z}$$

meet rule:

$$\frac{X \to Y \qquad \overline{Y \to Y \sqcap Z}^{Y \sqcap Z \leq Y}}{X \to Y \sqcap Z}$$

reduction rule: In order to prove this rule we recall that $Y \dot{-} X \leq Y$ holds.

$$\frac{X \to Y \qquad \overline{Y \to Y \dot{-} X}^{Y \dot{-} X \leq Y}}{X \to Y \dot{-} X}$$

subattribute rule:

$$\frac{X \to Y \qquad \overline{Y \to Z}^{Z \leq Y}}{X \to Z}$$

general extension rule:

$$\frac{\dfrac{\overline{U \to X}^{X \leq U} \quad \dfrac{X \to Y}{X \to X \sqcup Y}}{U \to X \sqcup Y} \qquad \overline{X \sqcup Y \to V}^{V \leq X \sqcup Y}}{U \to V}$$

general transitivity rule:

$$\frac{\dfrac{\dfrac{\dfrac{\overline{W \to X}^{X \leq W} \quad \dfrac{X \to Y}{X \to X \sqcup Y}}{W \to X \sqcup Y} \qquad \overline{X \sqcup Y \to U}^{U \leq X \sqcup Y}}{W \to U} \qquad U \to V}{W \to V} \qquad \dfrac{\overline{W \to V \sqcup W} \qquad \overline{V \sqcup W \to Z}^{Z \leq V \sqcup W}}{W \to Z}$$

□

3.1.4 Completeness

The idea for the completeness proof follows the lines of the original proof for the RDM [15]. A two-element instance r with $\models_r \Sigma^*$ and $\not\models_r X \to Y$ is constructed for any $X \to Y \notin \Sigma^+$. The proof needs a bit more effort than its original but still remains constructive.

Lemma 3.12. *Let $N \in \mathcal{N}A$. There are $t_1, t_2 \in dom(N)$ such that $\pi_X^N(t_1) \neq \pi_X^N(t_2)$ holds on all X with $\lambda_N \neq X \leq N$.*

Proof. We prove this lemma by induction on N. For $N = \lambda$ there is nothing to show. If $N = A$ is a flat attribute, then the only $\lambda \neq X \leq N$ is $X = A$. In this case, $t_1 = a$ and $t_2 = a'$ with $a, a' \in dom(A)$ and $a \neq a'$ are chosen. If $N = L(N_1, \ldots, N_k)$, then there are $t_1^i, t_2^i \in dom(N_i)$ with $\pi_{M_i}^{N_i}(t_1^i) \neq \pi_{M_i}^{N_i}(t_2^i)$ on all $\lambda_{N_i} \neq X_i \leq N_i$ for all $i = 1, \ldots, k$. Define $t_1 = (t_1^1, \ldots, t_1^k), t_2 = (t_2^1, \ldots, t_2^k) \in dom(N)$. For $\lambda_N \neq X \leq N$ we have $X = L(X_1, \ldots, X_k)$

with $X_i \neq \lambda_{N_i}$ for some $i \in \{1, \ldots, k\}$. This implies that $\pi_X^N(t_1) = (\pi_{X_1}^{N_1}(t_1^1), \ldots, \pi_{X_k}^{N_k}(t_1^k)) \neq (\pi_{X_1}^{N_1}(t_2^1), \ldots, \pi_{X_k}^{N_k}(t_2^k)) = \pi_X^N(t_2)$. It remains to consider the case where $N = L[N']$. In this case we define $t_1 = [\,], t_2 = [n'] \in dom(N)$ with $n' \in dom(N')$. For $\lambda \neq X \leq N$ follows $X = L[X']$ with $X' \leq N'$ and $\pi_X^N(t_1) = [\,] \neq [\pi_{X'}^{N'}(n')] = \pi_X^N(t_2)$. This concludes the proof. □

EXAMPLE 3.6. Suppose $N = L(A, K[B], M[\lambda])$. In this case, $t_1 = (a, [\,], [\,])$ and $t_2 = (a', [b], [ok])$ with different $a, a' \in dom(A), b \in dom(B)$ have different projections on all subattributes of N different from $L(\lambda, \lambda, \lambda) = \lambda_N$. □

The following lemma shows how to find two elements of $dom(N)$ that have exactly coincident projections on an arbitrary, but fixed subattribute X of N.

Lemma 3.13. *Let $N \in \mathcal{NA}$. For each $X \in Sub(N)$ there are $t_1, t_2 \in dom(N)$ such that $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ holds for $Y \in Sub(N)$ if and only if $Y \leq X$.*

Proof. If $X = \lambda_N$, then we apply Lemma 3.12. We assume that $X \neq \lambda_N$ from now on and proceed by induction on N. If $N = A$ is a flat attribute, then $X = A$ and we define $t_1 = a = t_2$ with some $a \in dom(A)$. Consider the case where $N = L(N_1, \ldots, N_k)$ and let $X = L(X_1, \ldots, X_k)$ with $X_i \leq N_i$ for $i = 1, \ldots, k$. For $i = 1, \ldots, k$ there are $t_1^i, t_2^i \in dom(N_i)$ with $\pi_{Y_i}^{N_i}(t_1^i) = \pi_{Y_i}^{N_i}(t_2^i)$ if and only if $Y_i \leq X_i$ holds, by hypothesis. We define $t_1 = (t_1^1, \ldots, t_1^k)$ and $t_2 = (t_2^1, \ldots, t_2^k)$ with $t_1, t_2 \in dom(N)$. It follows that $Y \leq X$ if and only if $Y = L(Y_1, \ldots, Y_k)$ with $Y_i \leq X_i$ for $i = 1, \ldots, k$ if and only if $\pi_{Y_i}^{N_i}(t_1^i) = \pi_{Y_i}^{N_i}(t_2^i)$ for $i = 1, \ldots, k$ if and only if $\pi_Y^N(t_1) = \pi_Y^N(t_2)$. It remains to consider the case where $N = L[N']$. Consequently, $X = L[X']$ with $X' \leq N'$. Then there are some $t_1', t_2' \in dom(N')$ such that $\pi_{Y'}^{N'}(t_1') = \pi_{Y'}^{N'}(t_2')$ if and only if $Y' \leq X'$ by hypothesis. Defining $t_1 = [t_1'], t_2 = [t_2'] \in dom(N)$ we infer that $\lambda \neq Y \leq X$ if and only if $Y = L[Y']$ with $Y' \leq X'$ if and only if $\pi_{Y'}^{N'}(t_1') = \pi_{Y'}^{N'}(t_2')$ if and only if $[\pi_{Y'}^{N'}(t_1')] = [\pi_{Y'}^{N'}(t_2')]$ if and only if $\pi_Y^N(t_1) = \pi_Y^N(t_2)$. The case $Y = \lambda$ is trivial. □

EXAMPLE 3.7. Suppose $N = L(A, K[M(B, O[C])])$ and $X = L(\lambda, K[M(B, O[\lambda])])$. Choose $t_1 = (a, [(b, [c])])$ and $t_2 = (a', [(b, [c'])])$ with different $a, a' \in dom(A)$, $b \in dom(B)$ and different $c, c' \in dom(C)$. In this case, the projections $\pi_X^N(t_1)$ and $\pi_X^N(t_2)$ are both equal to $(ok, [(b, [ok])])$, but projections of t_1 and t_2 on any proper superattribute of X are different from one another. □

The following result shows that FDs in the presence of records and lists can be captured by a natural generalisation of Armstrong's well-known axioms.

Theorem 3.14. *The generalised Armstrong axioms are sound and complete for the implication of functional dependencies in the presence of records and lists.*

Proof. It remains to show the completeness. Let N be a nested attribute, Σ a set of FDs defined on N. We need to show that $\Sigma^* \subseteq \Sigma^+$ holds. Let $X \to Y \notin \Sigma^+$. We will show that $X \to Y \notin \Sigma^*$ by defining some $r \subseteq dom(N)$ with $\models_r \Sigma^*$ and $\not\models_r X \to Y$. Let

$X^+ = \bigsqcup\{Z \mid X \to Z \in \Sigma^+\}$. It follows that $Y \not\leq X^+$. Otherwise we had $X^+ \dashrightarrow Y \in \Sigma^+$ by the reflexivity axiom, $X \to X^+ \in \Sigma^+$ by the join rule and also $X \to Y \in \Sigma^+$ by the transitivity rule. Using Lemma 3.13 we choose $r = \{t_1, t_2\} \subseteq dom(N)$ such that

$$\pi_Z^N(t_1) = \pi_Z^N(t_2) \qquad \text{if and only if} \qquad Z \leq X^+ \quad . \tag{1}$$

Since $X \leq X^+$ and $Y \not\leq X^+$ we have $\pi_X^N(t_1) = \pi_X^N(t_2)$, but $\pi_Y^N(t_1) \neq \pi_Y^N(t_2)$. This shows that $\not\models_r X \to Y$. Let $U \to V \in \Sigma$. If $U \not\leq X^+$, then $\pi_U^N(t_1) \neq \pi_U^N(t_2)$ by equation (1), and $\models_r U \to V$. If $U \leq X^+$, then $\pi_U^N(t_1) = \pi_U^N(t_2)$ by equation (1). It follows that $X^+ \to U \in \Sigma^+$ by the reflexivity axiom. From $X \to X^+, X^+ \dashrightarrow U, U \to V \in \Sigma^+$ follows $X \to V \in \Sigma^+$ which means that $V \leq X^+$ by definition of X^+. Again, equation (1) implies that $\pi_V^N(t_1) = \pi_V^N(t_2)$ holds. This shows $\models_r U \to V$. From $\models_r \Sigma$ follows immediately $\models_r \Sigma^*$ which completes the proof. □

EXAMPLE 3.8. Take again Factor(Integer,Prime[Number],Exponent[Number]), X = Factor(Prime[λ]) and X^+ = Factor(Prime[λ],Exponent[λ]). The two tuples t_1 = $(12, [2,3], [2,1])$ and $t_2 = (15, [3,5], [1,1])$ have projections which are equal on exactly X^+. □

Similar to the RDM, the generalised Armstrong axioms are also minimal for the implication of FDs in the presence of records and lists.

Theorem 3.15. *The generalised Armstrong axioms are minimal for the implication of functional dependencies in the presence of records and lists.*

Proof. Let R be the *reflexivity axiom*, and let $\mathfrak{R}$ consist of the extension and transitivity rule. Let $N = \lambda$, $\Sigma = \emptyset$ and $\sigma = \lambda \to \lambda$. Then $\Sigma^+_{\mathfrak{R}} = \emptyset$, but σ can be inferred by R.

Let R be the *extension rule*, and let $\mathfrak{R}$ consist of the reflexivity axiom and the transitivity rule. Let $N = L(A, B)$, $\Sigma = \{L(A, \lambda) \to L(\lambda, B)\}$ and $\sigma = L(A, \lambda) \to L(A, B)$. We present $\Sigma^+_{\mathfrak{R}}$ by the following table where the row names denote the left-hand side X, and the column names denote the right-hand side Y of an FD $X \to Y$. An FD $X \to Y$ belongs to $\Sigma^+_{\mathfrak{R}}$ if and only if the entry at row X and column Y is a cross $\times$. We compute

	$L(\lambda,\lambda)$	$L(A,\lambda)$	$L(\lambda,B)$	$L(A,B)$
$L(\lambda,\lambda)$	$\times$			
$L(A,\lambda)$	$\times$	$\times$	$\times$	
$L(\lambda,B)$	$\times$		$\times$	
$L(A,B)$	$\times$	$\times$	$\times$	$\times$

and see that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, as $L(A, \lambda) \sqcup L(\lambda, B) = L(A, B)$ we conclude that σ can be inferred from Σ using the extension rule.

Let R be the *transitivity rule*, and let $\mathfrak{R}$ consist of the reflexivity axiom and the extension rule. Let $N = L(A, B)$, $\Sigma = \{L(\lambda, \lambda) \to L(A, \lambda), L(A, \lambda) \to L(\lambda, B)\}$ and $\sigma = L(\lambda, \lambda) \to$

$L(\lambda, B)$. We compute

	$L(\lambda,\lambda)$	$L(A,\lambda)$	$L(\lambda,B)$	$L(A,B)$
$L(\lambda,\lambda)$	×	×		
$L(A,\lambda)$	×	×	×	×
$L(\lambda,B)$	×		×	
$L(A,B)$	×	×	×	×

and see that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, σ can be inferred from Σ using the transitivity rule. □

3.1.5 Dependencies for Keys

As an application of Theorem 3.14 we consider keys which are an important concept in databases. In relational databases, a key is a set of attributes on which no pair of tuples of a legal instance coincides. That is, values on the key attributes determine a tuple uniquely. We write $X < Y$ if and only if $X \leq Y$ and $X \neq Y$. In this case, X is called a *proper subattribute* of Y.

Definition 3.16. Let $N \in \mathcal{N}\mathrm{A}$ be a nested attribute and Σ a set of FDs on N. A subattribute $X \in Sub(N)$ is called a *superkey for* N with respect to Σ if and only if $\Sigma \models X \to N$ holds. In case there is not any proper subattribute X' of X which is also a superkey for N with respect to Σ we call X a *minimal key for* N with respect to Σ.

An FD $X \to N \in \Sigma^*$ is called a *key dependency for* N *with respect to* Σ if and only if X is a minimal key for N with respect to Σ. The set of all key dependencies for N with respect to Σ is denoted by Σ_{key}. □

We obtain the following characterisation as a consequence of Theorem 3.14. The generalised Armstrong axioms are again denoted by $\mathfrak{R}$.

Corollary 3.17. *Let N be a nested attribute, $X \in Sub(N)$, and Σ a set of FDs on N. The following statements are equivalent.*

1. *X is a superkey for N with respect to Σ,*
2. *for each $r \subseteq dom(N)$ with $\models_r \Sigma$ and for every two $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$ we have $t_1 = t_2$,*
3. *$X \to N \in \Sigma^*$,*
4. *$X \to N \in \Sigma^+$,*
5. *the semantic closure $X^* = \bigsqcup\{Y \mid X \to Y \in \Sigma^*\}$ of X with respect to Σ is N, i.e., $X^* = N$,*
6. *the syntactic closure $X^+_{\mathfrak{R}} = \bigsqcup\{Y \mid X \to Y \in \Sigma^+_{\mathfrak{R}}\}$ of X with respect to Σ and $\mathfrak{R}$ is N, i.e., $X^+_{\mathfrak{R}} = N$.* □

An FD $X \to N$, defined on the nested attribute N, is called a *superkey dependency for* N. We can then consider the implication for the class $\mathcal{C}$ of superkey dependencies. Given some nested attribute N and a set Σ of superkey dependencies for N we might ask which other superkey dependencies are implied by Σ? The next result shows that once the keys have been specified there are only trivial extensions derivable.

Theorem 3.18. *The following inference rules*

$$\frac{}{N \to N} \qquad \frac{X \to N}{X \sqcup_N Y \to N}$$

(key axiom) *(key augmentation)*

are minimal, sound and complete for the implication of superkey dependencies in the presence of records and lists.

Proof. Let N be a nested attribute and Σ a set of superkey dependencies for N. The soundness of the rules is immediate. For the completeness, take some $X \to N \notin \Sigma^+$ and choose $r = \{t_1, t_2\} \subseteq dom(N)$ such that

$$\pi_Y^N(t_1) = \pi_Y^N(t_2) \qquad \text{if and only if} \qquad Y \leq X \tag{2}$$

according to Lemma 3.13. If $N \leq X$, then $X = N$ and $X \to N \in \Sigma^+$ by the key axiom. It follows that t_1 and t_2 are different and $\not\models_r X \to N$. It remains to show that $\models_r K \to N$ for all superkey dependencies $K \to N \in \Sigma$. Suppose $K \leq X$. Then $X = K \sqcup X$ and from $K \to N \in \Sigma$ follows $X \to N \in \Sigma^+$ by the key augmentation rule, a contradiction. Therefore, K is not a subattribute of X, and $\pi_K^N(t_1) \neq \pi_K^N(t_2)$ by equation (2), and thus $\models_r K \to N$.

The minimality of the rules is also straightforward. For the independence of the key axiom let $\mathfrak{R}$ consist of the key augmentation rule only, $N = \lambda$ and $\Sigma = \emptyset$. Consequently, $\lambda \to \lambda \notin \Sigma_{\mathfrak{R}}^+$, but $\lambda \to \lambda$ can be derived using the *key axiom*. For the independence of the *key augmentation rule* let $\mathfrak{R}$ consist of the key axiom and take for instance $N = L(A, B)$, and $\Sigma = \{L(\lambda, \lambda) \to N\}$. It follows that $L(A, \lambda) \to N \notin \Sigma_{\mathfrak{R}}^+$, but $L(A, \lambda) \to N$ can be derived using the key augmentation rule. □

3.2 Implication Problem

The implication problem for FDs is to decide whether for an arbitrary nested attribute N, an arbitrary set Σ of FDs on N and an arbitrary FD σ on N, $\Sigma \models \sigma$ holds. For functional dependencies, the problem coincides with the finite implication problem $\Sigma \models_f \sigma$. According to Theorem 3.14 the FD σ is implied by Σ if and only if σ is derivable from Σ using the generalised Armstrong axioms. However, finding a derivation is arduous. The goal of this section is to solve the implication problem efficiently.

3.2.1 The Closure

Similar as in the RDM [29] we introduce the notion of syntactic closure for a nested attribute with respect to a given set of functional dependencies and a set of inference rules. This notion has already been used in the proof of Theorem 3.14 and in Corollary 3.17.

Definition 3.19. Let N be a nested attribute, $X \in Sub(N)$, Σ a set of FDs defined on N, and $\mathfrak{R}$ a set of inference rules. The *syntactic closure* $X^+_{\mathfrak{R}} \in Sub(N)$ of X with respect to Σ and $\mathfrak{R}$ is $X^+_{\mathfrak{R}} = \bigsqcup\{Y \mid X \to Y \in \Sigma^+_{\mathfrak{R}}\}$. □

As usual, once the set $\mathfrak{R}$ of inference rules has been fixed the index $\mathfrak{R}$ can be dropped from $X^+_{\mathfrak{R}}$, i.e., we simply write X^+. For the remainder of this chapter $\mathfrak{R}$ denotes the generalised Armstrong axioms from Definition 3.8.

In order to solve the implication problem $\Sigma \models X \to Y$ it is sufficient to determine X^+.

Lemma 3.20. *Let Σ be a set of FDs, all defined on some nested attribute N, and $X \to Y$ an FD on N. Then $X \to Y \in \Sigma^+$ if and only if $Y \leq X^+$.*

Proof. If $X \to Y \in \Sigma^+$, then $Y \in \{Z \mid X \to Z \in \Sigma^+\}$. Consequently, $Y \leq X^+$ by the property of a join.

Suppose $Y \leq X^+$. According to the join rule from Lemma 3.11 we have $X \to X^+ \in \Sigma^+$. Reflexivity implies $X^+ \to Y \in \Sigma^+$, and transitivity results in $X \to Y \in \Sigma^+$. □

3.2.2 A first Approach

We will now present a first algorithm which determines the closure X^+ of a given $X \in Sub(N)$ with respect to a set Σ of FDs on N. The algorithm generalises the one proposed in [29] for solving the implication problem in the RDM.

Algorithm 3.2.1 (Nested Attribute Closure I)

Input: $\boldsymbol{X \in Sub(N)}$, set $\boldsymbol{\Sigma}$ of FDs on $\boldsymbol{N}$

Output: closure $\boldsymbol{X^+_{alg}}$ of $\boldsymbol{X}$ with respect to $\boldsymbol{\Sigma}$

Method:

```
      VAR X_old, X_new ∈ Sub(N);
(1)   X_new := X;
(2)   REPEAT
(3)     X_old := X_new;
(4)     FOR each U → V ∈ Σ DO
(5)       IF U ≤ X_new THEN
(6)         X_new := X_new ⊔_N V;
(7)       ENDIF;
(8)     ENDDO;
(9)   UNTIL X_new := X_old;
(10)  X^+_alg := X_new;
(11)  RETURN(X^+_alg);
```

□

We illustrate Algorithm 3.2.1 with an example.

EXAMPLE 3.9. Let $N = L(K[A], B, C, M[D], O[P(E, F)])$, $X = L(K[\lambda], B, O[P(E)])$, and Σ given by

1. L(M[λ]) $\rightarrow$ L(M[D]),
2. L(O[λ]) $\rightarrow$ L(K[λ]),
3. L(O[P(F)]) $\rightarrow$ L(K[A]),
4. L(B,C) $\rightarrow$ L(O[P(E,F)]), and
5. $\lambda \rightarrow$ L(C).

Suppose that in each run through the REPEAT loop between lines (2) and (9) of Algorithm 3.2.1 the FDs are processed in the order listed above. While this order has a direct impact on the intermediate results, the final result X^+_{alg} is independent from the order the FDs are processed. After the first run, $X_{\text{new}} = L(K[\lambda], B, C, O[P(E)])$. The second run yields $X_{\text{new}} = L(K[\lambda], B, C, O[P(E, F)])$ and the third run $X_{\text{new}} = L(K[A], B, C, O[P(E, F)])$. The fourth run does not change anything, i.e., $X^+_{\text{alg}} = L(K[A], B, C, O[P(E, F)])$. □

It is the first objective to show that Algorithm 3.2.1 works correctly.

Theorem 3.21. *Algorithm 3.2.1 is correct, i.e.,* $X^+_{alg} = X^+$.

Proof. We show first that $X^+_{\text{alg}} \leq X^+$ holds. We proceed by induction on the number j of runs through the REPEAT loop from line (2) to (9). If $j = 0$, then we have $X_{\text{new}} := X$ by line (1). However, $X \leq X^+$ holds in any case as $X \rightarrow X \in \Sigma^+$ by the reflexivity axiom.

Let $j > 0$. The hypothesis says that after j runs through the REPEAT loop we have $X_{\text{new}} \leq X^+$. Consider now the $j + 1$-st run through line (2) to (9) which computes the join of X_{new} and V in line (6) whenever $U \rightarrow V \in \Sigma$ and $U \leq X_{\text{new}}$ hold. From $U \leq X_{\text{new}}$ and the hypothesis $X_{\text{new}} \leq X^+$ we know that $U \leq X^+$ holds as well. Lemma 3.20 implies $X \rightarrow U \in \Sigma^+$. We conclude that $X \rightarrow V \in \Sigma^+$ by transitivity. Consequently, $V \leq X^+$ and thus $X_{\text{new}} = X_{\text{new}} \sqcup V \leq X^+$ after the $j + 1$-st run. We have shown that $X_{\text{new}} \leq X^+$ is invariant under the REPEAT loop, hence $X^+_{\text{alg}} \leq X^+$.

Since $\leq$ is a partial order, it remains to show that $X^+ \leq X^+_{\text{alg}}$ holds as well. Since the definition of X^+ depends on Σ^+, we assume that there is a chain

$$\Sigma = \Sigma_0 \subset \Sigma_1 \subset \ldots \subset \Sigma_k = \Sigma^+$$

where every Σ_i results from Σ_{i-1} by application of a single inference rule of the generalised Armstrong axioms. We will use induction on i to show the following:

$$\text{if } Y \rightarrow Z \in \Sigma_i \text{ and } Y \leq X^+_{\text{alg}}, \text{ then } Z \leq X^+_{\text{alg}}. \tag{3}$$

Then we conclude for $i = k$ that $Z \leq X^+_{\text{alg}}$ follows from $Y \rightarrow Z \in \Sigma^+$ and $Y \leq X^+_{\text{alg}}$. Hence, $X^+ \leq X^+_{\text{alg}}$ for $Y = X$ and $Z = X^+$.

It remains to show (3). If $i = 0$, we assume that $Y \rightarrow Z \in \Sigma$. If $Y \leq X^+_{\text{alg}}$, then $Y \leq X_{\text{new}}$ at some point. The REPEAT loop computes $X_{\text{new}} := X_{\text{new}} \sqcup Z$ in line (6) and we obtain $Z \leq X^+_{\text{alg}}$ as stated. If $i > 0$, then $\Sigma_i - \Sigma_{i-1}$ contains exactly one $Y \rightarrow Z$. There is nothing to show for any functional dependencies in Σ_{i-1} (hypothesis). Thus, we consider only $Y \rightarrow Z$ and distinguish between three cases.

- If $Y \to Z$ results from applying the reflexivity axiom, then $Z \leq Y$. Since $Y \leq X^+_{\text{alg}}$ holds by assumption, we obtain $Z \leq X^+_{\text{alg}}$ by the transitivity of $\leq$.
- If $Y \to Z$ results from applying the extension rule, then $Z = Y \sqcup U$ with $Y \to U \in \Sigma_{i-1}$. However, $Y \leq X^+_{\text{alg}}$ implies $U \leq X^+_{\text{alg}}$ by hypothesis, and therefore also $Z \leq X^+_{\text{alg}}$ as Z is the join of Y and U.
- If $Y \to Z$ results from applying the transitivity rule, then there is some $U \in Sub(N)$ with $Y \to U \in \Sigma_{i-1}$ and $U \to Z \in \Sigma_{i-1}$. If $Y \leq X^+_{\text{alg}}$, then we conclude $U \leq X^+_{\text{alg}}$, and subsequently $Z \leq X^+_{\text{alg}}$ by hypothesis. □

EXAMPLE 3.10. Consider Example 3.2 again. As input for Algorithm 3.2.1 we choose $X =$ Factor(Integer) and Σ as the set of four FDs given in Example 3.2. The only FD $U \to V \in \Sigma$ with $U \leq X$ is Factor(Integer) $\to$ Factor(Prime[Number], Exponent[Number]). Therefore, X_{new} becomes Factor(Integer, Prime[Number], Exponent[Number]) already. Since X_{new} is already maximal, further runs through the REPEAT loop cannot add anything new. The output is therefore Factor(Integer, Prime[Number], Exponent[Number]). □

We continue our example of the GenBank database.

EXAMPLE 3.11. Suppose that nested attribute N and the set Σ of FDs on N are specified as in Example 3.3. Suppose further that one is interested in the closure of the subattribute

$$X = \text{DNA(Origin[Base],Gene(Start,End))}.$$

We use Algorithm 3.2.1 to determine X^+. The following sequence contains the updates of X_{new} by the FDs in Σ. We choose to select the FDs in the exact order they are given in Example 3.3.

- FD1 gives $X_{\text{new}} =$ DNA(Origin[Base],Count(A,C,G,T),Gene(Start,End)),
- the FDs of the second and third item do not add anything new to X_{new},
- FD4 gives $X_{\text{new}} =$ DNA(Origin[Base],Count(A,C,G,T),Gene(Start,End,Sub[Nucleo])), and
- FD5 results in $X_{\text{new}} = N$, i.e., there cannot be any more changes

As it turns out, DNA(Origin[Base],Gene(Start,End)) is a minimal key for N with respect to Σ. A further minimal key is given by DNA(Origin[Base],Gene(Start,Translation[λ])) since DNA(Gene(Translation[λ])) $\to$ DNA(Gene(Sub[λ])) and DNA(Gene(Start,Sub[λ])) $\to$ DNA(Gene(Start,End)) are in Σ. □

3.2.3 A different Perspective

FDs on some nested attribute N have been defined as expressions $X \to Y$ with $X, Y \in Sub(N)$. Alternatively, we can view FDs on N as expressions $X \to Y$ where $X, Y \in \mathcal{C}_N$ and $(\mathcal{C}_N, \subseteq, \cup, \cap, \dot{-}_{\mathcal{C}_N}, J_N)$ is the Brouwerian algebra of closed subsets of the PO-space on the join-irreducible elements J_N of $Sub(N)$. A set $r \subseteq dom(N)$ *satisfies* the functional dependency $X \to Y$ on N, denoted by $\models_r X \to Y$, if and only if $\pi^N_B(t_1) = \pi^N_B(t_2)$ for all $B \in Y$ whenever $\pi^N_A(t_1) = \pi^N_A(t_2)$ for all $A \in X$ holds for any $t_1, t_2 \in r$. This view

can be justified in the following sense. Lemma 3.9 shows that for all $r \subseteq dom(N)$ we have $\models_r X \to Y$ for $X, Y \in \mathcal{C}_N$ if and only if $\models_r \bigsqcup X \to \bigsqcup Y$ in terms of Definition 3.1. For the remainder of our investigations on the implication problem we will consider FDs from the perspective of closed subsets of $\mathcal{C}_N$. The generalised Armstrong axioms are then given by

$$\frac{}{X \to Y}\ Y \subseteq X, \qquad \frac{X \to Y}{X \to X \cup Y}, \qquad \frac{X \to Y,\ Y \to Z}{X \to Z}.$$

These inference rules use set inclusion and set union and look exactly like Armstrong's original axioms. However, they are defined on $(\mathcal{C}_N, \subseteq, \cup, \cap, \dot{-}_{\mathcal{C}_N}, J_N)$. The closure X^+ of some $X \in \mathcal{C}_N$ with respect to a given set of FDs on N is then $\cup\{Y \mid X \to Y \in \Sigma^+\}$. Algorithm 3.2.1 translates to the following algorithm.

Algorithm 3.2.2 (Nested Attribute Closure II)

Input: $X \in \mathcal{C}_N$, set Σ of FDs on N

Output: closure X^+_{alg} of X with respect to Σ

Method:

```
        VAR X_old, X_new ∈ C_N;
(1)     X_new := X;
(2)     REPEAT
(3)       X_old := X_new;
(4)       FOR each U → V ∈ Σ DO
(5)         IF U ⊆ X_new THEN
(6)           X_new := X_new ∪ V;
(7)         ENDIF;
(8)       ENDDO;
(9)     UNTIL X_new := X_old;
(10)    X+_alg := X_new;
(11)    RETURN(X+_alg);
```

□

The correctness of Algorithm 3.2.2 follows immediately from Theorem 3.21 and 2.11. We will now study the time complexity of Algorithm 3.2.2 in terms of the size of the input N and Σ. The size of N, denoted by n, is defined as the number of join-irreducible elements of N, i.e., $n = \mid J_N \mid$. The size s of Σ is defined as the number of its elements, i.e., $s = \mid \Sigma \mid$.

Theorem 3.22. *Algorithm 3.2.2 terminates in time* $\mathcal{O}(n \cdot s \cdot \min\{n, s\})$.

Proof. In each step of the REPEAT loop between line (2) and (9), the inner FOR loop between line (4) and (8) is executed exactly s times. The inclusion test $U \subseteq X_{new}$ takes at most n operations. The same holds for all operations within the IF branch in line (5) to (7). Therefore, $\mathcal{O}(n \cdot s)$ operations are necessary for the inner FOR loop.

The REPEAT loop between line (2) and (9) itself is executed at most $n+1$ times since X_{new} cannot have more than n elements and at least one element is added in each step. Moreover, the loop is executed at most $s+1$ times as every element in Σ can contribute to the extension of X_{new} at most once. It follows that Algorithm 3.2.2 indeed terminates in time $\mathcal{O}(n \cdot s \cdot \min\{n, s\})$. □

3.2.4 A linear time Algorithm

The time complexity of Algorithm 3.2.2 can be improved. Every FD in Σ is used at most once but every run through the REPEAT loop considers all FDs in Σ again. Therefore, Algorithm 3.2.2 can be optimised.

The idea of the optimised Algorithm 3.2.3 is the following: for every $A \in J_N$ the set of FDs $U \to V \in \Sigma$ with $A \in U$ is stored as *In(A)* in an array *In*. Initially, $In(A) = \{U \to V \in \Sigma \mid A \in U\}$ in line (11). Moreover, one considers for every dependency $U \to V$ the number of elements in U which have not been added to the closure X^+_{alg}. This is done using an array Ar with $Ar(U \to V) = \mid U \mid$ initially in line (6). The set X_q contains join-irreducible subattributes A which will be added to the closure, and $Ar(U \to V)$ is decremented whenever $A \in U$, see line (20). If $Ar(U \to V)$ becomes 0, every element of V that is not already in the closure is added to X_q in line (22).

Algorithm 3.2.3 (Optimised Nested Attribute Closure)

Input: $\boldsymbol{N} \in \mathcal{N}$A, set $\boldsymbol{\Sigma}$ of FDs on $\boldsymbol{N}$, $\boldsymbol{X} \in \mathcal{C}_{\boldsymbol{N}}$

Output: the closure $\boldsymbol{X}^+_{\text{alg}}$ of $\boldsymbol{X}$ with respect to $\boldsymbol{\Sigma}$

Method:

```
     VAR X_q, X' ⊆ J_N,
     Ar: Array Σ of INTEGER,
     In: Array J_N of sets of FDs;

(1)  X' := ∅; X_q := X;
(2)  FOR each A ∈ J_N DO
(3)    In(A) := ∅;
(4)  ENDDO;
(5)  FOR each U → V ∈ Σ DO
(6)    Ar(U → V) := | U |;
(7)    IF U = ∅ THEN
(8)      X_q := X_q ∪ V;
(9)    ELSE
(10)     FOR each A ∈ U DO
(11)       In(A) := In(A) ∪ {U → V};
(12)     ENDDO;
(13)   ENDIF;
(14) ENDDO;
```

(15) **WHILE** $X_q \neq \emptyset$ **DO**
(16) **SELECT** $A \in X_q$;
(17) $X_q := X_q - \{A\}$;
(18) $X' := X' \cup \{A\}$;
(19) **FOR each** $U \to V \in In(A)$ **DO**
(20) $Ar(U \to V) := Ar(U \to V) - 1$;
(21) **IF** $Ar(U \to V) = 0$ **THEN**
(22) $X_q := X_q \cup (V - X')$;
(23) **ENDIF**;
(24) **ENDDO**;
(25) **ENDDO**;
(26) $X^+_{\text{alg}} := X'$;
(27) **RETURN**(X^+_{alg});

□

We will prove that Algorithm 3.2.3 works correctly and in linear time in the size $n \cdot s$ of the input.

Theorem 3.23. *Algorithm 3.2.3 is correct and terminates in time* $\mathcal{O}(n \cdot s)$.

Proof. We argue first that X^+_{alg} is indeed a closed set in the PO-space on $(J_N, \leq)$. Therefore, we must show that $X' \in \mathcal{C}_N$ in line (26) of Algorithm 3.2.3. X' inherits all the attributes from X_q at some point. However, X_q is initialised as $X \in \mathcal{C}_N$, and if some new attributes are added to X_q, then these are attributes in $V - X'$ where $V \in \mathcal{C}_N$ as the right-hand side of an FD. In line (26), X' is therefore the finite union of closed subsets, i.e., closed as well.

For the correctness proof itself we show $X^+_{\text{alg}} \subseteq X^+$ and $X^+ \subseteq X^+_{\text{alg}}$. The first inclusion follows from the initialisation of X^+_{alg} and $X_q \in \mathcal{C}_N$. In fact, only subattributes from X_q are added to X^+_{alg}. On the other hand X_q is extended in case $Ar(U \to V) = 0$. This, however, is only possible if every subattribute in U is already included in X_q, i.e., $U \subseteq X^+$. An application of the transitivity rule shows $V \subseteq X^+$.

For the inverse inclusion $X^+ \subseteq X^+_{\text{alg}}$ we consider, as in the proof of Theorem 3.21, the chain

$$\Sigma = \Sigma_0 \subset \Sigma_1 \subset \ldots \subset \Sigma_k = \Sigma^+ \quad ,$$

where every Σ_i results from Σ_{i-1} by application of a single inference rule of the generalised Armstrong axioms. If Σ_i is such a set and $Y \to Z \in \Sigma_i$ with $Y \subseteq X^+_{\text{alg}}$, we show $Z \subseteq X^+_{\text{alg}}$. The claim $X^+ \subseteq X^+_{\text{alg}}$ follows then as in the proof of Theorem 3.21.

We proceed by induction on i. If $i = 0$, then we assume that $Y \to Z \in \Sigma$. If $Y \subseteq X'$, then every subattribute in Y belongs to X_q at some point in time. This means, every subattribute in Y will be selected during the WHILE loop and $Ar(Y \to Z)$ will eventually become 0. In this case $Z - X'$ is added to X_q and indirectly to X'. This implies $Z \subseteq X'$ as claimed. For $i > 0$ the set $\Sigma_i - \Sigma_{i-1}$ contains exactly one $Y \to Z$. The statement can then be proven as in the proof of Theorem 3.21.

In order to prove the complexity, we first look at the initialisation loop between line (5) and line (14). This FOR loop is executed s times. The inner FOR loop between line (10) and line (12) is executed exactly $\mid U \mid$ times. Since adding a functional dependency to $In(A)$ can be done in constant time if $In(A)$ is represented in an appropriate way as a list, and since $\mid U \mid \leq n$ we obtain a complexity of $\mathcal{O}(n \cdot s)$ for the initialisation.

The WHILE loop between line (15) and line (25) is executed at most n times since every join-irreducible subattribute can be selected at most once. The inner FOR loop between line (19) and (24) is executed at most s times. Clearly, if $Ar(U \rightarrow V) = 0$, then all the subattributes in U have been considered and $U \rightarrow V$ cannot occur any further in the algorithm. Hence, the IF test evaluates to *true* at most s times. Since the union operation on X_q takes $\mathcal{O}(n)$ time, it follows that the statement $X_q := X_q \cup (V - X')$ takes on the whole also $\mathcal{O}(n \cdot s)$ time. Hence the algorithm has time complexity $\mathcal{O}(n \cdot s)$. □

We illustrate Algorithm 3.2.3 with an example.

EXAMPLE 3.12. We use the same input as in Example 3.9, i.e., $N = L(K[A], B, C, M[D], O[P(E, F)])$, $X = L(K[\lambda], B, O[P(E)])$, and Σ is given by

- L(M[λ]) $\rightarrow$ L(M[D]),
- L(O[λ]) $\rightarrow$ L(K[λ]),
- L(O[P(F)]) $\rightarrow$ L(K[A]),
- L(B,C) $\rightarrow$ L(O[P(E,F)]), and
- $\lambda \rightarrow$ L(C).

We would like to compute X^+ by means of Algorithm 3.2.3. Therefore, we translate the instance above into an input instance for Algorithm 3.2.3. The set J_N of join-irreducible subattributes of N consists of

$$Y_1 = \text{L(K[A])}, Y_2 = \text{L(K[}\lambda\text{])}, Y_3 = \text{L(B)}, Y_4 = \text{L(C)}, Y_5 = \text{L(M[D])},$$
$$Y_6 = \text{L(M[}\lambda\text{])}, Y_7 = \text{L(O[P(E)])}, Y_8 = \text{L(O[P(F)])} \text{ and } Y_9 = \text{L(O[}\lambda\text{])}.$$

The instance above is then translated into the following input for Algorithm 3.2.3, using Theorem 2.11.

- $J_N = Y_1Y_2Y_3Y_4Y_5Y_6Y_7Y_8Y_9$,
- $X = Y_2Y_3Y_7Y_9$,
- Σ consists of
  - $Y_6 \rightarrow Y_5Y_6$,
  - $Y_9 \rightarrow Y_2$,
  - $Y_8Y_9 \rightarrow Y_1Y_2$,
  - $Y_3Y_4 \rightarrow Y_7Y_8Y_9$, and
  - $\emptyset \rightarrow Y_4$.

The precomputations yield the following:

- $X' = \emptyset$, $X_q = Y_2Y_3Y_7Y_9$

- $Ar(Y_6 \to Y_5Y_6) = 1$, $Ar(Y_9 \to Y_2) = 1$, $Ar(Y_8Y_9 \to Y_1Y_2) = 2$, $Ar(Y_3Y_4 \to Y_7Y_8Y_9) = 2$, and $Ar(\emptyset \to Y_4) = 0$ implies $X_q = Y_2Y_3Y_4Y_7Y_9$
- $In(Y_3) = \{Y_3Y_4 \to Y_7Y_8Y_9\}$, $In(Y_4) = \{Y_3Y_4 \to Y_7Y_8Y_9\}$, $In(Y_6) = \{Y_6 \to Y_5Y_6\}$, $In(Y_8) = \{Y_8Y_9 \to Y_1Y_2\}$ and $In(Y_9) = \{Y_9 \to Y_2, Y_8Y_9 \to Y_1Y_2\}$.

We show the respective values for every run through the WHILE loop:

1. $A = Y_2$: $X_q = Y_3Y_4Y_7Y_9$, $X' = Y_2$,
2. $A = Y_3$: $X_q = Y_4Y_7Y_9$, $X' = Y_2Y_3$, $Ar(Y_3Y_4 \to Y_7Y_8Y_9) = 1$,
3. $A = Y_4$: $X_q = Y_7Y_9$, $X' = Y_2Y_3Y_4$, $Ar(Y_3Y_4 \to Y_7Y_8Y_9) = 0$, and therefore $X_q = Y_7Y_8Y_9$
4. $A = Y_7$: $X_q = Y_8Y_9$, $X' = Y_2Y_3Y_4Y_7$,
5. $A = Y_8$: $X_q = Y_9$, $X' = Y_2Y_3Y_4Y_7Y_8$, $Ar(Y_8Y_9 \to Y_1Y_2) = 1$,
6. $A = Y_9$: $X_q = \emptyset$, $X' = Y_2Y_3Y_4Y_7Y_8Y_9$, $Ar(Y_9 \to Y_2) = 0$ causes no change to X_q, $Ar(Y_8Y_9 \to Y_1Y_2) = 0$ implies $X_q = Y_1$,
7. $A = Y_1$: $X_q = \emptyset$, $X' = Y_1Y_2Y_3Y_4Y_7Y_8Y_9$

We have output $X^+_{alg} = Y_1Y_2Y_3Y_4Y_7Y_8Y_9$ which corresponds to L(K[A],B,C,O[P(E,F)]). □

In the literature, $\mathcal{O}(n \cdot s)$ is usually considered as the order of the input. From this point of view, Algorithm 3.2.3 is a linear time algorithm for the computation of the closure of a nested attribute.

Theorem 3.24. *The implication problem for functional dependencies in the presence of records and lists is decidable in linear time.* □

3.2.5 Applications

Algorithm 3.2.3 can be applied to other problems important for database design, for instance to eliminate redundant FDs. Let Σ be a set of FDs on some nested attribute N. An FD $\sigma \in \Sigma$ is *redundant* in Σ if and only if $(\Sigma - \{\sigma\})^+ = \Sigma^+$. A *non-redundant cover of* Σ is a set Θ of FDs on N where $\Theta^+ = \Sigma^+$ and Θ does not contain any redundant FD. In order to determine if σ is redundant in Σ one can test whether $\sigma \in (\Sigma - \{\sigma\})^+$ holds. The following algorithm finds a subset $\Theta \subseteq \Sigma$ that is a non-redundant cover of Σ.

Algorithm 3.2.4 (Non-Redundant Covers)

Input: $\boldsymbol{N} \in \boldsymbol{\mathcal{N}}\mathbf{A}$, set $\boldsymbol{\Sigma}$ of FDs on $\boldsymbol{N}$

Output: a non-redundant cover $\boldsymbol{\Theta}$ of $\boldsymbol{\Sigma}$

Method:

(1) $\Theta := \Sigma$;
(2) **FOR each** $\sigma \in \Sigma$ **DO**
(3) **IF** $\sigma \in (\Theta - \{\sigma\})^+$ **THEN** $\Theta := \Theta - \{\sigma\}$;
(4) **ENDDO**;
(5) **RETURN**(Θ);

□

Note that Θ will always be a subset of Σ although this is not required by the definition of a non-redundant cover. The result is dependent on the selection order of σ in line (2) of Algorithm 3.2.4.

Theorem 3.25. *Algorithm 3.2.4 computes a non-redundant cover for a set Σ of FDs on some nested attribute N in time $\mathcal{O}(n \cdot s^2)$.* □

Recall that $X \in Sub(N)$ is called a *superkey* for N with respect to a given set Σ of FDs on N if and only if $\Sigma \models X \to N$ holds. This means that X is a superkey for N if and only if $N \leq X^+$. From the view of closed subsets, $X \in \mathcal{C}_N$ is called a *superkey* for N with respect to a given set Σ of FDs on N if and only if $\Sigma \models X \to J_N$ holds. This is equivalent to the condition $J_N \subseteq X^+$.

Algorithm 3.2.5 (Superkey)

Input: $\boldsymbol{N} \in \boldsymbol{N}$A, set $\boldsymbol{\Sigma}$ of FDs on $\boldsymbol{N}$, $\boldsymbol{X} \in \boldsymbol{\mathcal{C}_N}$

Output: $\begin{cases} \text{yes} & \text{, if } \boldsymbol{X} \text{ is a superkey for } \boldsymbol{N} \text{ with respect to } \boldsymbol{\Sigma} \\ \text{no} & \text{, else} \end{cases}$

Method:

(1) Compute X^+_{alg} using Algorithm 3.2.3 with input (N, Σ, X);
(2) **IF** $J_N \subseteq X^+_{\text{alg}}$ **THEN RETURN**(yes)
(3) **ELSE RETURN**(No);

□

Theorem 3.26. *Algorithm 3.2.5 decides in time $\mathcal{O}(n \cdot s)$ whether $X \in \mathcal{C}_N$ is a superkey for N with respect to a set Σ of FDs defined on N.* □

3.3 Nested List Normal Form

We will now return to the view of FDs as given by Definition 3.1. One key objective in the research on dependency theory is the development of well-designed database schema proposals, and justification of these proposals by formally proving the equivalence to desirable semantic properties, for instance the absence of certain processing difficulties. In this section we will propose the Nested List normal form for nested attributes, defined in terms of FDs. We will show that this proposal can be characterised by the absence of redundancies and abnormal update behavior as well as simplified integrity checking.

In the context of nested attributes join-irreducible subattributes will be called basis attributes.

Definition 3.27. Let N be a nested attribute. The *subattribute basis* $SubB(N)$ of N is the set of join-irreducible elements of $(Sub(N), \leq, \sqcup, \sqcap, \dot{-}, N)$. Every attribute in $SubB(N)$ is called a *basis attribute of* N. Let $MaxB(N)$ denote the maximal basis attributes of N with respect to $\leq$, and $NMaxB(N)$ the non-maximal basis attributes of N with respect to $\leq$. □

3.3.1 Trivial FDs

Suppose we are given some nested attribute N. As in the RDM, there are some FDs on N which are satisfied by every $r \subseteq dom(N)$. We call these FDs *trivial.*

Lemma 3.28. *Let N be a nested attribute. An FD $X \to Y$ on N is trivial if and only if $Y \leq X$ holds.*

Proof. If $Y \leq X$ holds, then the soundness of the *reflexivity axiom* from Definition 3.8 shows that every $r \subseteq dom(N)$ satisfies the FD $X \to Y$.

Let $X \to Y$ be a trivial dependency on N and suppose $Y \not\leq X$ holds. Define $r = \{t_1, t_2\} \subseteq dom(N)$ by

$$\pi_W^N(t_1) = \pi_W^N(t_2) \qquad \text{if and only if} \qquad W \leq X \tag{4}$$

according to Lemma 3.13. Since $X \leq X$ and $Y \not\leq X$ hold, it follows that $\not\models_r X \to Y$ by equation (4). This, however, contradicts the triviality of $X \to Y$. Hence, $Y \leq X$ must hold indeed. □

EXAMPLE 3.13. Examples for trivial FDs on Factor(Integer, Prime[Number], Exponent[Number]) from Example 3.2 are

Factor(Prime[Number]) → Factor(Prime[λ]),
Factor(Integer,Prime[Number], Exponent[λ]) → Factor(Integer, Exponent[λ]), or
Factor(Prime[Number],Exponent[Number]) → Factor(Exponent[Number]). □

3.3.2 The Notion of Redundancy

In the RDM, the definition of redundancy is based on viewing FDs not only as integrity constraints on a relation, but also as representing the fundamental units of information for retrieving and updating the data in a relation. This interpretation of the semantics of the information stored in a relation was implicit in the original study of normalisation by Codd [70], and has since been used in many aspects of database theory. A relation schema is defined to be redundant with respect to a given set of FDs if there exists a relation over the schema which satisfies all these FDs and which has at least two tuples which are identical on a fact. If we formalise this notion of redundancy, which goes back to [30], in the framework of nested attributes, then we obtain the following definition. Let N be a nested attribute and Σ a set of FDs on N. We call N *redundant with respect to* Σ if and only if there is some $r \subseteq dom(N)$ with $\models_r \Sigma$ and there are some $t_1, t_2 \in r$ with $t_1 \neq t_2$ and $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$ for some non-trivial FD $X \to Y \in \Sigma$. Intuitively, this notion of redundancy seems to make perfect sense.

EXAMPLE 3.14. Take a look at the FD

Factor(Prime[λ]) → Factor(Exponent[λ])

from Example 3.2. This is obviously a non-trivial FD. The elements

(72,[2,3],[3,2]) and
(108,[2,3],[2,3])

coincide on Factor(Prime[λ],Exponent[λ]), i.e., the FD above causes some redundancy according to the definition above. □

The last example shows that our current definition of redundancy is not really appropriate anymore. That is, the FD

$$\text{Factor(Prime}[\lambda]) \rightarrow \text{Factor(Exponent[Number])}$$

is not satisfied by the instance of Example 3.14 and, consequently, redundancy would need to be defined in terms of the non-maximal basis attribute Factor(Exponent[λ]). This, however, appears to be impossible as the fact represented by Factor(Exponent[λ]) will always be implicitly represented by Factor(Exponent[Number]). The point here is that the information in a non-maximal basis attribute Y cannot be separated from the information in any (maximal) basis attribute Z with $Y \leq Z$. This motivates the following definition.

Definition 3.29. Let $N \in \mathcal{NA}$ be a nested attribute and Σ a set of FDs on N. Let $\Sigma_{\text{inev}} \subseteq \Sigma^+$ denote the set of all $X \rightarrow Y \in \Sigma^+$ where

- $Y \leq X$ holds or
- Y is a non-maximal basis attribute of N.

The elements of the closure Σ^+_{inev} of Σ_{inev} under derivation with respect to the generalised Armstrong axioms are called *inevitable FDs* on N with respect to Σ. □

In the same way that trivial FDs were not considered as redundancy-causing dependencies in the RDM, inevitable FDs will not be considered as redundancy-causing dependencies in our framework. The following lemma characterises inevitable FDs which are derivable from a given set of FDs.

Lemma 3.30. *Let $N \in \mathcal{NA}$, Σ a set of FDs on N and $X \rightarrow Y \in \Sigma^+$. We have $X \rightarrow Y \in \Sigma^+_{inev}$ if and only if every $M \in MaxB(N)$ with $M \leq Y$ also satisfies $M \leq X$.*

Proof. Let $X \rightarrow Y \in \Sigma^+_{\text{inev}}$. Consider the proper chain

$$\Sigma_{\text{inev}} = \Sigma_0 \subset \Sigma_1 \subset \cdots \subset \Sigma_k = \Sigma^+_{\text{inev}}$$

where Σ_j results from Σ_{j-1}, for $1 \leq j \leq k$, by single application of one of the generalised Armstrong axioms from Definition 3.8. We proceed by induction on j. If $j = 0$ and $X \rightarrow Y \in \Sigma_{\text{inev}}$, then $Y \leq X$ or $Y \in NMaxB(N)$. In both cases the claim follows immediately.

Assume now that this property holds for all elements in Σ_j for some $j \geq 0$. Consider the single $X \rightarrow Y \in \Sigma_{j+1} - \Sigma_j$. If $X \rightarrow Y$ has been inferred using the *reflexivity axiom*, then $Y \leq X$ and the property holds again. If the *extension rule* was used, then $Y = X \sqcup Y'$ and $X \rightarrow Y' \in \Sigma_j$. If $M \in MaxB(N) \cap MaxB(Y)$, then $M \in MaxB(X)$ or $M \in MaxB(Y')$. In the latter case we can apply hypothesis and conclude that $M \in MaxB(X)$ as well. It

remains to consider the case where $X \to Y$ has been inferred using the *transitivity rule*, i.e, from $X \to Z, Z \to Y \in \Sigma_j$. If $M \in MaxB(N) \cap MaxB(Y)$, then $M \in MaxB(Z)$ by hypothesis applied to $Z \to Y \in \Sigma_j$, and then $M \in MaxB(X)$ by hypothesis applied to $X \to Z \in \Sigma_j$.

Conversely, we show that $X \to Y \in \Sigma^+_{\text{inev}}$, if $X \to Y \in \Sigma^+$ and for all $M \in MaxB(N)$ with $M \leq Y$ also $M \leq X$ holds. Let $Y_1 = \sqcup(MaxB(Y) \cap MaxB(N))$. Since $(MaxB(Y) \cap MaxB(N)) \subseteq (MaxB(X) \cap MaxB(N))$, it follows that $Y_1 \leq X$ and therefore $X \to Y_1 \in \Sigma_{\text{inev}}$. Note that $MaxB(Y) - MaxB(N) \subseteq NMaxB(N)$ holds. This implies $X \to Y' \in \Sigma_{\text{inev}}$ for every $Y' \in MaxB(Y) - MaxB(N)$. This gives $X \to Y_2 \in \Sigma^+_{\text{inev}}$ for $Y_2 = \sqcup(MaxB(Y) - MaxB(N))$ by the join rule. Applying the join rule again gives $X \to Y \in \Sigma^+_{\text{inev}}$ where $Y = Y_1 \sqcup Y_2$. □

Note that in Lemma 3.30, the condition that every $M \in MaxB(N)$ with $M \leq Y$ implies that $M \leq X$ holds is equivalent to $Y^{cc} \leq X$. We are now prepared to define a better notion of redundancy for nested attributes *in terms of FDs*. Informally, every FD which is not inevitable may cause redundancies.

Definition 3.31. Let N be a nested attribute and Σ a set of FDs on N. We call N *redundant with respect to* Σ if and only if there is some $r \subseteq dom(N)$ with $\models_r \Sigma$ and there are some $t_1, t_2 \in r$ with $t_1 \neq t_2$ and $\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$ for some FD $X \to Y \in \Sigma$ which is not inevitable on N with respect to Σ. □

EXAMPLE 3.15. According to Definition 3.31 of redundancy, the FD

$$\text{Factor(Prime}[\lambda]) \to \text{Factor(Exponent}[\lambda])$$

from Example 3.14 does not cause redundancies anymore. In fact, this FD is inevitable. □

Definition 3.31 considers only FDs in Σ itself. As in the RDM one might define redundancy with respect to all logical consequences of Σ, i.e., Σ^*. That is, N is called *redundant with respect to* Σ^* if and only if there is some $r \subseteq dom(N)$ with $\models_r \Sigma^*$ and there are some $t_1, t_2 \in r$ with $t_1 \neq t_2$ and $\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$ for some FD $X \to Y \in \Sigma^*$ which is not inevitable on N with respect to Σ. It can be proven, as in the RDM, that both notions are in fact the same. Note that according to Theorem 3.14 redundancy with respect to Σ^* means redundancy with respect to Σ^+.

Theorem 3.32. *Let N be a nested attribute and Σ a set of FDs on N. Then N is redundant with respect to Σ if and only if N is redundant with respect to Σ^*.*

Proof. It is easy to see that redundancy of N with respect to Σ is sufficient for the redundancy of N with respect to Σ^* since $\models_r \Sigma$ implies $\models_r \Sigma^*$ and $\Sigma \subseteq \Sigma^*$. It remains to show that redundancy of N with respect to Σ is also a necessary condition for N to be redundant with respect to Σ^*. Therefore, we assume that N is *non-redundant* with respect to Σ. This means that for all $r \subseteq dom(N)$ with $\models_r \Sigma$ and for all $t_1, t_2 \in r$ with

$\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$ for some $X \to Y \in \Sigma$, which is not inevitable on N with respect to Σ, follows $t_1 = t_2$. We will show that N is non-redundant with respect to Σ^+. Let therefore

$$\Sigma = \Sigma_0 \subset \Sigma_1 \subset \ldots \subset \Sigma_k = \Sigma^+$$

be a chain where Σ_j results from Σ_{j-1} by application of one of the generalised Armstrong axioms. What we show, in fact, is that Σ can be replaced by Σ_j. We proceed by induction on j. For $j = 0$ there is nothing to show. Let $j > 0$, i.e., $\Sigma_j - \Sigma_{j-1}$ consists of exactly one dependency $X \to Y$. If $X \to Y$ has been inferred using the *reflexivity axiom*, then $Y \leq X$ which means that $X \to Y$ is trivial and, therefore, also inevitable. The non-redundancy of N with respect to Σ_j follows therefore from the hypothesis that N is non-redundant with respect to Σ_{j-1} since there is nothing to show for $X \to Y$. Consider the case where $X \to Y$ has been inferred using the *extension rule*, i.e., $Y = X \sqcup Y'$ with $X \to Y' \in \Sigma_{j-1}$. If $X \to Y$ is inevitable, the statement follows from the hypothesis. Assume therefore that $X \to Y$ is not inevitable and suppose $\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$ for some $t_1, t_2 \in r$ with $r \subseteq dom(N)$ and $\models_r \Sigma_j$. It follows that $\pi^N_X(t_1) = \pi^N_X(t_2)$ and since $\models_r \Sigma_{j-1}$, and in particular $\models_r X \to Y'$, we obtain that $\pi^N_{X \sqcup Y'}(t_1) = \pi^N_{X \sqcup Y'}(t_2)$ holds. If $X \to Y'$ was inevitable, the extension rule would imply that $X \to Y$ is inevitable, too. Therefore, $X \to Y'$ is not inevitable. Now, we can apply the hypothesis and conclude that $t_1 = t_2$ holds. It follows that N is non-redundant with respect to Σ_j. Finally, consider the case where $X \to Y$ has been derived using the *transitivity rule* with $X \to Z$, $Z \to Y \in \Sigma_{j-1}$. Again, we assume that $\models_r \Sigma_j$ for some $r \subseteq dom(N)$ and $\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$ holds with $X \to Y$ not being inevitable. Since $\models_r \Sigma_{j-1}$, we conclude that $\pi^N_{X \sqcup Z}(t_1) = \pi^N_{X \sqcup Z}(t_2)$ and $\pi^N_{Z \sqcup Y}(t_1) = \pi^N_{Z \sqcup Y}(t_2)$ hold as well. If $X \to Z$ and $Z \to Y$ were both inevitable, then $X \to Y$ would be inevitable, too. It follows that at least one of $X \to Z$ or $Z \to Y$ is not inevitable. In either case we can apply hypothesis, and consequently $t_1 = t_2$. Again, N is not redundant with respect to Σ_j. This concludes the proof. □

Note that by Definition 3.29 an FD σ, defined on some nested attribute, is inevitable with respect to Σ if and only if σ is inevitable with respect to $\Sigma^* = \Sigma^+$ (Σ_j for any $j = 0, \ldots, k$).

The last theorem shows that the notion of redundancy is invariant under the choice of equivalent sets of FDs.

3.3.3 Boyce-Codd and Nested List Normal Form

The Boyce-Codd Normal Form has been introduced in [72] and intensively studied since then. A relation schema R is in BCNF if and only if it is non-redundant with respect to the set of FDs on R. One might therefore say that a well-designed relation schema should be in BCNF. The following definition extends BCNF to the framework of nested attributes.

Definition 3.33. Let N be some nested attribute and Σ a set of FDs on N. We say that N is in *Boyce-Codd Normal Form (BCNF)* with respect to Σ if and only if every $X \to Y \in \Sigma^*$ is trivial or X is a superkey for N with respect to Σ. □

We might now ask whether BCNF for a nested attribute is a sufficient and necessary condition for the non-redundancy of N. Clearly, a nested attribute in BCNF is non-redundant in the sense of Definition 3.31. The converse, however, is false.

EXAMPLE 3.16. Consider Example 3.2 again. We have seen that, according to Definition 3.31, the nested attribute N = Factor(Integer,Prime[Number],Exponent[Number]) is non-redundant with respect to the FDs given in Example 3.2. That is, every FD has a superkey on the left-hand side or is inevitable. On the other hand, however, N is not in BCNF with respect to the FDs given. That is, the FD Factor(Prime[λ]) $\rightarrow$ Factor(Exponent[λ]) is not trivial nor is Factor(Prime[λ]) a superkey for N. □

Example 3.16 shows that BCNF is not a necessary property for non-redundant nested attributes. Thus, BCNF is too strong to characterise non-redundant nested attributes and, therefore, we would like to find a weaker normal form.

Definition 3.34. Let N be some nested attribute and Σ a set of FDs on N. We say that N is in *Nested List Normal Form (NLNF)* with respect to Σ if and only if every $X \rightarrow Y \in \Sigma^*$ is an inevitable dependency on N with respect to Σ or X is a superkey for N with respect to Σ. □

Corollary 3.35. *Nested List Normal Form is strictly weaker than Boyce-Codd Normal Form.*

Proof. Every nested attribute that is in BCNF with respect to Σ, is also in NLNF with respect to Σ. This is due to the fact that every trivial FD is also inevitable. Since there are inevitable FDs which are not trivial, there are examples of nested attributes which are in NLNF, but not in BCNF with respect to some Σ. Such an example is given in Example 3.2. □

3.3.4 NLNF - The same fact is only stored once

We show that NLNF captures exactly those nested attributes which are non-redundant in the sense of Definition 3.31. This is a first and important semantic justification for NLNF.

Theorem 3.36. *Let N be a nested attribute and Σ a set of FDs on N. Then is N non-redundant with respect to Σ^* if and only if N is in NLNF with respect to Σ.*

Proof. Assume that N is in NLNF with respect to Σ. If N was redundant with respect to Σ^*, then there would be some $r \subseteq dom(N)$ with $\models_r \Sigma^*$ and $t_1, t_2 \in r$, $t_1 \neq t_2$ with $\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$ for some FD $X \rightarrow Y \in \Sigma^*$ which is not inevitable. In particular, $\pi^N_X(t_1) = \pi^N_X(t_2)$ holds as $X \leq X \sqcup Y$. Since N is in NLNF and $X \rightarrow Y$ is not inevitable it follows that X is a superkey for N. This implies $t_1 = t_2$ which is a contradiction. Therefore, N must be non-redundant with respect to Σ^*.

Assume N is non-redundant with respect to Σ^*. Let $X \rightarrow Y \in \Sigma^*$ be an FD which is not inevitable. Non-redundancy of N with respect to Σ^* implies that $t_1 = t_2$ for all

$t_1, t_2 \in r \subseteq dom(N)$ with $\models_r \Sigma^*$ and $\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$. This means that $X \sqcup Y$ is a superkey for N. If $\pi^N_X(t_1) = \pi^N_X(t_2)$ holds for some $t_1, t_2 \in r$, then $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ holds as well since $\models_r X \to Y$. According to the extension rule we then have $\models_r X \to X \sqcup Y$. This implies $\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$ and $t_1 = t_2$ follows. In other words, X is already a superkey for N. Since this is true for every $X \to Y \in \Sigma^*$ which is not inevitable we conclude that N is in NLNF with respect to Σ. □

3.3.5 Characterising NLNF

Given some nested attribute N and some set Σ of FDs on N, how can we decide whether N is in NLNF? According to Definition 3.34 one needs to examine whether every $X \to Y$ implied by Σ, i.e. every $X \to Y$ in Σ^*, is inevitable or whether X is a superkey for N with respect to Σ. This is not very practical, although the implication problem for FDs is efficiently decidable. However, we will show now that inspecting every FD in Σ suffices.

Theorem 3.37. *Let N be a nested attribute and Σ a set of FDs on N. N is in NLNF with respect to Σ if and only if every $X \to Y \in \Sigma$ is inevitable on N with respect to Σ or X is a superkey for N with respect to Σ.*

Proof. Obviously, if every $X \to Y$ in Σ^* is an inevitable dependency or X is a superkey, then the same is true for every FD in Σ since $\Sigma \subseteq \Sigma^*$. It is therefore sufficient to show that every $X \to Y$ in Σ^+ has superkey X or is an inevitable dependency, if the same is true for every FD in Σ. Consider again the proper chain

$$\Sigma = \Sigma_0 \subset \Sigma_1 \subset \cdots \subset \Sigma_k = \Sigma^+$$

where Σ_j results from Σ_{j-1}, $j > 0$, by a single application of one of the generalised Armstrong axioms. We show that there is already some FD in Σ_{j-1} which is not inevitable and where X is not a superkey, if there is some $X' \to Y'$ in Σ_j which is not inevitable and where X' is not a superkey. Let $j > 0$ and $X \to Y \in \Sigma_j - \Sigma_{j-1}$ not inevitable and X not a superkey. Since $X \to Y$ is not inevitable, it is in particular not a trivial dependency. This means $X \to Y$ has not been derived by the *reflexivity rule* according to Lemma 3.28.

Assume that $X \to Y$ has been derived by means of the *extension rule*, i.e., $Y = X \sqcup Y'$ and $X \to Y' \in \Sigma_{j-1}$. Obviously, $X \to Y'$ cannot be inevitable since $X \to Y$ would immediately be inevitable too. Moreover, X is not a superkey by assumption. Hence, $X \to Y' \in \Sigma_{j-1}$ is not inevitable and X is not a superkey.

Assume that $X \to Y$ has been derived by means of the *transitivity rule*, i.e., $X \to Z, Z \to Y \in \Sigma_{j-1}$. By definition of inevitable dependencies, at least one of $X \to Z, Z \to Y$ cannot be inevitable. On the other hand, neither X nor Z are superkeys. X is not a superkey by assumption. If Z was a superkey, then $X \to Z, Z \to N \in \Sigma^+$ and therefore also $X \to N \in \Sigma^+$ by transitivity. This means that X would be superkey, a contradiction. It is now immediate that one of $X \to Z, Z \to Y \in \Sigma_{j-1}$ is neither inevitable nor has a superkey on the left-hand side.

We have therefore shown that if there is an FD in Σ^+ which is not inevitable and where the left-hand side is not a superkey, then there is already an FD in Σ with this property. This concludes the proof. □

Theorem 3.37 tells us that NLNF is invariant under derivation (implication) of FDs, and therefore also invariant under different choices of equivalent sets of FDs. This guarantees that one is able to check efficiently whether a given nested attribute is non-redundant with respect to Σ^*.

EXAMPLE 3.17. The nested attribute Factor(Integer,Prime[Number],Exponent[Number]) is in NLNF with respect to the set Σ of FDs that are given in Example 3.2. Every functional dependency $X \to Y \in \Sigma$ is inevitable or X is a superkey for Factor(Integer,Prime[Number],Exponent[Number]) with respect to Σ. □

EXAMPLE 3.18. The nested attribute

DNA(Origin[Base],Count(A,C,G,T),Gene(Start,End,Sub[Nucleo],Translation[Amino]))

is not in NLNF with respect to the set Σ of FDs given in Example 3.3. The FD

$$\text{DNA(Origin[Base])} \to \text{DNA(Count(A,C,G,T))}$$

is neither inevitable nor is DNA(Origin[Base]) a superkey for the underlying nested attribute with respect to Σ. □

We will now give yet another characterisation of NLNF which will, in particular, give us a different proof of Theorem 3.37. The result extends a well-known result from [105] for relational databases. In order to verify whether a nested attribute N in NLNF satisfies all FDs given, one simply needs to check whether N satisfies all key dependencies and all inevitable FDs. This makes integrity checking more efficient and is another justification why nested attributes in NLNF are well-designed. Unlike the RDM where one simply needs to inspect all key dependencies for relation schemata in BCNF, one still needs to deal with all inevitable FDs when a nested attribute in NLNF is given. This is the price for introducing lists.

Theorem 3.38. *Let N be a nested attribute and Σ a set of FDs on N. N is in NLNF with respect to Σ if and only if every $r \subseteq dom(N)$ with $\models_r \Sigma_{key} \cup \Sigma^+_{inev}$ implies $\models_r \Sigma$.*

Proof. Assume there is some $r \subseteq dom(N)$ with $\models_r \Sigma_{\text{key}} \cup \Sigma^+_{\text{inev}}$, but $\not\models_r \Sigma$. Then there is some $X \to Y \in \Sigma$ which cannot be inevitable and where X is not a superkey. Since $\Sigma \subseteq \Sigma^*$, N cannot be in NLNF with respect to Σ.

Vice versa, assume that N is not in NLNF with respect to Σ. Then there is some $X \to Y \in \Sigma^+$ which is not inevitable and where X is not a superkey. We show that there is some $r \subseteq dom(N)$ with $\models_r \Sigma_{\text{key}} \cup \Sigma^+_{\text{inev}}$, but $\not\models_r \Sigma$. We define the closure $X^+_{\text{inev}} = \bigsqcup\{Z \mid X \to Z \in \Sigma^+_{\text{inev}}\}$ of X with respect to inevitable FDs. According to Lemma 3.13 we define some $r \subseteq dom(N)$ with $r = \{t, t'\}$ by

$$\pi^N_W(t') = \pi^N_W(t) \quad \text{if and only if} \quad W \leq X^+_{\text{inev}} \quad .$$

We show first that $\models_r \Sigma_{\text{key}}$. Let K be an arbitrary minimal key for N. Since X is not a superkey for N we have $X^+_{\text{inev}} \leq X^+ < N$. This implies that X^+_{inev} cannot be a superkey neither. Consequently, $K \not\leq X^+_{\text{inev}}$, and therefore $\pi^N_K(t') \neq \pi^N_K(t)$ by definition of r.

We show that $\models_r \Sigma^+_{\text{inev}}$ holds. Let $U \to V \in \Sigma^+_{\text{inev}}$. If $U \not\leq X^+_{\text{inev}}$, then $\pi^N_U(t') \neq \pi^N_U(t)$ and $\models_r U \to V$. Suppose $U \leq X^+_{\text{inev}}$ and, therefore, $\pi^N_U(t') = \pi^N_U(t)$. It follows from the soundness of the join rule that $X \to X^+_{\text{inev}} \in \Sigma^+_{\text{inev}}$. We have $X^+_{\text{inev}} \to U \in \Sigma^+_{\text{inev}}$ by reflexivity. Consequently, $X \to U \in \Sigma^+_{\text{inev}}$ by transitivity, too. Since $U \to V \in \Sigma^+_{\text{inev}}$, we derive $X \to V \in \Sigma^+_{\text{inev}}$ as well. This means $V \leq X^+_{\text{inev}}$ and we conclude $\pi^N_V(t') = \pi^N_V(t)$. Hence, $\models_r U \to V$.

We show finally that $\not\models_r \Sigma$. If $Y \leq X^+_{\text{inev}}$ held we would infer $X^+_{\text{inev}} \to Y \in \Sigma^+_{\text{inev}}$ by reflexivity, and $X \to Y \in \Sigma^+_{\text{inev}}$ by transitivity since also $X \to X^+_{\text{inev}} \in \Sigma^+_{\text{inev}}$ holds. This, however, is a contradiction. Therefore, $Y \not\leq X^+_{\text{inev}}$ and as $X \leq X^+_{\text{inev}}$ holds as well, it follows that $\pi^N_X(t') = \pi^N_X(t)$ and $\pi^N_Y(t') \neq \pi^N_Y(t)$. We conclude $\not\models_r \Sigma^*$ and consequently $\not\models_r \Sigma$. □

According to Theorem 3.38, if N is not in NLNF with respect to Σ, then there is some $r \subseteq dom(N)$ with $\models_r \Sigma_{\text{key}} \cup \Sigma^+_{\text{inev}}$, but $\not\models_r \Sigma$. This means there is some FD in Σ which is not inevitable and where the left-hand side is not a superkey. This gives an alternative proof for Theorem 3.37.

3.3.6 Update Anomalies

In the RDM, a relation schema in BCNF does not have any update anomalies. This is another justification why relation schemata should be in BCNF [42]. We will demonstrate that nested attributes in NLNF behave very similar. However, the next example reveals a fundamental difference.

EXAMPLE 3.19. Reconsider Example 3.2 with

$$\text{Factor(Integer,Prime[Number],Exponent[Number])}.$$

Recall that this nested attribute is non-redundant with respect to the FDs given. Say our database simply consists of the tuple

$$(12,[2,3],[2,1])$$

and the tuple $(35,[5,7],[1,1,0])$ is about to be inserted. Then obviously all key dependencies are still satisfied by the new relation, but the FD

$$\text{Factor(Prime}[\lambda]) \to \text{Factor(Exponent}[\lambda])$$

is not satisfied. Hence, it is insufficient to examine key dependencies only. □

Example 3.19 shows that, in general, the absence of redundancy for a nested attribute does not imply the absence of insertion anomalies. Therefore, it cannot be expected that nested attributes in NLNF do not have update anomalies. We define, however, strong update anomalies in the context of nested attributes. The main difference to the RDM is that updated relations which define any strong anomaly do not only satisfy all key dependencies on the nested attribute, but also all inevitable FDs. Deletion anomalies cannot occur with FDs and are therefore not defined.

Definition 3.39. Let N be a nested attribute and Σ a set of FDs on N.

1. We say that N has a *strong insertion anomaly* if and only if there is some $r \subseteq dom(N)$ with $\models_r \Sigma$ and some $t \notin r$ with $\models_{r \cup \{t\}} \Sigma_{\text{key}} \cup \Sigma^+_{\text{inev}}$, but $\not\models_{r \cup \{t\}} \Sigma$.
2. We say that N has a *strong replacement anomaly*
   - *of type 1* if and only if there is some $r \subseteq dom(N)$ with $\models_r \Sigma$ and some $t \in r$ and $t' \in dom(N)$ with $\pi^N_K(t) = \pi^N_K(t')$ for some minimal key K on N and $\models_{r-\{t\}\cup\{t'\}} \Sigma_{\text{key}} \cup \Sigma^+_{\text{inev}}$ and $\not\models_{r-\{t\}\cup\{t'\}} \Sigma$ hold.
   - *of type 2* if and only if there is some $r \subseteq dom(N)$ with $\models_r \Sigma$ and some $t \in r$ and $t' \in dom(N)$ with $\pi^N_K(t) = \pi^N_K(t')$ for some distinguished minimal key K on N and $\models_{r-\{t\}\cup\{t'\}} \Sigma_{\text{key}} \cup \Sigma^+_{\text{inev}}$ and $\not\models_{r-\{t\}\cup\{t'\}} \Sigma$ hold.
   - *of type 3* if and only if there is some $r \subseteq dom(N)$ with $\models_r \Sigma$ and some $t \in r$ and $t' \in dom(N)$ with $\pi^N_K(t) = \pi^N_K(t')$ for all minimal keys K on N and $\models_{r-\{t\}\cup\{t'\}} \Sigma_{\text{key}} \cup \Sigma^+_{\text{inev}}$ and $\not\models_{r-\{t\}\cup\{t'\}} \Sigma$ hold.

We say that N has a *strong update anomaly* if and only if N has a strong insertion or a strong replacement anomaly of some type. □

EXAMPLE 3.20. Consider the nested attribute

Paper(Lecturer, Course, Textbook)

together with the FDs

Paper(Lecturer, Course) → Paper(Textbook) and
Paper(Textbook) → Paper(Course).

A small snapshot over this nested attribute is the following:

(Kleene, Model Theory, Handbook of Mathematical Logic),
(Mostowski, Model Theory, Handbook of Mathematical Logic).

An insertion of the tuple

(Church, Recursion Theory, Handbook of Mathematical Logic)

leads to a new snapshot which satisfies all key dependencies and all inevitable dependencies (there are none), but the FD

Paper(Textbook) → Paper(Course)

is now violated. This defines an insertion anomaly. Consider now the snapshot

(Kleene, Axiomatic Set Theory, A course in Mathematical Logic),
(Church, Recursion Theory, Handbook of Mathematical Logic).

Replacing the element (Kleene, Axiomatic Set Theory, A course in Mathematical Logic) by (Kleene, Axiomatic Set Theory, Handbook of Mathematical Logic) leads to another snapshot which satisfies all key dependencies, but the FD

$$\text{Paper(Textbook)} \rightarrow \text{Paper(Course)}$$

is again violated. This defines therefore a replacement anomaly of type 1. If the minimal key Paper(Lecturer, Course) is the distinguished minimal key, then we even have a type-2 replacement anomaly. □

The next theorem generalises a result from [105]. It shows that NLNF is an exact condition for the absence of strong insertion anomalies.

Theorem 3.40. *Let N be a nested attribute and Σ a set of FDs on N. Then is N in NLNF if and only if N does not have any strong insertion anomaly.*

Proof. This follows from the proof of Theorem 3.38. In fact, if N has a strong insertion anomaly, then there must be some $X \rightarrow Y \in \Sigma \subseteq \Sigma^+$ which is not inevitable and where X is not a superkey. Consequently, N cannot be in NLNF. Vice versa, if N is not in NLNF, then there is some $X \rightarrow Y \in \Sigma^+$ which is not inevitable and where X is no superkey. We can now define $t, t' \in dom(N)$ exactly as we did in the proof of Theorem 3.38. Take then for instance $r = \{t'\}$ which obviously satisfies $\models_r \Sigma$. The proof of Theorem 3.38 shows then that $\models_{r \cup \{t\}} \Sigma_{\text{key}} \cup \Sigma^+_{\text{inev}}$, but $\not\models_{r \cup \{t\}} \Sigma$. □

NLNF is also an exact condition for the absence of type-1 replacement anomalies. This is an extension of a well-known result in relational databases [279].

Theorem 3.41. *Let N be a nested attribute and Σ a set of FDs on N. Then is N in NLNF if and only if N does not have any strong replacement anomaly of type 1.*

Proof. Obviously is N not in NLNF if N has a strong replacement anomaly of type 1. Let us assume that N is not in NLNF. Then there is some $X \rightarrow Y \in \Sigma^+$ which is not inevitable and where X is not a superkey. It follows by Lemma 3.30 that there is some $M \in MaxB(N)$ with $M \in MaxB(Y)$ and $M \notin MaxB(X)$. That means $X \rightarrow M \in \Sigma^+$ and $X \rightarrow M \notin \Sigma^+_{\text{inev}}$, again by Lemma 3.30. Let $X^+_M = \sqcup(\{Z \in SubB(N) : X \rightarrow Z \in \Sigma^+\} - \{M\})$. Define $t_0, t' \in dom(N)$ with

$$\pi^N_Z(t_0) = \pi^N_Z(t') \qquad \text{if and only if} \qquad Z \leq X^+_M.$$

Moreover, define $t \in dom(N)$ by $\pi^N_M(t) = \pi^N_M(t_0)$ and $\pi^N_Z(t) = \pi^N_Z(t')$ for all $Z \in MaxB(N) - \{M\}$. Since $M \in MaxB(N)$ the element t is well-defined. It follows then that

$$\pi^N_Z(t_0) = \pi^N_Z(t) \qquad \text{if and only if} \qquad Z \leq X^+.$$

Let $r = \{t_0, t\}$. We show first that $\models_r \Sigma$. Let $U \rightarrow V \in \Sigma$, and suppose $U \leq X^+$. We need to show that $V \leq X^+$ as well. We obtain $X^+ \rightarrow U \in \Sigma^+$ by the reflexivity axiom, and $X \rightarrow X^+ \in \Sigma^+$ by the join rule. Applying the transitivity rule a few times shows $X \rightarrow V \in \Sigma^+$. This means, by definition of X^+, that $V \leq X^+$.

Show next that $\models_{r - \{t\} \cup \{t'\}} \Sigma_{\text{key}} \cup \Sigma^+_{\text{inev}}$. Let K be some minimal key for N. From $K \leq X^+_M$ follows $K \leq X^+$, but X is not a superkey. This is a contradiction, i.e., $K \not\leq X^+_M$ which

means that $\pi_K^N(t_0) \neq \pi_K^N(t')$. Let $U \to V \in \Sigma^+_{\text{inev}}$. Assume further that $\pi_U^N(t_0) = \pi_U^N(t')$. If $M \leq V$, then $M \leq U$ by Lemma 3.30 which is a contradiction as $\pi_M^N(t_0) \neq \pi_M^N(t')$. Consequently, $M \not\leq V$ and therefore $V \leq X_M^+$. This shows $\pi_V^N(t_0) = \pi_V^N(t')$, i.e., $\models_{\{t_0,t'\}} U \to V$.

It is obvious that $\not\models_{r-\{t\}\cup\{t'\}} \Sigma$ since $X \leq X_M^+$ ($M \notin MaxB(X)$), but $M \not\leq X_M^+$, i.e., $\not\models_{r-\{t\}\cup\{t'\}} X \to M$.

It remains to show that there is some minimal key K such that $\pi_K^N(t) = \pi_K^N(t')$. Let $N_M = \sqcup(MaxB(N) - \{M\})$. From $X \to M \in \Sigma^+$ follows $X \sqcup N_M \to M \sqcup N_M \in \Sigma^+$. This is equivalent to $N_M \to N \in \Sigma^+$ since $X \leq N_M$ and $M \sqcup N_M = N$. It follows that N_M is a superkey, i.e., there is some minimal key $K \leq N_M$. As $\pi_{N_M}^N(t) = \pi_{N_M}^N(t')$ holds by definition of t it follows that $\pi_K^N(t) = \pi_K^N(t')$.

Consequently, there is some $r = \{t_0, t\} \subseteq dom(N)$ with $\models_r \Sigma$ and there are $t \in r, t' \in dom(N)$ with $\pi_K^N(t) = \pi_K^N(t')$ for some minimal key K for N such that $\models_{r-\{t\}\cup\{t'\}} \Sigma_{\text{key}} \cup \Sigma^+_{\text{inev}}$ and $\not\models_{r-\{t\}\cup\{t'\}} \Sigma$ hold. This means that N has a replacement anomaly of type 1. □

The following theorem also generalises a well-known result from [279].

Theorem 3.42. *Let N be a nested attribute and Σ a set of FDs on N. Then is N in NLNF if and only if N does not have any strong replacement anomaly of type 2.*

Proof. Obviously is N not in NLNF if N has a strong replacement anomaly of type 2. Let's assume that N is not in NLNF. The existence of some $X \to M \in \Sigma^+ - \Sigma^+_{\text{inev}}$ with $M \in MaxB(N)$, $M \notin MaxB(X)$ and where X is not a superkey for N follows as in the proof of Theorem 3.41. Let K be some distinguished minimal key for N.

If $M \notin MaxB(K)$, then we can proceed exactly as in the proof of Theorem 3.41. It remains to consider the case where $M \in MaxB(K)$. Let $Q \leq \sqcup(SubB(K) - MaxB(N))$ maximal with respect to $\leq$ and the property that $X \sqcup Q$ is not a superkey. Note that λ is not a superkey since K is a minimal key and $M \leq K$. Further define $G = ((X \sqcap K) \sqcup Q)^+$. Define $t_0, t, t' \in dom(N)$ with

1. $\pi_Z^N(t) = \pi_Z^N(t_0)$ if and only if $Z \leq G$,
2. $\pi_Z^N(t') = \pi_Z^N(t_0)$ if and only if $Z \leq (X \sqcup G)^+_{\text{inev}}$,
3. $\pi_K^N(t) = \pi_K^N(t')$.

We show that t_0, t, t' are well-defined, and in particular that t and t' can be chosen to coincide on K. The first two properties imply that $\pi_G^N(t) = \pi_G^N(t')$, in particular $\pi_{X\sqcap K}^N(t) = \pi_{X\sqcap K}^N(t')$ and $\pi_Q^N(t) = \pi_Q^N(t')$. We show that $SubB((X \sqcup G)^+_{\text{inev}}) - SubB(G)$ is disjoint to $SubB(K)$. Assume there is some $B \in SubB((X \sqcup G)^+_{\text{inev}}) - SubB(G)$ with $B \in SubB(K)$. It follows immediately that $B \notin SubB(X) - SubB(G)$ since $X \sqcap K \leq G$. This leaves us with $B \in SubB(K) - SubB(G)$ and $B \in NMaxB(N)$, or equivalently $B \in SubB(K) - MaxB(N)$ and $B \notin SubB(G)$. By definition of Q follows that $X \sqcup Q \sqcup B$ is a superkey. From $B \in SubB((X \sqcup G)^+_{\text{inev}})$ follows $B \leq (X \sqcup Q)^+$. Therefore, $X \sqcup Q$ is already a superkey, a contradiction to the choice of Q.

The claim is that t_0, t, t' define a replacement anomaly of type 2. It is rather easy to show that $\models_{\{t_0,t\}} \Sigma$ and $\models_{\{t_0,t'\}} \Sigma^+_{\text{inev}}$ hold (the tuples coincide on closed sets, respectively).

Assume t_0 and t' coincide on some minimal key K'. Then $K' \leq (X \sqcup G)^+_{\text{inev}}$, and $X \sqcup G \to K' \in \Sigma^+$. This implies that $X \sqcup Q \to K' \in \Sigma^+$ holds, i.e., $X \sqcup Q$ is some superkey, a contradiction to the choice of Q. Consequently, t_0 and t' differ on every minimal key K'.

It remains to show that $\not\models_{\{t_0,t'\}} X \to M$ holds. First of all, $\pi^N_X(t_0) = \pi^N_X(t')$ since $X \leq (X \sqcup G)^+_{\text{inev}}$. Recall that $M \leq K, M \notin SubB(X)$, and $M \in MaxB(N)$. From $M \in MaxB(N)$ follows $M \not\leq Q$ and therefore $M \not\leq (X \sqcap K) \sqcup Q$. Moreover, if $(X \sqcap K) \sqcup Q \to M \in \Sigma^+$ held, then K would not be a minimal key. It follows that $M \not\leq G$. From $M \in MaxB(N), M \notin MaxB(G), M \notin MaxB(X)$ and $X \sqcup G \to M \in \Sigma^+$ follows immediately $X \sqcup G \to M \notin \Sigma^+_{\text{inev}}$ by Lemma 3.30. This means $M \not\leq (X \sqcup G)^+_{\text{inev}}$ and therefore $\pi^N_M(t') \neq \pi^N_M(t_0)$. This concludes the proof. □

It remains to study strong type 3 replacement anomalies.

Lemma 3.43. *Let N be a nested attribute and Σ a set of FDs on N. If N is in NLNF, then N does not have any strong replacement anomaly of type 3.* □

Unlike the case of strong type 1 and strong type 2 replacement anomalies, the converse of Lemma 3.43 does not hold in general.

EXAMPLE 3.21. Consider again the nested attribute

Paper(Lecturer, Course, Textbook)

together with the FDs

Paper(Lecturer, Course) → Paper(Textbook) and
Paper(Textbook) → Paper(Course).

This nested attribute is not in NLNF with respect to the FDs given. However, the nested attribute does not have any strong replacement anomalies of type 3. In fact, Paper(Lecturer, Course) and Paper(Lecturer, Textbook) are both minimal keys. Consequently, every (modified) tuple t' with $\pi^N_K(t) = \pi^N_K(t')$ for all minimal keys K must be equal to t. This implies immediately that $r - \{t\} \cup \{t'\} = r$ cannot satisfy both $\models_r \Sigma$ and $\not\models_r \Sigma$ simultaneously. □

The final result follows immediately from the previous theorems.

Theorem 3.44. *A nested attribute in NLNF does not have any strong update anomalies. Strong replacement anomalies of type 1 coincide with strong replacement anomalies of type 2.* □

The results for strong update anomalies and NLNF are the same as for update anomalies and BCNF in the RDM. In summary, the results obtained for NLNF generalise all results from Theorem 1.4 for BCNF in the RDM.

3.4 Decomposition into NLNF

So far we have presented the Nested List Normal Form as a goal that is desirable to achieve in the database design process. We now tackle the problem how to actually obtain NLNF. This is an important problem since, in general, it cannot be expected that the first design of a large and complex database schema is already optimised. Having achieved an agreement on a suitable database schema that meets the requirements of all parties involved in the lengthy design process, one wants to avoid starting all over again just to satisfy design criteria. It is much more desirable to provide automatic tools that transform a first design into an equivalent database schema which is in normal form. Of couse, it cannot be expected at all that such tools exist.

There are two competing approaches to relational database design: the *decomposition* approach [70] and the *synthesis* approach [41, 49]. For a discussion and comparison of these two approaches see [102, 181]. In this section, we will focus on applying the decomposition approach to NLNF.

3.4.1 FDs and Decompositions

The first desirable property of a decomposition in relational databases is that it be a lossless join decomposition with respect to the given set of FDs. Informally, a decomposition $\{R_1, \ldots, R_n\}$ of a relation schema R is called lossless with respect to a given set Σ of FDs on R if every relation that satisfies all FDs in Σ is the natural join of its projections on the subschemata R_i, i.e., $r = \pi_{R_1}(r) \bowtie \cdots \bowtie \pi_{R_n}(r)$. This implies that one can project a relation onto the subschemata and then join the projections without losing or adding any information. In order to generalise this desirable property, we define the generalised natural join within our framework.

Definition 3.45. Let $N \in \mathcal{NA}$ and $X, Y \in Sub(N)$. Let $r_1 \subseteq dom(X)$ and $r_2 \subseteq dom(Y)$. Then $r_1 \bowtie r_2 = \{t \in dom(X \sqcup Y) \mid \exists t_1 \in r_1, t_2 \in r_2 . \pi_X^{X \sqcup Y}(t) = t_1 \text{ and } \pi_Y^{X \sqcup Y}(t) = t_2\}$ is called the *generalised natural join* $r_1 \bowtie r_2$ *of* r_1 and r_2. □

We will now show that any instance $r \subseteq dom(N)$ that satisfies an FD $X \to Y$ on N can be decomposed into its projections on $X \sqcup Y$ and $X \sqcup Y^C$ without loss of information. The *projection* $\pi_X(r)$ of $r \subseteq dom(N)$ on $X \in Sub(N)$ is defined as $\{\pi_X^N(t) \mid t \in r\}$.

Theorem 3.46. *Let $N \in \mathcal{NA}$, $r \subseteq dom(N)$ and $X \to Y$ an FD on N. If $\models_r X \to Y$, then $r = \pi_{X \sqcup Y}(r) \bowtie \pi_{X \sqcup Y^C}(r)$.*

Proof. One can see that $r \subseteq \pi_{X \sqcup Y}(r) \bowtie \pi_{X \sqcup Y^C}(r)$ is always satisfied. Let $t \in \pi_{X \sqcup Y}(r) \bowtie \pi_{X \sqcup Y^C}(r)$ and $\models_r X \to Y$. We show that $t \in r$. There are $t_1' \in \pi_{X \sqcup Y}(r)$ and $t_2' \in \pi_{X \sqcup Y^C}(r)$ with $t_1' = \pi_{X \sqcup Y}^N(t)$ and $t_2' = \pi_{X \sqcup Y^C}^N(t)$. That means there are $t_1, t_2 \in r$ with $t_1' = \pi_{X \sqcup Y}^N(t_1)$ and $t_2' = \pi_{X \sqcup Y^C}^N(t_2)$, i.e., $\pi_{X \sqcup Y}^N(t) = \pi_{X \sqcup Y}^N(t_1)$ and $\pi_{X \sqcup Y^C}^N(t) = \pi_{X \sqcup Y^C}^N(t_2)$. In particular, $\pi_X^N(t_1) = \pi_X^N(t_2)$ and as $t_1, t_2 \in r$ with $\models_r X \to Y$ holds, we conclude $\pi_Y^N(t_1) = \pi_Y^N(t_2)$ as well. Therefore, $\pi_{X \sqcup Y}^N(t_1) = \pi_{X \sqcup Y}^N(t_2)$ by Lemma 3.9 and therefore also $\pi_{X \sqcup Y}^N(t) = \pi_{X \sqcup Y}^N(t_2)$. Since also $\pi_{X \sqcup Y^C}^N(t) = \pi_{X \sqcup Y^C}^N(t_2)$ holds we conclude $t = t_2$ by Lemma 3.9. This means, $t \in r$. □

Theorem 3.46 suggests that the decomposition approach may be successfully applied to the class of FDs in the context of lists. It is important to note that the converse of Theorem 3.46 is wrong. Consider $N = L(A, B, C)$ with $r = \{(a, b, c), (a, b', c)\}$ and different $b, b' \in dom(B)$. Obviously, $\not\models_r L(A) \rightarrow L(B)$, but $\pi_{L(A,B)}(r) = \{(a, b, ok), (a, b', ok)\}$ and $\pi_{L(A,C)}(r) = \{(a, ok, c)\}$, i.e., $r = \pi_{L(A,B)}(r) \bowtie \pi_{L(A,C)}(r)$.

3.4.2 The Decomposition Algorithm

Given some nested attribute N and a set Σ of FDs defined on N, the decomposition approach aims at finding a set of subattributes of N each of which is in NLNF with respect to the corresponding set of all implied FDs on that subattribute. Moreover, any instance of N that satisfies all the FDs in Σ is the generalised natural join of its projections on all the subattributes, i.e., every valid database on N can be decomposed without loss of information.

Definition 3.47. Let $N \in \mathcal{N}\mathrm{A}$, $N_1, \ldots, N_k \in Sub(N)$, and Σ a set of FDs defined on N. The set $\{N_1, \ldots, N_k\}$ is called a *lossless join decomposition of N with respect to Σ* if and only if $N = \bigsqcup\{N_1, \ldots, N_k\}$ and $r = \pi_{N_1}(r) \bowtie \cdots \bowtie \pi_{N_k}(r)$ holds for all $r \subseteq dom(N)$ with $\models_r \Sigma$. The set $\{N_1, \ldots, N_k\}$ is called a *lossless NLNF decomposition of N with respect to Σ* if and only if $\{N_1, \ldots, N_k\}$ is a lossless join decomposition of N with respect to Σ and N_i is in NLNF with respect to $\pi_{N_i}(\Sigma^+)$ for every $i = 1, \ldots, k$, and where $\pi_M(\Sigma) = \{X \rightarrow Y \in \Sigma \mid X \sqcup Y \leq M\}$. □

The lossless join property guarantees that the information of any legal instance over the original nested attribute can be obtained by joining the information on all decomposed subattributes. An NLNF decomposition guarantees moreover that every decomposed subattribute is in NLNF with respect to the projected sets of dependencies.

We will now show that it is possible to obtain a lossless NLNF decomposition for any given nested attribute N and any given set of FDs on N. Whenever an FD in the current state of the output schema violates NLNF, the decomposition algorithm removes the cause for this violation of NLNF by replacing the offending parent subattribute by two of its proper child subattributes which can be joined losslessly to reconstruct their parent.

Algorithm 3.4.1 (Lossless NLNF decomposition)

Input: $\boldsymbol{N \in \mathcal{N}}\mathrm{A}$, set $\boldsymbol{\Sigma}$ of FDs on $\boldsymbol{N}$

Output: set $\boldsymbol{\mathcal{S} = \{(N_1, \Sigma_1), \ldots, (N_k, \Sigma_k)\}}$ where $\boldsymbol{\Sigma_i}$ is set of FDs on $\boldsymbol{N_i \in Sub(N)}$ and $\boldsymbol{\{N_1, \ldots, N_k\}}$ is lossless NLNF decomposition of $\boldsymbol{N}$ with respect to $\boldsymbol{\Sigma}$

Method:

VAR $X, Y \in Sub(N)$

DECOMPOSE(N,Σ)

(1) **BEGIN**
(2) **IF** N in NLNF with respect to Σ, **THEN** $\mathcal{S} := \{(N, \Sigma)\}$;
(3) **ELSE**
(4) **LET** $X \to Y \in \Sigma$ be not inevitable on N with respect to Σ and $\Sigma \not\models X \to N$;
(5) $N_1 := X \sqcup Y$;
(6) $\mathcal{S} := \mathbf{DECOMPOSE}(N_1, \pi_{N_1}(\Sigma^+))$;
(7) $N_2 := X \sqcup Y^C$;
(8) $\mathcal{S} := \mathcal{S} \cup \mathbf{DECOMPOSE}(N_2, \pi_{N_2}(\Sigma^+))$;
(9) **ENDIF**;
(10) **RETURN**($\mathcal{S}$);
(11) **END**;

□

Theorem 3.48. *Algorithm 3.4.1 is correct.*

Proof. The algorithm terminates with an NLNF decomposition. In each decomposition step $N_1 \sqcup N_2 = N$, i.e., $N = \bigsqcup\{N_1, \ldots, N_k\}$ upon termination of the algorithm. Moreover, the algorithm continues the decomposition process whenever N_i is not in NLNF with respect to $\pi_{N_i}(\Sigma^+)$. Therefore, $\{N_1, \ldots, N_k\}$ is an NLNF decomposition.

It remains to show that $r = \pi_{N_1}(r) \bowtie \cdots \bowtie \pi_{N_k}(r)$ holds for any $r \subseteq dom(N)$ with $\models_r \Sigma$ as well. Therefore, we consider the case where $X \to Y$ is some FD on M which is not inevitable on M with respect to $\pi_M(\Sigma^+)$ and where X is not a superkey for M with respect to $\pi_M(\Sigma^+)$. Since $\models_{\pi_M(r)} X \to Y$, it follows that $\pi_M(r) = \pi_{X \sqcup Y}(\pi_M(r)) \bowtie \pi_{X \sqcup Y^C}(\pi_M(r))$ by Theorem 3.46. This, however, is equivalent to $\pi_M(r) = \pi_{X \sqcup Y}(r) \bowtie \pi_{X \sqcup Y^C}(r)$. This shows that the NLNF decomposition $\{N_1, \ldots, N_k\}$ is indeed lossless. □

One may replace line (5) of Algorithm 3.4.1 by $N_1 := X \sqcup Y^{CC}$. This would eliminate those non-maximal basis attributes of N in Y which do not have a superattribute in Y that is also a maximal basis attribute of N. Such non-maximal basis attributes are also subattributes of Y^C, anyway. Since $X \to Y$ is not inevitable, at least one maximal basis attribute of N is a subattribute of Y by Lemma 3.30, i.e., $Y^{CC} \neq \lambda$. The resulting algorithm is still correct as $Y^{CC} \sqcup Y^C = N$ and $\models_r X \to Y$ implies $\models_r X \to Y^{CC}$ by the subattribute rule as $Y^{CC} \leq Y$. We illustrate Algorithm 3.4.1 with the following abstract example. We say that a set Σ of FDs is *covered* by another set Θ of FDs, if every FD in Σ is implied by Θ.

EXAMPLE 3.22. Suppose $N = L(A, K[M(B, C, D)], P[Q[R(E, F)]])$ with $\Sigma = \{L(A) \to L(K[M(B, C)], P[\lambda]),\ L(K[M(D)]) \to L(P[Q[R(E)]])\}$. Obviously, N is not in NLNF with respect to Σ. Neither of the two given FDs is inevitable nor are the two left-hand sides superkeys for N with respect to Σ. We choose to decompose along $L(A) \to L(K[M(B, C)], P[\lambda])$ and obtain new nested attributes $N_1 = L(A, K[M(B, C, \lambda)], P[\lambda])$ and $N' = L(A, K[M(\lambda, \lambda, D)], P[Q[R(E, F)]])$. The projection of Σ on N_1 is covered by $L(A) \to L(K[M(B, C)], P[\lambda])$, the projection on N' covered by $\{L(K[M(D)]) \to L(P[Q[R(E)]]); L(A) \to L(K[\lambda], P[\lambda])\}$. One can see that N_1 is in NLNF with respect

to $\pi_{N_1}(\Sigma^+)$ as $L(A)$ is a superkey for N_1. However, N' is not in NLNF with respect to $\pi_{N'}(\Sigma^+)$ since $L(K[M(D)]) \rightarrow L(P[Q[R(E)]])$ is not inevitable and $L(K[M(D)])$ is not a superkey for N' with respect to $\pi_{N'}(\Sigma^+)$. The next decomposition step gives $N_2 = L(\lambda, K[M(\lambda, \lambda, D)], P[Q[R(E, \lambda)]])$ and $N_3 = L(A, K[M(\lambda, \lambda, D)], P[Q[R(\lambda, F)]])$. Now, $\pi_{N_2}(\Sigma^+)$ is covered by $L(K[M(D)]) \rightarrow L(P[Q[R(E)]])$, i.e., N_2 is in NLNF with respect to $\pi_{N_2}(\Sigma^+)$. Furthermore,$\pi_{N_3}(\Sigma^+)$ is covered by $L(A) \rightarrow L(K[\lambda], P[\lambda])$ which is inevitable on N_3 with respect to $\pi_{N_3}(\Sigma^+)$, i.e., N_3 is in NLNF with respect to $\pi_{N_3}(\Sigma^+)$. The output of Algorithm 3.4.1 is therefore $\{(N_1, \pi_{N_1}(\Sigma^+)), (N_2, \pi_{N_2}(\Sigma^+)), (N_3, \pi_{N_3}(\Sigma^+))\}$. See Figure 3.1 for an illustration. □

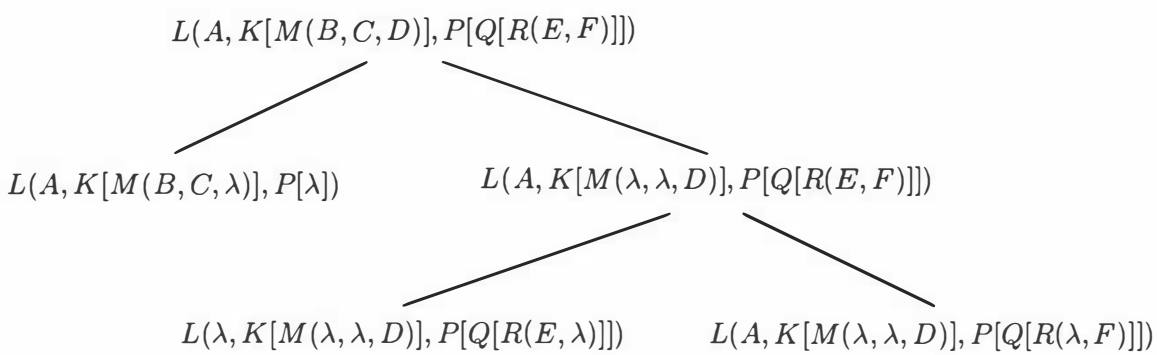

Fig. 3.1. NLNF decomposition Tree of Example 3.22.

EXAMPLE 3.23. Example 3.18 has shown that the nested attribute $N =$

DNA(Origin[Base],Count(A,C,G,T),Gene(Start,End,Sub[Nucleo],Translation[Amino]))

is not in NLNF with respect to the set Σ of FDs from Example 3.3. Since

$$\text{DNA(Origin[Base])} \rightarrow \text{DNA(Count(A,C,G,T))}$$

is neither inevitable nor is DNA(Origin[Base]) a superkey for N with respect to Σ, we decompose N into $N_1' =$ DNA(Origin[Base],Count(A,C,G,T)) and $N_2' =$ DNA(Origin[Base], Gene(Start,End,Sub[Nucleo],Translation[Amino])). The FD

$$\text{DNA(Origin[}\lambda\text{],Count(A,C,G))} \rightarrow \text{DNA(Count(T))}$$

is in $\pi_{N_1'}(\Sigma^+)$, but is neither inevitable nor is DNA(Origin[λ],Count(A,C,G)) a superkey for N_1' with respect to $\pi_{N_1'}(\Sigma^+)$. Therefore, we decompose N_1' into $N_1 =$ DNA(Origin[λ],Count(A,C,G,T)) and $N_2 =$ DNA(Origin[Base],Count(A,C,G)). The projected FDs $\pi_{N_1}(\Sigma^+)$ are covered by the set Σ_1 with the following FDs:

- DNA(Origin[λ],Count(A,C,G)) $\rightarrow$ DNA(Count(T)),
- DNA(Origin[λ],Count(A,C,T)) $\rightarrow$ DNA(Count(G)),
- DNA(Origin[λ],Count(A,G,T)) $\rightarrow$ DNA(Count(C)),

- DNA(Origin[λ],Count(C,G,T)) $\rightarrow$ DNA(Count(A)), and
- DNA(Count(A,C,G,T)) $\twoheadrightarrow$ DNA(Origin[λ]).

The projected FDs $\pi_{N_2}(\Sigma^+)$ are covered by Σ_2 = {DNA(Origin[Base]) $\rightarrow$ DNA(Count(A,C,G))}. N_1 is in NLNF with respect to Σ_1 and N_2 is in NLNF with respect to Σ_2. The FD

$$\text{DNA(Gene(Sub[Nucleo])) } \rightarrow \text{ DNA(Gene(Translation[Amino]))}$$

is an element of $\pi_{N_2'}(\Sigma^+)$, but is neither inevitable nor is DNA(Gene(Sub[Nucleo])) a superkey for N_2' with respect to $\pi_{N_2'}(\Sigma^+)$. Therefore, we decompose N_2' into N_3 = DNA(Gene(Sub[Nucleo],Translation[Amino])) and N_3' = DNA(Origin[Base],Gene(Start,End,Sub[Nucleo])). The projected FDs $\pi_{N_3}(\Sigma^+)$ are covered by the set Σ_3 with the following FDs:

- DNA(Gene(Sub[Nucleo])) $\rightarrow$ DNA(Gene(Translation[Amino])),
- DNA(Gene(Sub[λ])) $\twoheadrightarrow$ DNA(Gene(Translation[λ])), and
- DNA(Gene(Translation[λ])) $\rightarrow$ DNA(Gene(Sub[λ])).

N_3 is again in NLNF with respect to Σ_3. The FD

$$\text{DNA(Gene(Start,Sub[}\lambda\text{])) } \rightarrow \text{ DNA(Gene(End))}$$

is in $\pi_{N_3'}(\Sigma^+)$, but is neither inevitable nor is DNA(Gene(Start,Sub[λ])) a superkey for N_3' with respect to $\pi_{N_3'}(\Sigma^+)$. We decompose N_3' into N_4 = DNA(Gene(Start,End,Sub[λ])) and N_5 = DNA(Origin[Base],Gene(Start,Sub[Nucleo])). The projected FDs $\pi_{N_4}(\Sigma^+)$ are covered by the set Σ_4 with the following FDs:

- DNA(Gene(Start,Sub[λ])) $\rightarrow$ DNA(Gene(End)),
- DNA(Gene(End,Sub[λ])) $\rightarrow$ DNA(Gene(Start)), and
- DNA(Gene(Start,End)) $\rightarrow$ DNA(Gene(Sub[λ])).

N_4 is in NLNF with respect to Σ_4. The projected FDs $\pi_{N_5}(\Sigma^+)$ are covered by the set Σ_5 = {DNA(Origin[Base],Gene(Start,Sub[λ])) $\rightarrow$ DNA(Gene(Sub[Nucleo]))}. N_5 is in NLNF with respect to Σ_5. The output of Algorithm 3.4.1 is therefore $\{(N_1, \Sigma_1)), (N_2, \Sigma_2), (N_3, \Sigma_3)), (N_4, \Sigma_4)), (N_5, \Sigma_5)\}$. See Figure 3.2 for an illustration. □

For relational databases it is well-known that any relation schema with any set of FDs defined on it can be decomposed into subschemata that are all in BCNF with respect to the projected sets of FDs. In the presence of lists, however, the situation is different. Of course, one can modify Definition 3.47 of lossless NLNF decomposition to lossless BCNF decomposition for any nested attributes. Consider the nested attribute $N = L[A]$ where the set Σ of FDs on N simply consists of the single FD $\lambda \twoheadrightarrow L[\lambda]$. The FD is not trivial and λ is not a superkey for N with respect to Σ. Consequently, N is not in BCNF with respect to Σ. However, any decomposition of $L[A]$ must contain the nested attribute $L[A]$ itself. Therefore, no lossless BCNF decomposition of $L[A]$ with respect to Σ exists.

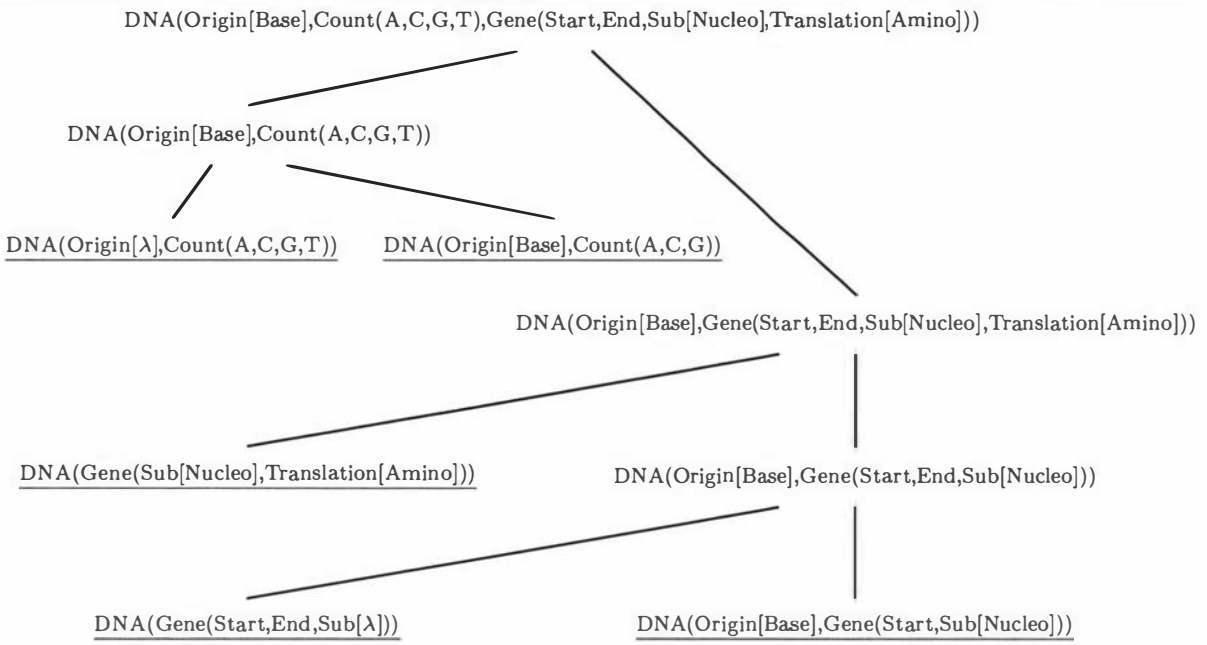

Fig. 3.2. NLNF decomposition Tree of Example 3.23.

3.4.3 Problems with NLNF decomposition

Algorithm 3.4.1 generalises the well-known BCNF decomposition algorithm for relational databases, see for instance [181, p.270]. It follows that the NLNF decomposition algorithm causes at least as many problems as its relational counterpart. The first problem is that Algorithm 3.4.1 does not execute in time polynomial in the sizes of N and Σ since computing a cover of $\pi_{N_i}(\Sigma^+)$ is intractable [33]. Changing the computations of $\pi_{N_1}(\Sigma^+)$ and $\pi_{N_2}(\Sigma^+)$ in lines (6) and (8) of Algorithm 3.4.1, respectively, to polynomial-time computations of $\pi_{N_1}(\Sigma)$ and $\pi_{N_2}(\Sigma)$ in the size of Σ leads to an algorithm which may not always output an NLNF decomposition. For example, let $\Sigma = \{L(A) \to L(B), L(B) \to L(C)\}$ be a set of FDs defined on $N = L(A, B, C, D)$. Then, $\pi_M(\Sigma)$ contains only trivial FDs for $M = L(A, C, D)$, but the FDs in $\pi_M(\Sigma^+)$ are covered by $\{L(A) \to L(C)\}$. It follows that if the FD, $L(A) \to L(B)$ is chosen at line (4) of Algorithm 3.4.1, then M, which is not in NLNF with respect to $\pi_M(\Sigma^+)$, is in the output decomposition. Furthermore, the cardinality of the decomposition returned by Algorithm 3.4.1 may be exponential in the cardinality of N [181, p. 271]. While checking whether N itself is in NLNF with respect to Σ can be done in polynomial time in the size of N and Σ (Theorem 3.37 and Lemma 3.30), checking whether a proper subattribute $N_i \in Sub(N)$ is in NLNF with respect to $\pi_{N_i}(\Sigma^+)$ is harder. The following theorem follows from Corollary 3 in [29].

Theorem 3.49. *Let $N \in \mathcal{NA}$ and Σ a set of FDs on N. The problem of deciding whether an arbitrary $N_i \in Sub(N)$ is in NLNF with respect to $\pi_{N_i}(\Sigma^+)$ is coNP-complete.*

Proof (Sketch). The problem of deciding whether an arbitrary $N_i \in Sub(N)$ is not in NLNF with respect to $\pi_{N_i}(\Sigma^+)$ is in *NP*. According to Theorem 3.37 one guesses non-

deterministically an FD $X \to Y \in \pi_{N_i}(\Sigma^+)$ and verifies in polynomial time that $X \to Y$ is not inevitable on N_i with respect to $\pi_{N_i}(\Sigma^+)$ and that X is not a superkey for N_i with respect to $\pi_{N_i}(\Sigma^+)$. Following Lemma 3.30, $X \to Y$ is not inevitable, if there is some $Y' \in MaxB(N_i)$ with $Y' \leq Y$ and $Y' \not\leq X$.

It remains to show that the problem of deciding whether an arbitrary $N_i \in Sub(N)$ is not in NLNF with respect to $\pi_{N_i}(\Sigma^+)$ is NP-hard. One can use the polynomial-time reduction of the hitting set problem [122] in [29, p.55-57] to the decision problem whether an arbitrary subschema of a relation schema is not in BCNF with respect to the corresponding projected set of given FDs. This is possible since every relational subschema can be represented using only null, flat and record-valued attributes, and NLNF and BCNF are equivalent in the absence of lists. Note that in this case inevitable FDs are simply trivial FDs. □

For relational databases a polynomial-time algorithm in the sizes of a relation schema R and Σ that outputs a lossless BCNF decomposition with respect to Σ has been proposed in [270]. It is the subject of future research to generalise this algorithm to the context of lists.

We have seen that it is always possible to achieve a lossless NLNF decomposition. Unfortunately, losslessness is not the only desirable property of a decomposition. An output $\{(N_1, \Sigma_1), \ldots, (N_k, \Sigma_k)\}$ should only be considered equivalent to (N, Σ) in case the semantic information in $\bigcup_{i=1}^{k} \Sigma_i$ is equivalent to the semantic information in Σ. This means that it is not only necessary not to lose any information regarding the database itself, but also not to lose any information regarding the semantic properties that this database carries. In other words, the dependencies must have been preserved at the end of the decomposition process.

Definition 3.50. Let $N \in \mathcal{NA}$ and Σ a set of FDs on N. A lossless join decomposition $\{N_1, \ldots, N_k\}$ of N is called *dependency-preserving* with respect to Σ if and only if $\Sigma^* = \left(\bigcup_{i=1}^{k} \pi_{N_i}(\Sigma^*)\right)^*$. □

We can see that the lossless NLNF decomposition in Example 3.22 is indeed dependency-preserving. What about the decomposition of our GenBank example?

EXAMPLE 3.24. Consider the decomposition of the GenBank database from Example 3.23. Define Θ as $\bigcup_{i=1}^{5} \pi_{N_i}(\Sigma^+)$. The decomposition is dependency-preserving if and only if $\Sigma^+ \subseteq \Theta^+$. All FDs in Σ are also in Θ except

$$\text{DNA(Origin[Base])} \to \text{DNA(Count(A,C,G,T))}$$

and

$$\text{DNA(Origin[Base],Gene(Start,End))} \to \text{DNA(Gene(Sub[Nucleo]))}.$$

The closure of DNA(Origin[Base]) with respect to Θ^+ is DNA(Origin[Base],Count(A,C,G,T)), i.e., the first FD is also in Θ^+. The closure of DNA(Origin[Base],Gene(Start,End)) with respect to Θ^+ is again N, i.e., the second FD is in Θ^+, too. Consequently, $\Sigma^+ \subseteq \Theta^+$ holds and the decomposition of the GenBank example is dependency-preserving. □

Unfortunately, our examples are exceptions. For relational databases it has been shown in [26, 29, 273] that there may be no decomposition of a relation schema into BCNF that is dependency-preserving. This negative result carries immediately over to the framework of lists, see Theorem 3 of [29].

Theorem 3.51. *There are nested attributes N and sets Σ of FDs on N for which no dependency-preserving and lossless NLNF decomposition exists.*

Proof. Let $N = L(A, B, C)$ and $\Sigma = \{L(A, B) \to L(C), L(C) \to L(B)\}$. By a brute force examination of Σ^+ it can be shown that $L(A, B) \to L(C)$ is in every non-redundant cover of Σ. Therefore, in any dependency-preserving and lossless join decomposition of N with respect to Σ, one of the subattributes of N must be $L(A, B, C)$, but this nested attribute is not in NLNF. □

Following [29], and using the same polynomial-time reduction of the hitting set problem [122] as in the proof of Theorem 3.49 it can be shown that the problem whether there exists a dependency-preserving and lossless NLNF decomposition for an arbitrary nested attribute is *NP*-hard.

For relational databases, an exponential algorithm in the size of Σ which decides the problem whether there is a dependency-preserving and lossless BCNF decomposition can be found in [214]. A method of guaranteeing a dependency-preserving decomposition which is in BCNF was proposed in [157], wherein it was shown that by adding attributes to R and FDs to Σ it is always possible to obtain a BCNF dependency-preserving decomposition of the augmented schema with respect to the augmented set of FDs.

In summary, obtaining a dependency-preserving and lossless NLNF decomposition is in general an unrealistic goal. In relational database theory, research on the third normal form (3NF) [186, 304] has shown that a lossless join decomposition that preserves dependencies can always be found [41, 49]. Note, however, that 3NF cannot guarantee the absence of redundancies. It is again subject of future research to extend these results to the framework of lists.

Further open problems that warrant future research are discussed in Section 6.2.

Chapter 4

Functional and Multi-valued Dependencies in the Presence of Lists

According to [87], functional dependencies constitute about two thirds of all uni-relational dependencies used in practical applications. A further important class of relational dependencies are so called multi-valued dependencies (MVDs). The class of FDs and MVDs covers around 75 percent of all uni-relational dependencies in practice [87]. It is the goal of this chapter to extend the theory of MVDs from the relational data model to the presence of null, flat, record-, and list-valued attributes.

We have seen in the previous chapter that an instance satisfying the FD $X \to Y$ can be decomposed into $X \sqcup Y$ and $X \sqcup Y^{\mathcal{C}}$ without losing information. Since such a lossless decomposition of some instance does not always imply that this instance satisfies a respective FD the question arises whether there is a class of dependencies that precisely captures this property. For relational databases, an affirmative answer to that question is given by the class of MVDs. They subsume the class of FDs and may also cause redundancies in the representation of data and abnormal update behavior. It is therefore desirable to investigate the impact of the list constructor on the class of MVDs as well.

In this chapter we will formally introduce MVDs in the presence of lists. We will show that an instance satisfies the MVD $X \twoheadrightarrow Y$ precisely when this instance is the generalised natural join of $X \sqcup Y$ and $X \sqcup Y^{\mathcal{C}}$. Then we study axiomatisability and implication problem for the class $\mathcal{C}$ of FDs and MVDs in the presence of records and lists. Here, a surprising difference to the RDM is revealed. MVDs imply non-trivial FDs in the context of lists which is impossible in relational databases. Using this fact and the algebraic framework from Chapter 2 the theory of FDs and MVDs can be generalised from the RDM to the presence of lists. Next, the independence of the inference rules is studied, and further interesting differences to the RDM are revealed. Subsequently, the role of the Brouwerian complement rule is investigated. This rule is special because it does not have a counterpart in the context of FDs. Finally, a provably-correct and polynomial-time algorithm for deciding implication of FDs and MVDs in the presence of lists is proposed. The algorithm naturally extends the well-known membership algorithm of Beeri [27].

The axiomatisation of FDs and MVDs is published in [146], the axiomatisation of MVDs

in [142], and the membership algorithm for FDs and MVDs appears in [141].

4.1 Axiomatisation

Multi-valued dependencies have been independently introduced in [86, 103, 303]. They have been axiomatised in [32]. In this section we will extend the generalised Armstrong axioms to obtain a finite axiomatisation for the class $\mathcal{C}$ of FDs and MVDs in the presence of null, flat, record- and list-valued attributes. This axiomatisation is a natural extension of the axiomatisation in the relational case (compare for instance to [220, pp. 80,81]). A fundamental difference will be the fact that the non-trivial FD $X \to Y \sqcap Y^{\mathcal{C}}$ is implied by the MVD $X \twoheadrightarrow Y$.

4.1.1 Definition and First Results

As it was the case for FDs, the algebraic framework from Chapter 2 allows to naturally extend the definition of multi-valued dependencies from the RDM.

Definition 4.1. Let $N \in \mathcal{N}\mathrm{A}$ be a nested attribute. A *multi-valued dependency on* N is an expression of the form $X \twoheadrightarrow Y$ where $X, Y \in Sub(N)$. A set $r \subseteq dom(N)$ *satisfies* the multi-valued dependency $X \twoheadrightarrow Y$ on N if and only if for all values $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$ there is a value $t \in r$ with $\pi^N_{X \sqcup Y}(t) = \pi^N_{X \sqcup Y}(t_1)$ and $\pi^N_{X \sqcup Y^{\mathcal{C}}}(t) = \pi^N_{X \sqcup Y^{\mathcal{C}}}(t_2)$. □

Intuitively, an instance r exhibits the MVD $X \twoheadrightarrow Y$ whenever the value on X determines the set of values on Y independently from the set of values on $Y^{\mathcal{C}}$. If there are two elements t_1, t_2 in r with the same projections on X, then there is also an element in r which has the same projection on $X \sqcup Y$ as t_1 and the same projection on $X \sqcup Y^{\mathcal{C}}$ as t_2. We will illustrate the concept of an MVD by the following example.

EXAMPLE 4.1. Consider Example 3.1 where Pubcrawl(Person,Visit[Drink(Beer,Pub)]) was the nested attribute. Suppose the snapshot r is now extended to

{ (Sven, [(Lübzer, Deanos), (Kindl, Highflyers)]),
(Sven, [(Kindl, Deanos), (Lübzer, Highflyers)]),
(Klaus-Dieter, [(Guiness, Irish Pub), (Speights, 3Bar),(Guiness, Irish Pub)]),
(Klaus-Dieter, [(Kölsch, Irish Pub), (Bönnsch, 3Bar), (Guiness, Irish Pub)]),
(Klaus-Dieter, [(Guiness, Highflyers), (Speights, Deanos), (Guiness, 3Bar)]),
(Klaus-Dieter, [(Kölsch, Highflyers), (Bönnsch, Deanos), (Guiness, 3Bar)]),
(Sebastian, [ ]) }.

Obviously, the FD Pubcrawl(Person) → Pubcrawl(Visit[Drink(Pub)]) is not satisfied by r, and neither is the FD Pubcrawl(Person) → Pubcrawl(Visit[Drink(Beer)]). However, $\models_r$ Pubcrawl(Person) ↠ Pubcrawl(Visit[Drink(Pub)]). This MVD says informally that a person has preferred lists of pubs, e.g. according to the weekday, and preferred lists of beers, e.g. according to the mood that person is in. Since a weekday is independent from

the mood of a person, all possible combinations of these lists can occur. That is, the lists of pubs a person visits is independent from the lists of beers that person drinks. It appears that $\models_r$ Pubcrawl(Person) $\rightarrow$ Pubcrawl(Visit[λ]) holds as well. This means informally that in this snapshot each person visits a fixed number of pubs (and drinks a fixed number of beers). □

The intuitive meaning of satisfaction of an MVD from the RDM is here naturally extended to more complex objects, in this case any nesting that involves null, flat, record- or list-valued attributes.

EXAMPLE 4.2. Consider Example 2.3 where the nested attribute

Align(St1[Seq1],St2[Seq2],Num1(Occ1,Nuc1),Num2(Occ2,Nuc2),Comp[Pair(N1,N2)])

was used as a description of a database that compares two nucleotide sequences each having a certain characteristic. The constraint that was informally described in Section 1.2.2 is now formally specified as

$$\text{Align(St1[Seq1],St2[Seq2],Num1(Occ1,Nuc1),Num2(Occ2,Nuc2))} \twoheadrightarrow \text{Align(Comp[Pair(N1,}\lambda\text{)]).}$$

Note that neither the FD

$$\text{Align(St1[Seq1],St2[Seq2],Num1(Occ1,Nuc1),Num2(Occ2,Nuc2))} \rightarrow \text{Align(Comp[Pair(N1,}\lambda\text{)])}$$

nor the FD

$$\text{Align(St1[Seq1],St2[Seq2],Num1(Occ1,Nuc1),Num2(Occ2,Nuc2))} \rightarrow \text{Align(Comp[Pair(}\lambda\text{,N2)])}$$

hold in general. □

EXAMPLE 4.3. Consider Example 2.4 where the nested attribute

Halftoning(Brightness,Input[Level],Output[Bit])

was used to represent a database that stores possible output regions for input regions of a certain brightness. The constraints that were informally described in Section 1.2.2 are formally specified as

$$\text{Halftoning(Input[}\lambda\text{])} \rightarrow \text{Halftoning(Output[}\lambda\text{])}, \text{ and}$$
$$\text{Halftoning(Output[}\lambda\text{])} \rightarrow \text{Halftoning(Input[}\lambda\text{])},$$

and

$$\text{Halftoning(Brightness,Input[}\lambda\text{])} \twoheadrightarrow \text{Halftoning(Input[Level])}.$$

□

4.1.2 Trivial MVDs

We would like to describe MVDs that are satisfied by every instance in a syntactically convenient form. Recall that a dependency σ on some nested attribute N is called trivial if and only if $\models_r \sigma$ for every $r \subseteq dom(N)$. We characterise trivial MVDs.

Lemma 4.2. *Let $N \in \mathcal{N}A$ and $X \twoheadrightarrow Y$ a multi-valued dependency on N. Then is $X \twoheadrightarrow Y$ trivial if and only if $Y \leq X$ or $X \sqcup Y = N$.*

Proof. We show first that $X \twoheadrightarrow Y$ is trivial, if $Y \leq X$ or $X \sqcup Y = N$. Let $r \subseteq dom(N)$ and $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. If there is some $t \in r$ with $\pi^N_{X \sqcup Y}(t) = \pi^N_{X \sqcup Y}(t_1)$ and $\pi^N_{X \sqcup Y^C}(t) = \pi^N_{X \sqcup Y^C}(t_2)$, then $\models_r X \twoheadrightarrow Y$. If $Y \leq X$, then take $t = t_2$. If $X \sqcup Y = N$, or equivalently $Y^C \leq X$, then take $t = t_1$.

Let now be $Y \not\leq X$ and $X \sqcup Y \neq N$, i.e., $Y^C \not\leq X$. We show that there is some $r \subseteq dom(N)$ with $\not\models_r X \twoheadrightarrow Y$. Define $r = \{t_1, t_2\}$ by

$$\pi^N_Z(t_1) \quad = \quad \pi^N_Z(t_2) \qquad \text{if and only if} \qquad Z \quad \leq \quad X$$

using Lemma 3.13. For $\models_r X \twoheadrightarrow Y$ there must be some $t \in r$ with $\pi^N_{X \sqcup Y}(t) = \pi^N_{X \sqcup Y}(t_1)$ and $\pi^N_{X \sqcup Y^C}(t) = \pi^N_{X \sqcup Y^C}(t_2)$. If $t = t_1$, then the second condition is violated since $Y^C \not\leq X$. If $t = t_2$, then the first condition is violated since $Y \not\leq X$. Hence, $\not\models_r X \twoheadrightarrow Y$. □

4.1.3 MVDs are Binary Join Dependencies

Fagin proves in [103] that MVDs "provide a necessary and sufficient condition for a relation to be decomposable into two of its projections without loss of information (in the sense that the original relation is guaranteed to be the join of the two projections)."

We will now prove that MVDs still have the same property in the presence of null, flat, record- and list-valued attributes. In this sense, $r \subseteq dom(N)$ satisfies the MVD $X \twoheadrightarrow Y$ exactly when r is the lossless generalised join of its projections on $X \sqcup Y$ and $X \sqcup Y^C$, i.e., $r = \pi_{X \sqcup Y}(r) \bowtie \pi_{X \sqcup Y^C}(r)$.

Theorem 4.3. *Let $N \in \mathcal{N}A$, $r \subseteq dom(N)$ and $X \twoheadrightarrow Y$ a multi-valued dependency on N. Then is $X \twoheadrightarrow Y$ satisfied by r if and only if $r = \pi_{X \sqcup Y}(r) \bowtie \pi_{X \sqcup Y^C}(r)$.*

Proof. Let $r_1 = \pi_{X \sqcup Y}(r)$ and $r_2 = \pi_{X \sqcup Y^C}(r)$. Note that $r \subseteq r_1 \bowtie r_2$ is always satisfied.

First, let $\models_r X \twoheadrightarrow Y$. We show that $r_1 \bowtie r_2 \subseteq r$. Let $t \in r_1 \bowtie r_2$. Then there are $t_1, t_2 \in r$ with

$$\pi^N_{X \sqcup Y}(t) = \pi^N_{X \sqcup Y}(t_1) \quad \text{and} \quad \pi^N_{X \sqcup Y^C}(t) = \pi^N_{X \sqcup Y^C}(t_2).$$

From $\pi^N_X(t_1) = \pi^N_X(t_2)$ and $\models_r X \twoheadrightarrow Y$ follows the existence of some $t_3 \in r$ with $\pi^N_{X \sqcup Y}(t_3) = \pi^N_{X \sqcup Y}(t_1)$ and $\pi^N_{X \sqcup Y^C}(t_3) = \pi^N_{X \sqcup Y^C}(t_2)$. Consequently, $\pi^N_{X \sqcup Y}(t) = \pi^N_{X \sqcup Y}(t_3)$ and $\pi^N_{X \sqcup Y^C}(t) = \pi^N_{X \sqcup Y^C}(t_3)$. It follows that $t = t_3 \in r$ by Lemma 3.9 and, therefore, $r_1 \bowtie r_2 \subseteq r$.

Let now $r = r_1 \bowtie r_2$ and $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. Let $t'_1 \in r_1$ with $t'_1 = \pi^N_{X \sqcup Y}(t_1)$ and $t'_2 \in r_2$ with $t'_2 = \pi^N_{X \sqcup Y^C}(t_2)$. Since $r_1 \bowtie r_2 = r$, there is some $t \in r$ with $\pi^N_{X \sqcup Y}(t) = t'_1$ and $\pi^N_{X \sqcup Y^C}(t) = t'_2$. This shows $\models_r X \twoheadrightarrow Y$. □

Theorem 4.3 shows that the class of MVDs characterises precisely the situation when an instance over a nested attribute is decomposable into two of its projections without loss of information.

EXAMPLE 4.4. If one applies Theorem 4.3 to Example 4.2 and the snapshot r that was given in Section 1.2.2, then the nested attribute

Align(St1[Seq1],St2[Seq2],Num1(Occ1,Nuc1),Num2(Occ2,Nuc2),Comp[Pair(N1,N2)])

is decomposed into

Align(St1[Seq1],St2[Seq2],Num1(Occ1,Nuc1),Num2(Occ2,Nuc2),Comp[Pair(N1,λ)])

and

Align(St1[Seq1],St2[Seq2],Num1(Occ1,Nuc1),Num2(Occ2,Nuc2),Comp[Pair(λ,N2)]).

The snapshot r is then the generalised natural join of

([A,A],[A,A,T],(3,A),(2,T),[(A,ok),(A,ok),(C,ok),(G,ok),(A,ok)]),
([A,A],[A,A,T],(3,A),(2,T),[(A,ok),(A,ok),(T,ok),(G,ok),(A,ok)]),
([C,G],[C],(2,G),(1,A),[(C,ok),(G,ok),(G,ok),(C,ok)]),
([C,G],[C],(2,G),(1,A),[(C,ok),(G,ok),(C,ok),(G,ok)]),

and

([A,A],[A,A,T],(3,A),(2,T),[(ok,A),(ok,A),(ok,T),(ok,C),(ok,T)]),
([A,A],[A,A,T],(3,A),(2,T),[(ok,A),(ok,A),(ok,T),(ok,T),(ok,C)]),
([C,G],[C],(2,G),(1,A),[(ok,C),(ok,A),(ok,T),(ok,T)]),
([C,G],[C],(2,G),(1,A),[(ok,C),(ok,T),(ok,A),(ok,T)]),
([C,G],[C],(2,G),(1,A),[(ok,C),(ok,C),(ok,A),(ok,C)]).

□

4.1.4 Sound Inference Rules

Before introducing inference rules for FDs and MVDs we will prove the correctness of some algebraic formulae that will be useful in proving the soundness of some of the rules.

Lemma 4.4. *Let $N \in \mathcal{NA}$ and $X, Y \in Sub(N)$. Then the following equations hold:*

1. $X \sqcup (Y \dot{-} X) = X \sqcup Y$,
2. $X = X^{\mathcal{CC}} \sqcup (X \sqcap X^{\mathcal{C}})$,
3. $(X \sqcap Y)^{\mathcal{C}} = X^{\mathcal{C}} \sqcup Y^{\mathcal{C}}$, *and*
4. $(X \sqcup Y)^{\mathcal{C}} \leq X^{\mathcal{C}} \sqcap Y^{\mathcal{C}}$.

Proof. Firstly, $(Y \dot{-} X) \leq X \sqcup Y$ is equivalent to $Y \leq X \sqcup Y$ which is certainly true. As also $X \leq X \sqcup Y$ holds we conclude $X \sqcup (Y \dot{-} X) \leq X \sqcup Y$. Vice versa, $Y \dot{-} X \leq Y \dot{-} X$ is equivalent to $Y \leq X \sqcup (Y \dot{-} X)$. This implies $X \sqcup Y \leq X \sqcup (Y \dot{-} X)$ and the first equation follows.

Recall that $X^{cc} \leq X$ and $X \sqcup X^c = N$ hold. Furthermore, every Brouwerian algebra is distributive. From

$$\begin{aligned} X &= X \sqcap N \\ &= (X^{cc} \sqcup X) \sqcap (X^{cc} \sqcup X^c) \\ &= X^{cc} \sqcup (X \sqcap X^c) \end{aligned}$$

follows then the second equation.

Since $X \sqcap Y \leq X$ and $X \sqcap Y \leq Y$ hold, we conclude $X^c \leq (X \sqcap Y)^c$ and $Y^c \leq (X \sqcap Y)^c$. Consequently, $X^c \sqcup Y^c \leq (X \sqcap Y)^c$. On the other hand, $(X \sqcap Y)^c \leq X^c \sqcup Y^c$ as

$$(X \sqcap Y) \sqcup X^c \sqcup Y^c = (X \sqcup X^c \sqcup Y^c) \sqcap (Y \sqcup X^c \sqcup Y^c) = N$$

holds. The third law follows.

Finally, $(X \sqcup Y)^c \leq X^c \sqcap Y^c$ follows from

$$X \sqcup Y \sqcup (X^c \sqcap Y^c) = (X \sqcup Y \sqcup X^c) \sqcap (X \sqcup Y \sqcup Y^c) = N$$

and this concludes the proof of this lemma. □

Note that the proofs of Lemma 4.4 hold in any Brouwerian algebra, i.e., the laws are also satisfied by any Brouwerian algebra.

A sound and complete set of inference rules for FDs and MVDs has been provided in [32]. We will show in this section that natural extensions of the (sound and complete) rules from [220, p.80,81] are also sound in the presence of record and list type. Apart from these rules there is a further sound rule which allows to derive the non-trivial FD $X \to Y \sqcap Y^c$ from the MVD $X \twoheadrightarrow Y$.

Proposition 4.5. *The following inference rules*

$$\frac{}{X \to Y}\, Y \leq X \quad \textit{(reflexivity axiom)} \qquad \frac{X \to Y}{X \to X \sqcup Y} \quad \textit{(extension rule)} \qquad \frac{X \to Y, Y \to Z}{X \to Z} \quad \textit{(transitivity rule)}$$

$$\frac{X \to Y}{X \twoheadrightarrow Y} \quad \textit{(implication rule)} \qquad \frac{X \twoheadrightarrow Y}{X \twoheadrightarrow Y^c} \quad \textit{(Brouwerian complement rule)} \qquad \frac{X \twoheadrightarrow Y}{W \sqcup X \twoheadrightarrow V \sqcup Y}\, V \leq W \quad \textit{(augmentation rule)}$$

$$\frac{X \twoheadrightarrow Y, Y \twoheadrightarrow Z}{X \twoheadrightarrow (Z \dot{-} Y)} \quad \textit{(pseudo-transitivity rule)} \qquad \frac{X \twoheadrightarrow Y, Y \to Z}{X \to (Z \dot{-} Y)} \quad \textit{(mixed pseudo-transitivity rule)} \qquad \frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow (Y \sqcup Z)} \quad \textit{(multi-valued join rule)}$$

$$\frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow (Z \dot{-} Y)} \quad \textit{(pseudo-difference rule)} \qquad \frac{X \twoheadrightarrow Y}{X \to Y \sqcap Y^c} \quad \textit{(mixed meet rule)} \qquad \frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow (Y \sqcap Z)} \quad \textit{(multi-valued meet rule)}$$

are sound for the implication of FDs and MVDs in the presence of records and lists.

Proof. The correctness of the first three rules has already been proven in Proposition 3.10.

For a proof of the *implication rule* let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. One has to show that there is a $t \in r$ with $\pi^N_{X \cup Y}(t) = \pi^N_{X \cup Y}(t_1)$ and $\pi^N_{X \cup Y^C}(t) = \pi^N_{X \cup Y^C}(t_2)$. The premise implies $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ and, therefore, $\pi^N_{X \cup Y}(t_1) = \pi^N_{X \cup Y}(t_2)$ holds as well. Since also $\pi^N_{X \cup Y^C}(t_2) = \pi^N_{X \cup Y^C}(t_2)$ holds we can choose $t = t_2$. Another proof argument goes as follows. From $\models_r X \to Y$ follows $r = \pi_{X \cup Y}(r) \bowtie \pi_{X \cup Y^C}(r)$ by Theorem 3.46. However, Theorem 4.3 implies that $\models_r X \twoheadrightarrow Y$ holds as well.

In order to prove the *Brouwerian complement rule* let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. The premise implies that there is some $t \in r$ with $\pi^N_{X \cup Y}(t) = \pi^N_{X \cup Y}(t_1)$ and $\pi^N_{X \cup Y^C}(t) = \pi^N_{X \cup Y^C}(t_2)$. As $Y^{CC} \leq Y$ holds, we obtain $\pi^N_{X \cup Y^{CC}}(t) = \pi^N_{X \cup Y^{CC}}(t_1)$ and $\pi^N_{X \cup Y^C}(t) = \pi^N_{X \cup Y^C}(t_2)$. This shows $\models_r X \twoheadrightarrow Y^C$.

As to the *augmentation rule*, take $t_1, t_2 \in r$ with $\pi^N_{W \cup X}(t_1) \overset{(a)}{=} \pi^N_{W \cup X}(t_2)$. Since, in particular, $\pi^N_X(t_1) = \pi^N_X(t_2)$ holds, the premise tells us that there is some $t \in r$ with

$$\pi^N_{X \cup Y}(t) \overset{(b)}{=} \pi^N_{X \cup Y}(t_1) \quad \text{and} \quad \pi^N_{X \cup Y^C}(t) \overset{(c)}{=} \pi^N_{X \cup Y^C}(t_2).$$

Obviously, both $W \dot{-} (X \sqcup Y) \leq X \sqcup Y^C$ and $W \dot{-} (X \sqcup Y) \leq W$ hold. Using first (c) and then (a) we infer

$$\pi^N_{W \dot{-} (X \cup Y)}(t) = \pi^N_{W \dot{-} (X \cup Y)}(t_2) = \pi^N_{W \dot{-} (X \cup Y)}(t_1).$$

From $(X \sqcup Y) \sqcup (W \dot{-} (X \sqcup Y)) = W \sqcup X \sqcup Y$ and $V \leq W$ follows

$$\pi^N_{W \cup X \cup V \cup Y}(t) = \pi^N_{W \cup X \cup V \cup Y}(t_1).$$

Moreover, $W \dot{-} (X \sqcup Y^C) \leq X \sqcup Y$ and $W \dot{-} (X \sqcup Y^C) \leq W$ hold. Using now first (b) and then (a) we infer

$$\pi^N_{W \dot{-} (X \cup Y^C)}(t) = \pi^N_{W \dot{-} (X \cup Y^C)}(t_1) = \pi^N_{W \dot{-} (X \cup Y^C)}(t_2).$$

Due to (c) and the last equation we conclude $\pi^N_{W \cup X \cup Y^C}(t) = \pi^N_{W \cup X \cup Y^C}(t_2)$. Since $(V \sqcup Y)^C \leq Y^C$ holds as well, we obtain

$$\pi^N_{W \cup X \cup (V \cup Y)^C}(t) = \pi^N_{W \cup X \cup (V \cup Y)^C}(t_2).$$

This proves $\models_r W \sqcup X \twoheadrightarrow V \sqcup Y$.

For a proof of the *pseudo-transitivity rule* consider $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. Since $\models_r X \twoheadrightarrow Y$ holds, there is some $\bar{t} \in dom(r)$ with

$$\pi^N_{X \cup Y}(\bar{t}) = \pi^N_{X \cup Y}(t_1) \quad \text{and} \quad \pi^N_{X \cup Y^C}(\bar{t}) \overset{(d)}{=} \pi^N_{X \cup Y^C}(t_2).$$

In particular, $\pi^N_Y(\bar{t}) = \pi^N_Y(t_1)$ since $Y \leq X \sqcup Y$. As $\models_r Y \twoheadrightarrow Z$ we conclude that there is some $t \in r$ with

$$\pi^N_{Y \cup Z}(t) \overset{(e)}{=} \pi^N_{Y \cup Z}(\bar{t}) \quad \text{and} \quad \pi^N_{Y \cup Z^C}(t) \overset{(f)}{=} \pi^N_{Y \cup Z^C}(t_1).$$

Furthermore, $\pi^N_X(\bar{t}) = \pi^N_X(t_1) = \pi^N_X(t_2)$ together with (e) and (f) implies $\pi^N_X(t) = \pi^N_X(t_1) = \pi^N_X(t_2)$. Therefore, we conclude from (f) and $(Z \dot{-} Y)^{\mathcal{C}} \leq Y \sqcup Z^{\mathcal{C}}$ that

$$\pi^N_{X \sqcup (Z \dot{-} Y)^{\mathcal{C}}}(t) = \pi^N_{X \sqcup (Z \dot{-} Y)^{\mathcal{C}}}(t_1) \text{ holds.}$$

Applying $Z \dot{-} Y \leq Y^{\mathcal{C}}$ to (d) and $Z \dot{-} Y \leq Z$ to (e) results in

$$\pi^N_{X \sqcup (Z \dot{-} Y)}(\bar{t}) = \pi^N_{X \sqcup (Z \dot{-} Y)}(t_2) \quad \text{and} \quad \pi^N_{Z \dot{-} Y}(t) = \pi^N_{Z \dot{-} Y}(\bar{t}).$$

Altogether, this yields

$$\pi^N_{X \sqcup (Z \dot{-} Y)}(t) = \pi^N_{X \sqcup (Z \dot{-} Y)}(t_2)$$

and proves $\models_r X \twoheadrightarrow (Z \dot{-} Y)$.

In order to prove the *mixed pseudo-transitivity rule* we take again $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. Since $\models_r X \twoheadrightarrow Y$ holds, there is a $t \in r$ with

$$\pi^N_{X \sqcup Y}(t) = \pi^N_{X \sqcup Y}(t_1) \quad \text{and} \quad \pi^N_{X \sqcup Y^{\mathcal{C}}}(t) \overset{(g)}{=} \pi^N_{X \sqcup Y^{\mathcal{C}}}(t_2).$$

In particular, $\pi^N_Y(t) = \pi^N_Y(t_1)$ implies $\pi^N_Z(t) \overset{(h)}{=} \pi^N_Z(t_1)$ as $\models_r Y \rightarrow Z$ holds. Applying first $Z \dot{-} Y \leq Z$ to (h) and then $Z \dot{-} Y \leq Y^{\mathcal{C}}$ to (g) shows

$$\pi^N_{Z \dot{-} Y}(t_1) = \pi^N_{Z \dot{-} Y}(t) = \pi^N_{Z \dot{-} Y}(t_2)$$

and, therefore, $\models_r X \rightarrow (Z \dot{-} Y)$.

As to the *multi-valued join rule*, take $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. Since $\models_r X \twoheadrightarrow Y$, there is some $t' \in r$ with

$$\pi^N_{X \sqcup Y}(t') \overset{(i)}{=} \pi^N_{X \sqcup Y}(t_1) \quad \text{and} \quad \pi^N_{X \sqcup Y^{\mathcal{C}}}(t') \overset{(j)}{=} \pi^N_{X \sqcup Y^{\mathcal{C}}}(t_2).$$

According to the soundness of the augmentation rule $\models_r X \twoheadrightarrow Z$ implies $\models_r X \sqcup Y \twoheadrightarrow Y \sqcup Z$. Equation (i) guarantees the existence of some $t \in r$ with

$$\pi^N_{X \sqcup Y \sqcup Z}(t) = \pi^N_{X \sqcup Y \sqcup Z}(t_1)$$

and

$$\pi^N_{X \sqcup Y \sqcup (Y \sqcup Z)^{\mathcal{C}}}(t) \overset{(k)}{=} \pi^N_{X \sqcup Y \sqcup (Y \sqcup Z)^{\mathcal{C}}}(t').$$

Since $X \sqcup (Y \sqcup Z)^{\mathcal{C}} \leq X \sqcup Y \sqcup (Y \sqcup Z)^{\mathcal{C}}$ holds, we infer

$$\pi^N_{X \sqcup (Y \sqcup Z)^{\mathcal{C}}}(t) \overset{(l)}{=} \pi^N_{X \sqcup (Y \sqcup Z)^{\mathcal{C}}}(t').$$

from (k). Moreover, $(Y \sqcup Z)^{\mathcal{C}} \leq Y^{\mathcal{C}}$ and (j) imply

$$\pi^N_{X \sqcup (Y \sqcup Z)^{\mathcal{C}}}(t') \overset{(m)}{=} \pi^N_{X \sqcup (Y \sqcup Z)^{\mathcal{C}}}(t_2).$$

Therefore, (l) and (m) together result in

$$\pi^N_{X \sqcup (Y \sqcup Z)^{\mathcal{C}}}(t) = \pi^N_{X \sqcup (Y \sqcup Z)^{\mathcal{C}}}(t_2)$$

which proves $\models_r X \twoheadrightarrow Y \sqcup Z$.

For a proof of the *pseudo-difference rule* let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. From $\models_r X \twoheadrightarrow Y$ follows the existence of some $\bar{t} \in r$ with

$$\pi^N_{X \sqcup Y}(\bar{t}) = \pi^N_{X \sqcup Y}(t_1) \quad \text{and} \quad \pi^N_{X \sqcup Y^{\mathcal{C}}}(\bar{t}) \overset{(n)}{=} \pi^N_{X \sqcup Y^{\mathcal{C}}}(t_2).$$

Equation (n) and $Z \dot{-} Y \leq Y^{\mathcal{C}}$ result in

$$\pi^N_{X \sqcup (Z \dot{-} Y)}(\bar{t}) \overset{(o)}{=} \pi^N_{X \sqcup (Z \dot{-} Y)}(t_2).$$

Since also $\pi^N_X(\bar{t}) = \pi^N_X(t_1)$ and $\models_r X \twoheadrightarrow Z$ hold, there is some $t \in r$ with

$$\pi^N_{X \sqcup Z}(t) \overset{(p)}{=} \pi^N_{X \sqcup Z}(\bar{t}) \quad \text{and} \quad \pi^N_{X \sqcup Z^{\mathcal{C}}}(t) \overset{(q)}{=} \pi^N_{X \sqcup Z^{\mathcal{C}}}(t_1).$$

Applying $Z \dot{-} Y \leq Z$ to (p) and applying (o) subsequently gives

$$\pi^N_{X \sqcup (Z \dot{-} Y)}(t) = \pi^N_{X \sqcup (Z \dot{-} Y)}(t_2).$$

Due to $\pi^N_Y(\bar{t}) = \pi^N_Y(t_1)$ and the construction of t we derive $\pi^N_Y(t) = \pi^N_Y(t_1)$ and $\pi^N_{X \sqcup Y \sqcup Z^{\mathcal{C}}}(t) = \pi^N_{X \sqcup Y \sqcup Z^{\mathcal{C}}}(t_1)$ by equation (q). As also $(Z \dot{-} Y)^{\mathcal{C}} \leq Y \sqcup Z^{\mathcal{C}}$ holds, this finally leads to

$$\pi^N_{X \sqcup (Z \dot{-} Y)^{\mathcal{C}}}(t) = \pi^N_{X \sqcup (Z \dot{-} Y)^{\mathcal{C}}}(t_1)$$

and this proves $\models_r X \twoheadrightarrow (Z \dot{-} Y)$.

For a proof of the *mixed meet rule* let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. Applying the premise gives us some $t \in r$ with $\pi^N_{X \sqcup Y}(t) = \pi^N_{X \sqcup Y}(t_1)$ and $\pi^N_{X \sqcup Y^{\mathcal{C}}}(t) = \pi^N_{X \sqcup Y^{\mathcal{C}}}(t_2)$. As $Y \sqcap Y^{\mathcal{C}} \leq Y, Y^{\mathcal{C}}$ holds by definition of the meet we derive

$$\pi^N_{Y \sqcap Y^{\mathcal{C}}}(t_1) = \pi^N_{Y \sqcap Y^{\mathcal{C}}}(t) = \pi^N_{Y \sqcap Y^{\mathcal{C}}}(t_2)$$

which proves $\models_r X \to Y \sqcap Y^{\mathcal{C}}$.

Finally, for a proof of the *multi-valued meet rule* let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. From the soundness of the implication rule, Brouwerian complement rule, multi-valued join rule and mixed meet rule follows

$$\models_r X \twoheadrightarrow \underbrace{\underbrace{\left(Y^{\mathcal{C}} \sqcup Z^{\mathcal{C}}\right)^{\mathcal{C}}}_{=(Y \sqcap Z)^{\mathcal{C}\mathcal{C}}} \sqcup \left(Y \sqcap Y^{\mathcal{C}}\right) \sqcup \left(Z \sqcap Z^{\mathcal{C}}\right)}_{=W}.$$

Consequently, there is some $t \in r$ with

$$\pi^N_{X \sqcup W}(t_1) = \pi^N_{X \sqcup W}(t) \qquad \text{and} \qquad \pi^N_{X \sqcup W^{\mathcal{C}}}(t_2) = \pi^N_{X \sqcup W^{\mathcal{C}}}(t). \tag{5}$$

Now $Y \sqcap Z \leq W$ holds since

$$\begin{aligned} Y \sqcap Z &= (Y \sqcap Z)^{\mathcal{CC}} \sqcup ((Y \sqcap Z) \sqcap (Y \sqcap Z)^{\mathcal{C}}) \\ &= (Y \sqcap Z)^{\mathcal{CC}} \sqcup ((Y \sqcap Z) \sqcap (Y^{\mathcal{C}} \sqcup Z^{\mathcal{C}})) \\ &= (Y \sqcap Z)^{\mathcal{CC}} \sqcup (Y \sqcap Z \sqcap Y^{\mathcal{C}}) \sqcup (Z \sqcap Y \sqcap Z^{\mathcal{C}}) \\ &\leq W. \end{aligned}$$

Herein, the first equation follows from Lemma 4.4. Consequently,

$$\pi^N_{X \sqcup (Y \sqcap Z)}(t_1) = \pi^N_{X \sqcup (Y \sqcap Z)}(t).$$

Moreover, $W^{\mathcal{C}} \leq (Y \sqcap Z)^{\mathcal{C}}$ since $W \sqcup (Y \sqcap Z)^{\mathcal{C}} = N$ holds. We show that also $(Y \sqcap Z)^{\mathcal{C}} \leq W^{\mathcal{C}}$, i.e., $(Y \sqcap Z) \sqcup W^{\mathcal{C}} = N$ holds. It suffices to show that every $V \in MaxB(N)$ satisfies $V \leq (Y \sqcap Z) \sqcup W^{\mathcal{C}}$. If $V \leq Y \sqcap Z$ there is nothing to show. If $V \not\leq Y \sqcap Z$, then also $V \not\leq W$ as $V \in MaxB(N)$ and $(Y \sqcap Y^{\mathcal{C}}) \sqcup (Z \sqcap Z^{\mathcal{C}})$ contains only non-maximal basis attributes of N. Consequently, $V \leq W^{\mathcal{C}}$. It follows that $(Y \sqcap Z)^{\mathcal{C}} = W^{\mathcal{C}}$, and

$$\pi^N_{X \sqcup (Y \sqcap Z)^{\mathcal{C}}}(t_2) = \pi^N_{X \sqcup (Y \sqcap Z)^{\mathcal{C}}}(t).$$

This shows that $\models_r X \twoheadrightarrow (Y \sqcap Z)$ holds. □

Unless stated otherwise $\mathfrak{R}$ denotes the set of inference rules from Proposition 4.5. It is easy to see that all these rules, except the mixed meet rule, are natural extensions of rules in the RDM (compare [220, p. 80,81]). Interpreting the mixed meet rule in relational databases means that the trivial FD $X \rightarrow \emptyset$ can be derived from the MVD $X \twoheadrightarrow Y$, and is therefore not needed. As $Y \sqcap Y_N^{\mathcal{C}}$ is in general different from λ_N in a Brouwerian algebra, the mixed meet rule is no longer trivial. In fact, it gives the set of inference rules a distinctive Brouwerian flavour.

In what follows, it is important to emphasise the importance of the mixed meet rule. It says informally that $\models_r X \twoheadrightarrow Y$ implies that two elements of r which are coincident on X will also coincide on all non-maximal basis attributes of N that are in $SubB(Y)$.

EXAMPLE 4.5. Consider again Example 4.1. The r given there satisfies the MVD

$$\text{Pubcrawl(Person)} \twoheadrightarrow \text{Pubcrawl(Visit[Drink(Pub)])}.$$

According to the mixed meet rule this implies also that r satisfies the FD

$$\text{Pubcrawl(Person)} \rightarrow \text{Pubcrawl(Visit[Drink(Pub)])} \sqcap \text{Pubcrawl(Visit[Drink(Beer)])}$$

which is Pubcrawl(Person) $\rightarrow$ Pubcrawl(Visit[λ]). This shows that each person visits a fixed number of pubs (and drinks a fixed number of beers), as Visit[λ] represents the length of the list of visits. □

The example indicates how the mixed meet rule might help a database designer to specify or not specify MVDs. If the corresponding FD Pubcrawl(Person) $\rightarrow$ Pubcrawl(Visit[λ]) is not considered meaningful for the application in mind, then the MVD Pubcrawl(Person) $\twoheadrightarrow$ Pubcrawl(Visit[Drink(Pub)]) cannot be meaningful neither.

4.1.5 Dependency Basis

We generalise the notion of a dependency basis from the RDM [32] to our framework. Therefore, we repeat first the notion of a dependency basis of an attribute set $X \subseteq R$ with respect to a given set Σ of FDs and MVDs on the relation schema R. The family $Dep(X) = \{Y \mid X \twoheadrightarrow Y \in \Sigma^+\}$ is closed under Boolean operations according to the union, intersection and difference rule from Theorem 1.6. Therefore, it contains a unique subfamily with the following properties:

1. The empty set is not an element of the subfamily.
2. Each pair of sets in the subfamily is disjoint.
3. Each set in $Dep(X)$ is a union of sets from the subfamily.

This subfamily consists of all atoms in $(Dep(X), \subseteq, \emptyset)$, and is called the dependency basis of X with respect to Σ.

We will now extend this notion to our framework. Let $N \in \mathcal{NA}$, $X \in Sub(N)$ and Σ be a set of FDs and MVDs on N. Let $Dep(X)$ be the set of all $Y \in Sub(N)$ with $X \twoheadrightarrow Y \in \Sigma^+$ and $X^+ = \bigsqcup\{Y \mid X \to Y \in \Sigma^+\}$. Consider the set $Dep'(X) = \{Y \mid Y \in Dep(X) \text{ and } Y^{cc} = Y\}$ which consists of all those subattributes $Y \in Dep(X)$ which are a join of maximal basis attributes of N. According to [202] such subattributes are called *regular*. $(Dep(X), \leq, \sqcup, \sqcap, \dot{-}, N)$ is a Brouwerian algebra according to the multi-valued join, multi-valued meet and pseudo-difference rule. The regular elements of a Brouwerian algebra form a Boolean algebra [202, Theorem 4.5]. Thus $(Dep'(X), \leq, \sqcup, \sqcap', (\cdot)^c, \lambda_N, N)$ is a Boolean algebra where $\sqcap'$ is defined by $Y \sqcap' Z = (Y \sqcap Z)^{cc}$ for all $Y, Z \in Dep'(X)$. For $Y, Z \in Dep'(X)$ we have $Y = (Y \sqcap' Z) \sqcup (Y \dot{-} Z)$ since $Y \dot{-} Z = Y \sqcap' Z^c$ holds. Moreover, there is no maximal basis attribute of N that is a subattribute of both $Y \sqcap' Z$ and $Y \dot{-} Z$, i.e., $(Y \sqcap' Z) \sqcap' (Y \dot{-} Z) = \lambda_N$. This shows that there is a unique subset $X^M \subseteq Dep'(X)$ with the following properties:

1. $\lambda_N \notin X^M$.
2. For all distinct $Y, Z \in X^M$ we have $Y \sqcap' Z = \lambda_N$.
3. For all $W \in Dep(X)$ we have $W = \bigsqcup \mathcal{Y}$ for some $\mathcal{Y} \subseteq X^M$.

Consequently, X^M consists of all $\leq$-minimal elements of $(Dep'(X) - \{\lambda_N\})$, i.e., X^M is the set of all atoms of $(Dep'(X), \leq, \lambda_N)$. Note that $\{MaxB(W) \mid W \in X^M\}$ is the partition of $MaxB(N)$ which is generated by $\{MaxB(Y^{cc}) \mid Y \in Dep(X)\}$.

Definition 4.6. Let $N \in \mathcal{NA}$, $X \in Sub(N)$ and Σ be a set of FDs and MVDs on N. The *dependency basis* of X with respect to Σ is $DepB(X) = SubB(X^+) \cup X^M$. □

We will illustrate the notion of a dependency basis by the following example.

EXAMPLE 4.6. Let $N = L(A, K[M(B, C, D)])$ and $\Sigma = \{L(A) \twoheadrightarrow L(K[M(B)])\}$. The Boolean algebra carried by $Dep'(L(A))$ is shown in Figure 4.1. The set X^M consists of $L(A)$, $L(K[M(B)])$, and $L(K[M(C, D)])$ which are the atoms of the Boolean algebra. The dependency basis of $L(A)$ consists of $L(A)$, $L(K[\lambda])$, $L(K[M(B)])$, and $L(K[M(C, D)])$.
□

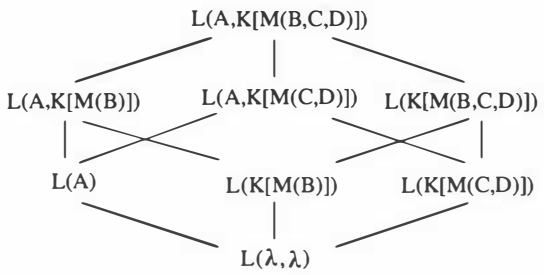

Fig. 4.1. The Boolean algebra of $L(A)^M$

Lemma 4.7. *Let $N \in \mathcal{N}A$, Σ a set of FDs and MVDs on N, and $X \in Sub(N)$. If $W \in X^M$, then $W \in Dep(X)$.*

Proof. This is immediate as $X^M \subseteq Dep'(X)$ and $Dep'(X) \subseteq Dep(X)$. □

We can now show that an MVD $X \twoheadrightarrow Y$ is derivable from Σ if and only if the right-hand side Y is the join over some elements of the dependency basis of X with respect to Σ. This extends a result from [32].

Proposition 4.8. *Let $N \in \mathcal{N}$A and Σ as set of functional and multi-valued dependencies on N. Then*

1. *$X \twoheadrightarrow Y \in \Sigma^+$ if and only if $Y = \sqcup Z$ for some $Z \subseteq DepB(X)$*
2. *$X \rightarrow Y \in \Sigma^+$ if and only if $Y \leq X^+$.*

Proof. The second property is obvious. Let $Y \in Dep(X)$. Lemma 4.4 shows that $Y = Y^{cc} \sqcup (Y \sqcap Y^c)$ holds. From $Y \in Dep(X)$ follows $Y^{cc} \in Dep'(X)$ and therefore $Y^{cc} = \sqcup Z_1$ for some $Z_1 \subseteq X^M$. Moreover, $Y \in Dep(X)$ means that $X \twoheadrightarrow Y \in \Sigma^+$ and thus $X \rightarrow Y \sqcap Y^c \in \Sigma^+$ by the mixed meet rule. It follows that $Y \sqcap Y^c \leq X^+$ and, therefore, $Y \sqcap Y^c = \sqcup Z_2$ for some $Z_2 \subseteq SubB(X^+)$. Hence, $Z = Z_1 \cup Z_2 \subseteq DepB(X)$ and $Y = \sqcup Z$.

Assume now that $Y = \sqcup Z$ holds for some $Z \subseteq DepB(X)$. Then $Z = Z_1 \cup Z_2$ with $Z_1 \subseteq SubB(X^+)$ and $Z_2 \subseteq X^M$. It follows that $\sqcup Z_1 = Y_1 \leq X^+$, and the reflexivity rule gives $X^+ \rightarrow Y_1 \in \Sigma^+$. According to the join rule for FDs we have $X \rightarrow X^+ \in \Sigma^+$ and the transitivity rule implies $X \rightarrow Y_1 \in \Sigma^+$. Finally, the implication rule gives $X \twoheadrightarrow Y_1 \in \Sigma^+$. Furthermore, if $Z_2 = \{V_1, \ldots, V_m\} \subseteq X^M$, then $X \twoheadrightarrow V_i \in \Sigma^+$ for $1 \leq i \leq m$ by Lemma 4.7. Applying the multi-valued join rule gives $X \twoheadrightarrow Y \in \Sigma^+$. □

4.1.6 Completeness

Proving the completeness result for functional and multi-valued dependencies will involve the construction of some finite instance r_X which satisfies all dependencies in Σ but does not satisfy any FD or MVD $\sigma \notin \Sigma^+$ with left-hand side X. This instance will initially contain two elements t_1, t_2 which are coincident on exactly all attributes which are functionally determined by X. Afterwards new elements are generated and added to r_X by exhaustively combining values from t_1 on the join of some $\mathcal{W} \subseteq X^M$ and the values from t_2 on the join of

$X^M - \mathcal{W}$. Note, however, that for different $W, W' \in X^M$ the meet $W \sqcap W'$ is not necessarily equal to λ_N. The construction above becomes possible, if one can show that $W \sqcap W' \leq X^+$ for any different $W, W' \in X^M$.

Definition 4.9. Let $N \in \mathcal{N}\mathrm{A}$, $X' \subseteq MaxB(N)$ and $X = \sqcup X'$. A basis attribute $Y \in SubB(X)$ is *possessed by* X if and only if every basis attribute $Z \in SubB(N)$ with $Y \leq Z$ is also a subattribute of X ($Z \leq X$). □

It follows that $SubB(W \sqcap W')$ with different $W, W' \in X^M$ contains only basis attributes of W or W' which are neither possessed by W nor by W'.

EXAMPLE 4.7. Let $K[L(M[N(A,B)],C)] \in \mathcal{N}\mathrm{A}$, and $X = K[L(M[N(A,B)])]$. Then X does possess $K[L(M[N(\lambda,\lambda],\lambda)]$, but does *not* possess $K[L(\lambda,\lambda)]$. For an illustration see also Figure 4.2. □

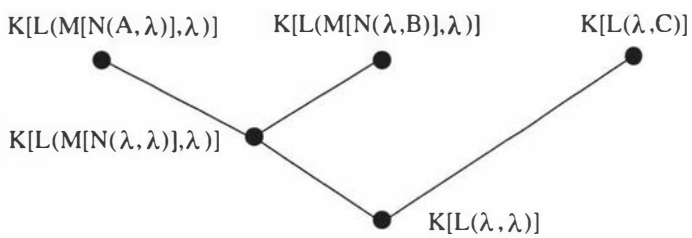

Fig. 4.2. The subattribute basis of $K[L(M[N(A,B)],C)]$

A basis attribute of N is not possessed by some X exactly if it is also a subattribute of $X_N^{\mathcal{C}}$. According to the mixed meet rule it follows that basis attributes which are not possessed by any element in X^M are functionally determined by X.

Lemma 4.10. *Let $N \in \mathcal{N}A$, $X' \subseteq MaxB(N)$, $X = \sqcup X'$ and $Y \in SubB(X)$. Then is Y possessed by X if and only if $Y \notin SubB(X^{\mathcal{C}})$.*

Proof. Let Y be possessed by X. Assume that $Y \in SubB(X^{\mathcal{C}})$. It follows that there is some $Z \in MaxB(X^{\mathcal{C}}) \subseteq MaxB(N)$ with $Y \leq Z$. Since Y is possessed by X we also have $Z \in SubB(X)$, and as $Z \in MaxB(N)$ also $Z \in MaxB(X)$. This is a contradiction since $MaxB(X) \cap MaxB(X^{\mathcal{C}}) = \emptyset$ (note that $X = X^{\mathcal{CC}}$ in this case). Consequently, $Y \notin SubB(X^{\mathcal{C}})$.

If Y is not possessed by X, then there is some $Z \in SubB(N)$ with $Y \leq Z$ and $Z \notin SubB(X)$. Since $SubB(N) = SubB(X) \cup SubB(X^{\mathcal{C}})$ we must have that $Z \in SubB(X^{\mathcal{C}})$, and therefore also $Y \in SubB(X^{\mathcal{C}})$. □

Corollary 4.11. *Let $N \in \mathcal{N}A$, $X \in Sub(N)$, and Σ a set of functional and multi-valued dependencies on N. Then for every $W \in X^M$ and every $Y \in SubB(W)$ that is not possessed by W follows that $Y \leq X^+$.*

Proof. Since Y is not possessed by W, we have $Y \in SubB(W^{\mathcal{C}})$ by Lemma 4.10 and therefore $Y \leq W \sqcap W^{\mathcal{C}}$. Lemma 4.7 implies that $X \twoheadrightarrow W \in \Sigma^+$ holds, and using the mixed meet rule we infer $X \rightarrow W \sqcap W^{\mathcal{C}} \in \Sigma^+$. The reflexivity rule implies $W \sqcap W^{\mathcal{C}} \rightarrow Y \in \Sigma^+$ since $Y \leq W \sqcap W^{\mathcal{C}}$. Consequently, $X \rightarrow Y \in \Sigma^+$ by the transitivity rule. This means $Y \leq X^+$. □

Suppose $DepB(X) = SubB(X^+) \cup \{W_{0,1}, \ldots, W_{0,m}, W_1, \ldots, W_k\}$ with $W_{0,i} \leq X^+$ for $i = 1, \ldots, m$ and $W_1, \ldots, W_k \not\leq X^+$. We have seen that $SubB(W_i \sqcap W_j)$, $i \neq j$, contains only basis attributes of W_i or W_j neither possessed by W_i nor by W_j. It follows that $X \rightarrow W_i \sqcap W_j \in \Sigma^+$ holds.

Assume now that there are two elements $t_1, t_2 \in dom(N)$ which coincide on at least all subattributes of X^+. It is then easy to see that one can substitute the values of t_1 on all subattributes of some given $W_i \in X^M$ for the corresponding values of t_2 and end up with an element in $dom(N)$.

Lemma 4.12. *Let $N \in \mathcal{N}A$, Σ a set of functional and multi-valued dependencies on N and $X \in Sub(N)$. Let $DepB(X) = SubB(X^+) \cup X^M$ with $X^M = \{W_{0,1}, \ldots, W_{0,m}, W_1, \ldots, W_k\}$ and $W_{0,i} \leq X^+$ for $1 \leq i \leq m$ and $W_1, \ldots, W_k \not\leq X^+$. Let $t_1, t_2 \in dom(N)$ with $\pi^N_W(t_1) = \pi^N_W(t_2)$, if $W \leq X^+$. Then for all $W = \sqcup W'$ with $W' \subseteq \{W_1, \ldots, W_k\}$ there is some $t \in dom(N)$ with $\pi^N_W(t) = \pi^N_W(t_1)$ and $\pi^N_{W^C}(t) = \pi^N_{W^C}(t_2)$.* □

Corollary 4.11 shows that every basis attribute which is not possessed by any $W_i \in X^M$ is functionally determined by X, i.e., elements in $dom(N)$ with the same value on X will also coincide on all basis attributes which are not possessed by any $W_i \in X^M$. The statement from Lemma 4.12 is therefore equivalent to the fact that there is some $t \in dom(N)$ with

$$\pi^N_A(t) = \begin{cases} \pi^N_A(t_1) & \text{, if } A \text{ is possessed by some } W_i \in W' \\ \pi^N_A(t_2) & \text{, else} \end{cases}$$

and where A is any element of $SubB(N)$. We are now prepared to prove the main result of this section.

Theorem 4.13. *The set of rules from Proposition 4.5 is complete for the finite implication of FDs and MVDs in the presence of records and lists.*

Proof. Let N be an arbitrary nested attribute and Σ an arbitrary set of FDs and MVDs on N. Due to Proposition 4.5 it remains to show completeness in the finite sense, i.e., $\Sigma^*_{\text{fin}} \subseteq \Sigma^+$. Let $X \in Sub(N)$. Let $DepB(X) = SubB(X^+) \cup X^M$ with $X^M = \{W_{0,1}, \ldots, W_{0,m}, W_1, \ldots, W_k\}$ and $W_{0,i} \leq X^+$ for $i = 1, \ldots, m$ and $W_1, \ldots, W_k \not\leq X^+$. Take $t_1, t_2 \in dom(N)$ defined by

$$\pi^N_W(t_1) = \pi^N_W(t_2) \qquad \text{if and only if} \qquad W \leq X^+.$$

Recall that such t_1, t_2 exist according to Lemma 3.13. Define an instance $r \subseteq dom(N)$ with $t_1, t_2 \in r$ and add for every $W = \sqcup W'$ with $W' \subseteq \{W_1, \ldots, W_k\}$ the $t \in dom(N)$ with

$\pi^N_W(t) = \pi^N_W(t_1)$ and $\pi^N_{W^C}(t) = \pi^N_{W^C}(t_2)$ from Lemma 4.12 to r. Obviously, r has exactly 2^k elements, and if $\pi^N_{W_l}(t_i) \neq \pi^N_{W_l}(t_j)$, then also $\pi^N_W(t_i) \neq \pi^N_W(t_j)$ on all $W \leq W_l$ which are possessed by W_l.
We will show that $\models_r \Sigma$. Then, for $X \to Y \in \Sigma^*_{\text{fin}}$ we have $\models_r X \to Y$. Since all elements of r coincide on X^+, r can only satisfy $X \to Y$ if all elements of r also coincide on Y. It follows by construction of r that $Y \leq X^+$. Proposition 4.8 implies $X \to Y \in \Sigma^+$. For $X \twoheadrightarrow Y \in \Sigma^*_{\text{fin}}$ we have $\models_r X \twoheadrightarrow Y$. Again by construction, r can only satisfy $X \twoheadrightarrow Y$ if $Y = X_0 \sqcup W_{i_1} \sqcup \cdots \sqcup W_{i_m}$ with $X_0 \leq X^+$ and $1 \leq i_1 < \cdots < i_m \leq k$. Therefore, $X \twoheadrightarrow Y \in \Sigma^+$ by Proposition 4.8.

It remains to show that $\models_r \Sigma$ holds.

1. Suppose $U \to V \in \Sigma$. Define

$$W = \bigsqcup\{W_i \mid \exists U'.U' \leq U \text{ and } U' \text{ is possessed by } W_i\}.$$

It follows that $U \leq X^+ \sqcup W$ since every subattribute of U that is possessed by some W_i is also a subattribute of W and every subattribute of U that is not possessed by any W_i is a subattribute of X^+. The reflexivity rule implies $X^+ \sqcup W \to U \in \Sigma^+$. Using the transitivity rule gives $X^+ \sqcup W \to V \in \Sigma^+$.
Take $t_i, t_j \in r$ with $\pi^N_U(t_i) = \pi^N_U(t_j)$. All elements of r are equal on X^+. Assume $\pi^N_W(t_i) \neq \pi^N_W(t_j)$. Then there is some W_l with $\pi^N_{W_l}(t_i) \neq \pi^N_{W_l}(t_j)$ and some subattribute $U' \leq U$ which is possessed by W_l. Consequently, $\pi^N_{U'}(t_i) \neq \pi^N_{U'}(t_j)$ and, therefore, $\pi^N_U(t_i) \neq \pi^N_U(t_j)$ too, a contradiction. It follows that $\pi^N_W(t_i) = \pi^N_W(t_j)$ holds and therefore $\pi^N_{X^+ \sqcup W}(t_i) = \pi^N_{X^+ \sqcup W}(t_j)$ as well. Now, $X^+ \sqcup W$ is the join of elements in $DepB(X)$, i.e., $X \twoheadrightarrow X^+ \sqcup W \in \Sigma^+$ by Proposition 4.8. Hence, we infer $X \to (V \dot{-} (X^+ \sqcup W)) \in \Sigma^+$ by the mixed pseudo-transitivity rule. Proposition 4.8 implies $V \dot{-} (X^+ \sqcup W) \leq X^+$, and therefore $\pi^N_{V \dot{-} (X^+ \sqcup W)}(t_i) = \pi^N_{V \dot{-} (X^+ \sqcup W)}(t_j)$. Since $V \leq (X^+ \sqcup W) \sqcup (V \dot{-} (X^+ \sqcup W))$ holds we obtain $\pi^N_V(t_i) = \pi^N_V(t_j)$. This proves $\models_r U \to V$.
2. Suppose $U \twoheadrightarrow V \in \Sigma$. Define again

$$W = \bigsqcup\{W_i \mid \exists U'.U' \leq U \text{ and } U' \text{ is possessed by } W_i\}.$$

As before, $U \leq X^+ \sqcup W$ holds. From $U \twoheadrightarrow V \in \Sigma$ and $\lambda \leq X^+ \sqcup W$ follows $X^+ \sqcup W \twoheadrightarrow V \in \Sigma^+$ by the augmentation rule.
Take $t_i, t_j \in r$ with $\pi^N_U(t_i) = \pi^N_U(t_j)$. Again, the construction of r implies $\pi^N_{X^+ \sqcup W}(t_i) = \pi^N_{X^+ \sqcup W}(t_j)$. Since $X \twoheadrightarrow X^+ \sqcup W \in \Sigma^+$ holds by Proposition 4.8, the pseudo-transitivity rule allows to derive $X \twoheadrightarrow (V \dot{-} (X^+ \sqcup W)) \in \Sigma^+$. Therefore, $V \dot{-} (X^+ \sqcup W)$ is the join of some elements in $DepB(X)$ by Proposition 4.8. By construction of r there is some $t \in r$ with

$$\pi^N_{X^+ \sqcup W \sqcup (V \dot{-} (X^+ \sqcup W))}(t) = \pi^N_{X^+ \sqcup W \sqcup (V \dot{-} (X^+ \sqcup W))}(t_i)$$

and

$$\pi^N_{X^+ \sqcup W \sqcup (V \dot{-} (X^+ \sqcup W))^C}(t) = \pi^N_{X^+ \sqcup W \sqcup (V \dot{-} (X^+ \sqcup W))^C}(t_j).$$

As $U, V \leq X^+ \sqcup W \sqcup (V \dot{-} (X^+ \sqcup W))$ hold we have $U \sqcup V \leq X^+ \sqcup W \sqcup (V \dot{-} (X^+ \sqcup W))$ and therefore $\pi^N_{U \sqcup V}(t) = \pi^N_{U \sqcup V}(t_i)$. From $V \dot{-} (X^+ \sqcup W) \leq V$ follows

$$V^{\mathcal{C}} \leq (V \dot{-} (X^+ \sqcup W))^{\mathcal{C}} \quad .$$

But then $U \sqcup V^{\mathcal{C}} \leq X^+ \sqcup W \sqcup (V \dot{-} (X^+ \sqcup W))^{\mathcal{C}}$ and thus $\pi^N_{U \sqcup V^{\mathcal{C}}}(t) = \pi^N_{U \sqcup V^{\mathcal{C}}}(t_j)$. This proves $\models_r U \twoheadrightarrow V$. □

The construction for maximal basis attributes of N is based on the relational theory. The rest follows from the algebraic framework and the fact that non-maximal basis attributes of N that are not possessed by any $W \in X^M$ are already functionally determined by X. This is due to the mixed meet rule.

Corollary 4.14. *Finite and unrestricted implication coincide for the class of FDs and MVDs in the presence of records and lists.*

Proof. According to Proposition 4.5 the inference rules are sound for the (unrestricted) implication of FDs and MVDs. This shows that $\Sigma^+ \subseteq \Sigma^*$. Theorem 4.13 has just shown that $\Sigma^*_{\text{fin}} \subseteq \Sigma^+$ holds. This implies

$$\Sigma^+ \subseteq \Sigma^* \subseteq \Sigma^*_{\text{fin}} \subseteq \Sigma^+$$

and $\Sigma^* = \Sigma^*_{\text{fin}}$ follows for an arbitrary nested attribute N and an arbitrary set Σ of FDs and MVDs on Σ. □

4.2 Minimality

In this section the independence of the inference rules from Proposition 4.5 is studied. That is, we will investigate whether the sound and complete set of inference rules from Proposition 4.5 is minimal in the sense of Definition 3.7. The goal of this section is to identify a minimal subset of the rules from Proposition 4.5. For technical reasons we first derive the auto-complement rule

$$\frac{X \twoheadrightarrow Y}{X \twoheadrightarrow Z} Z \leq Y \sqcap Y^{\mathcal{C}}.$$

The proof of Lemma 4.15 shows in particular the soundness of this rule.

Lemma 4.15. *The auto-complement rule is not independent from {reflexivity axiom, transitivity rule, implication rule, mixed meet rule}.*

Proof.

$$\dfrac{\dfrac{\dfrac{X \twoheadrightarrow Y}{X \to Y \sqcap Y^{\mathcal{C}}} \qquad \dfrac{}{Y \sqcap Y^{\mathcal{C}} \to Z}\, Z \leq Y \sqcap Y^{\mathcal{C}}}{X \to Z}}{X \twoheadrightarrow Z}$$

□

The following lemmata show that pseudo-difference rule, multi-valued meet rule as well as augmentation rule cannot be used to infer any further MVDs in the presence of the remaining inference rules. In the corresponding proofs inference schemata are identified which can be used instead of an application of the respective rule.

Lemma 4.16. *The pseudo-difference rule is not independent from {Brouwerian complement rule, auto-complement rule, join rule}.*

Proof. According to Theorem 4.2(viii) in [202] we have $(Z \dot{-} Y)^{cc} = (Z^{cc} \sqcap Y^c)^{cc}$. Applying the third equation from Lemma 4.4 we obtain $(Z \dot{-} Y)^{cc} = (Z^c \sqcup Y^{cc})^c$.

Furthermore, $(Z \dot{-} Y) \sqcap (Z \dot{-} Y)^c \leq (Z \sqcap Y^c) \sqcap (Y \sqcup Z^c) = (Z \sqcap Y^c \sqcap Y) \sqcup (Z \sqcap Y^c \sqcap Z^c) \leq (Y^c \sqcap Y) \sqcup (Z^c \sqcap Z)$. As $(Z \sqcap Z^c) \sqcup (Y \sqcap Y^c)$ contains only non-maximal basis attributes of the underlying nested attribute N, all maximal basis attributes of N are in $((Z \sqcap Z^c) \sqcup (Y \sqcap Y^c))^c$, i.e., $((Z \sqcap Z^c) \sqcup (Y \sqcap Y^c))^c = N$. This shows that

$$(Z \dot{-} Y) \sqcap (Z \dot{-} Y)^c \leq ((Y \sqcap Y^c) \sqcup (Z \sqcap Z^c)) \sqcap ((Y \sqcap Y^c) \sqcup (Z \sqcap Z^c))^c \tag{6}$$

holds as well.

$$\dfrac{\dfrac{\dfrac{\dfrac{X \twoheadrightarrow Z}{X \twoheadrightarrow Z^c} \quad \dfrac{X \twoheadrightarrow Y}{X \twoheadrightarrow Y^c}}{X \twoheadrightarrow Z^c \sqcup Y^{cc}}}{X \twoheadrightarrow (Z^c \sqcup Y^{cc})^c} \quad \dfrac{\dfrac{\dfrac{X \twoheadrightarrow Y}{X \twoheadrightarrow Y \sqcap Y^c}\, {\scriptstyle Y \sqcap Y^c \leq Y \sqcap Y^c} \quad \dfrac{X \twoheadrightarrow Z}{X \twoheadrightarrow Z \sqcap Z^c}\, {\scriptstyle Z \sqcap Z^c \leq Z \sqcap Z^c}}{X \twoheadrightarrow (Y \sqcap Y^c) \sqcup (Z \sqcap Z^c)}}{X \twoheadrightarrow (Z \dot{-} Y) \sqcap (Z \dot{-} Y)^c}\, {\scriptstyle (6)}}{X \twoheadrightarrow \underbrace{(Z \dot{-} Y)^{cc} \sqcup ((Z \dot{-} Y) \sqcap (Z \dot{-} Y)^c)}_{= Z \dot{-} Y}}$$

□

Lemma 4.17. *The multi-valued meet rule is not independent from {Brouwerian complement rule, auto-complement rule, multi-valued join rule}.*

Proof. The following derivation tree applies the third equation of Lemma 4.4, and uses distributivity.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{X \twoheadrightarrow Y}{X \twoheadrightarrow Y^c} \quad \dfrac{X \twoheadrightarrow Z}{X \twoheadrightarrow Z^c}}{X \twoheadrightarrow Y^c \sqcup Z^c}}{X \twoheadrightarrow (Y^c \sqcup Z^c)^c} \quad \dfrac{X \twoheadrightarrow Y}{X \twoheadrightarrow Y \sqcap Y^c \sqcap Z}\, {\scriptstyle Y \sqcap Y^c \sqcap Z \leq Y \sqcap Y^c}}{X \twoheadrightarrow (Y^c \sqcup Z^c)^c \sqcup (Y \sqcap Y^c \sqcap Z)} \quad \dfrac{X \twoheadrightarrow Z}{X \twoheadrightarrow Z \sqcap Z^c \sqcap Y}\, {\scriptstyle Z \sqcap Z^c \sqcap Y \leq Z \sqcap Z^c}}{X \twoheadrightarrow \underbrace{(Y^c \sqcup Z^c)^c}_{= (Y \sqcap Z)^{cc}} \sqcup \underbrace{(Y \sqcap Y^c \sqcap Z) \sqcup (Z \sqcap Z^c \sqcap Y)}_{= (Y \sqcap Z) \sqcap (Y \sqcap Z)^c}}{X \twoheadrightarrow Y \sqcap Z}$$

□

Lemma 4.18. *The augmentation rule follows from {reflexivity axiom, pseudo-transitivity rule, multi-valued join rule, implication rule}.*

Proof. Note that $Y = Y \sqcap (Y \sqcup X) = ((Y \dot{-} X) \sqcup Y) \sqcap ((Y \dot{-} X) \sqcup X) = (Y \dot{-} X) \sqcup (Y \sqcap X)$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{X \sqcup W \to X}\,X \leq X \sqcup W}{X \sqcup W \twoheadrightarrow X} \qquad X \twoheadrightarrow Y \quad \dfrac{}{X \sqcup W \to Y \sqcap X}\,Y \sqcap X \leq X \sqcup W}{X \sqcup W \twoheadrightarrow Y \dot{-} X} \qquad \dfrac{}{X \sqcup W \twoheadrightarrow Y \sqcap X}}{X \sqcup W \twoheadrightarrow Y} \qquad \dfrac{\dfrac{}{X \sqcup W \to V}\,V \leq W \leq X \sqcup W}{X \sqcup W \twoheadrightarrow V}}{X \sqcup W \twoheadrightarrow Y \sqcup V}$$

□

It follows from the previous lemmata that reflexivity axiom, extension rule, transitivity rule, implication rule, Brouwerian complement rule, pseudo-transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule and mixed meet rule form already a sound and complete set of inference rules for the implication of FDs and MVDs. We are now going to show that this is in fact a minimal set, i.e., each of the rules is independent from the others.

Lemma 4.19. *The reflexivity axiom is independent from the set $\mathfrak{R}$ = {extension rule, transitivity rule, implication rule, Brouwerian complement rule, pseudo-transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule, mixed meet rule}.*

Proof. The reflexivity axiom is the only inference rule that allows one to infer dependencies from the empty set. □

Lemma 4.20. *The extension rule is independent from the set $\mathfrak{R}$ = {reflexivity axiom, transitivity rule, implication rule, Brouwerian complement rule, pseudo-transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule, mixed meet rule}.*

Proof. Let $N = L(A, B)$, $\Sigma = \{L(A) \to L(B)\}$ and $\sigma = L(A) \to L(A, B)$. The closure of Σ under $\mathfrak{R}$ is represented by the following tables in the following way. An FD $X \to Y$ is in the closure $\Sigma^+_{\mathfrak{R}}$ if and only if there appears a cross $\times$ in row X and column Y of the left table. Correspondingly, an MVD $X \twoheadrightarrow Y$ is in the closure $\Sigma^+_{\mathfrak{R}}$ if and only if there appears a cross $\times$ in row X and column Y of the right table.

$\to$	λ	$L(A)$	$L(B)$	$L(A,B)$
λ	×			
$L(A)$	×	×	×	
$L(B)$	×		×	
$L(A,B)$	×	×	×	×

$\twoheadrightarrow$	λ	$L(A)$	$L(B)$	$L(A,B)$
λ	×			×
$L(A)$	×	×	×	×
$L(B)$	×	×	×	×
$L(A,B)$	×	×	×	×

It can be seen that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, σ can be inferred from Σ using the extension rule. □

Lemma 4.21. *The transitivity rule is independent from the set* $\mathfrak{R} = \{$*reflexivity axiom, extension rule, implication rule, Brouwerian complement rule, pseudo-transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule, mixed meet rule*$\}$.

Proof. Let $N = L(A, B)$, $\Sigma = \{\lambda \to L(A), L(A) \to L(B)\}$ and $\sigma = \lambda \to L(B)$. The closure of Σ under $\mathfrak{R}$ is represented by the following tables.

$\to$	λ	$L(A)$	$L(B)$	$L(A,B)$
λ	×	×		
$L(A)$	×	×	×	×
$L(B)$	×		×	
$L(A,B)$	×	×	×	×

$\twoheadrightarrow$	λ	$L(A)$	$L(B)$	$L(A,B)$
λ	×	×	×	×
$L(A)$	×	×	×	×
$L(B)$	×	×	×	×
$L(A,B)$	×	×	×	×

It can be seen that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, σ can be inferred from Σ using the transitivity rule. □

Lemma 4.22. *The implication rule is independent from the set* $\mathfrak{R} = \{$*reflexivity axiom, extension rule, transitivity rule, Brouwerian complement rule, pseudo-transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule, mixed meet rule*$\}$.

Proof. Let $N = \lambda$, $\Sigma = \emptyset$ and $\sigma = \lambda \twoheadrightarrow \lambda$. The closure of Σ under $\mathfrak{R}$ is represented by the following tables.

$\to$	λ
λ	×

$\twoheadrightarrow$	λ
λ	

It can be seen that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, σ can be inferred from Σ using first the reflexivity axiom to infer $\lambda \to \lambda$, and subsequently the implication rule. □

Lemma 4.23. *The Brouwerian complement rule is independent from the set* $\mathfrak{R} = \{$*reflexivity axiom, extension rule, transitivity rule, implication rule, pseudo-transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule, mixed meet rule*$\}$.

Proof. Suppose we interpret $X \twoheadrightarrow Y$ as "X functionally determines Y", and consider the set of FDs on a nested attribute N. Under this interpretation, implication rule, pseudo-transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule and mixed meet rule are all still valid, but the Brouwerian complement rule is not. Hence, it cannot be logically implied by the set given. □

Lemma 4.24. *The pseudo-transitivity rule is independent from the set* $\mathfrak{R} = \{$*reflexivity axiom, extension rule, transitivity rule, implication rule, Brouwerian complement rule, mixed pseudo-transitivity rule, multi-valued join rule, mixed meet rule*$\}$.

Proof. Let $N = L(A, B, C)$, $\Sigma = \{\lambda \twoheadrightarrow L(A), L(A) \twoheadrightarrow L(B)\}$ and $\sigma = \lambda \twoheadrightarrow L(B)$. The closure of Σ under $\mathfrak{R}$ is represented by the following tables

$\rightarrow$	λ	$L(A)$	$L(B)$	$L(C)$	$L(A,B)$	$L(A,C)$	$L(B,C)$	$L(A,B,C)$
λ	×							
$L(A)$	×	×						
$L(B)$	×		×					
$L(C)$	×			×				
$L(A,B)$	×	×	×		×			
$L(A,C)$	×	×		×		×		
$L(B,C)$	×		×	×			×	
$L(A,B,C)$	×	×	×	×	×	×	×	×

and

$\twoheadrightarrow$	λ	$L(A)$	$L(B)$	$L(C)$	$L(A,B)$	$L(A,C)$	$L(B,C)$	$L(A,B,C)$
λ	×	•					•	×
$L(A)$	×	×	○	○	○	○	×	×
$L(B)$	×		×			×		×
$L(C)$	×			×	×			×
$L(A,B)$	×	×	×	×	×	×	×	×
$L(A,C)$	×	×	×	×	×	×	×	×
$L(B,C)$	×	×	×	×	×	×	×	×
$L(A,B,C)$	×	×	×	×	×	×	×	×

A filled circle • in line X and column Y indicates that $X \twoheadrightarrow Y$ follows from the given MVD $\lambda \twoheadrightarrow L(A)$, and a ○ in line X and column Y indicates that $X \twoheadrightarrow Y$ follows from the given MVD $L(A) \twoheadrightarrow L(B)$. This shows that $\lambda \twoheadrightarrow L(B) \notin \Sigma^+_{\mathfrak{R}}$, but σ can be derived using the pseudo-transitivity rule. □

Lemma 4.25. *The mixed pseudo-transitivity rule is independent from the set* $\mathfrak{R} = \{$*reflexivity axiom, extension rule, transitivity rule, implication rule, Brouwerian complement rule, pseudo-transitivity rule, multi-valued join rule, mixed meet rule*$\}$.

Proof. Let $N = L(A, B)$, $\Sigma = \{\lambda \twoheadrightarrow L(A), L(A) \rightarrow L(B)\}$ and $\sigma = \lambda \rightarrow L(B)$. The closure of Σ under $\mathfrak{R}$ is represented by the following tables.

$\rightarrow$	λ	$L(A)$	$L(B)$	$L(A,B)$
λ	×			
$L(A)$	×	×	×	×
$L(B)$	×		×	
$L(A,B)$	×	×	×	×

$\twoheadrightarrow$	λ	$L(A)$	$L(B)$	$L(A,B)$
λ	×	×	×	×
$L(A)$	×	×	×	×
$L(B)$	×	×	×	×
$L(A,B)$	×	×	×	×

It can be seen that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, σ can be inferred from Σ using the mixed pseudo-transitivity rule. □

In order to show the independence of the multi-valued join rule, we make use of the fact that non-maximal basis attributes cannot be represented as the Brouwerian complement of any subattribute.

Lemma 4.26. *The multi-valued join rule is independent from the set* $\mathfrak{R} = \{$*reflexivity axiom, extension rule, transitivity rule, implication rule, Brouwerian complement rule, pseudo-transitivity rule, mixed pseudo-transitivity rule, mixed meet rule*$\}$.

Proof. Let $N = L(A, K[B])$, $\Sigma = \{\lambda \twoheadrightarrow L(A), \lambda \twoheadrightarrow L(K[\lambda])\}$ and $\sigma = \lambda \twoheadrightarrow L(A, K[\lambda])$. The closure of Σ under $\mathfrak{R}$ is represented by the following tables

$\rightarrow$	λ	$L(A)$	$L(K[\lambda])$	$L([K(B)])$	$L(A, K[\lambda])$	$L(A, K[B])$
λ	×		○			
$L(A)$	×	×	○		○	
$L(K[\lambda])$	×		×			
$L(K[B])$	×		×	×		
$L(A, K[\lambda])$	×	×	×		×	
$L(A, K[B])$	×	×	×	×	×	×

and

$\twoheadrightarrow$	λ	$L(A)$	$L(K[\lambda])$	$L([K(B)])$	$L(A, K[\lambda])$	$L(A, K[B])$
λ	×	●	○	●		×
$L(A)$	×	×	○	×	○	×
$L(K[\lambda])$	×	●	×	●		×
$L(K[B])$	×	×	×	×		×
$L(A, K[\lambda])$	×	×	×	×	×	×
$L(A, K[B])$	×	×	×	×	×	×

A filled circle ● in line X and column Y indicates that $X \twoheadrightarrow Y$ follows from the given MVD $\lambda \twoheadrightarrow L(A)$, whereas a circle ○ in line X and column Y indicates that $X \rightarrow Y$ or $X \twoheadrightarrow Y$ follows from the given MVD $\lambda \twoheadrightarrow L(K[B])$. One can see that $\lambda \twoheadrightarrow L(A, K[\lambda]) \notin \Sigma^+_{\mathfrak{R}}$, but σ can be derived using the multi-valued join rule. □

Lemma 4.27. *The mixed meet rule is independent from the set* $\mathfrak{R} = \{$*reflexivity axiom, extension rule, transitivity rule, implication rule, Brouwerian complement rule, pseudo-transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule*$\}$.

Proof. Let $N = L[K(A, B)]$, $\Sigma = \{\lambda \twoheadrightarrow L[K(A)]\}$ and $\sigma = \lambda \twoheadrightarrow L[\lambda]$. We use σ instead of $\lambda \rightarrow L[\lambda]$ for technical reasons. The closure of Σ under $\mathfrak{R}$ is represented by the following tables

$\rightarrow$	λ	$L[\lambda]$	$L[K(A)]$	$L[K(B)]$	$L[K(A,B)]$
λ	$\times$				
$L[\lambda]$	$\times$	$\times$			
$L[K(A)]$	$\times$	$\times$	$\times$		
$L[K(B)]$	$\times$	$\times$		$\times$	
$L[K(A,B)]$	$\times$	$\times$	$\times$	$\times$	$\times$

and

$\twoheadrightarrow$	λ	$L[\lambda]$	$L[K(A)]$	$L[K(B)]$	$L[K(A,B)]$
λ	$\times$		$\bullet$	$\bullet$	$\times$
$L[\lambda]$	$\times$	$\times$	$\bullet$	$\bullet$	$\times$
$L[K(A)]$	$\times$	$\times$	$\times$	$\times$	$\times$
$L[K(B)]$	$\times$	$\times$	$\times$	$\times$	$\times$
$L[K(A,B)]$	$\times$	$\times$	$\times$	$\times$	$\times$

as before. A filled circle $\bullet$ in line X and column Y indicates that $X \twoheadrightarrow Y$ follows from the given MVD $\lambda \twoheadrightarrow L[K(A)]$. One can see that $\lambda \rightarrow L[\lambda] \notin \Sigma^+_{\mathfrak{R}}$, but it can be derived using the mixed meet rule. Furthermore $\sigma \notin \Sigma^+_{\mathfrak{R}}$, but σ can be derived by applying the implication rule to $\lambda \rightarrow L[\lambda]$. □

The combination of the previous lemmata gives the following result.

Theorem 4.28. *Reflexivity axiom, extension rule, transitivity rule, implication rule, Brouwerian complement rule, pseudo-transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule and mixed meet rule form a minimal, sound and complete set of inference rules for the implication of FDs and MVDs in the presence of records and lists.* □

Theorem 4.28 is somewhat surprising. We have seen that in the presence of lists the multi-valued join rule is independent from the rest of the rules in Theorem 4.28. For relational databases, however, it was proven in [204] that the multi-valued join rule is logically implied by a corresponding subset of the rules above. The fact that this is not the case in the presence of lists results from the existence of some subattributes which are not the Brouwerian complement of any other subattributes, e.g. $L[\lambda]$ is not the Brouwerian complement of any subattribute of $L[A]$.

4.3 Brouwerian-Complement Rule

The Brouwerian complement rule is the analogue of the complementation rule for MVDs in relational databases. There are a few papers [32, 46, 204] which point at the significance of the complementation rule. In fact, it is the only rule that does not have a direct analogue in the axiomatisation of FDs since it is the only rule that takes into account the context

of the dependencies, that is, the underlying relation schema R. The rest of the inference rules apply independently of whatever relation schema the attributes are embedded in.

The situation is again slightly different in the presence of lists. Here, the Brouwerian complement rule is not the only context-sensitive rule. The mixed meet rule depends on the underlying nested attribute as well. That is, the FD $X \to Y \sqcap Y_N^{\mathcal{C}}$ can be inferred from the MVD $X \twoheadrightarrow Y$.

EXAMPLE 4.8. Let $Y = L(\lambda, K[\lambda])$, $N_1 = L(A, K[\lambda])$ and $N_2 = L(A, K[B])$. In these cases, $Y_{N_1}^{\mathcal{C}} = L(A, \lambda)$ and $Y_{N_2}^{\mathcal{C}} = L(A, K[B])$. It follows that $Y \sqcap_{N_1} Y_{N_1}^{\mathcal{C}} = L(\lambda, \lambda)$, but $Y \sqcap_{N_2} Y_{N_2}^{\mathcal{C}} = L(\lambda, K[\lambda])$. □

We delay a detailed study of the mixed meet rule to future research, and focus for the remainder of this section on the role of the Brouwerian complement rule. It is interesting to study whether one can obtain a (minimal) sound and complete set of inference rules that does not include the Brouwerian complement rule. A further reason for this might appear in the future when the algorithmic synthesis of nested attributes will be studied. This will cause at least as many problems as in relational databases [29, 204].

Given a nested attribute N we introduce the N-axiom $\dfrac{}{\lambda \twoheadrightarrow N}$ which is sound by the definition of MVDs. It is a very weak form of the Brouwerian complement rule, and corresponds to the so-called R-axiom $\dfrac{}{\emptyset \twoheadrightarrow R}$ for a relation schema R in relational databases (see [46]). We will show in this section that the Brouwerian complement rule in the minimal set of inference rules from Theorem 4.28 can be replaced by the N-axiom and still maintain minimality.

First of all, we will show that N-axiom and Brouwerian complement rule are equivalent in the presence of reflexivity axiom, implication rule and pseudo-transitivity rule.

Lemma 4.29. *The Brouwerian complement rule is not independent from {reflexivity axiom, N-axiom, implication rule, pseudo-transitivity}. Furthermore, the N-axiom is not independent from {reflexivity axiom, Brouwerian complement rule, implication rule}.*

Proof. The derivation tree for the first statement is given by

$$\dfrac{X \twoheadrightarrow Y \qquad \dfrac{\dfrac{\dfrac{}{Y \to \lambda}^{\lambda \leq Y}}{Y \twoheadrightarrow \lambda} \qquad \dfrac{}{\lambda \twoheadrightarrow N}}{Y \twoheadrightarrow N}}{X \twoheadrightarrow \underbrace{N \dot{-} Y}_{=Y^{\mathcal{C}}}}$$

and the proof for the second statement is given by

$$\dfrac{\dfrac{\dfrac{}{\lambda \to \lambda}^{\lambda \leq \lambda}}{\lambda \twoheadrightarrow \lambda}}{\lambda \twoheadrightarrow N} \quad .$$

□

While all elements of $\mathfrak{R} = \{$reflexivity axiom, extension rule, transitivity rule, implication rule, pseudo-transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule, mixed meet rule$\}$ are still sound when the MVDs are interpreted as FDs, the N-axiom is not. This shows the independence of the N-axiom from $\mathfrak{R}$ above.

Let $\mathfrak{R} = \{$reflexivity axiom, extension rule, transitivity rule, implication rule, N-axiom, pseudo-transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule, mixed meet rule$\}$. It follows from Lemma 4.29 and the lemmata from the previous section that R is independent from $\mathfrak{R} - \{R\}$, if $R \in \{$extension rule, transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule, mixed meet rule$\}$. We deal with the remaining cases in the following lemma.

Lemma 4.30. *Let $\mathfrak{R} = \{$reflexivity axiom, extension rule, transitivity rule, implication rule, N-axiom, pseudo-transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule, mixed meet rule$\}$. For $R \in \{$reflexivity axiom, implication rule, pseudo-transitivity rule$\}$ it follows that R is independent from $\mathfrak{R} - \{R\}$.*

Proof. The independence of the reflexivity axiom R from $\mathfrak{R} - \{R\}$ follows from the fact that there is no other axiom to infer any trivial FDs.

Let R be the implication rule, $N = A, \Sigma = \emptyset$ and $\sigma = A \twoheadrightarrow \lambda$. Then $\Sigma^+_{\mathfrak{R}-\{R\}} = \{\lambda \rightarrow \lambda, A \rightarrow \lambda, A \rightarrow A, \lambda \twoheadrightarrow A\}$, but σ can be derived by the implication rule.

Let R be the pseudo-transitivity rule, $N = L(A,B), \Sigma = \{\lambda \twoheadrightarrow L(A), L(A) \twoheadrightarrow L(B)\}$ and $\sigma = \lambda \twoheadrightarrow L(B)$. The closure of Σ under $\mathfrak{R} - \{R\}$ is represented by the following two tables.

$\rightarrow$	λ	$L(A)$	$L(B)$	$L(A,B)$
λ	×			
$L(A)$	×	×		
$L(B)$	×		×	
$L(A,B)$	×	×	×	×

$\twoheadrightarrow$	λ	$L(A)$	$L(B)$	$L(A,B)$
λ	×	○		×
$L(A)$	×	×	•	•
$L(B)$	×		×	
$L(A,B)$	×	×	×	×

A filled circle • in line X and column Y indicates that $X \twoheadrightarrow Y$ follows from the given MVD $L(A) \twoheadrightarrow L(B)$, and a ○ in line X and column Y indicates that $X \rightarrow Y$ follows from the given MVD $\lambda \twoheadrightarrow L(A)$. One can see that $\sigma \notin \Sigma^+_{\mathfrak{R}-\{R\}}$, but σ can be derived from Σ by the pseudo-transitivity rule. □

We obtain the following result.

Theorem 4.31. *Reflexivity axiom, extension rule, transitivity rule, implication rule, N-axiom, pseudo-transitivity rule, mixed pseudo-transitivity rule, multi-valued join rule, and mixed meet rule form a minimal, sound and complete set of inference rules for the implication of FDs and MVDs in the presence of records and lists.* □

4.4 Implication Problem

The implication problem for FDs and MVDs is to decide whether for an arbitrary nested attribute N, an arbitrary set Σ of FDs and MVDs on N and an arbitrary FD or MVD σ on N, $\Sigma \models \sigma$ holds. According to Corollary 4.14 finite and unrestricted implication problem coincide. We will now present a provably-correct algorithm for deciding the implication problem. It is shown that the algorithm works in polynomial time in the number of basis attributes of the underlying nested attribute and the number of FDs and MVDs given.

4.4.1 The Algorithm

Proposition 4.8 suggests a strategy for deciding the implication problem for the class of FDs and MVDs. An MVD $X \twoheadrightarrow Y$ is implied by a set Σ of FDs and MVDs if and only if Y is the join of some elements of the dependency basis of X with respect to Σ. Furthermore, an FD $X \rightarrow Y$ logically follows from Σ if and only if Y is a subattribute of the nested attribute closure X^+ of X with respect to Σ. The implication problem $\Sigma \models \sigma$ where σ is an FD or MVD with left-hand side X reduces therefore to the problem of computing the dependency basis $DepB(X)$ for any X and with respect to any set Σ in the sense of Definition 4.6. The following algorithm extends the well-known membership algorithm for relational databases proposed in [27].

The idea of this algorithm is to start with trivial values for nested attribute closure X^+ (in the algorithm X_{new}) and X^M (in the algorithm DB_{new}) in line (1) and (2). The algorithm then tries to further extend X^+ and decompose X^M as long as there are any changes to one of them after considering all FDs and MVDs in Σ (REPEAT loop from line (3) to line (26)). As it will turn out in the correctness proof later it is sufficient to consider all the FDs and MVDs in Σ. That is, any dependencies in $\Sigma^+ - \Sigma$ do not need to be considered. This leads ultimately to a polynomial-time algorithm. If $X \rightarrow \widetilde{V} \in \Sigma^+$ at some stage, then X^+ is extended by $\widetilde{V}$ in line (11). Furthermore, every maximal basis attribute of N that is a subattribute of $\widetilde{V}$ becomes a new singleton in X^M, and all previous $W \in X^M$ are reduced to $(W \dot{-} \widetilde{V})^{CC}$ in line (12). If $X \twoheadrightarrow \widetilde{V} \in \Sigma^+$ at some stage, then X^+ is extended by $\widetilde{V} \sqcap \widetilde{V}^C$ in line (19) according to the mixed meet rule. Suppose there is some $W \in X^M$ which shares at least some maximal basis attribute of N with $\widetilde{V}$ $((\widetilde{V} \sqcap W)^{CC} \neq \lambda)$, but not all of the maximal basis attributes of N which are subattributes of W are also subattributes of $\widetilde{V}$ $((\widetilde{V} \sqcap W)^{CC} \neq W)$. Such a W is then replaced by two new elements, $(\widetilde{V} \sqcap W)^{CC}$ and $(W \dot{-} \widetilde{V})^{CC}$ of X^M in line (22). That means a finer partition has been found according to pseudo-difference and multi-valued meet rule. More details of the algorithm can be found in the correctness proof.

Algorithm 4.4.1 (Nested Attribute Closure and Dependency Basis)
Input: $\boldsymbol{N} \in \boldsymbol{N}\text{A}$, $\boldsymbol{X} \in \boldsymbol{Sub(N)}$, set $\boldsymbol{\Sigma}$ of FDs and MVDs on $\boldsymbol{N}$
Output: $\boldsymbol{X}^+_{\text{alg}}$ and $\text{DepB}_{\text{alg}}(\boldsymbol{X})$
Method:

VAR $DB_{\text{new}}, DB_{\text{old}} \subseteq Sub(N)$, $X_{\text{new}}, X_{\text{old}}, U, V, W, \overline{U}, \widetilde{V}, U' \in Sub(N)$;

(1) $X_{\text{new}} := X$;
(2) $DB_{\text{new}} := MaxB(X^{cc}) \cup \{X^c\}$;
(3) **REPEAT**
(4) $X_{\text{old}} := X_{\text{new}}$;
(6) $DB_{\text{old}} := DB_{\text{new}}$;
(7) **FOR each** $U \rightarrow V \in \Sigma$ **DO**
(8) $\overline{U} := \bigsqcup\{W \in DB_{\text{new}} \mid \exists U'.U' \text{ possessed by } W, U' \not\leq X_{\text{new}}, U' \leq U\}$;
(9) $\widetilde{V} := V \dot{-} \overline{U}$;
(10) **IF** $\widetilde{V} \neq \lambda$ **THEN BEGIN**
(11) $X_{\text{new}} := X_{\text{new}} \sqcup \widetilde{V}$;
(12) $DB_{\text{new}} := \{(W \dot{-} \widetilde{V})^{cc} \mid W \in DB_{\text{new}}, (W \dot{-} \widetilde{V})^{cc} \neq \lambda\} \cup MaxB(\widetilde{V}^{cc})$;
(13) **END**;
(14) **ENDDO**;
(15) **FOR each** $U \twoheadrightarrow V \in \Sigma$ **DO**
(16) $\overline{U} := \bigsqcup\{W \in DB_{\text{new}} \mid \exists U'.U' \text{ possessed by } W, U' \not\leq X_{\text{new}}, U' \leq U\}$;
(17) $\widetilde{V} := V \dot{-} \overline{U}$;
(18) **IF** $\widetilde{V} \neq \lambda$ **THEN BEGIN**
(19) $X_{\text{new}} := X_{\text{new}} \sqcup (\widetilde{V} \sqcap \widetilde{V}^c)$;
(20) **FOR each** $W \in DB_{\text{new}}$ **DO**
(21) **IF** $(\widetilde{V} \sqcap W)^{cc} \neq \lambda$ **AND** $(\widetilde{V} \sqcap W)^{cc} \neq W$ **THEN**
(22) $DB_{\text{new}} := (DB_{\text{new}} - \{W\}) \cup \{(\widetilde{V} \sqcap W)^{cc}, (W \dot{-} \widetilde{V})^{cc}\}$;
(23) **ENDDO**;
(24) **END**;
(25) **ENDDO**;
(26) **UNTIL** $(X_{\text{new}} = X_{\text{old}})$ **AND** $(DB_{\text{new}} = DB_{\text{old}})$;
(27) $X^+_{\text{alg}} := X_{\text{new}}$; $X^M_{\text{alg}} := DB_{\text{new}}$;
(28) $DepB_{\text{alg}}(X) := SubB(X^+_{\text{alg}}) \cup X^M_{\text{alg}}$;
(29) **RETURN**$(X^+_{\text{alg}}, DepB_{\text{alg}}(X))$;

□

In order to become more familiar with Algorithm 4.4.1 we present an example.

EXAMPLE 4.9. Suppose the input for Algorithm 4.4.1 is as follows:

- $N = L_1(L_2[L_3[L_4(A, B, C)]], L_5[L_6(D, E)], L_7(F, L_8[L_9(G, L_{10}[H])], I))$,
- $U_1 = L_1(L_5[\lambda], L_7(F, L_8[L_9(G)], I))$, $V_1 = L_1(L_2[L_3[L_4(C)]], L_5[L_6(E)])]$,
- $U_2 = L_1(L_2[L_3[\lambda]], L_7(F))$, $V_2 = L_1(L_2[L_3[L_4(A)]], L_7(L_8[L_9(G)], I))$,
- $U_3 = L_1(L_7(F, L_8[L_9(L_{10}[\lambda])]))$, $V_3 = L_1(L_2[L_3[\lambda]], L_5[L_6(D)])$,
- $\Sigma = \{U_1 \twoheadrightarrow V_1, U_2 \rightarrow V_2, U_3 \twoheadrightarrow V_3\}$ and $X = L_1(L_7(F, L_8[L_9(L_{10}[H])]))$.

After initialisation we have

- $X_{\text{new}} = X$, and
- $DB_{\text{new}} = \{L_1(L_2[L_3[L_4(A,B,C)]], L_5[L_6(D,E)], L_7(L_8[L_9(G)], I)); \quad L_1(L_7(F)); L_1(L_7(L_8[L_9(L_{10}[H])]))\}$.

The initial state is illustrated in Figure 4.3. Functionally determined basis attributes are marked with a circle, remaining maximal basis attributes appear in different boxes according to their membership.

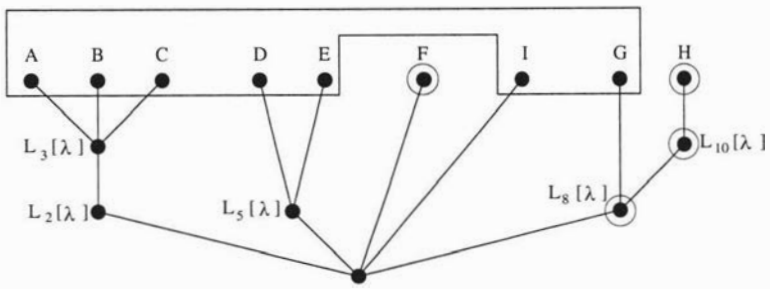

Fig. 4.3. Initialisation for $DepB_{\text{alg}}(X)$.

The first pass through the REPEAT loop between line (3) and (26) yields the following intermediate results:

1. $U_2 \rightarrow V_2$:
   $\overline{U} = L_1(L_2[L_3[L_4(A,B,C)]], L_5[L_6(D,E)], L_7(L_8[L_9(G)], I))$,
   $\widetilde{V} = \lambda$ and therefore no changes
2. $U_1 \twoheadrightarrow V_1$:
   $\overline{U} = L_1(L_2[L_3[L_4(A,B,C)]], L_5[L_6(D,E)], L_7(L_8[L_9(G)], I))$,
   $\widetilde{V} = \lambda$ and therefore no changes
3. $U_3 \twoheadrightarrow V_3$:
   $\overline{U} = \lambda$, $\widetilde{V} = V_3$,
   $X_{\text{new}} = L_1(L_2[L_3[\lambda]], L_5[\lambda], L_7(F, L_8[L_9(L_{10}[H])]))$
   $DB_{\text{new}} = \{L_1(L_2[L_3[L_4(A,B,C)]], L_5[L_6(E)], L_7(L_8[L_9(G)], I)); \quad L_1(L_7(F)); L_1(L_7(L_8[L_9(L_{10}[H])])); L_1(L_5[L_6(D)])\}$

The second pass through the REPEAT loop yields the following intermediate results:

1. $U_2 \rightarrow V_2$:
   $\overline{U} = \lambda$, $\widetilde{V} = V_2$,
   $X_{\text{new}} = L_1(L_2[L_3[L_4(A)]], L_5[\lambda], L_7(F, L_8[L_9(G, L_{10}[H])], I))$,
   $DB_{\text{new}} = \{L_1(L_2[L_3[L_4(A)]]); L_1(L_7(L_8[L_9(G)])); L_1(L_7(I)); L_1(L_2[L_3[L_4(B,C)]], L_5[L_6(E)]); L_1(L_7(F)); L_1(L_7(L_8[L_9(L_{10}[H])])); L_1(L_5[L_6(D)])\}$
2. $U_1 \twoheadrightarrow V_1$:
   $\overline{U} = \lambda$, $\widetilde{V} = V_1$,
   $X_{\text{new}} = L_1(L_2[L_3[L_4(A)]], L_5[\lambda], L_7(F, L_8[L_9(G, L_{10}[H])], I))$,

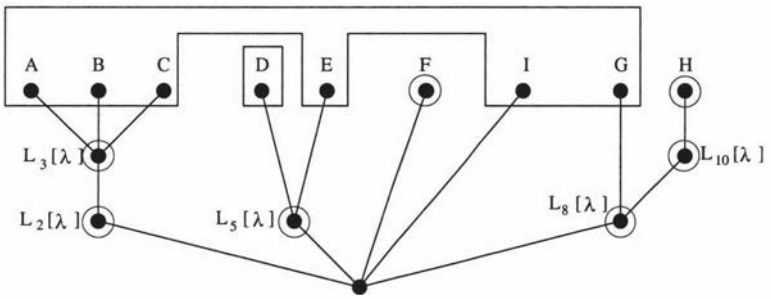

Fig. 4.4. $DepB_{\text{alg}}(X)$ after its first Update.

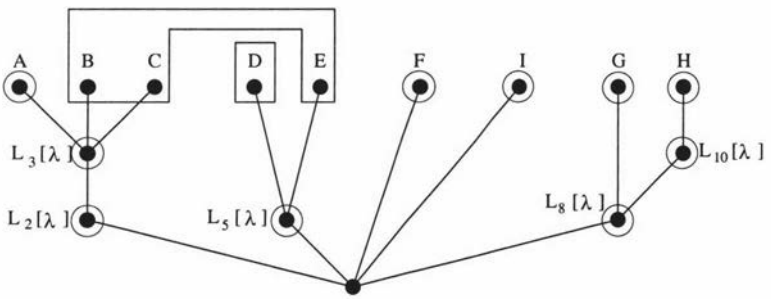

Fig. 4.5. $DepB_{\text{alg}}(X)$ after its second Update.

$DB_{\text{new}} = \{L_1(L_2[L_3[L_4(A)]]);\ L_1(L_7(L_8[L_9(G)]));\ L_1(L_7(I));\ L_1(L_2[L_3[L_4(B)]]);$
$L_1(L_2[L_3[L_4(C)]], L_5[L_6(E)]);\ L_1(L_7(F));\ L_1(L_7(L_8[L_9(L_{10}[H])]));\ L_1(L_5[L_6(D)])\}$

3. $U_3 \twoheadrightarrow V_3$:
   $\overline{U} = \lambda$, $\overline{V} = V_3$ and therefore no changes.

The next pass through the REPEAT loop yields nothing new. We therefore have the output

- $X^+_{\text{alg}} = L_1(L_2[L_3[L_4(A)]], L_5[\lambda], L_7(F, L_8[L_9(G, L_{10}[H])], I))$ and
- $DepB_{\text{alg}}(X) = \{L_1(L_2[\lambda]);\ L_1(L_2[L_3[\lambda]]);\ L_1(L_2[L_3[L_4(A)]]);\ L_1(L_5[\lambda]);\ L_1(L_7(F));$
  $L_1(L_7(L_8[\lambda]));\ L_1(L_7(L_8[L_9(G)]));\ L_1(L_7(L_8[L_9(L_{10}[\lambda])]));\ L_1(L_7(L_8[L_9(L_{10}[H])]));$
  $L_1(L_7(I));\ L_1(L_5[L_6(D)]);\ L_1(L_2[L_3[L_4(B)]]);\ L_1(L_2[L_3[L_4(C)]], L_5[L_6(E)])\}$.

Figure 4.6 illustrates $DepB_{\text{alg}}(X)$. □

4.4.2 Correctness

As mentioned beforehand one major objective of this section is to prove that Algorithm 4.4.1 is correct, i.e., $X^+_{\text{alg}} = X^+$ and $DepB_{\text{alg}}(X) = DepB(X)$. Before doing so we show a technical lemma which is a result of Brouwerian algebras that is not specific to $Sub(N)$.

Lemma 4.32. *Let $N \in \mathcal{NA}$ and $X, Y, Z \in Sub(N)$. Then $X \dot{-} Y = (X \dot{-} (Y \sqcup Z)) \sqcup ((X \sqcap Z) \dot{-} Y)$.*

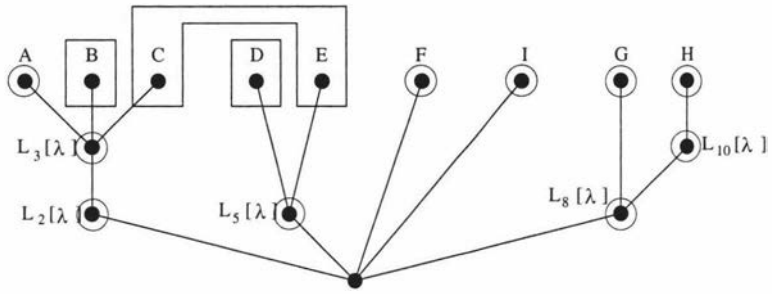

Fig. 4.6. Final State for $DepB_{\text{alg}}(X)$ from Example 4.9.

Proof. From

$$\begin{aligned} X \sqcup Y &= (X \sqcup Y \sqcup Z) \sqcap (X \sqcup Y) \\ &= (X \sqcup Y \sqcup Z) \sqcap (X \sqcup Y \sqcup (X \dot{-} (Y \sqcup Z))) \\ &= (Y \sqcup Z \sqcup (X \dot{-} (Y \sqcup Z))) \sqcap (X \sqcup Y \sqcup (X \dot{-} (Y \sqcup Z))) \\ &= ((Y \sqcup Z) \sqcap (X \sqcup Y)) \sqcup (X \dot{-} (Y \sqcup Z)) \\ &= Y \sqcup (X \sqcap Z) \sqcup (X \dot{-} (Y \sqcup Z)) \\ &= Y \sqcup ((X \sqcap Z) \dot{-} Y) \sqcup (X \dot{-} (Y \sqcup Z)) \end{aligned}$$

follows $X \leq Y \sqcup ((X \sqcap Z) \dot{-} Y) \sqcup (X \dot{-} (Y \sqcup Z))$, i.e., $X \dot{-} Y \leq ((X \sqcap Z) \dot{-} Y) \sqcup (X \dot{-} (Y \sqcup Z))$. It remains to show that $((X \sqcap Z) \dot{-} Y) \sqcup (X \dot{-} (Y \sqcup Z)) \leq X \dot{-} Y$ holds. This follows from $X \dot{-} (Y \sqcup Z) \leq X \dot{-} Y$ and $(X \sqcap Z) \dot{-} Y \leq X \dot{-} Y$. □

The next result shows that Algorithm 4.4.1 does not compute anything wrong, i.e., every element of $DepB_{\text{alg}}(X)$ is indeed in $Dep(X)$ and X^+_{alg} is a subattribute of X^+.

Lemma 4.33. *Let $N \in \mathcal{NA}$, $X \in Sub(N)$, Σ a set of FDs and MVDs on N and $(X^+_{alg}, DepB_{alg}(X))$ the output of Algorithm 4.4.1 with $DepB_{alg}(X) = SubB(X^+_{alg}) \cup X^M_{alg}$. Then*

- *$X \twoheadrightarrow W_j \in \Sigma^+$ for all $W_j \in DepB_{alg}(X)$ and*
- *$X \rightarrow X^+_{alg} \in \Sigma^+$ hold.*

Proof. The proof is done by induction on the number of passes through the two FOR loops between line (7) and line (25) of Algorithm 4.4.1. Let X^+_i and X^M_i denote X_{new} and DB_{new}, respectively, after the ith pass.

After initialisation we have $X^+_0 = X$ on one hand and $X \rightarrow X^+_0 \in \Sigma^+$ by the reflexivity rule. On the other hand $X^M_0 = MaxB(X^{cc}) \cup \{X^c\}$, i.e., $W_j \in X^M_0$ implies $W_j \in MaxB(X^{cc})$ or $W_j = X^c$. In the first case we have $X \rightarrow X \in \Sigma^+$ by reflexivity, which implies $X \rightarrow W_j \in \Sigma^+$ by the subattribute rule for FDs since $W_j \leq X$ and finally $X \twoheadrightarrow W_j \in \Sigma^+$ by the implication rule. In the second case we obtain again $X \rightarrow X \in \Sigma^+$ by reflexivity, then $X \twoheadrightarrow X \in \Sigma^+$ by implication and finally $X \twoheadrightarrow X^c \in \Sigma^+$ by the Brouwerian complement rule.

Let now $i > 0$ and $U \to V \in \Sigma$ the functional dependency selected in the next pass. Suppose that $X_{i+1}^+ \neq X_i^+$ or $X_{i+1}^M \neq X_i^M$ since there is nothing to show otherwise. First we have $X \to X_i^+ \in \Sigma^+$ by hypothesis. Using the implication rule we derive $X \twoheadrightarrow X_i^+ \in \Sigma^+$. Moreover, $X \twoheadrightarrow \overline{U} \in \Sigma^+$ by the join rule for multi-valued dependencies and the hypothesis. Applying again the join rule for multi-valued dependencies gives us $X \twoheadrightarrow X_i^+ \sqcup \overline{U} \in \Sigma^+$. On the other hand we have $U \leq \overline{U} \sqcup X_i^+$, i.e., $X_i^+ \sqcup \overline{U} \to U \in \Sigma^+$ by reflexivity. Since $U \to V \in \Sigma$ we can apply the transitivity rule to derive $X_i^+ \sqcup \overline{U} \to V \in \Sigma^+$. An application of the mixed pseudo-transitivity rule to $X \twoheadrightarrow X_i^+ \sqcup \overline{U}, X_i^+ \sqcup \overline{U} \to V \in \Sigma^+$ leaves us with $X \to (V \dot{-} (X_i^+ \sqcup \overline{U})) \in \Sigma^+$. The subattribute rule can be applied to $X \to X_i^+ \in \Sigma^+$ to infer $X \to (X_i^+ \sqcap V) \dot{-} \overline{U} \in \Sigma^+$. The join rule for functional dependencies is then applied to $X \to (V \dot{-} (X_i^+ \sqcup \overline{U})), X \to (X_i^+ \sqcap V) \dot{-} \overline{U} \in \Sigma^+$, and using Lemma 4.32 we obtain $X \to V \dot{-} \overline{U} \in \Sigma^+$. This is $X \to \widetilde{V} \in \Sigma^+$ by definition, and applying the join rule for functional dependencies to $X \to X_i^+ \in \Sigma^+$ tells us that $X \to X_i^+ \sqcup \widetilde{V} \in \Sigma^+$, i.e., $X \to X_{i+1}^+ \in \Sigma^+$ holds.

Let $W_j \in X_{i+1}^M = \{(W \dot{-} \widetilde{V})^{cc} \mid W \in X_i^M, (W \dot{-} \widetilde{V})^{cc} \neq \lambda\} \cup MaxB(\widetilde{V}^{cc})$. If $W_j \in X_i^M$, then there is nothing to show. Otherwise, if $W_j \leq \widetilde{V}^{cc} \leq \widetilde{V}$, then it follows $X \to W_j \in \Sigma^+$ by the subattribute rule from $X \to \widetilde{V} \in \Sigma^+$, and then $X \twoheadrightarrow W_j \in \Sigma^+$ by the implication rule. Let now be $W_j = (W \dot{-} \widetilde{V})^{cc}$ for some $W \in X_i^M$. We obtain $X \twoheadrightarrow \widetilde{V} \in \Sigma^+$ by the implication rule. Furthermore, $X \twoheadrightarrow W \in \Sigma^+$ by hypothesis which leads to $X \twoheadrightarrow W \dot{-} \widetilde{V} \in \Sigma^+$ by application of the pseudo-difference rule. Applying the Brouwerian complement rule twice gives us $X \twoheadrightarrow W_j \in \Sigma^+$. This shows $X \twoheadrightarrow W_j \in \Sigma^+$ for all $W_j \in X_{i+1}^M$, and completes the proof for the first FOR loop between line (7) and (14) where functional dependencies are selected.

Consider now the second FOR loop between line (15) and (25) where multi-valued dependencies are selected. Suppose $U \twoheadrightarrow V \in \Sigma$ is used in the next pass. As before we derive $X \twoheadrightarrow \overline{U} \sqcup X_i^+ \in \Sigma^+$. From $U \twoheadrightarrow V \in \Sigma$, $U \leq \overline{U} \sqcup X_i^+$ and $\lambda \leq \overline{U} \sqcup X_i^+$ it follows that $\overline{U} \sqcup X_i^+ \twoheadrightarrow V \in \Sigma^+$ by the augmentation rule. An application of the pseudo-transitivity rule leaves us with $X \twoheadrightarrow V \dot{-} (\overline{U} \sqcup X_i^+) \in \Sigma^+$. On the other hand we can derive $X \to (X_i^+ \sqcap V) \dot{-} \overline{U} \in \Sigma^+$, and $X \twoheadrightarrow (X_i^+ \sqcap V) \dot{-} \overline{U} \in \Sigma^+$ by the implication rule. Using now the join rule for multi-valued dependencies, and applying Lemma 4.32 gives us $X \twoheadrightarrow V \dot{-} \overline{U} \in \Sigma^+$ which is $X \twoheadrightarrow \widetilde{V} \in \Sigma^+$. It follows that $X \to \widetilde{V} \sqcap \widetilde{V}^C \in \Sigma^+$ by using the mixed meet rule. Applying the join rule for functional dependencies to $X \to X_i^+ \in \Sigma^+$ leads to $X \to X_i^+ \sqcup (\widetilde{V} \sqcap \widetilde{V}^C) \in \Sigma^+$, i.e., $X \to X_{i+1}^+ \in \Sigma^+$.

Let now $W_j \in X_i^M$ and $\lambda \neq (\widetilde{V} \sqcap W_j)^{cc} \neq W_j$. Then follows $X \twoheadrightarrow W_j \in \Sigma^+$ by hypothesis. From $X \twoheadrightarrow \widetilde{V} \in \Sigma^+$ follows $X \twoheadrightarrow W_j \sqcap \widetilde{V} \in \Sigma^+$ by the multi-valued meet rule on one hand, and $X \twoheadrightarrow W_j \dot{-} \widetilde{V} \in \Sigma^+$ by the pseudo-difference rule on the other hand. A double application of the Brouwerian complement rule leads to $X \twoheadrightarrow (W_j \sqcap \widetilde{V})^{cc} \in \Sigma^+$ and $X \twoheadrightarrow (W_j \dot{-} \widetilde{V})^{cc} \in \Sigma^+$, respectively. This proves that $X \twoheadrightarrow W_j \in \Sigma^+$ for all $W_j \in X_{i+1}^M$, and completes the proof. □

Consider the proper chain $\Sigma = \Sigma^0 \subset \Sigma^1 \subset \cdots \subset \Sigma^n = \Sigma^+$ where Σ^i results from Σ^{i-1} by adding exactly one functional or multi-valued dependency which is not in Σ^{i-1} and can

be derived by applying one of the inference rules from Proposition 4.5 to dependencies in Σ^{i-1}. On the way to showing the correctness of Algorithm 4.4.1 we have to justify that it is sufficient to consider FDs and MVDs in Σ. That is, we need to show that dependencies in $\Sigma^+ - \Sigma$ do not alter the dependency basis. Suppose the algorithm does not only select $U \to V, U \twoheadrightarrow V \in \Sigma$ within line (7) and (15), respectively, but all FDs and MVDs from some fixed Σ^i instead. Denoting the respective output by $(X^+_{\text{alg},i}, DepB_{\text{alg},i}(X))$, we define

$$\begin{aligned}(\Sigma^i)^+_{\text{alg}} \quad = \quad & \{X \to Y \mid Y \leq X^+_{\text{alg},i}\} \cup \\ & \{X \twoheadrightarrow Y \mid Y = \sqcup Z \text{ for some } Z \subseteq DepB_{\text{alg},i}(X)\}.\end{aligned}$$

Then it is obvious that $\Sigma^+_{\text{alg}} = (\Sigma^0)^+_{\text{alg}} \subseteq (\Sigma^1)^+_{\text{alg}} \subseteq \cdots \subseteq (\Sigma^n)^+_{\text{alg}}$ holds. We are going to show that $\Sigma^i \subseteq (\Sigma^i)^+_{\text{alg}}$ for every $i = 0, \ldots, n$, and $\Sigma^+_{\text{alg}} = (\Sigma^1)^+_{\text{alg}}$. Therefore $\Sigma^+ \subseteq (\Sigma^n)^+_{\text{alg}} = \cdots = (\Sigma^1)^+_{\text{alg}} = \Sigma^+_{\text{alg}}$ holds. Since also $\Sigma^+_{\text{alg}} \subseteq \Sigma^+$ by Lemma 4.33 and Proposition 4.8, we have indeed shown that $\Sigma^+_{\text{alg}} = \Sigma^+$.

Lemma 4.34. *For every $i = 0, \ldots, n$ we have $\Sigma^i \subseteq (\Sigma^i)^+_{alg}$.*

Proof. Let $X \to Y \in \Sigma^i$ be arbitrary. Then we choose to compute $X^+_{\text{alg},i}$ and $DepB_{\text{alg},i}(X)$ using Algorithm 4.4.1 where functional and multi-valued dependencies in line (7) and (15), respectively, are selected from Σ^i. Due to initialisation we have $X \leq X_{\text{new}}$ at any time during the computation. Since $X \to Y \in \Sigma^i$, $X \to Y$ will be selected at some point. In this case, $\overline{X} = \lambda$ since every X' with $X' \leq X$ also satisfies $X' \leq X_{\text{new}}$. Consequently, $\widetilde{Y} = Y \dot{-} \lambda = Y$ and, therefore, $X_{\text{new}} := X_{\text{new}} \sqcup Y$ which implies $Y \leq X^+_{\text{alg},i}$. This, however, means that $X \to Y \in (\Sigma^i)^+_{\text{alg}}$.

Let $X \twoheadrightarrow Y \in \Sigma^i$ be arbitrary. Again, we choose to compute $X^+_{\text{alg},i}$ and $DepB_{\text{alg},i}(X) = SubB(X^+_{\text{alg},i}) \cup X^M_{\text{alg},i}$ using Algorithm 4.4.1 where functional and multi-valued dependencies in line (7) and (15), respectively, are selected from Σ^i. Due to initialisation we have $X \leq X_{\text{new}}$ at any time during the computation. Since $X \twoheadrightarrow Y \in \Sigma^i$, $X \twoheadrightarrow Y$ will be selected at some point. In this case, $\overline{X} = \lambda$ since every X' with $X' \leq X$ also satisfies $X' \leq X_{\text{new}}$. Consequently, $\widetilde{Y} = Y \dot{-} \lambda = Y$ and, therefore, $X_{\text{new}} := X_{\text{new}} \sqcup (Y \sqcap Y^C)$. This implies $Y \sqcap Y^C \leq X^+_{\text{alg},i}$. Moreover, $Y^{CC} = \bigsqcup\{W \in DB_{\text{new}} \mid (Y \sqcap W)^{CC} \neq \lambda\}$ after the inner FOR-loop between line (20) and (23). It follows that $Y^{CC} = \bigsqcup\{W \in X^M_{\text{alg},i} \mid (Y \sqcap W)^{CC} \neq \lambda\}$. Since $Y = Y^{CC} \sqcup (Y \sqcap Y^C)$, we conclude that $Y = \sqcup Z$ for some $Z \subseteq DepB_{\text{alg},i}(X)$ and, therefore, $X \twoheadrightarrow Y \in (\Sigma^i)^+_{\text{alg}}$. □

The following lemma is a reminder of some algebraic properties of Brouwerian algebras.

Lemma 4.35. *Let $N \in \mathcal{NA}$ and $X, Y, Z \in Sub(N)$. Then*

1. $(X \dot{-} Y) \dot{-} Z = X \dot{-} (Y \sqcup Z)$,
2. $X \dot{-} Y \leq Y^C$,
3. $X^C \dot{-} Y \leq (X \dot{-} Y)^C$, *and*
4. $X^C \dot{-} Y = (X \sqcup Y)^C$.

Proof. Consider the first equation. As $X \leq X \sqcup Y \sqcup Z = (X \dot{-} (Y \sqcup Z)) \sqcup Y \sqcup Z$ we have $X \dot{-} Y \leq (X \dot{-} (Y \sqcup Z)) \sqcup Z$ and this means $(X \dot{-} Y) \dot{-} Z \leq (X \dot{-} (Y \sqcup Z))$. Vice versa, $X \leq X \sqcup Y \sqcup Z = Z \sqcup Y \sqcup (X \dot{-} Y) = Z \sqcup Y \sqcup ((X \dot{-} Y) \dot{-} Z)$ and, consequently, $X \dot{-} (Y \sqcup Z) \leq (X \dot{-} Y) \dot{-} Z$.

The second law follows immediately from $X \leq Y \sqcup Y^{\mathcal{C}} = N$.

The third law follows from the fact that $X \sqcup Y \sqcup (X \dot{-} Y)^{\mathcal{C}} = (X \dot{-} Y) \sqcup Y \sqcup (X \dot{-} Y)^{\mathcal{C}} = N$ holds. Therefore, $X^{\mathcal{C}} \leq Y \sqcup (X \dot{-} Y)^{\mathcal{C}}$ and $X^{\mathcal{C}} \dot{-} Y \leq (X \dot{-} Y)^{\mathcal{C}}$.

$X^{\mathcal{C}} \dot{-} Y \leq (X \sqcup Y)^{\mathcal{C}}$ since $X \sqcup Y \sqcup (X \sqcup Y)^{\mathcal{C}} = N$ holds. Moreover, $N = X \sqcup Y \sqcup X^{\mathcal{C}} = X \sqcup Y \sqcup (X^{\mathcal{C}} \dot{-} Y)$, i.e., $(X \sqcup Y)^{\mathcal{C}} \leq X^{\mathcal{C}} \dot{-} Y$ as well. This shows the fourth law. □

The next lemma shows that FDs and MVDs in $\Sigma^+ - \Sigma$ do not have any impact on the dependency basis.

Lemma 4.36. $\Sigma^+_{alg} = (\Sigma^1)^+_{alg}$

Proof. Let $N \in \mathcal{NA}$, $X \in Sub(N)$ and Σ a set of FDs and MVDs on N. We show that the functional or multi-valued dependency in $\Sigma^1 - \Sigma$ does not affect the output $(X^+_{\text{alg}}, DepB_{\text{alg}}(X))$ of Algorithm 4.4.1. Therefore, we consider every inference rule from Theorem 4.28 in turn.

Suppose we have $\dfrac{}{U \to V}\ V \leq U$, i.e., $\{U \to V\} = \Sigma^1 - \Sigma$. Note that $U \leq \overline{U} \sqcup X^+_{\text{alg}}$ since every subattribute of U that is possessed by some $W_i \in X^M_{\text{alg}}$ is a subattribute of X^+_{alg} or a subattribute of $\overline{U}$ and every subattribute of U that is not possessed by any $W_i \in X^M_{\text{alg}}$ is a subattribute of X^+_{alg}. From $V \leq U \leq \overline{U} \sqcup X^+_{\text{alg}}$ follows $\widetilde{V} = V \dot{-} \overline{U} \leq X^+_{\text{alg}}$. Since every $W \in MaxB((X^+_{\text{alg}})^{\mathcal{CC}})$ satisfies $\{W\} \in X^M_{\text{alg}}$ we have $(W_i \dot{-} \widetilde{V})^{\mathcal{CC}} = \lambda$ or $(W_i \dot{-} \widetilde{V})^{\mathcal{CC}} = W_i$ for all $W_i \in X^M_{\text{alg}}$. That proves the lemma for this case.

Suppose $\dfrac{U \to V}{U \to U \sqcup V}$ with $\{U \to U \sqcup V\} = \Sigma^1 - \Sigma$. Since $U \to V \in \Sigma$ we know that $V \dot{-} \overline{U} \leq X^+_{\text{alg}}$. Furthermore, $U \leq \overline{U} \sqcup X^+_{\text{alg}}$ implies $U \dot{-} \overline{U} \leq X^+_{\text{alg}}$. Then we compute

$$\widetilde{U \sqcup V} = (U \sqcup V) \dot{-} \overline{U} = (U \dot{-} \overline{U}) \sqcup (V \dot{-} \overline{U}) \leq X^+_{\text{alg}},$$

where the second equation follows from Theorem 2.3(10). As before, it follows that $(W_i \dot{-} \widetilde{U \sqcup V})^{\mathcal{CC}} = \lambda$ or $(W_i \dot{-} \widetilde{U \sqcup V})^{\mathcal{CC}} = W_i$ for all $W_i \in X^M_{\text{alg}}$.

Suppose $\dfrac{U \to V, V \to W}{U \to W}$ with $\{U \to W\} = \Sigma^1 - \Sigma$. We know that $V \dot{-} \overline{U}, W \dot{-} \overline{V} \leq X^+_{\text{alg}}$, i.e., $V \leq \overline{U} \sqcup X^+_{\text{alg}}$ and $W \leq \overline{V} \sqcup X^+_{\text{alg}}$. We need to show that $W \dot{-} \overline{U} \leq X^+_{\text{alg}}$, or $W \leq \overline{U} \sqcup X^+_{\text{alg}}$ equivalently. We show $\overline{V} \leq \overline{U}$ first. Let $W_i \in X^M_{\text{alg}}$ be a subattribute of $\overline{V}$. Then there is some $Z \in SubB(V)$ with $Z \not\leq X^+_{\text{alg}}$ and Z is possessed by W_i. Since $V \leq \overline{U} \sqcup X^+_{\text{alg}}$ it follows that $Z \in SubB(\overline{U})$ and as Z is possessed by W_i it must be the case that $W_i \leq \overline{U}$ holds as well. This shows $\overline{V} \leq \overline{U}$. It is now obvious that $W \leq \overline{V} \sqcup X^+_{\text{alg}} \leq \overline{U} \sqcup X^+_{\text{alg}}$ holds. As before, it follows that $(W_i \dot{-} \widetilde{W})^{\mathcal{CC}} = \lambda$ or $(W_i \dot{-} \widetilde{W})^{\mathcal{CC}} = W_i$ for all $W_i \in X^M_{\text{alg}}$.

Suppose $\dfrac{U \to V}{U \twoheadrightarrow V}$ with $\{U \twoheadrightarrow V\} = \Sigma^1 - \Sigma$. First we know that $V \dot{-} \overline{U} \leq X^+_{\text{alg}}$ which already implies

$$(V \dot{-} \overline{U}) \sqcap (V \dot{-} \overline{U})^{\mathcal{C}} \leq V \dot{-} \overline{U} \leq X^+_{\text{alg}}.$$

Moreover, we also know that $(W_i \dot{-} (V \dot{-} \overline{U}))^{\mathcal{CC}} = \lambda$ or $(W_i \dot{-} (V \dot{-} \overline{U}))^{\mathcal{CC}} = W_i$ holds for all $W_i \in X^M_{\text{alg}}$. This is equivalent to saying that $((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = W_i$ or $((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = \lambda$ holds for all $W_i \in X^M_{\text{alg}}$.

Suppose $\dfrac{U \twoheadrightarrow V}{U \twoheadrightarrow V^{\mathcal{C}}}$ with $\{U \twoheadrightarrow V^{\mathcal{C}}\} = \Sigma^1 - \Sigma$. First we show that $(V^{\mathcal{C}} \dot{-} \overline{U}) \sqcap (V^{\mathcal{C}} \dot{-} \overline{U})^{\mathcal{C}} \leq X^+_{\text{alg}}$ follows from $(V \dot{-} \overline{U}) \sqcap (V \dot{-} \overline{U})^{\mathcal{C}} \leq X^+_{\text{alg}}$. Note that $\overline{U}$ is the join of $W_i \in X^M_{\text{alg}}$, i.e., $\overline{U} \sqcap \overline{U}^{\mathcal{C}} \leq X^+_{\text{alg}}$ holds as well. In the following we use the facts that $V^{\mathcal{C}} \dot{-} \overline{U} \leq (V \dot{-} \overline{U})^{\mathcal{C}}$ and $V^{\mathcal{C}} \dot{-} \overline{U} = (\overline{U} \sqcup V)^{\mathcal{C}}$ hold. It follows

$$\begin{aligned}
(V^{\mathcal{C}} \dot{-} \overline{U}) \sqcap (V^{\mathcal{C}} \dot{-} \overline{U})^{\mathcal{C}} &= (V^{\mathcal{C}} \dot{-} \overline{U}) \sqcap (\overline{U} \sqcup V)^{\mathcal{CC}} \\
&\leq (V^{\mathcal{C}} \dot{-} \overline{U}) \sqcap (\overline{U} \sqcup V) \\
&= (V^{\mathcal{C}} \dot{-} \overline{U}) \sqcap (\overline{U} \sqcup (V \dot{-} \overline{U})) \\
&= ((V^{\mathcal{C}} \dot{-} \overline{U}) \sqcap \overline{U}) \sqcup ((V^{\mathcal{C}} \dot{-} \overline{U}) \sqcap (V \dot{-} \overline{U})) \\
&\leq (\overline{U}^{\mathcal{C}} \sqcap \overline{U}) \sqcup ((V \dot{-} \overline{U})^{\mathcal{C}} \sqcap (V \dot{-} \overline{U})) \\
&\leq X^+_{\text{alg}}.
\end{aligned}$$

We know that

$$((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = W_i \quad \text{or} \quad ((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = \lambda \tag{7}$$

holds for all $W_i \in X^M_{\text{alg}}$. Suppose there is some $W_i \in X^M_{\text{alg}}$ with $\lambda \neq (W_i \sqcap (V^{\mathcal{C}} \dot{-} \overline{U}))^{\mathcal{CC}} \neq W_i$. Then there is some $X \in MaxB(W_i)$ with $X \in MaxB(V^{\mathcal{C}} \dot{-} \overline{U})$. This implies $X \in MaxB(V^{\mathcal{C}})$ and $X \notin MaxB(\overline{U})$, and $X \in MaxB(V^{\mathcal{C}})$ implies $X \notin MaxB(V^{\mathcal{CC}})$. Therefore, $X \notin MaxB((V \dot{-} \overline{U})^{\mathcal{CC}})$. Furthermore, there is some $Y \in MaxB(W_i)$ with $Y \notin MaxB(V^{\mathcal{C}} \dot{-} \overline{U})$. This implies $Y \notin MaxB(V^{\mathcal{C}})$ or ($Y \in MaxB(V^{\mathcal{C}})$ and $Y \in MaxB(\overline{U})$). As $X \notin MaxB(\overline{U})$ and $X, Y \in MaxB(W_i)$ we must also have $Y \notin MaxB(\overline{U})$. This leaves us with $Y \notin MaxB(V^{\mathcal{C}})$. We conclude $Y \in MaxB(V^{\mathcal{CC}})$ and $Y \in MaxB((V \dot{-} \overline{U})^{\mathcal{CC}})$. The fact that $X, Y \in MaxB(W_i)$ with $X \notin MaxB((V \dot{-} \overline{U})^{\mathcal{CC}})$ and $Y \in MaxB((V \dot{-} \overline{U})^{\mathcal{CC}})$ is obviously a contradiction to (7). Hence, $((V^{\mathcal{C}} \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = W_i$ or $((V^{\mathcal{C}} \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = \lambda$ holds for all $W_i \in X^M_{\text{alg}}$.

Suppose $\dfrac{U \twoheadrightarrow V, V \twoheadrightarrow W}{U \twoheadrightarrow W \dot{-} V}$ with $\{U \twoheadrightarrow W \dot{-} V\} = \Sigma^1 - \Sigma$. We show first that $((W \dot{-} V) \dot{-} \overline{U}) \sqcap ((W \dot{-} V) \dot{-} \overline{U})^{\mathcal{C}} \leq X^+_{\text{alg}}$. We know that $(V \dot{-} \overline{U}) \sqcap (V \dot{-} \overline{U})^{\mathcal{C}} \leq X^+_{\text{alg}}$ and $(W \dot{-} \overline{V}) \sqcap (W \dot{-} \overline{V})^{\mathcal{C}} \leq X^+_{\text{alg}}$. In what follows we prove that

$$(W \dot{-} V) \dot{-} \overline{U} \leq (W \dot{-} \overline{V}) \sqcap \overline{U}^{\mathcal{C}} \sqcap (V \dot{-} \overline{U})^{\mathcal{C}} \tag{8}$$

holds. For this purpose it is proven that $\overline{V} \leq V \sqcup \overline{U}$ holds. Let $Z \in MaxB(\overline{V} \dot{-} \overline{U})$, i.e., $Z \in MaxB(\overline{V})$ and $Z \notin SubB(\overline{U})$ (note that Z is a maximal basis attribute of N).

According to the definition of $\overline{V}$ it follows that there is some $Y \leq V$ which is possessed by some $W_i \in X^M_{\text{alg}}$, $Z \leq W_i$, and $Y \not\leq X^+_{\text{alg}}$. Since $Z \notin SubB(\overline{U})$ and $Z \leq W_i$ we must have $W_i \not\leq \overline{U}$, and $Y \notin SubB(\overline{U})$. Since $Y \leq V$ we derive $Y \in SubB(V \dot{-} \overline{U})$ and as $Y \not\leq X^+_{\text{alg}}$, but $(V \dot{-} \overline{U}) \sqcap (V \dot{-} \overline{U})^{\mathcal{C}} \leq X^+_{\text{alg}}$, it follows that $Y \in MaxB((V \dot{-} \overline{U})^{\mathcal{CC}})$. Consequently, $((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} \neq \lambda$ which leaves us with $((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = W_i$. From $Z \leq W_i$ follows now $Z \leq ((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} \leq (V \dot{-} \overline{U}) \sqcap W_i \leq V \dot{-} \overline{U} \leq V$. Hence, $\overline{V} \dot{-} \overline{U} \leq V$, or equivalently, $\overline{V} \leq V \sqcup \overline{U}$.

We now compute $(W \dot{-} V) \dot{-} \overline{U} = W \dot{-} (V \sqcup \overline{U}) \leq W \dot{-} \overline{V}$ where the first equality follows from Lemma 4.35 and the second relationship follows from $\overline{V} \leq V \sqcup \overline{U}$ and Theorem 2.3(3). Due to the same laws, $V \dot{-} \overline{U} \leq V \sqcup \overline{U}$ implies $W \dot{-} (V \sqcup \overline{U}) \leq W \dot{-} (V \dot{-} \overline{U})$ and therefore $(W \dot{-} V) \dot{-} \overline{U} = W \dot{-} (V \sqcup \overline{U}) \leq W \dot{-} (V \dot{-} \overline{U}) \leq (V \dot{-} \overline{U})^{\mathcal{C}}$. Finally, $(W \dot{-} V) \dot{-} \overline{U} \leq \overline{U}^{\mathcal{C}}$, and (8) follows.

Next, we compute $((W \dot{-} V) \dot{-} \overline{U})^{\mathcal{C}} \leq \overline{U} \sqcup (W \dot{-} V)^{\mathcal{C}} \leq \overline{U} \sqcup V \sqcup W^{\mathcal{C}} = \overline{U} \sqcup (V \dot{-} \overline{U}) \sqcup W^{\mathcal{C}} \leq \overline{U} \sqcup (V \dot{-} \overline{U}) \sqcup (W \dot{-} \overline{V})^{\mathcal{C}}$. This shows

$$((W \dot{-} V) \dot{-} \overline{U})^{\mathcal{C}} \leq \overline{U} \sqcup (V \dot{-} \overline{U}) \sqcup (W \dot{-} \overline{V})^{\mathcal{C}}. \tag{9}$$

According to (8) and (9) we obtain

$$\begin{aligned} ((W \dot{-} V) \dot{-} \overline{U}) \sqcap ((W \dot{-} V) \dot{-} \overline{U})^{\mathcal{C}} &\leq (W \dot{-} \overline{V}) \sqcap \overline{U}^{\mathcal{C}} \sqcap (V \dot{-} \overline{U})^{\mathcal{C}} \sqcap \left(\overline{U} \sqcup (V \dot{-} \overline{U}) \sqcup (W \dot{-} \overline{V})^{\mathcal{C}}\right) \\ &\leq (\overline{U} \sqcap \overline{U}^{\mathcal{C}}) \sqcup ((V \dot{-} \overline{U}) \sqcap (V \dot{-} \overline{U})^{\mathcal{C}}) \sqcup ((W \dot{-} \overline{V}) \sqcap (W \dot{-} \overline{V})^{\mathcal{C}}) \\ &\leq X^+_{\text{alg}}. \end{aligned}$$

It remains to show that $(\widetilde{W \dot{-} V} \sqcap W_i)^{\mathcal{CC}} = W_i$ or $(\widetilde{W \dot{-} V} \sqcap W_i)^{\mathcal{CC}} = \lambda$ holds for all $W_i \in X^M_{\text{alg}}$. We know that

$$\begin{aligned} &\left(((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = W_i \text{ or } ((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = \lambda\right) \text{ and} \\ &\left(((W \dot{-} \overline{V}) \sqcap W_i)^{\mathcal{CC}} = W_i \text{ or } ((W \dot{-} \overline{V}) \sqcap W_i)^{\mathcal{CC}} = \lambda\right) \end{aligned} \tag{10}$$

hold for all $W_i \in X^M_{\text{alg}}$. Assume there is some $W_i \in X^M_{\text{alg}}$ with $(\widetilde{W \dot{-} V} \sqcap W_i)^{\mathcal{CC}} \neq W_i$ and $(\widetilde{W \dot{-} V} \sqcap W_i)^{\mathcal{CC}} \neq \lambda$. From $(\widetilde{W \dot{-} V} \sqcap W_i)^{\mathcal{CC}} \neq \lambda$ follows the existence of some $Y \in MaxB(W_i)$ with $Y \in MaxB((\widetilde{W \dot{-} V})^{\mathcal{CC}})$. As $\widetilde{W \dot{-} V} = W \dot{-} (\overline{U} \sqcup V)$ holds we infer that $Y \in MaxB(W^{\mathcal{CC}})$ and $Y \notin MaxB((\overline{U} \sqcup V)^{\mathcal{CC}})$. The fact that $\overline{V} \leq V \sqcup \overline{U}$ holds tells us that $Y \notin MaxB(\overline{V})$, i.e., $Y \in MaxB((W \dot{-} \overline{V})^{\mathcal{CC}})$ and therefore $((W \dot{-} \overline{V}) \sqcap W_i)^{\mathcal{CC}} \neq \lambda$. On the other hand $V \dot{-} \overline{U} \leq V \leq \overline{U} \sqcup V$, i.e., $Y \notin MaxB((V \dot{-} \overline{U})^{\mathcal{CC}})$ neither. This shows $((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} \neq W_i$.

It follows from (10) that $((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = \lambda$ and $((W \dot{-} \overline{V}) \sqcap W_i)^{\mathcal{CC}} = W_i$ must both hold. Our assumption that there is some $W_i \in X^M_{\text{alg}}$ with $(\widetilde{W \dot{-} V} \sqcap W_i)^{\mathcal{CC}} \neq W_i$ and $(\widetilde{W \dot{-} V} \sqcap W_i)^{\mathcal{CC}} \neq \lambda$ implies the existence of some $X \in MaxB(W_i)$ with $X \in MaxB((\widetilde{W \dot{-} V})^{\mathcal{CC}})$ and $X \notin MaxB((V \dot{-} \overline{U})^{\mathcal{CC}})$. Furthermore, there must be some $Y \in MaxB(W_i)$ with $Y \notin MaxB((\widetilde{W \dot{-} V})^{\mathcal{CC}})$ and $Y \in MaxB((W \dot{-} \overline{V})^{\mathcal{CC}})$. From $X \in MaxB((\widetilde{W \dot{-} V})^{\mathcal{CC}})$ follows

$X \notin MaxB(\overline{U})$ as $\widetilde{W \dot{-} V} = (W \dot{-} V) \dot{-} \overline{U}$, and consequently, $Y \notin MaxB(\overline{U})$ neither. Recall that

$$\widetilde{W \dot{-} V} = W \dot{-} (\overline{U} \sqcup V) = ((\overline{V} \sqcap W) \sqcup (W \dot{-} \overline{V})) \dot{-} (\overline{U} \sqcup (V \dot{-} \overline{U})).$$

From $Y \in MaxB((W \dot{-} \overline{V})^{\mathcal{CC}})$ and $Y \notin MaxB((\widetilde{W \dot{-} V})^{\mathcal{CC}})$ as well as $Y \notin MaxB(\overline{U})$ follows $Y \in MaxB((V \dot{-} \overline{U})^{\mathcal{CC}})$. This contradicts $((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = \lambda$, and our assumption must have been wrong. Therefore, $(\widetilde{W \dot{-} V} \sqcap W_i)^{\mathcal{CC}} = W_i$ or $(\widetilde{W \dot{-} V} \sqcap W_i)^{\mathcal{CC}} = \lambda$ holds indeed for all $W_i \in X^M_{\text{alg}}$.

Suppose $\dfrac{U \twoheadrightarrow V, V \rightarrow W}{U \rightarrow W \dot{-} V}$ with $\{U \rightarrow W \dot{-} V\} = \Sigma^1 - \Sigma$. We need to show that $(W \dot{-} V) \dot{-} \overline{U} \leq X^+_{\text{alg}}$, or $W \leq \overline{U} \sqcup V \sqcup X^+_{\text{alg}}$. We know that $W \dot{-} \overline{V} \leq X^+_{\text{alg}}$, $(V \dot{-} \overline{U}) \sqcap (V \dot{-} \overline{U})^{\mathcal{C}} \leq X^+_{\text{alg}}$ and $((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = W_i$ or $((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = \lambda$ holds for all $W_i \in X^M_{\text{alg}}$. As in the previous case one shows that $\overline{V} \leq V \sqcup \overline{U}$ holds. The proof for this case follows: $W \leq \overline{V} \sqcup X^+_{\text{alg}} \leq \overline{U} \sqcup V \sqcup X^+_{\text{alg}}$, which means $(W \dot{-} V) \dot{-} \overline{U} \leq X^+_{\text{alg}}$.

Suppose $\dfrac{U \twoheadrightarrow V, U \twoheadrightarrow W}{U \twoheadrightarrow V \sqcup W}$ with $\{U \twoheadrightarrow V \sqcup W\} = \Sigma^1 - \Sigma$. We show first that $((V \sqcup W) \dot{-} \overline{U}) \sqcap ((V \sqcup W) \dot{-} \overline{U})^{\mathcal{C}} \leq X^+_{\text{alg}}$. Note that $V \dot{-} \overline{U} \leq (V \sqcup W) \dot{-} \overline{U}$ and $W \dot{-} \overline{U} \leq (V \sqcup W) \dot{-} \overline{U}$ imply that $((V \sqcup W) \dot{-} \overline{U})^{\mathcal{C}} \leq (V \dot{-} \overline{U})^{\mathcal{C}} \sqcap (W \dot{-} \overline{U})^{\mathcal{C}}$ holds (according to Theorem 2.3(3)). We then have $((V \sqcup W) \dot{-} \overline{U}) \sqcap ((V \sqcup W) \dot{-} \overline{U})^{\mathcal{C}} \leq ((V \dot{-} \overline{U}) \sqcup (W \dot{-} \overline{U})) \sqcap (V \dot{-} \overline{U})^{\mathcal{C}} \sqcap (W \dot{-} \overline{U})^{\mathcal{C}} \leq ((V \dot{-} \overline{U}) \sqcap (V \dot{-} \overline{U})^{\mathcal{C}}) \sqcup ((W \dot{-} \overline{U}) \sqcap (W \dot{-} \overline{U})^{\mathcal{C}}) \leq X^+_{\text{alg}}$ by hypothesis. It remains to show that $(\widetilde{V \sqcup W} \sqcap W_i)^{\mathcal{CC}} = W_i$ or $(\widetilde{V \sqcup W} \sqcap W_i)^{\mathcal{CC}} = \lambda$ holds for all $W_i \in X^M_{\text{alg}}$ where $\widetilde{V \sqcup W} = (V \sqcup W) \dot{-} \overline{U} = (V \dot{-} \overline{U}) \sqcup (W \dot{-} \overline{U})$. We know that

$$\begin{aligned} &\big(((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = W_i \text{ or } ((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = \lambda\big) \text{ and} \\ &\big((W \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = W_i \text{ or } ((W \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = \lambda\big) \end{aligned} \tag{11}$$

hold for all $W_i \in X^M_{\text{alg}}$. Assume there is some $W_i \in X^M_{\text{alg}}$ with $(\widetilde{V \sqcup W} \sqcap W_i)^{\mathcal{CC}} \neq \lambda$ and $(\widetilde{V \sqcup W} \sqcap W_i)^{\mathcal{CC}} \neq W_i$. Then there is some $X \in MaxB(W_i)$ with $X \in MaxB((\widetilde{V \sqcup W})^{\mathcal{CC}})$, and there is some $Y \in MaxB(W_i)$ with $Y \notin MaxB((\widetilde{V \sqcup W})^{\mathcal{CC}})$. The latter implies $Y \notin MaxB((V \dot{-} \overline{U})^{\mathcal{CC}})$ and $Y \notin MaxB((W \dot{-} \overline{U})^{\mathcal{CC}})$, and $((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} \neq W_i$ and $((W \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} \neq W_i$ follow, respectively. On the other hand $X \in MaxB((\widetilde{V \sqcup W})^{\mathcal{CC}})$ implies $X \in MaxB((V \dot{-} \overline{U})^{\mathcal{CC}})$ which contradicts $((V \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = \lambda$, or $X \in MaxB((W \dot{-} \overline{U})^{\mathcal{CC}})$ which contradicts $((W \dot{-} \overline{U}) \sqcap W_i)^{\mathcal{CC}} = \lambda$. In any case, this contradicts (11) and our assumption must have been wrong.

Finally, suppose $\dfrac{U \twoheadrightarrow V}{U \rightarrow V \sqcap V^{\mathcal{C}}}$ with $\{U \rightarrow V \sqcap V^{\mathcal{C}}\} = \Sigma^1 - \Sigma$. We need to show that $(V \sqcap V^{\mathcal{C}}) \dot{-} \overline{U} \leq X^+_{\text{alg}}$ knowing that $(V \dot{-} \overline{U}) \sqcap (V \dot{-} \overline{U})^{\mathcal{C}} \leq X^+_{\text{alg}}$ holds. From $V^{\mathcal{C}} \dot{-} \overline{U} \leq (V \dot{-} \overline{U})^{\mathcal{C}}$ follows

$$(V \sqcap V^{\mathcal{C}}) \dot{-} \overline{U} \leq (V \dot{-} \overline{U}) \sqcap (V^{\mathcal{C}} \dot{-} \overline{U}) \leq (V \dot{-} \overline{U}) \sqcap (V \dot{-} \overline{U})^{\mathcal{C}} \leq X^+_{\text{alg}}$$

where the first relationship follows since $(V \sqcap V^{\mathcal{C}}) \dot{-} \overline{U}$ is not only a subattribute of $V \dot{-} \overline{U}$, but also of $V^{\mathcal{C}} \dot{-} \overline{U}$. This completes the proof. □

We obtain the following result by putting Lemma 4.33, 4.34, 4.36 and the comments after Lemma 4.33 together.

Theorem 4.37. *Let $N \in \mathcal{N}A$, $X \in Sub(N)$ and Σ a set of FDs and MVDs on N. Then Algorithm 4.4.1 always terminates and computes nested attribute closure X^+ and dependency basis $DepB(X)$ for X with respect to Σ.*

Proof. It remains to show that Algorithm 4.4.1 always terminates. After the initialisation and after each pass of the REPEAT loop between line (3) and (26) the set $MaxBasis=\{MaxB(Z) \mid Z \in DB_{\text{new}}\}$ is a partition of $MaxB(N)$. Consequently, the number of sets in any such partition is at most $\mid MaxB(N) \mid$, the number of maximal basis attributes of N. However, after each pass through the REPEAT loop (except the last) the partition *MaxBasis* is refined, and the number of sets in it increases, or the number of elements in $NMaxB(X_{\text{new}})$ increases. It follows, that the REPEAT loop is executed at most $\mid SubB(N) \mid = \mid MaxB(N) \mid + \mid NMaxB(N) \mid$ times, and therefore the algorithm terminates. □

EXAMPLE 4.10. We continue the example of the halftoning application. Recall that the underlying nested attribute N is Halftoning(Brightness,Input[Level],Output[Bit]) and the set Σ of FDs and MVDs is specified as in Example 4.3. We may now ask whether the MVD

$$\text{Halftoning(Brightness,Output}[\lambda]) \twoheadrightarrow \text{Halftoning(Output[Bit])}$$

is a logical consequence of Σ. We apply Algorithm 4.4.1 to the input (N, X, Σ) where X = Halftoning(Brightness,Output[λ]). The algorithm has output

$$\begin{aligned} X^+_{\text{alg}} &= \text{Halftoning(Brightness,Input}[\lambda]\text{,Output}[\lambda]), \text{ and} \\ DepB_{\text{alg}}(X) &= \{\ \text{Halftoning(Brightness), Halftoning(Input}[\lambda]), \text{Halftoning(Output}[\lambda]), \\ &\qquad \text{Halftoning(Input[Level]), Halftoning(Output[Bit])}\}. \end{aligned}$$

As Halftoning(Output[Bit]) is an element of $DepB_{\text{alg}}(X)$ and according to the correctness of Algorithm 4.4.1, the MVD above must indeed be a logical consequence of Σ. □

4.4.3 Complexity

We will now show that implication of FDs and MVDs can be decided in polynomial time. Let $N \in \mathcal{N}$A, $X \in Sub(N)$ and Σ a set of FDs and MVDs on N be the input for Algorithm 4.4.1. We use n to denote the size of N, that is $n = \mid SubB(N) \mid$ is the number of basis attributes of N. Furthermore, let s denote the number of dependencies given by Σ, i.e., $s = \mid \Sigma \mid$. $SubB(X)$ and $MaxB(X)$ can be computed in time $\mathcal{O}(n)$. Moreover, union and intersection of sets can be computed in time $\mathcal{O}(n)$ as well. This follows from the fact that $SubB(X \sqcup Y) = SubB(X) \cup SubB(Y)$ and $SubB(X \sqcap Y) = SubB(X) \cap SubB(Y)$, i.e., join and meet operation are linear in n, as well. The pseudo-difference, and therefore the Brouwerian complement operation as well, can be easily implemented in quadratic time:

```
(1)   SubB(X÷Y) := SubB(X);
(2)   FOR each A ∈ SubB(X) DO
(3)     IF A ∈ SubB(Y) THEN SubB(X÷Y) := SubB(X÷Y) − {A};
(4)   ENDDO;
(5)   FOR each A ∈ SubB(X÷Y) DO
(6)     SubB(X÷Y) := SubB(X÷Y) ∪ SubB(A);
(7)   ENDDO;
```

i.e. in time $\mathcal{O}(n^2)$. This implementation reflects exactly the definition of the pseudo-difference operation $\div_{\mathcal{C}_N}$ in the Brouwerian algebra $(\mathcal{C}_N, \subseteq, \cup, \cap, \div_{\mathcal{C}_N}, J_N)$. First, the set difference $SubB(X) - SubB(Y)$ is applied as usual, followed by closing the result downwards with respect to $\leq$.

Next we will write down an algorithm which computes $\overline{U} := \bigsqcup\{W \in DB_{\text{new}} \mid \exists U'.U' \text{ is possessed by } W, U' \not\leq X_{\text{new}}, U' \leq U\}$. Recall that, according to Lemma 4.10, the subattribute U' is possessed by some $W \in DB_{\text{new}}$ if and only if $U' \in SubB(W)$ and $U' \notin SubB(W^{\mathcal{C}})$.

```
(1)   Ū := λ;
(2)   WHILE (SubB(U) ≠ ∅) AND (DB_new ≠ ∅) DO
(3)     SELECT U' ∈ SubB(U);
(4)     SubB(U) := SubB(U) − {U'};
(5)     IF U' ∉ SubB(X_new) THEN
(6)       FOR each W ∈ DB_new DO
(7)         IF (U' ∈ SubB(W)) AND (U' ∉ SubB(W^C)) THEN
(8)           Ū := Ū ⊔ W;
(9)           DB_new := DB_new − {W};
(10)        ENDIF;
(11)      ENDDO;
(12)    ENDIF;
(13)  ENDDO;
```

This demonstrates that $\overline{U}$ can be computed in time $\mathcal{O}(n^3)$. Let us now look at the time complexity to refine X_{new} and DB_{new}, respectively. First consider the case where this refinement has been triggered by a functional dependency $U \to V \in \Sigma$, i.e., line (7) to (14). If $\widetilde{V} \neq \lambda$ in line (10), then $X_{\text{new}} := X_{\text{new}} \sqcup \widetilde{V}$ in line (11) which takes time in $\mathcal{O}(n)$. In order to compute $DB_{\text{new}} := \{(W \div \widetilde{V})^{\mathcal{CC}} \mid W \in DB_{\text{new}}, (W \div \widetilde{V})^{\mathcal{CC}} \neq \lambda\} \cup MaxB(\widetilde{V}^{\mathcal{CC}})$ in line (12) we need to compute $(W \div \widetilde{V})^{\mathcal{CC}}$ in time $\mathcal{O}(n^2)$ for every $W \in DB_{\text{new}}$. Since DB_{new} has at most $\mid MaxB(N) \mid$ elements this takes time in $\mathcal{O}(n^3)$. Computing $MaxB(\widetilde{V}^{\mathcal{CC}})$ and forming the union is in $\mathcal{O}(n^2)$. Consider now the case where the refinement is triggered by some multi-valued dependency $U \twoheadrightarrow V \in \Sigma$, i.e., line (15) to (25). If $\widetilde{V} \neq \lambda$ in line (18), then $X_{\text{new}} := X_{\text{new}} \sqcup (\widetilde{V} \sqcap \widetilde{V}^{\mathcal{C}})$ in line (19) which takes time in $\mathcal{O}(n^2)$. As the computation of $(\widetilde{V} \sqcap W)^{\mathcal{CC}}$ and $(W \div \widetilde{V})^{\mathcal{CC}}$ takes time in $\mathcal{O}(n^2)$ the inner FOR loop between line (20) and (23) for the refinement of DB_{new} takes $\mathcal{O}(n^3)$ steps.

It follows that each pass through the REPEAT loop between line (3) and (26) takes time in $\mathcal{O}(n^3 \cdot s)$. As we have seen before, the REPEAT loop is executed at most n times. Therefore, the time complexity of Algorithm 4.4.1 is $\mathcal{O}(n^4 \cdot s)$.

Theorem 4.38. *Let $N \in \mathcal{N}A$, Σ a set of FDs and MVDs on N and σ an FD or MVD on N. The implication problem whether $\Sigma \models \sigma$ holds can be decided in time $\mathcal{O}(n^4 \cdot s)$.*

Proof. Let σ be the FD $X \to Y$. Algorithm 4.4.1 computes the attribute set closure X^+ in time $\mathcal{O}(n^4 \cdot s)$. It follows that $\Sigma \models X \to Y$ if and only if $Y \leq X^+$ according to Proposition 4.8. To decide whether $Y \leq X^+$ holds takes time in $\mathcal{O}(n)$.

Let σ be the MVD $X \twoheadrightarrow Y$. Algorithm 4.4.1 computes the dependency basis $DepB(X) = SubB(X^+) \cup X^M$ in time $\mathcal{O}(n^4 \cdot s)$. It follows that $\Sigma \models X \twoheadrightarrow Y$ if and only if $Y = \sqcup Z$ for some $Z \subseteq DepB(X)$ according to Proposition 4.8. To decide whether Y is the join of some elements in $DepB(X)$ takes time in $\mathcal{O}(n^2)$. That is, Y' is the join of those $W \in DepB(X)$ with $W \leq Y$. If $Y \leq Y'$ holds as well, then Y is indeed the join of some elements in $DepB(X)$, otherwise it is not. This proves the theorem. □

4.4.4 Applications

The algorithms in Section 3.2.5 can now be generalised to cover the more general class of FDs and MVDs. Algorithm 3.2.4 can be extended to compute non-redundant covers for sets Σ of FDs and MVDs.

Algorithm 4.4.2 (Non-Redundant Covers)

Input: $\boldsymbol{N} \in \boldsymbol{\mathcal{N}}$A, set $\boldsymbol{\Sigma}$ of FDs and MVDs on $\boldsymbol{N}$

Output: a non-redundant cover $\boldsymbol{\Theta}$ of $\boldsymbol{\Sigma}$

Method:

(1) $\Theta := \Sigma$;
(2) **FOR each** $\sigma \in \Sigma$ **DO**
(3) **IF** $\sigma \in (\Theta - \{\sigma\})^+$ **THEN** $\Theta := \Theta - \{\sigma\}$;
(4) **ENDDO**;
(5) **RETURN**(Θ);

□

Theorem 4.39. *Algorithm 4.4.2 computes a non-redundant cover for a set Σ of FDs and MVDs on some nested attribute N in time $\mathcal{O}(n^4 \cdot s^2)$.* □

In the same way, we can generalise Algorithm 3.2.5 to decide whether a given subattribute of N is a superkey for N with respect to a set of FDs and MVDs defined on N. Recall that X is a superkey for N with respect to a set Σ of FDs and MVDs on N if and only if $\Sigma \models X \to N$ holds. This, however, is equivalent to $N \leq X^+_{\text{alg}}$.

Algorithm 4.4.3 (Superkey)

Input: $\boldsymbol{N} \in \boldsymbol{N}$A, set $\boldsymbol{\Sigma}$ of FDs and MVDs on $\boldsymbol{N}$, $\boldsymbol{X} \in \boldsymbol{Sub(N)}$

Output: $\begin{cases} \text{yes} & \text{, if } \boldsymbol{X} \text{ is a superkey for } \boldsymbol{N} \text{ with respect to } \boldsymbol{\Sigma} \\ \text{no} & \text{, else} \end{cases}$

Method:

(1) Compute X^+_{alg} using Algorithm 4.4.1 with input (N, Σ, X);
(2) **IF** $N \leq X^+_{alg}$ **THEN RETURN**(yes)
(3) **ELSE RETURN**(No);

□

Theorem 4.40. *Algorithm 4.4.3 decides in time $\mathcal{O}(n^4 \cdot s)$ whether $X \in Sub(N)$ is a superkey for N with respect to a set Σ of FDs and MVDs defined on N.* □

4.5 The Class of Multi-valued Dependencies

Although the more general class of FDs and MVDs has been the object of study in this chapter it is also interesting to investigate the class of MVDs itself. The proofs of previous sections can be used to obtain minimal axiomatisations and solve the implication problem efficiently for the class of MVDs.

4.5.1 Axiomatisation

A sound and complete set of inference rules for MVDs in the context of relational databases has been provided in [32]. In Section 4.2 we have seen that the mixed meet rule and implication rule, i.e.,

$$\frac{X \twoheadrightarrow Y}{X \to Y \sqcap Y^C} \quad \text{and} \quad \frac{X \to Y}{X \twoheadrightarrow Y},$$

imply the soundness of the auto-complement rule

$$\frac{X \twoheadrightarrow Y}{X \twoheadrightarrow Z} Z \leq Y \sqcap Y^C.$$

The non-triviality of its side condition $Z \leq Y \sqcap Y^C$ gives the following axiomatisation again a distinctive Brouwerian touch.

Theorem 4.41. *The following inference rules*

$$\frac{}{X \twoheadrightarrow Y} Y \leq X \quad \textit{(reflexivity)} \qquad \frac{X \twoheadrightarrow Y}{W \sqcup X \twoheadrightarrow V \sqcup Y} V \leq W \quad \textit{(augmentation)} \qquad \frac{X \twoheadrightarrow Y, Y \twoheadrightarrow Z}{X \twoheadrightarrow (Z \dot{-} Y)} \quad \textit{(pseudo-transitivity)}$$

$$\frac{X \twoheadrightarrow Y}{X \twoheadrightarrow Y^C} \quad \textit{(Brouwerian complement)} \qquad \frac{X \twoheadrightarrow Y}{X \twoheadrightarrow Z} Z \leq Y \sqcap Y^C \quad \textit{(auto-complement)} \qquad \frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow (Y \sqcup Z)} \quad \textit{(multi-valued join)}$$

$$\frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow (Z \dot{-} Y)} \quad \textit{(pseudo-difference)} \qquad \frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow (Y \sqcap Z)} \quad \textit{(multi-valued meet)}$$

are sound and complete for the implication of MVDs in the presence of records and lists.

□

It is easy to see that all rules from Theorem 4.41 apart from the auto-complement rules are natural extensions of rules in the RDM (compare [220, p. 80,81]). Interpreting the auto-complement rule in relational databases means that the trivial MVD $X \twoheadrightarrow \emptyset$ can be derived from the MVD $X \twoheadrightarrow Y$, and is therefore not needed.

The soundness of the inference rules in Theorem 4.41 has already been proven. In order to show the completeness of these rules one can proceed as in the proof of Theorem 4.13. The instance that is used in this proof is generated from two tuples that coincide exactly on the closure X^+ of X with respect to the set Σ of FDs and MVDs given. It remains to clarify what X^+ looks like when Σ does not contain any FDs.

Lemma 4.42. *Let N be a nested attribute, $X \in Sub(N)$ and Σ be a set of MVDs defined on N. It follows that $X^+ = X \sqcup \bigsqcup\{Y \sqcap Y^C : X \twoheadrightarrow Y \in \Sigma^+\}$.*

Proof. The functional closure X^+ of X under all inference rules for FDs and MVDs is defined by $X^+ = \bigsqcup\{Y : X \to Y \in \Sigma^+\}$. It is rather easy to see that $X \sqcup \bigsqcup\{Y \sqcap Y^C : X \twoheadrightarrow Y \in \Sigma^+\} \leq X^+$. First, $X \to X \in \Sigma^+$ by the reflexivity axiom, and then $X \to (Y \sqcap Y^C) \in \Sigma^+$ for every $X \twoheadrightarrow Y \in \Sigma^+$ by the mixed meet rule. It remains to show that $X^+ \leq X \sqcup \bigsqcup\{Y \sqcap Y^C : X \twoheadrightarrow Y \in \Sigma^+\}$ holds as well. Consider the proper chain $\Sigma = \Sigma^0 \subset \Sigma^1 \subset \cdots \subset \Sigma^n = \Sigma^+$ where Σ^i results from Σ^{i-1} by adding exactly one functional or multi-valued dependency which is not in Σ^{i-1} and can be derived by applying one of the inference rules from Theorem 4.13 to dependencies in Σ^{i-1}. We further define $X_i^+ = \bigsqcup\{Y : X \to Y \in \Sigma^i\}$ and show $X_i^+ \leq X \sqcup \bigsqcup\{Y \sqcap Y^C : X \twoheadrightarrow Y \in \Sigma^+\}$ for all i by induction. The case $i = 0$ is trivial as $\Sigma^0 = \Sigma$ does not contain any FDs, i.e., $X_0^+ = \lambda$. We exhibit now every inference rule in turn, considering only inference rules which have an FD in the conclusion.

Reflexivity Rule: Say $X \to Y \in (\Sigma^{i+1} - \Sigma^i)$ with $Y \leq X$. We conclude by hypothesis that $Y \leq X \leq X \sqcup \bigsqcup\{Y \sqcap Y^C : X \twoheadrightarrow Y \in \Sigma^+\}$.

Extension Rule: Suppose now that $X \to Y \in (\Sigma^{i+1} - \Sigma^i)$ where $Y = X \sqcup Z$ and $X \to Z \in \Sigma^i$ holds. The hypothesis tells us that $Z \leq X \sqcup \bigsqcup\{Y \sqcap Y^C : X \twoheadrightarrow Y \in \Sigma^+\}$ holds. This, however, implies also $Y = X \sqcup Z \leq X \sqcup \bigsqcup\{Y \sqcap Y^C : X \twoheadrightarrow Y \in \Sigma^+\}$.

Transitivity Rule: Assume that $X \to Y \in (\Sigma^{i+1} - \Sigma^i)$ where $X \to Z, Z \to Y \in \Sigma^i$ hold. We conclude by hypothesis that

$$Z \leq X \sqcup \bigsqcup\{V \sqcap V^C : X \twoheadrightarrow V \in \Sigma^+\} \quad \text{and} \quad Y \leq Z \sqcup \bigsqcup\{W \sqcap W^C : Z \twoheadrightarrow W \in \Sigma^+\}$$

hold. This implies $Y \leq X \sqcup \bigsqcup\{V \sqcap V^C : X \twoheadrightarrow V \in \Sigma^+\} \sqcup \bigsqcup\{W \sqcap W^C : Z \twoheadrightarrow W \in \Sigma^+\}$. We show that $\bigsqcup\{W \sqcap W^C : Z \twoheadrightarrow W \in \Sigma^+\} \leq Z \sqcup \bigsqcup\{V \sqcap V^C : X \twoheadrightarrow V \in \Sigma^+\}$

holds which implies $Y \leq X \sqcup \bigsqcup\{V \sqcap V^{\mathcal{C}} : X \twoheadrightarrow V \in \Sigma^+\}$. Let $Z \twoheadrightarrow W \in \Sigma^+$. From $X \rightarrow Z \in \Sigma^+$ follows $X \twoheadrightarrow Z \in \Sigma^+$ by the implication rule. From $X \twoheadrightarrow Z, Z \twoheadrightarrow W \in \Sigma^+$ follows $X \twoheadrightarrow (W \dot{-} Z) \in \Sigma^+$ by means of the pseudo-transitivity rule. Recall that $W \leq W \sqcup Z = Z \sqcup (W \dot{-} Z)$ and $W^{\mathcal{C}} \leq (W \dot{-} Z)^{\mathcal{C}} \leq Z \sqcup (W \dot{-} Z)^{\mathcal{C}}$ hold. It is then easy to see that

$$W \sqcap W^{\mathcal{C}} \leq (Z \sqcup (W \dot{-} Z)) \sqcap (Z \sqcup (W \dot{-} Z)^{\mathcal{C}}) = Z \sqcup ((W \dot{-} Z) \sqcap (W \dot{-} Z)^{\mathcal{C}})$$

holds as well which completes this case.

Mixed Pseudo-Transitivity Rule: Suppose $X \rightarrow (Y \dot{-} Z) \in (\Sigma^{i+1} - \Sigma^i)$ where $X \twoheadrightarrow Z, Z \rightarrow Y \in \Sigma^i$ hold. We conclude by hypothesis that

$$Y \leq Z \sqcup \bigsqcup\{W \sqcap W^{\mathcal{C}} : Z \twoheadrightarrow W \in \Sigma^+\}$$

holds. We need to show that $Y \dot{-} Z \leq X \sqcup \bigsqcup\{V \sqcap V^{\mathcal{C}} : X \twoheadrightarrow V \in \Sigma^+\}$ holds which is equivalent to $Y \leq X \sqcup Z \sqcup \bigsqcup\{V \sqcap V^{\mathcal{C}} : X \twoheadrightarrow V \in \Sigma^+\}$. We show that $\bigsqcup\{W \sqcap W^{\mathcal{C}} : Z \twoheadrightarrow W \in \Sigma^+\} \leq Z \sqcup \bigsqcup\{V \sqcap V^{\mathcal{C}} : X \twoheadrightarrow V \in \Sigma^+\}$. Let $Z \twoheadrightarrow W \in \Sigma^+$. From $X \twoheadrightarrow Z \in \Sigma^+$ follows $X \twoheadrightarrow (W \dot{-} Z) \in \Sigma^+$ by the multi-valued pseudo-transitivity rule. As in the previous case follows then $W \sqcap W^{\mathcal{C}} \leq Z \sqcup ((W \dot{-} Z) \sqcap (W \dot{-} Z)^{\mathcal{C}})$. We conclude

$$\begin{aligned} Y &\leq Z \sqcup \bigsqcup\{W \sqcap W^{\mathcal{C}} : Z \twoheadrightarrow W \in \Sigma^+\} \\ &\leq Z \sqcup \bigsqcup\{V \sqcap V^{\mathcal{C}} : X \twoheadrightarrow V \in \Sigma^+\} \\ &\leq X \sqcup Z \sqcup \bigsqcup\{V \sqcap V^{\mathcal{C}} : X \twoheadrightarrow V \in \Sigma^+\} \end{aligned}$$

which we had to prove.

Mixed Meet Rule: It remains to consider the case where $X \rightarrow Y \sqcap Y^{\mathcal{C}} \in (\Sigma^{i+1} - \Sigma^i)$ where $X \twoheadrightarrow Y \in \Sigma^i$ holds. It follows then immediately that $Y \sqcap Y^{\mathcal{C}} \leq X \sqcup \bigsqcup\{V \sqcap V^{\mathcal{C}} : X \twoheadrightarrow V \in \Sigma^+\}$ holds. This concludes the proof. □

4.5.2 Minimality

The proofs in Section 4.2 show that the inference rules from Theorem 4.41 are not minimal. In particular, it follows from Lemmata 4.15, 4.16, 4.17 and 4.18 that pseudo-difference rule, meet rule and augmentation rule are logically implied by reflexivity axiom, Brouwerian complement rule, pseudo-transitivity rule, multi-valued join rule and auto-complement rule. The independence of these rules from one another follows directly from the proofs of Lemmata 4.19, 4.23, 4.24, 4.26 and 4.27.

Theorem 4.43. *Reflexivity axiom, Brouwerian complement rule, pseudo-transitivity rule, multi-valued join rule and auto-complement rule form a minimal, sound and complete set of inference rules for the implication of MVDs in the presence of records and lists.* □

Theorem 4.43 reveals again a difference to relational databases. It has been proven in [204] that reflexivity axiom, complementation rule and pseudo-transitivity rule form a minimal, sound and complete set. The auto-complement rule is of course not needed in

relational databases, but the join rule is logically implied by {reflexivity axiom, complementation rule, pseudo-transitivity rule}. However, the situation is different in the presence of lists.

As in the more general case of FDs and MVDs one can replace the Brouwerian complement rule in Theorem 4.43 by the N-axiom $\frac{}{\lambda \twoheadrightarrow N}$. The proofs of Section 4.3 show that Brouwerian complement rule and N-axiom are equivalent in the presence of reflexivity axiom and pseudo-transitivity rule. Moreover, the independence of reflexivity axiom, N-axiom, pseudo-transitivity rule, multi-valued join rule and auto-complement rule from one another follows from the comments and proofs in Section 4.3 as well.

Theorem 4.44. *Reflexivity axiom, N-axiom, pseudo-transitivity rule, multi-valued join rule and auto-complement rule form a minimal, sound and complete set of inference rules for the implication of MVDs in the presence of records and lists.* □

Remark. Although the set of inference rules in Theorem 4.44 is minimal in the sense of Definition 3.7 the side condition $Y \leq X$ in the reflexivity axiom $\frac{}{X \twoheadrightarrow Y} Y \leq X$ can be weakened to $Y \leq X, Y \in SubB(N)$. Let us call

$$\frac{}{X \twoheadrightarrow Y} Y \leq X, Y \in SubB(N)$$

the membership-axiom. We show that the reflexivity axiom follows from {membership-axiom, N-axiom, pseudo-transitivity rule, join rule}.

If $N = \lambda$, then the only instance of the reflexivity axiom is $\lambda \twoheadrightarrow \lambda$ which is in this case also an instance of the N-axiom. We can therefore assume that $N \neq \lambda$. Suppose $X \neq \lambda$, say $A \in \{Z \leq X \mid Z \in SubB(N)\}$. We proceed by induction on the number n of elements in $\{Z \leq Y \mid Z \in SubB(N)\}$. If $n = 0$, then the inference schema is

$$\frac{\frac{}{X \twoheadrightarrow A}{\scriptstyle A \leq X, A \in SubB(N)} \quad \frac{}{A \twoheadrightarrow A}{\scriptstyle A \leq A, A \in SubB(N)}}{X \twoheadrightarrow \lambda}$$

where the pseudo-transitivity rule is used in the last step. Suppose $Y = \bigsqcup\{A_1, \ldots, A_n, A_{n+1}\}$. Note that $A_i \leq X$ for $i = 1, \ldots, n+1$ as $Y \leq X$. We then have the following inference schema

$$\frac{\frac{}{X \twoheadrightarrow \bigsqcup\{A_1, \ldots, A_n\}}{\scriptstyle \bigsqcup\{A_1, \ldots, A_n\} \leq X} \quad \frac{}{X \twoheadrightarrow A_{n+1}}{\scriptstyle A_{n+1} \leq X, A_{n+1} \in SubB(N)}}{X \twoheadrightarrow Y}$$

where the join rule is applied in the last step. It remains to consider the case where $X = \lambda$. Note that $\frac{}{N \twoheadrightarrow N} N \leq N$ follows from the previous case as $N \neq \lambda$.

$$\frac{\frac{}{\lambda \twoheadrightarrow N} \quad \frac{}{N \twoheadrightarrow N}{\scriptstyle N \leq N}}{\lambda \twoheadrightarrow \lambda}$$

The pseudo-transitivity rule is again applied in the last step. This shows that the reflexivity axiom follows from {membership-axiom, N-axiom, pseudo-transitivity rule, join rule}.

Following a similar line of reasoning one can show that the auto-complement rule can be weakened to

$$\frac{X \twoheadrightarrow Y}{X \twoheadrightarrow Z} Z \leq Y \sqcap Y^{\mathcal{C}}, Z \in SubB(N).$$

The following inference rules

$$\frac{}{X \twoheadrightarrow Y} Y \leq X, Y \in SubB(N) \qquad \frac{}{\lambda \twoheadrightarrow N} \qquad \frac{X \twoheadrightarrow Y, Y \twoheadrightarrow Z}{X \twoheadrightarrow (Z \dot{-} Y)}$$

$$\frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow (Y \sqcup Z)} \qquad \frac{X \twoheadrightarrow Y}{X \twoheadrightarrow Z} Z \leq Y \sqcap Y^{\mathcal{C}}, Z \in SubB(N)$$

are minimal, sound and complete for the implication of MVDs in the presence of records and lists. □

4.5.3 Implication Problem

Finally, Algorithm 4.4.1 can be used to compute the dependency basis of some attribute $X \in Sub(N)$ with respect to a set Σ of MVDs given. However, lines (7) to (14) can be omitted as FDs do not need to be considered. This results in the following algorithm.

Algorithm 4.5.1 (Dependency Basis)

Input: $\boldsymbol{N} \in \boldsymbol{N}\mathrm{A}$, $\boldsymbol{X} \in \boldsymbol{Sub(N)}$, set $\boldsymbol{\Sigma}$ of MVDs on $\boldsymbol{N}$

Output: $\mathrm{DepB}_{\mathrm{alg}}(\boldsymbol{X})$

Method:

VAR $DB_{\mathrm{new}}, DB_{\mathrm{old}} \subseteq Sub(N)$, $X_{\mathrm{new}}, X_{\mathrm{old}}, U, V, W, \overline{U}, \widetilde{V}, U' \in Sub(N)$;

(1) $X_{\mathrm{new}} := X$;
(2) $DB_{\mathrm{new}} := MaxB(X^{\mathcal{CC}}) \cup \{X^{\mathcal{C}}\}$;
(3) **REPEAT**
(4) $X_{\mathrm{old}} := X_{\mathrm{new}}$;
(5) $DB_{\mathrm{old}} := DB_{\mathrm{new}}$;
(6) **FOR each** $U \twoheadrightarrow V \in \Sigma$ **DO**
(7) $\overline{U} := \bigsqcup\{W \in DB_{\mathrm{new}} \mid \exists U'.U'$ possessed by $W, U' \not\leq X_{\mathrm{new}}, U' \leq U\}$;
(8) $\widetilde{V} := V \dot{-} \overline{U}$;
(9) **IF** $\widetilde{V} \neq \lambda$ **THEN BEGIN**
(10) $X_{\mathrm{new}} := X_{\mathrm{new}} \sqcup (\widetilde{V} \sqcap \widetilde{V}^{\mathcal{C}})$;
(11) **FOR each** $W \in DB_{\mathrm{new}}$ **DO**
(12) **IF** $(\widetilde{V} \sqcap W)^{\mathcal{CC}} \neq \lambda$ **AND** $(\widetilde{V} \sqcap W)^{\mathcal{CC}} \neq W$ **THEN**
(13) $DB_{\mathrm{new}} := (DB_{\mathrm{new}} - \{W\}) \cup \{(\widetilde{V} \sqcap W)^{\mathcal{CC}}, (W \dot{-} \widetilde{V})^{\mathcal{CC}}\}$;

(14) **ENDDO**;
(15) **END**;
(16) **ENDDO**;
(17) **UNTIL** $(X_{\text{new}} = X_{\text{old}})$ **AND** $(DB_{\text{new}} = DB_{\text{old}})$;
(18) $X^{+}_{\text{alg}} := X_{\text{new}}$; $X^{M}_{\text{alg}} := DB_{\text{new}}$;
(19) $DepB_{\text{alg}}(X) := SubB(X^{+}_{\text{alg}}) \cup X^{M}_{\text{alg}}$;
(20) **RETURN**$(DepB_{\text{alg}}(X))$;

□

The correctness of Algorithm 4.5.1 follows immediately from Theorem 4.37.

Theorem 4.45. *Let $N \in \mathcal{N}A$, $X \in Sub(N)$ and Σ a set of MVDs on N. Then Algorithm 4.5.1 always terminates and computes the dependency basis $DepB(X)$ for X with respect to Σ.* □

The complexity analysis of Algorithm 4.4.1 in Section 4.4.3 determines also the complexity of Algorithm 4.5.1. It follows that the complexity of the implication problem for the class of MVDs has essentially the same complexity as the class of FDs and MVDs, as expected.

Theorem 4.46. *Let $N \in \mathcal{N}A$, Σ a set of MVDs on N and σ an MVD on N. The implication problem whether $\Sigma \models \sigma$ holds can be decided in time $\mathcal{O}(n^4 \cdot s)$.* □

4.5.4 A different Perspective for MVDs

MVDs have been defined as expressions $X \twoheadrightarrow Y$ where $X, Y \in Sub(N)$. Alternatively, we can view MVDs as expressions $X \twoheadrightarrow Y$ where $X, Y \in \mathcal{C}_N$ and $(\mathcal{C}_N, \subseteq, \cup, \cap, \dot{-}_{\mathcal{C}_N}, J_N)$ is the Brouwerian algebra of closed subsets of the PO-space on the join-irreducible elements J_N of $Sub(N)$. A set $r \subseteq dom(N)$ *satisfies* the MVD $X \twoheadrightarrow Y$ on $\mathcal{C}_N$, denoted by $\models_r X \twoheadrightarrow Y$, if and only if there is a $t \in r$ with $\pi^N_B(t) = \pi^N_B(t_1)$ for all $B \in X \cup Y$ and $\pi^N_C(t) = \pi^N_C(t_2)$ for all $C \in X \cup (J_N \dot{-}_{\mathcal{C}_N} Y)$ whenever $\pi^N_A(t_1) = \pi^N_A(t_2)$ for all $A \in X$ holds for any $t_1, t_2 \in r$. This view can again be justified in the following sense. Lemma 3.9 shows that for all $r \subseteq dom(N)$ we have $\models_r X \twoheadrightarrow Y$ for $X, Y \in \mathcal{C}_N$ if and only if $\models_r \bigsqcup X \twoheadrightarrow \bigsqcup Y$ in terms of Definition 4.1. The minimal axiomatisation from Theorem 4.44 reads then as follows. The following inference rules

$$\frac{}{X \twoheadrightarrow Y}\, Y \subseteq X \qquad \frac{}{\emptyset \twoheadrightarrow J_N} \qquad \frac{X \twoheadrightarrow Y, Y \twoheadrightarrow Z}{X \twoheadrightarrow Z \dot{-}_{\mathcal{C}_N} Y}$$

$$\frac{X \twoheadrightarrow Y, X \twoheadrightarrow Z}{X \twoheadrightarrow Y \cup Z} \qquad \frac{X \twoheadrightarrow Y}{X \twoheadrightarrow Z}\, Z \in \mathcal{C}_N, Z \subseteq Y \cap (J_N \dot{-}_{\mathcal{C}_N} Y)$$

are minimal, sound and complete for the implication of MVDs in the presence of records and lists.

4.6 Related and Future Work

MVDs have been studied very well in relational databases. The next goal is the proposal of a nested list normal form for nested attributes with respect to the class of MVDs and the class of FDs and MVDs, to semantically justify this proposal and generalise the decomposition approach. Research papers that may be used as guidelines are [103, 133, 289, 290]. The proposal of such a normal form can be found in Section 6.2 of this thesis. It is desirable to improve the running time of Algorithm 4.4.1 for deciding the implication of FDs and MVDs. Substantial research on that subject has again been done for relational databases and the papers [98, 118, 135, 152, 173, 223, 239, 277] may give some more information. The paper [216] proposes algorithms how to obtain *reduced MVDs* and *minimal covers* of sets of MVDs for relational databases. The concept of a *pure set of FDs and MVDs* was introduced in [154]. An MVD $X \twoheadrightarrow Y$ of a set Σ of FDs and MVDs on a relation schema R is called pure iff it is non-trivial and neither $X \rightarrow Y$ nor $X \rightarrow (R - Y)$ are in Σ^+. A related definition aimed at factoring out MVDs which cannot be derived from FDs appears in the concept of an *envelope set* due to [301, 302] in a work on desirable 4NF decompositions. So-called *conflict-free MVDs* are introduced in [247]. MVDs of this class have the property that they allow a unique 4NF dependency preserving database schema. Moreover it is stated that non conflict-free sets of dependencies are inadequately specified. It is interesting to study these different notions in the context of complex object types.

Multi-valued dependencies have been the subject of *data mining*. In [242] two algorithms for the discovery of multi-valued dependencies from relations are presented. The top-down algorithm enumerates the hypotheses from the most general to more specific hypotheses which are checked on the input relation. The bottom-up algorithm first computes the invalid multi-valued dependencies. Starting with the most general dependencies, the algorithm iteratively refines the set of dependencies to conform with each particular invalid dependency. The implementation of the algorithms is analysed and some empirical results are presented. A different approach is proposed in [300].

Recent papers that study multi-valued dependencies in the context of XML are [286, 287]. The work in [286] introduces MVDs in XML (XMVDs) and justifies the definition by showing that for a general class of mappings from relations to XML, a relation satisfies an MVD if and only if the corresponding XML document satisfies the corresponding XMVD. As this justification of XMVDs already suggests, XMVDs provide semantics for XML documents that are exported or imported from relational databases. Therefore, XMVDs do not cover multi-valued dependencies among complex objects such as lists. The definition of XMVDs is again based on the notion of a path. The work in [287] proposes an extension of the well-known fourth normal form (4NF) from relational databases to XML in order to syntactically describe semantically well-designed XML documents with respect to XMVDs as studied in [286].

A conceptual treatment of MVDs is introduced in [266]. It is proposed that entity-relationship modelling techniques enable a more natural and intuitive way of handling MVDs. Based on the concept of *competing MVDs* it is proven in which case a unique entity-relationship schema representation exists. If MVDs are competing, then either one

of the competing schemata is chosen or an approximation which combines the competing schemata can be used.

For more comments on future work see Section 6.2. Let us finally look at a further example of MVDs among complex objects. Suppose we store nucleotide sequences together with certain genes that occur in it, i.e. sequences of amino acids, and together with a certain base and the sequence of positions in which that base appears in the original nucleotide sequence. We may use the nested attribute

Genes(Sequence[Nucleotide], Gene[Amino-Acid], Occurs(Base, Position[Number])).

There might be several genes encoded within the nucleotide sequence, and there are different bases together with a certain sequence of positions in which they occur. The set of genes, however, is independent from the set of bases and the corresponding sequence of occurrences. We therefore have the following MVDs

$$\text{Genes(Sequence[Nucleotide])} \twoheadrightarrow \text{Genes(Gene[Amino-Acid])} \text{ and}$$
$$\text{Genes(Sequence[Nucleotide])} \twoheadrightarrow \text{Genes(Occurs(Base, Position[Number]))}.$$

Moreover there are the FDs

$$\text{Genes(Sequence[Nucleotide],Occurs(Base))} \rightarrow \text{Genes(Occurs(Position[Number]))} \text{ and}$$
$$\text{Genes(Sequence[Nucleotide],Occurs(Position[Number]))} \rightarrow \text{Genes(Occurs(Base))}.$$

It appears that the chance of MVDs occurring among complex objects is as good as the chance of MVDs occurring among flat data. The techniques provided in this chapter may therefore help to cover more application domains.

Chapter 5

Functional Dependencies in the Presence of Lists, Sets and Multisets

This chapter is devoted to the study of FDs in the presence of multiple type constructors. In fact, the objective is to investigate all combinations of types illustrated in Figure 1.2.

We have seen in Chapter 3 that the theory of FDs can be generalised from the relational data model to the presence of the list constructor. The semantic behavior of such functional dependencies can be captured by a natural extension of Armstrong's axioms to null, flat, record- and list-valued attributes.

The goal of this chapter is to investigate the impact of set and multiset constructor on the theory of FDs. Both, sets and multisets do not impose an order on their elements. It turns out that the extension rule is no longer sound in general, which results in a more complex notion of an FD. A syntactical condition for pairs of subattributes is introduced that characterises those pairs X and Y for which the values of projections on X and Y uniquely determine the value of the projection on their join $X \sqcup Y$. This leads to a more sophisticated set of sound and complete inference rules. In order to prove the completeness result the standard technique of constructing a certain two-element instance is used whose elements coincide exactly on a set of subattributes which is closed under inference. The construction of such an instance, however, becomes difficult in the presence of sets and multisets. While in the case of sets a few combinatorial arguments are used, the case of multisets requires further studies of the algebraic structure of nested attributes. Having proven the completeness of the set of inference rules it is shown that they are all independent from one another. In this sense none of the rules can be omitted without losing completeness. The first main result of this chapter provides minimal axiomatisations for FDs in the presence of all combinations of record, list, set and multiset type in which at least the record type is present.

The second objective is to study the implication problem for FDs in all the different contexts of record, list, set and multiset type. A provably-correct algorithm for computing the nested attribute closure for a set of subattributes is proposed that works in polynomial time in the number of subattributes and the number of FDs given.

The axiomatisation of FDs can be found in [145], [139] contains the axiomatisation of

FDs in the presence of records and sets, and [140] discusses the implication problem of FDs in the presence of all type combinations above.

5.1 Axiomatisation

In this section axiomatisability of FDs is studied in the presence of null, flat, record-, list-, set- and multiset-valued attributes.

5.1.1 The Failure of the Extension Rule

We start with an example that reveals the difficulty of dealing with sets or multisets.

EXAMPLE 5.1. 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)}:

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

The second element of this set results from the first by simply switching opponents. We can see that $\models_r \text{Tennis}\{\text{Match}(\text{Winner})\} \rightarrow \text{Tennis}\{\text{Match}(\text{Loser})\}$ 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, $\not\models_r \text{Tennis}\{\text{Match}(\text{Winner})\} \rightarrow \text{Tennis}\{\text{Match}(\text{Winner, Loser})\}$ 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. □

Example 5.1 shows, in particular, that the current notion of functional dependency is insufficient in the context of sets and multisets. In general, values on subattributes do not determine the value on the join of these subattributes. This implies that instead of considering single subattributes as left- and right-hand sides of functional dependencies it becomes necessary to consider sets of subattributes. This motivates the following definition.

Definition 5.1. Let $N \in \mathcal{N}\mathrm{A}$ be a nested attribute. A *functional dependency on* N is an expression of the form $\mathcal{X} \rightarrow \mathcal{Y}$ where $\mathcal{X}, \mathcal{Y} \subseteq Sub(N)$ are non-empty. A set $r \subseteq dom(N)$ *satisfies* the functional dependency $\mathcal{X} \rightarrow \mathcal{Y}$ on N, denoted by $\models_r \mathcal{X} \rightarrow \mathcal{Y}$, 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 $t_1, t_2 \in r$. □

Clearly, the new definition of FDs has increased the level of expressiveness. The FD $\mathcal{X} \to \{L\{K(A,B)\}\}$ implies for instance the FD $\mathcal{X} \to \{L\{K(A,\lambda)\}, L\{K(\lambda,B)\}\}$ but not vice versa. In the same way the FD $\{L\{K(A,B)\}\} \to \mathcal{Y}$ is implied by the FD $\{L\{K(A,\lambda)\}, L\{K(\lambda,B)\}\} \to \mathcal{Y}$ but not vice versa.

The condition that $\mathcal{X}, \mathcal{Y} \subseteq Sub(N)$ are both non-empty is simply a matter of convenience and does not reduce the expressiveness. In fact, any two tuples have the same projection on λ. The FD $\mathcal{X} \to \emptyset$ is satisfied by any instance, but so is $\mathcal{X} \to \{\lambda\}$. On the other hand the FD $\emptyset \to \mathcal{Y}$ is satisfied by some instance r if and only if r satisfies $\{\lambda\} \to \mathcal{Y}$.

We are now able to formalise the constraints for the retailer example from Section 1.2.2.

EXAMPLE 5.2. Let N denote the nested attribute of Example 2.5 which was used as a schema for the retailer database. The set Σ of FDs on N, informally described in Section 1.2.2, can be formally specified as follows:

1. Sales(Day) $\to N$,
2. Sales(List[Order(Cart⟨Article(Title)⟩)]) $\to$ Sales(Sold{Product(Item)}),
3. Sales(List[Order(Cart⟨Article(Price)⟩)]) $\to$ Sales(List[Order(SubTotal)]),
4. Sales(List[Order(SubTotal)]) $\to$ Sales(Total),
5. Sales(List[Order(Customer(Name))]) $\to$ Sales(Sold{Product(CustName)}),
6. Sales(List[Order(Cart⟨Article(Title)⟩,Customer(Name))]) $\to$ Sales(Sold{Product(Item,CustName)}),
7. Sales(List[λ]) $\to$ Sales(NOrd), and Sales(NOrd) $\to$ Sales(List[λ]),
8. Sales(List[Order(Cart⟨λ⟩)]) $\to$ Sales(NProd),
9. Sales(List[Order(Cart⟨λ⟩,Customer(Address))]) $\to$ Sales(NShip). □

The notions of implication ($\models$) and derivability ($\vdash_{\mathfrak{R}}$) with respect to a set $\mathfrak{R}$ of inference rules can be defined as in Section 3.1.2 where $\mathcal{C}$ is now the class of FDs in the presence of null, flat, record-, list-, set- and multiset-valued attributes. As before it follows that finite and unrestricted implication coincide for this class of FDs.

5.1.2 Reconcilable Attributes

Example 5.1 shows that Definition 5.1 of a functional dependency $\mathcal{X} \to \mathcal{Y}$ on some nested attribute N cannot be simplified to an expression of the form $X \to Y$ with $X, Y \in Sub(N)$. That is, values of projections on subattributes X and Y do not determine the value of the projection on $X \sqcup Y$ in general. The reason for this is the set constructor, and the same reasoning applies to the multiset constructor. Before we introduce some inference rules for FDs we will give a sufficient condition when projections on subattributes X and Y do determine the projection on $X \sqcup Y$.

Definition 5.2. Let $N \in \mathcal{N}\mathrm{A}$. The subattributes $X, Y \in Sub(N)$ are *reconcilable* if and only if one of the following conditions is satisfied

- $Y \leq X$ or $X \leq Y$,

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

Given $X, Y \in Sub(N)$ that are reconcilable and some $t \in dom(N)$ the projections $\pi^N_X(t)$ and $\pi^N_Y(t)$ determine $\pi^N_{X \sqcup Y}(t)$.

Lemma 5.3. *Let $N \in \mathcal{NA}$, $X, Y \in Sub(N)$ reconcilable and $t_1, t_2 \in dom(N)$. If $\pi^N_X(t_1) = \pi^N_X(t_2)$ and $\pi^N_Y(t_1) = \pi^N_Y(t_2)$, then $\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$.*

Proof. We proceed by induction on the structure of N. If $Y \leq X$, then $X \sqcup Y = X$ and the statement follows from the assumption that $\pi^N_X(t_1) = \pi^N_X(t_2)$. If $X \leq Y$, then $X \sqcup Y = Y$ and the statement follows from the assumption that $\pi^N_Y(t_1) = \pi^N_Y(t_2)$. Let $N = L(N_1, \ldots, N_k), X = L(X_1, \ldots, X_k)$ and $Y = L(Y_1, \ldots, Y_k)$. Consequently, $t_1, t_2 \in dom(N)$ have the form $t_1 = (t_1^1, \ldots, t_k^1)$ and $t_2 = (t_1^2, \ldots, t_k^2)$ with $t_j^i \in dom(N_j)$ for $j = 1, \ldots, k$ and $i = 1, 2$. From $\pi^N_X(t_1) = \pi^N_X(t_2)$ follows $\pi^{N_i}_{X_i}(t_i^1) = \pi^{N_i}_{X_i}(t_i^2)$ for $i = 1, \ldots, k$ by definition of the projection function. Similarly follows $\pi^{N_i}_{Y_i}(t_i^1) = \pi^{N_i}_{Y_i}(t_i^2)$ for $i = 1, \ldots, k$ from $\pi^N_Y(t_1) = \pi^N_Y(t_2)$. The assumption that X and Y are reconcilable implies that X_i and Y_i are reconcilable for all $i = 1, \ldots, k$. Consequently, we conclude $\pi^{N_i}_{X_i \sqcup Y_i}(t_i^1) = \pi^{N_i}_{X_i \sqcup Y_i}(t_i^2)$ for $i = 1, \ldots, k$. This shows that

$$\begin{aligned} \pi^N_{X \sqcup Y}(t_1) &= (\pi^{N_1}_{X_1 \sqcup Y_1}(t_1^1), \ldots, \pi^{N_k}_{X_k \sqcup Y_k}(t_k^1)) \\ &= (\pi^{N_1}_{X_1 \sqcup Y_1}(t_1^2), \ldots, \pi^{N_k}_{X_k \sqcup Y_k}(t_k^2)) \\ &= \pi^N_{X \sqcup Y}(t_2) \end{aligned}$$

which we had to prove. It remains to consider the case where $N = L[N'], X = L[X'], Y = L[Y']$. Consequently, $t_1, t_2 \in dom(N)$ have the form $t_1 = [t_1^1, \ldots, t_k^1]$ and $t_2 = [t_1^2, \ldots, t_l^2]$ with $t_i^1, t_j^2 \in dom(N')$ for $i = 1, \ldots, k$ and $j = 1, \ldots, l$. From $\pi^N_X(t_1) = \pi^N_X(t_2)$ follows $k = l$ and $\pi^{N'}_{X'}(t_i^1) = \pi^{N'}_{X'}(t_i^2)$ for $i = 1, \ldots, k$ by definition of the projection function. Similarly follows $\pi^{N'}_{Y'}(t_i^1) = \pi^{N'}_{Y'}(t_i^2)$ for $i = 1, \ldots, k$ from $\pi^N_Y(t_1) = \pi^N_Y(t_2)$. The assumption that X and Y are reconcilable implies that X' and Y' are reconcilable. Consequently, we conclude $\pi^{N'}_{X' \sqcup Y'}(t_i^1) = \pi^{N'}_{X' \sqcup Y'}(t_i^2)$ for $i = 1, \ldots, k$. This shows that

$$\begin{aligned} \pi^N_{X \sqcup Y}(t_1) &= [\pi^{N'}_{X' \sqcup Y'}(t_1^1), \ldots, \pi^{N'}_{X' \sqcup Y'}(t_k^1)] \\ &= [\pi^{N'}_{X' \sqcup Y'}(t_1^2), \ldots, \pi^{N'}_{X' \sqcup Y'}(t_k^2)] \\ &= \pi^N_{X \sqcup Y}(t_2) \end{aligned}$$

which we had to prove. If N is a set-valued or multiset-valued attribute, then $X \leq Y$ or $Y \leq X$ according to Definition 5.2 of reconcilable subattributes. □

We will see later on that this condition is exact, i.e., if the projections on X and Y do determine the projection on $X \sqcup Y$, then X and Y are necessarily reconcilable. In [139], where only record and set type have been studied, the term semi-disjoint was used instead of the term reconcilable. In that setting reconcilability of two nested attributes $X, Y \in Sub(N)$ means that there are $X' \leq X, Y' \leq Y$ with $X' \sqcap Y' = \lambda_N$ and $X' \sqcup Y' = X \sqcup Y$.

Definition 5.4. The *generalised Armstrong axioms for FDs* are

$$\frac{}{\mathcal{X} \to \mathcal{Y}}\ \mathcal{Y} \subseteq \mathcal{X}, \quad \frac{}{\{X\} \to \{Y\}}\ Y \leq X, \quad \frac{\mathcal{X} \to \mathcal{Y}}{\mathcal{X} \to \mathcal{X} \cup \mathcal{Y}},$$

$$\frac{}{\{X, Y\} \to \{X \sqcup_N Y\}}\ X, Y \text{reconcilable}, \quad \frac{\mathcal{X} \to \mathcal{Y},\ \mathcal{Y} \to \mathcal{Z}}{\mathcal{X} \to \mathcal{Z}},$$

i.e., reflexivity axiom, subattribute axiom, extension rule, restricted join axiom and transitivity rule. □

5.1.3 Soundness and some useful Inference Rules

We show that all FDs that can be derived from a given set Σ of FDs using any of the rules from Definition 5.4 are also implied by Σ. This shows in particular that reconcilability is indeed a sufficient condition under which the join axiom is sound.

Proposition 5.5. *The generalised Armstrong axioms are sound for the implication of FDs in the presence of records, lists, sets and multisets.*

Proof. Let $N \in \mathcal{N}\mathrm{A}$ and $r \subseteq dom(N)$. First consider the *reflexivity axiom*, and let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$ for all $X \in \mathcal{X}$. Since $\mathcal{Y} \subseteq \mathcal{X}$ this implies that $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ holds also for all $Y \in \mathcal{Y}$.

For the *subattribute axiom* let again $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$. For $Y \leq X$ follows $\pi^N_Y = \pi^X_Y \circ \pi^N_X$ where $\circ$ denotes the composition of functions. Consequently, $\pi^N_Y(t_1) = \pi^X_Y(\pi^N_X(t_1)) = \pi^X_Y(\pi^N_X(t_2)) = \pi^N_Y(t_2)$.

In order to prove the *extension rule* let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$ for all $X \in \mathcal{X}$. Since $\models_r \mathcal{X} \to \mathcal{Y}$ holds, it follows that $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ holds for all $Y \in \mathcal{Y}$. Consequently, $\pi^N_Z(t_1) = \pi^N_Z(t_2)$ is true for all $Z \in \mathcal{X} \cup \mathcal{Y}$.

For the *restricted join axiom* let X and Y be reconcilable, and $r \subseteq dom(N)$. Let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$ and $\pi^N_Y(t_1) = \pi^N_Y(t_2)$. Lemma 5.3 shows that also $\pi^N_{X \sqcup Y}(t_1) = \pi^N_{X \sqcup Y}(t_2)$ holds. The soundness of the restricted join axiom follows.

For the proof of the *transitivity rule* let $t_1, t_2 \in r$ with $\pi^N_X(t_1) = \pi^N_X(t_2)$ for all $X \in \mathcal{X}$. Since $\models_r \mathcal{X} \to \mathcal{Y}$ holds, we infer $\pi^N_Y(t_1) = \pi^N_Y(t_2)$ for all $Y \in \mathcal{Y}$. Moreover, $\models_r \mathcal{Y} \to \mathcal{Z}$ which implies $\pi^N_Z(t_1) = \pi^N_Z(t_2)$ for all $Z \in \mathcal{Z}$. This proves that $\models_r \mathcal{X} \to \mathcal{Z}$ holds as well. □

Recall that the famous Armstrong axioms for the implication of FDs in the RDM consist of the reflexivity axiom, the extension rule and the transitivity rule with $\mathcal{X}, \mathcal{Y}$ and $\mathcal{Z}$ being sets of flat attribute names. Subattribute and restricted join axiom, however, are not needed in the RDM since flat attribute names are not comparable anyway, i.e., form an anti-chain. We derive a couple of sound inference rules from the generalised Armstrong axioms which will be needed in the completeness proof.

Lemma 5.6. *The following inference rules are derivable from the generalised Armstrong axioms, and hence are sound:*

$$\frac{}{\mathcal{X} \to \{\lambda\}} \quad \frac{\mathcal{X} \to \mathcal{Y}, \mathcal{X} \to \mathcal{Z}}{\mathcal{X} \to \mathcal{Y} \cup \mathcal{Z}} \quad \frac{\mathcal{X} \to \{Z\}}{\mathcal{X} \to \{Y\}}\ Y \leq Z \quad \frac{\mathcal{X} \to \mathcal{Z}}{\mathcal{X} \to \mathcal{Y}}\ \mathcal{Y} \subseteq \mathcal{Z}\ .$$

They are called λ-axiom, union rule, subattribute rule and subset rule, respectively.

Proof. Applications of any of these rules can be replaced by inferences using the generalised Armstrong axioms only.
λ-axiom: Every instantiation of $\mathcal{X} \to \{\lambda\}$ in any derivation tree is an FD according to Definition 5.1. We can therefore assume that there is some $X \in \mathcal{X}$ where X is used as a parameter for an element of $\mathcal{X}$.

$$\dfrac{\dfrac{}{\mathcal{X} \to \{X\}}\{X\} \subseteq \mathcal{X} \quad \dfrac{}{\{X\} \to \{\lambda\}}\lambda \leq X}{\mathcal{X} \to \{\lambda\}}$$

union rule:

$$\dfrac{\dfrac{\dfrac{}{\mathcal{X} \cup \mathcal{Y} \to \mathcal{X}}\mathcal{X} \subseteq \mathcal{X} \cup \mathcal{Y} \quad \mathcal{X} \to \mathcal{Z}}{\mathcal{X} \cup \mathcal{Y} \to \mathcal{Z}} \qquad \dfrac{\dfrac{}{\mathcal{X} \cup \mathcal{Y} \to \mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z}} \quad \dfrac{}{\mathcal{X} \cup \mathcal{Y} \cup \mathcal{Z} \to \mathcal{Y} \cup \mathcal{Z}}}{\mathcal{X} \cup \mathcal{Y} \to \mathcal{Y} \cup \mathcal{Z}}}{\dfrac{\mathcal{X} \to \mathcal{Y}}{\mathcal{X} \to \mathcal{X} \cup \mathcal{Y}} \qquad \mathcal{X} \to \mathcal{Y} \cup \mathcal{Z}}$$

subattribute rule:

$$\dfrac{\mathcal{X} \to \{Z\} \qquad \dfrac{}{\{Z\} \to \{Y\}}Y \leq Z}{\mathcal{X} \to \{Y\}}$$

subset rule:

$$\dfrac{\mathcal{X} \to \mathcal{Z} \qquad \dfrac{}{\mathcal{Z} \to \mathcal{Y}}\mathcal{Y} \subseteq \mathcal{Z}}{\mathcal{X} \to \mathcal{Y}}$$

□

If one chooses to permit empty sets $\mathcal{X}, \mathcal{Y}$ in the definition of FDs $\mathcal{X} \to \mathcal{Y}$, then the λ-axiom is not implied by the generalised Armstrong axioms from Definition 5.4. In this case, the λ-axiom needs to be included in this set to achieve completeness.

5.1.4 Completeness

The idea for the completeness proof follows again the original lines of reasoning: for every $\mathcal{X} \to \mathcal{Y} \notin \Sigma^+$ a two element instance $r = \{t_1, t_2\}$ is constructed which satisfies all FDs in Σ, but does not satisfy $\mathcal{X} \to \mathcal{Y}$. In fact, the projections of t_1 and t_2 will coincide on exactly those subattributes W which are in the closure $\mathcal{X}^+$ of $\mathcal{X}$ with respect to Σ. In order to construct this instance r we need some further preparations.

Definition 5.7. Let $N \in \mathcal{N}\mathrm{A}$. The identifying term $\tau_N(X)$ of $X \in Sub(N)$ is inductively defined as follows:

- $\tau_\lambda(\lambda) = ok$,
- $\tau_A(\lambda) = a, \tau_A(A) = a'$ with $a, a' \in dom(A)$ and $a \neq a'$ for $A \in \mathcal{U}$,

- $\tau_{L(N_1,\ldots,N_k)}(L(M_1,\ldots,M_k)) = (\tau_{N_1}(M_1),\ldots,\tau_{N_k}(M_k))$,
- $\tau_{L\{N\}}(L\{M\}) = \{\tau_N(M)\}$ and $\tau_{L\{N\}}(\lambda) = \emptyset$,
- $\tau_{L\langle N\rangle}(L\langle M\rangle) = \langle\tau_N(M)\rangle$ and $\tau_{L\{N\}}(\lambda) = \langle\,\rangle$,
- $\tau_{L[N]}(L[M]) = [\tau_N(M)]$ and $\tau_{L[N]}(\lambda) = [\,]$. □

Figure 5.1 shows the subattributes X of $K\{L(A, M[N(B,C)])\}$ together with their identifying terms.

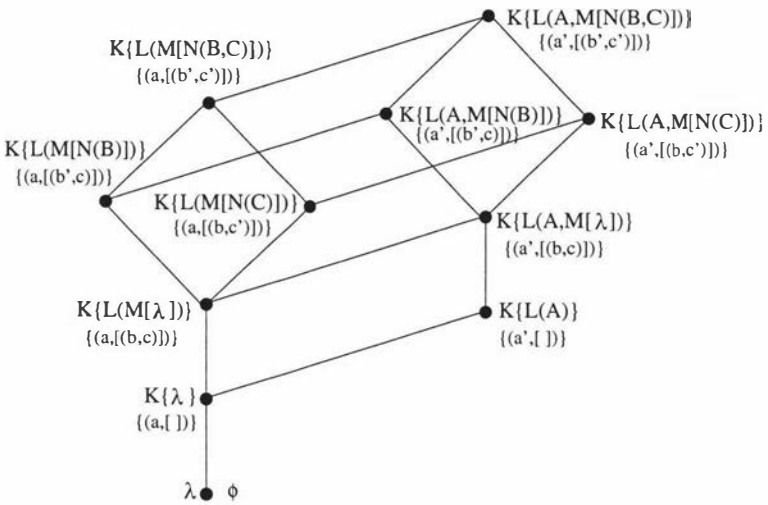

Fig. 5.1. Identifying Terms of the Algebra $K\{L(A, M[N(B,C)])\}$

The problem is now the construction of the two element instance where the difficult cases are set- and multiset-valued attributes. In order to deal with these cases some technical lemmata are proven first.

Technical Lemmata. We establish some results on the projection of identifying terms. If the projection of Y's identifying term on X is the same as the projection of X's identifying term on X, then is X necessarily a subattribute of Y:

Lemma 5.8. *Let $N \in \mathcal{NA}$ and $X, Y \in Sub(N)$. Then $\pi^N_X(\tau_N(Y)) = \pi^N_X(\tau_N(X))$ implies $X \leq Y$.*

Proof. We will show the contraposition by induction on N. From $X \not\leq Y$ follows $X \neq \lambda$.

Let $N = A$ be flat attribute. For $X \not\leq Y$ it remains to consider the case where $X = A$ and $Y = \lambda$. Then $\pi^N_X(\tau_N(Y)) = \tau_A(\lambda) = a$ and $\pi^N_X(\tau_N(X)) = \tau_A(A) = a'$. This shows $\pi^N_X(\tau_N(Y)) \neq \pi^N_X(\tau_N(X))$.

Let $N = L(N_1,\ldots,N_k), X = L(X_1,\ldots,X_k)$ and $Y = L(Y_1,\ldots,Y_k)$. From $X \not\leq Y$ follows $X_i \not\leq Y_i$ for some i with $1 \leq i \leq k$. We conclude that $\pi^{N_i}_{X_i}(\tau_{N_i}(Y_i)) \neq \pi^{N_i}_{X_i}(\tau_{N_i}(X_i))$ holds by hypothesis. However, since $\pi^N_X(\tau_N(Y)) = \pi^N_X(\tau_N(X))$ is equivalent to the fact that

$\pi^{N_j}_{X_j}(\tau_{N_j}(Y_j)) = \pi^{N_j}_{X_j}(\tau_{N_j}(X_j))$ holds for all $j = 1, \ldots, k$ the statement of the lemma follows for this case.

Let $N = L\{N'\}$. Then we distinguish between two cases. First, let $Y = \lambda$ and $X = L\{X'\}$. Then we have

$$\pi^N_X(\tau_N(X)) = \pi^{L\{N'\}}_{L\{X'\}}(\tau_{L\{N'\}}(L\{X'\})) = \pi^{L\{N'\}}_{L\{X'\}}(\{\tau_{N'}(X')\}) = \{\pi^{N'}_{X'}(\tau_{N'}(X'))\},$$

but

$$\pi^N_X(\tau_N(Y)) = \pi^{L\{N'\}}_{L\{X'\}}(\tau_{L\{N'\}}(\lambda)) = \pi^{L\{N'\}}_{L\{X'\}}(\emptyset) = \emptyset \quad .$$

It remains the case where $Y = L\{Y'\}$ and $X = L\{X'\}$. From $X' \not\leq Y'$ follows $\pi^{N'}_{X'}(\tau_{N'}(Y')) \neq \pi^{N'}_{X'}(\tau_{N'}(X'))$ by hypothesis. It follows that

$$\pi^N_X(\tau_N(Y)) = \{\pi^{N'}_{X'}(\tau_{N'}(Y'))\} \neq \{\pi^{N'}_{X'}(\tau_{N'}(X'))\} = \pi^N_X(\tau_N(X)) \quad .$$

The proof for the remaining cases of multiset- and list-valued attributes are completely analogous to the case of set-valued attributes. □

The projection of X's identifying term on Y is the projection of $X \sqcap Y$'s identifying term on Y:

Lemma 5.9. *For $N \in \mathcal{NA}$, $X, Y \in Sub(N)$ holds $\pi^N_Y(\tau_N(X)) = \pi^N_Y(\tau_N(X \sqcap Y))$.*

Proof. If $Y = \lambda$, then there is nothing to show. If $N = Y$, then $X \sqcap Y = X \sqcap N = X$. If $X \leq Y$, then $X \sqcap Y = X$. In both cases the lemma is obviously true.

We proceed by induction on N. The cases where $N = \lambda$ or N is a flat attribute follow from the considerations above. Suppose $N = L(N_1, \ldots, N_k)$, $Y = L(Y_1, \ldots, Y_k)$ and $X = L(X_1, \ldots, X_k)$. We compute

$$\begin{aligned}
\pi^N_Y(\tau_N(X)) &= (\pi^{N_1}_{Y_1}(\tau_{N_1}(X_1)), \ldots, \pi^{N_k}_{Y_k}(\tau_{N_k}(X_k))) \\
&= (\pi^{N_1}_{Y_1}(\tau_{N_1}(X_1 \sqcap Y_1), \ldots, \pi^{N_k}_{Y_k}(\tau_{N_k}(X_k \sqcap Y_k)) \\
&= \pi^N_Y(\tau_N(X \sqcap Y)).
\end{aligned}$$

Let $N = L\{N'\}, Y = L\{Y'\}$ and $X = L\{X'\}$. It follows

$$\begin{aligned}
\pi^N_Y(\tau_N(X \sqcap Y)) &= \pi^{L\{N'\}}_{L\{Y'\}}(\tau_{L\{N'\}}(L\{X'\} \sqcap L\{Y'\})) \\
&= \pi^{L\{N'\}}_{L\{Y'\}}(\tau_{L\{N'\}}(L\{X' \sqcap Y'\})) \\
&= \pi^{L\{N'\}}_{L\{Y'\}}(\{\tau_{N'}(X' \sqcap Y')\}) \\
&= \{\pi^{N'}_{Y'}(\tau_{N'}(X' \sqcap Y'))\} \\
&= \{\pi^{N'}_{Y'}(\tau_{N'}(X'))\} \\
&= \pi^{L\{N'\}}_{L\{Y'\}}(\{\tau_{N'}(X')\}) \\
&= \pi^{L\{N'\}}_{L\{Y'\}}(\tau_{L\{N'\}}(L\{X'\})) \\
&= \pi^N_Y(\tau_N(X)) \quad .
\end{aligned}$$

The proof for the remaining cases of multiset- and list-valued attributes are completely analogous to the case of set-valued attributes. □

The general construction of the desired two element instance is given in Lemma 5.14. While inductive arguments can be used for record- and list-valued attributes the elements for set- and multiset-valued attributes must be constructed directly. This is due to the notion of reconcilability.

The Case of Sets. The construction in the case of set-valued attributes $L\{P\}$ is based on the following idea. Given some ideal $\mathcal{Y}$ of subattributes of P, one element contains exactly the identifying terms of subattributes in $\mathcal{Y}$ while the other element contains the identifying terms of all subattributes of P.

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 $W \in \mathcal{X}$.*

Proof. Since $\mathcal{X} \neq \emptyset$ is an ideal we have $\lambda \in \mathcal{X}$. Let $\mathcal{X} = \{L\{X\} : X \in \mathcal{Y}\} \cup \{\lambda\}$ for some $\mathcal{Y} \subseteq Sub(P)$. Let $t_N = \{\tau_P(X) : X \leq P\}$ and $t'_N = \{\tau_P(X) : X \in \mathcal{Y}\}$. For $W = \lambda$ we obviously have $\pi^N_\lambda(t_N) = ok = \pi^N_\lambda(t'_N)$. Let now be $W = L\{V\}$. We need to show that

$$\{\pi^P_V(\tau_P(X)) : X \leq P\} = \{\pi^P_V(\tau_P(X)) : X \in \mathcal{Y}\} \text{ if and only if } V \in \mathcal{Y}$$

holds. It is always true that $\{\pi^P_V(\tau_P(X)) : X \in \mathcal{Y}\} \subseteq \{\pi^P_V(\tau_P(X)) : X \leq P\}$ holds since $\mathcal{Y} \subseteq Sub(P)$.

We show first that $V \in \mathcal{Y}$ implies $\{\pi^P_V(\tau_P(X)) : X \leq P\} \subseteq \{\pi^P_V(\tau_P(X)) : X \in \mathcal{Y}\}$. Let $V \in \mathcal{Y}$. We show that for all $X \leq P$ there is some $Y \in \mathcal{Y}$ with $\pi^P_V(\tau_P(X)) = \pi^P_V(\tau_P(Y))$. If $X \in \mathcal{Y}$, then obviously take $Y = X$. If $X \notin \mathcal{Y}$, then take $Y = X \sqcap V$. We conclude $\pi^P_V(\tau_P(X)) = \pi^P_V(\tau_P(Y))$ by Lemma 5.9. Certainly, $Y \in \mathcal{Y}$ since $\mathcal{Y}$ is an $\leq$-ideal.

It remains to show that $\{\pi^P_V(\tau_P(X)) : X \in \mathcal{Y}\} \subset \{\pi^P_V(\tau_P(X)) : X \leq P\}$, if $V \notin \mathcal{Y}$. Let $V \notin \mathcal{Y}$. Since $\mathcal{Y}$ is an ideal it follows that all $X \leq P$ with $V \leq X$ also satisfy $X \notin \mathcal{Y}$. Hence, $\tau_P(X) \in t_N$, but $\tau_P(X) \notin t'_N$ for all X with $V \leq X \leq P$. Suppose there was some $X \in \mathcal{Y}$ with $\pi^P_V(\tau_P(X)) = \pi^P_V(\tau_P(V))$. Using Lemma 5.8 we infer $V \leq X$ and therefore $\tau_P(X) \notin t'_N$. This is a contradiction since $\tau_P(X) \in t'_N$ for all $X \in \mathcal{Y}$ holds. Consequently, $\pi^P_V(\tau_P(X)) \neq \pi^P_V(\tau_P(V))$ for all $X \notin \mathcal{Y}$. We conclude that $\pi^P_V(\tau_P(V)) \in \{\pi^P_V(\tau_P(X)) : X \leq P\}$ and $\pi^P_V(\tau_P(V)) \notin \{\pi^P_V(\tau_P(X)) : X \in \mathcal{Y}\}$. This concludes the proof. □

EXAMPLE 5.3. Consider the nested attribute $N = K\{L(A, M[O(B, C)])\}$ together with the FDs $K\{L(A)\} \to K\{L(M[O(B)])\}$ and $K\{L(A)\} \to K\{L(M[O(C)])\}$. The closure $\mathcal{X}^+$ of $\mathcal{X} = K\{L(A)\}$ with respect to these FDs is illustrated in Figure 5.2.

We generate two elements t_N, t'_N which coincide exactly on the elements of $\mathcal{X}^+$. Following the proof of Lemma 5.10, $t_N = \{\tau_{L(A,M[O(B,C)])}(X) : X \leq L(A, M[O(B, C)])\}$ is

$$\{(a', [(b', c')]); (a, [(b', c')]); (a', [(b', c)]); (a', [(b, c')]); (a, [(b', c)]);$$
$$(a, [(b, c')]); (a', [(b, c)]); (a, [(b, c)]); (a', [\,]); (a, [\,])\}$$

and $t'_N = \{\tau_{L(A,M[O(B,C)])}(Y) : Y \in \mathcal{Y}\}$ is

$$\{(a, [(b', c)]); (a, [(b, c')]); (a, [(b, c)]); (a', [\,]); (a, [\,])\} \quad .$$

The projections $\pi^N_W(t)$ and $\pi^N_W(t')$ for $W \in Sub(N)$ are:

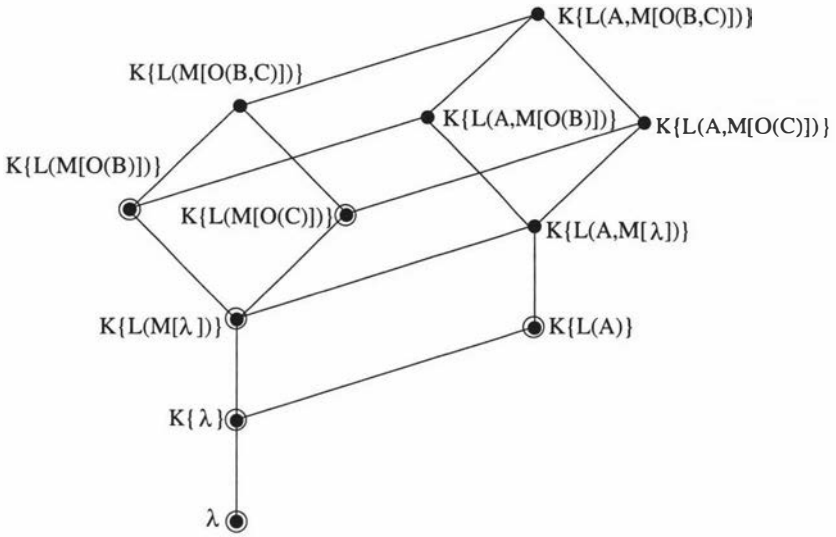

Fig. 5.2. The closure $\mathcal{X}^+$ of $\mathcal{X} = K\{L(A)\}$

W	$\pi^N_W(t_N)$	$\pi^N_W(t'_N)$
$K\{L(M[O(B)])\}$	$\{(ok, [(b', ok)]); (ok, [(b, ok)]); (ok, [\,])\}$	
$K\{L(M[O(C)])\}$	$\{(ok, [(ok, c')]); (ok, [(ok, c')]); (ok, [\,])\}$	
$K\{L(A)\}$	$\{(a', ok); (a, ok)\}$	
$K\{L(M[O(B,C)])\}$	$\{(ok, [(b, c)]); (ok, [(b', c)]), (ok, [(b, c')]);$ $(\mathbf{ok}, [(\mathbf{b'}, \mathbf{c'})]); (ok, [\,])\}$	$\{(ok, [(b, c)]); (ok, [(b', c)]);$ $(ok, [(b, c')]); (ok, [\,])\}$
$K\{L(A, M[\lambda])\}$	$\{(a, [(ok, ok)]); (a, [\,]);$ $(\mathbf{a'}, [(\mathbf{ok}, \mathbf{ok})]); (a', [\,])\}$	$\{(a, [(ok, ok)]);$ $(a, [\,]); (a', [\,])\}$

Indeed, t_N and t'_N coincide on all maximal elements of $\mathcal{X}^+$, and therefore on all elements of $\mathcal{X}^+$. Furthermore, t_N and t'_N deviate on all minimal attributes of $Sub(N)$ which are not in $\mathcal{X}^+$. □

The Case of Multisets. The strategy used for set-valued attributes does not work for multiset-valued attributes since multiple occurrences of projections do not vanish in a multiset. At this point it helps to look deeper into the structure of nested attributes. In fact, the relativised subalgebra $(Sub(X), \leq, \sqcup, \sqcap, \dot{-}, X)$ with respect to $X \sqcap X_1 \sqcap \cdots \sqcap X_k$ is Boolean where $X_1, \ldots, X_k$ are the $\leq$-maximal subattributes of $X \in Sub(N)$.

Let $X, Y \in Sub(N)$ with $X \leq Y$. Then $[X, Y] = \{Z \in Sub(N) : X \leq Z \leq Y\}$ is called an *interval* of $Sub(N)$, [9, 101].

Lemma 5.11. *Let $N \in \mathcal{N}A$ and $X \in Sub(N)$. Let $\{X_1, \ldots, X_k\}$ be the set of all $\leq$-maximal proper subattributes of X. Then $([0_X, X], \leq, \sqcap, \sqcup, \overline{(\cdot)}, 0_X, X)$ forms a Boolean algebra where $0_X = X \sqcap X_1 \sqcap \cdots \sqcap X_k$ and $\overline{Y} = (X \dot{-} Y) \sqcup 0_X$ for all $Y \in [0_X, X]$.*

Proof. The order $\leq$, meet $\sqcap$ and join $\sqcup$ in $([0_X, X], \leq, \sqcap, \sqcup)$ are the respective restrictions of order, meet and join from $(Sub(N), \leq, \sqcap, \sqcup)$ to $[0_X, X]$. $[0_X, X]$ is closed under meet $\sqcap$,

join $\sqcup$ and complement $\overline{(\cdot)}$. It remains to show that $\overline{Y} = (X \dot{-} Y) \sqcup 0_X$ defines indeed the complement of $Y \in [0_X, X]$.

We show that $Y \sqcap (X \dot{-} Y) \leq 0_X$ holds for all $Y \in [0_X, X]$. If $Y = X$, then $Y \sqcap (X \dot{-} Y) = \lambda_N \leq 0_X$. If $Y = 0_X$, then $Y \sqcap (X \dot{-} Y) = 0_X \leq 0_X$. For every other $Y \in [0_X, X]$ we have then $Y = X_{i_1} \sqcap \cdots \sqcap X_{i_n}$ where $\{1, \ldots, k\}$ is the disjoint union of the two non-empty sets $\{i_1, \ldots, i_n\}$ and $\{j_1, \ldots, j_m\}$. Since $(X_{i_1} \sqcap \cdots \sqcap X_{i_n}) \sqcup (X_{j_1} \sqcap \cdots \sqcap X_{j_m}) = (X_{i_1} \sqcup X_{j_1}) \sqcap \cdots \sqcap (X_{i_n} \sqcup X_{j_m}) = X$ holds (the join of two different maximal proper subattributes of X is always X) we have $X \dot{-} Y \leq X_{j_1} \sqcap \cdots \sqcap X_{j_m}$. We conclude that $Y \sqcap (X \dot{-} Y) \leq X_{i_1} \sqcap \cdots \sqcap X_{i_n} \sqcap X_{j_1} \sqcap \cdots \sqcap X_{j_m} = X_1 \sqcap \cdots \sqcap X_k = X \sqcap X_1 \sqcap \cdots \sqcap X_k = 0_X$.

It follows then that $Y \sqcup \overline{Y} = Y \sqcup (X \dot{-} Y) \sqcup 0_X = X \sqcup Y \sqcup 0_X = X$, and $Y \sqcap \overline{Y} = Y \sqcap ((X \dot{-} Y) \sqcup 0_X) = (Y \sqcap (X \dot{-} Y)) \sqcup (Y \sqcap 0_X) \leq 0_X \sqcup 0_X = 0_X$. This completes the proof. □

We are going to prove the existence of two elements which deviate in their projections on exactly all elements of a principal filter, i.e., on all elements in the shaded area of the left picture in Figure 5.3.

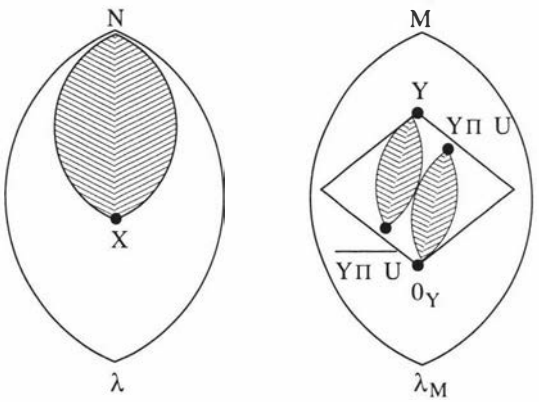

Fig. 5.3. Illustration of Lemma 5.12

The idea is to use a bijection between the intervals $[0_Y, Y \sqcap U]$ and $[\overline{Y \sqcap U}, Y]$. The meet of Y and some $\leq$-maximal subattribute U of Y that is not in the principal filter of Y, however, is always the complement of some atom. This is illustrated in the right-hand picture of Figure 5.3. One multiset contains the identifying terms of all attributes from the even levels of $([0_Y, Y], \leq)$, the other multiset contains the identifying terms of all attributes from the odd levels of $([0_Y, Y], \leq)$. The kth level of $([0_Y, Y], \leq)$ is defined as the set of all elements in $[0_Y, Y]$ that have distance k to 0_Y in the Hasse diagram of $([0_Y, Y], \leq)$, see also [9, 101].

Lemma 5.12. *Let $N = L\langle M \rangle \in \mathcal{NA}$ and $\lambda \neq X = L\langle Y \rangle \leq N$. Then there are $t_1, t_2 \in dom(N)$ with $\pi^N_W(t_1) \neq \pi^N_W(t_2)$ for $W \in Sub(N)$ if and only if $X \leq W$.*

Proof. Let $([0_Y, Y], \leq, \sqcap, \sqcup, \overline{(\cdot)}, 0_Y, Y)$ be the Boolean algebra according to Lemma 5.11 where $[0_Y, Y]$ contains 2^k elements. Let $\mathcal{L}_i$ denote the ith level of $([0_Y, Y], \leq)$ for $i = 0, \ldots, k$. Then we define $t_1 = \langle \tau_M(Z) \,:\, Z \in \mathcal{L}_i, i \text{ even} \rangle$ and $t_2 = \langle \tau_M(Z) \,:\, Z \in \mathcal{L}_i, i \text{ odd} \rangle$. Note that $t_2 = \langle\,\rangle$, if $k = 0$.

First, it follows that $\pi_Y^M(\tau_M(Y))$ is an element of either $\pi_X^N(t_1)$ or $\pi_X^N(t_2)$. If $\pi_Y^M(\tau_M(Y)) = \pi_Y^M(\tau_M(Z))$ held for some $Z \leq M$, then $Y \leq Z$ by Lemma 5.8. The elements t_1 and t_2, however, have only identifying terms of subattributes $Z \leq Y$ as members. We conclude that $\pi_Y^M(\tau_M(Y)) \neq \pi_Y^M(\tau_M(Z))$ for $Z < Y$. This shows that $\pi_X^N(t_1) \neq \pi_X^N(t_2)$, and therefore also $\pi_W^N(t_1) \neq \pi_W^N(t_2)$ whenever $X \leq W$.

It remains to show that $\pi_W^N(t_1) = \pi_W^N(t_2)$ holds whenever $X \not\leq W$ holds. It is sufficient to show that $\pi_V^N(t_1) = \pi_V^N(t_2)$ holds for all $\leq$-maximal subattributes $V \in Sub(N)$ with $X \not\leq V$. This is obvious if $V = \lambda$. Let therefore be $V = L\langle U \rangle$ where U is a $\leq$-maximal subattribute $U \in Sub(M)$ with $Y \not\leq U$.

We show first that $Y \sqcap U$ is always a $\leq$-maximal proper subattribute of Y. Suppose there is some Z with $Y \sqcap U < Z < Y$. If $U = Z \sqcup U$, then

$$U \sqcap Y = (Z \sqcup U) \sqcap Y = (Z \sqcap Y) \sqcup (U \sqcap Y) = Z \sqcup (U \sqcap Y) = Z$$

and this contradicts $U \sqcap Y < Z$. This shows $U < Z \sqcup U$. If $Y \leq Z \sqcup U$, then $Y \dot{-} Z \leq U \sqcap Y \leq Z$. This means $Y \leq Z$ which gives the contradiction $Z < Y \leq Z$. We conclude that $U < Z \sqcup U$ and $Y \not\leq Z \sqcup U$. This contradicts the $\leq$-maximality of U with $Y \not\leq U$ and shows that $Z = Y \sqcap U$ or $Z = Y$, i.e., $Y \sqcap U$ is indeed a $\leq$-maximal proper subattribute of Y. This implies that $Y \sqcap U$ is always the complement of an atom of $([0_Y, Y], \leq)$.

Let $[0_Y, Y \sqcap U], [\overline{Y \sqcap U}, Y]$ denote the intervals between 0_Y and $Y \sqcap U$, and $\overline{Y \sqcap U}$ and Y, respectively. The mapping $Z \mapsto Z \sqcup \overline{Y \sqcap U}$ from $[0_Y, Y \sqcap U]$ to $[\overline{Y \sqcap U}, Y]$ is bijective with inverse $Z \mapsto Z \sqcap (Y \sqcap U)$. Since $\overline{Y \sqcap U}$ is an atom we have $\tau_M(Z \sqcup \overline{Y \sqcap U}) \in t_2$ whenever $\tau_M(Z) \in t_1$, and vice versa. The situation is illustrated in the right picture of Figure 5.3.

It is now sufficient to show that $\pi_U^M(\tau_M(Z)) = \pi_U^M(\tau_M(Z \sqcup \overline{Y \sqcap U}))$ for $Z \in [0_Y, Y \sqcap U]$. We have

$$\begin{aligned}
\pi_U^M(\tau_M(Z)) &= \pi_U^M(\tau_M(Z \sqcup 0_Y)) && (Z = Z \sqcup 0_Y)\\
&= \pi_U^M(\tau_M((Z \sqcap U) \sqcup (\overline{Y \sqcap U} \sqcap Y \sqcap U))) && (Z = Z \sqcap U, \overline{Y \sqcap U} \sqcap Y \sqcap U = 0_Y)\\
&= \pi_U^M(\tau_M((Z \sqcap U) \sqcup (\overline{Y \sqcap U} \sqcap U))) && (\overline{Y \sqcap U} \sqcap Y = \overline{Y \sqcap U})\\
&= \pi_U^M(\tau_M((Z \sqcup \overline{Y \sqcap U}) \sqcap U)) && (\text{Distributivity})\\
&= \pi_U^M(\tau_M(Z \sqcup \overline{Y \sqcap U}))
\end{aligned}$$

where the last equation follows from Lemma 5.9. □

For the general construction we pick all $\leq$-minimal subattributes M_i that are not in the ideal $\mathcal{X}$ and form the union over all multisets given by the previous lemma on all generated principal filters. This is illustrated by Figure 5.4.

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 $\pi_W^N(t_N) = \pi_W^N(t'_N)$ if and only if $W \in \mathcal{X}$.*

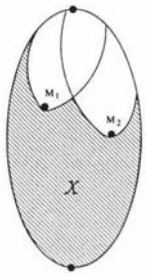

Fig. 5.4. Illustration of Lemma 5.13

Proof. Let $\{M_1, \ldots, M_n\} \subseteq Sub(N)$ be the set of all $\leq$-minimal subattributes of N with $M_i \notin \mathcal{X}$. Since $\lambda \in \mathcal{X}$ holds it follows that $M_i \neq \lambda$ for all $i = 1, \ldots, n$. According to Lemma 5.12, and for all $i = 1, \ldots, n$, there are $t_{M_i}, t'_{M_i} \in dom(N)$ with $\pi^N_Z(t_{M_i}) \neq \pi^N_Z(t'_{M_i})$ if and only if $M_i \leq Z$. Define $t_N = \bigcup_{i=1}^{n} t_{M_i}$ and $t'_N = \bigcup_{i=1}^{n} t'_{M_i}$, where the union is taken over multisets. If $W \in \mathcal{X}$ holds, then $M_i \not\leq W$ for all $i = 1, \ldots, n$ and, consequently $\pi^N_W(t_{M_i}) = \pi^N_W(t'_{M_i})$ holds for all $i = 1, \ldots, n$ as well. This implies $\pi^N_W(t_N) = \pi^N_W(t'_N)$. If $W \notin \mathcal{X}$ holds, then there is some j with $1 \leq j \leq n$ such that $M_j \leq W$ holds. The element $\pi^N_W(\tau_N(M_j))$, however, is member of exactly one of $\pi^N_W(t_N), \pi^N_W(t'_N)$ by the construction. This implies $\pi^N_W(t_N) \neq \pi^N_W(t'_N)$. Consequently, $\pi^N_W(t_N) = \pi^N_W(t'_N)$ if and only if $W \in \mathcal{X}$. □

EXAMPLE 5.4. We will illustrate the construction for multisets. Consider the nested attribute $N = L\langle M\rangle$ with $M = K(J[A], O\{P(B, Q\{C\})\})$. The structure of $(Sub(M), \leq)$ is illustrated in Figure 5.5 where labels have been omitted.

Let $\mathcal{X} = \{L\langle X\rangle \ : \ X \in \mathcal{Y}\}$ where $\mathcal{Y}$ is the ideal that consists of all subattributes of M which are circled in Figure 5.5. The $\leq$-minimal subattributes $V \in Sub(N)$ with $V \notin \mathcal{X}$ are $V_1 = L\langle K(J[\lambda], O\{P(B, Q\{\lambda\})\})\rangle$ and $V_2 = L\langle K(J[A], O\{P(\lambda, \lambda)\})\rangle$. The structures of $([K(\lambda, O\{P(\lambda, \lambda)\}), K(J[\lambda], O\{P(B, Q\{\lambda\})\})], \leq)$ and $([K(J[\lambda], \lambda), K(J[A], O\{P(\lambda, \lambda)\})], \leq)$ are illustrated in Figure 5.6.

According to Lemma 5.12 the following elements are chosen:

$$
\begin{aligned}
t'_1 &= \langle([\,], \{(b, \emptyset)\}); ([\,], \{(b', \{c\})\}); ([a], \{(b', \emptyset)\}); ([a], \{(b, \{c\})\})\rangle \\
t'_2 &= \langle([\,], \{(b', \emptyset)\}); ([\,], \{(b, \{c\})\}); ([a], \{(b, \emptyset)\}); ([a], \{(b', \{c\})\})\rangle \\
t''_1 &= \langle([a], \emptyset); ([a'], \{(b, \emptyset)\})\rangle \\
t''_2 &= \langle([a], \{(b, \emptyset)\}); ([a'], \emptyset)\rangle \quad .
\end{aligned}
$$

Finally, and according to Lemma 5.13 one chooses

$$
\begin{aligned}
t_N = t'_1 \cup t''_1 = \langle&([\,], \{(b, \emptyset)\}); ([\,], \{(b', \{c\})\}); ([a], \{(b', \emptyset)\}); ([a], \{(b, \{c\})\}); \\
&([a], \emptyset); ([a'], \{(b, \emptyset)\})\rangle \\
t'_N = t'_2 \cup t''_2 = \langle&([\,], \{(b', \emptyset)\}); ([\,], \{(b, \{c\})\}); ([\mathbf{a}], \{(\mathbf{b}, \emptyset)\}); ([a], \{(b', \{c\})\}); \\
&([\mathbf{a}], \{(\mathbf{b}, \emptyset)\}); ([a'], \emptyset)\rangle \quad .
\end{aligned}
$$

One can verify then that $\pi^N_W(t_N) = \pi^N_W(t'_N)$ for all $\leq$-maximal $W \in \mathcal{X}$, i.e., $W \in \{L\langle K(\lambda, O\{P(B, Q\{C\})\})\rangle, L\langle K(J[\lambda], O\{P(B, \lambda)\})\rangle, L\langle K(J[\lambda], O\{P(\lambda, Q\{\lambda\})\})\rangle, L\langle K(J[A], \lambda)\rangle\}$. Furthermore, $\pi^N_{V_1}(t_N) \neq \pi^N_{V_1}(t'_N)$ and $\pi^N_{V_2}(t_N) \neq \pi^N_{V_2}(t'_N)$. □

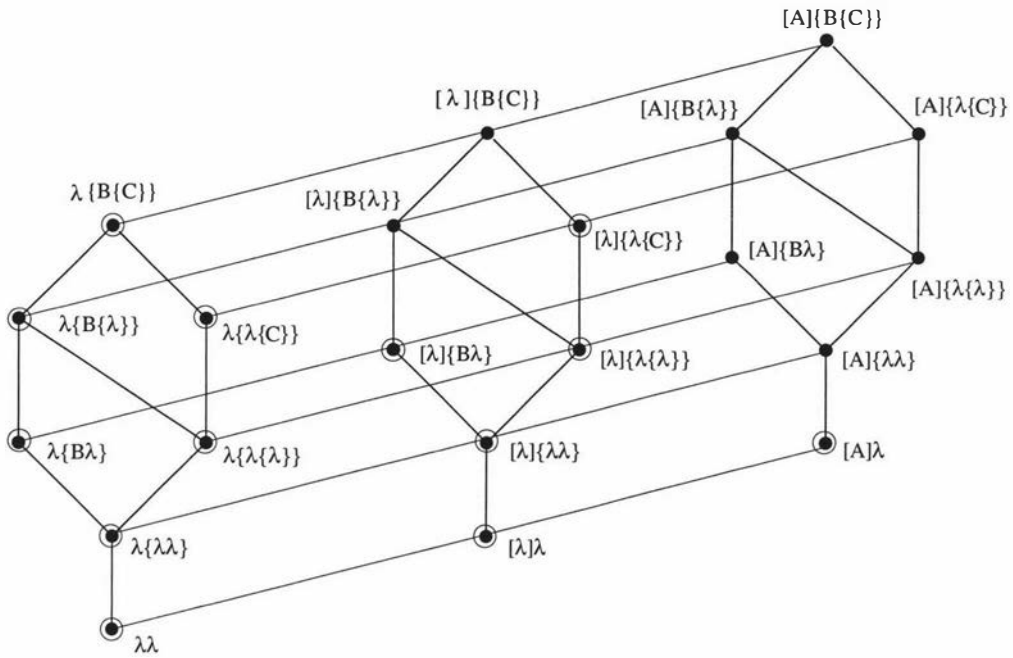

Fig. 5.5. The structure of $M = K(J[A], O\{P(B, Q\{C\})\})$

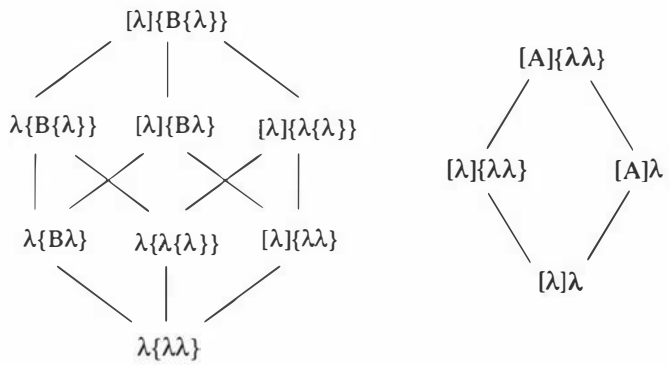

Fig. 5.6. The structure of Subalgebras in Example 5.4

The Main Lemma. The main lemma is now a mix of the previous lemmata on set- and multiset-valued attributes as well as induction arguments for record- and list-valued attributes.

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 $W \in \mathcal{X}$.*

Proof. The proof is done by induction on N. The case $N = \lambda$ is trivial. If $N = A$ is a flat attribute, then there are two cases $\mathcal{X} = \{\lambda\}$ and $\mathcal{X} = \{\lambda, A\}$ to consider. In the first case we choose $t_A = a, t'_A = a'$ with $a, a' \in dom(A)$ and $a \neq a'$, in the second case $t_A = a = t'_A$.

Consider now the case where $N = L(N_1, \ldots, N_k)$. For every $X \in \mathcal{X}$ we have $X = (X \sqcap L(N_1)) \sqcup \cdots \sqcup (X \sqcap L(N_k))$. Consequently, $\mathcal{X}_i = \{X \sqcap L(N_i) : X \in \mathcal{X}\}$ is a non-empty ideal in $Sub(L(N_i))$ for every $i = 1, \ldots, k$. Let $X_i, Y_i \in \mathcal{X}_i$ be reconcilable. Then $X_i = X \sqcap L(N_i)$ and $Y_i = Y \sqcap L(N_i)$ for some $X, Y \in \mathcal{X}$. Since $\mathcal{X}$ is an ideal it follows from $X_i \leq X$ and $Y_i \leq Y$ that $X_i, Y_i \in \mathcal{X}$, too. We conclude that $X_i \sqcup Y_i \in \mathcal{X}$ since $\mathcal{X}$ is closed under the join of reconcilable elements. Since $X_i \sqcup Y_i = (X \sqcup Y) \sqcap L(N_i) \in \mathcal{X}$ it follows that $(X_i \sqcup Y_i) \sqcap L(N_i) = X_i \sqcup Y_i \in \mathcal{X}_i$ by definition of $\mathcal{X}_i$. That is, $\mathcal{X}_i$ is also closed under the join of reconcilable elements. We know by hypothesis that for all $i = 1, \ldots, k$ there are $t_{L(N_i)}, t'_{L(N_i)} \in dom(L(N_i))$ with $\pi^{L(N_i)}_{L(W_i)}(t_{L(N_i)}) = \pi^{L(N_i)}_{L(W_i)}(t'_{L(N_i)})$ if and only if $L(W_i) \in \mathcal{X}_i$ holds. Now we choose $t_N = (t_{L(N_1)}, \ldots, t_{L(N_k)})$ and $t'_N = (t'_{L(N_1)}, \ldots, t'_{L(N_k)})$ and have the equivalence of $\pi^N_W(t_N) = \pi^N_W(t'_N)$ if and only if $W \in \mathcal{X}$ with $\pi^{L(N_i)}_{L(W_i)}(t_{L(N_i)}) = \pi^{L(N_i)}_{L(W_i)}(t'_{L(N_i)})$ if and only if $L(W_i) \in \mathcal{X}_i$ holds for $i = 1, \ldots, k$.

Suppose $N = L[N']$. Then $\mathcal{X} = \{L[M] : M \in \mathcal{Y}\} \cup \{\lambda\}$ for an ideal $\mathcal{Y} \subseteq Sub(N')$. If $\mathcal{Y} = \emptyset$, then $\mathcal{X} = \{\lambda\}$. Define $t_N = [], t'_N = [n'] \in dom(N)$ for some $n' \in dom(N')$. For $\lambda \neq W \in Sub(N)$, say $W = L[M']$, we have then $\pi^N_W(t_N) = [] \neq [\pi^{N'}_{M'}(n')] = \pi^N_W(t'_N)$. This implies $\pi^N_W(t_N) = \pi^N_W(t'_N)$ if and only if $W = \lambda$. Suppose $\mathcal{Y} \neq \emptyset$ and $X', Y' \in \mathcal{Y}$ are reconcilable. It follows that $L[X'], L[Y'] \in \mathcal{X}$ are also reconcilable. Consequently, $L[X' \sqcup Y'] = L[X'] \sqcup L[Y'] \in \mathcal{X}$ by assumption, and $X' \sqcup Y' \in \mathcal{Y}$. The hypothesis tells us that there are $t_{N'}, t'_{N'} \in dom(N')$ with $\pi^{N'}_{W'}(t_{N'}) = \pi^{N'}_{W'}(t'_{N'})$ if and only if $W' \in \mathcal{Y}$. We define $t_N = [t_{N'}], t'_N = [t'_{N'}] \in dom(N)$. First, $\pi^N_\lambda(t_N) = \pi^N_\lambda(t'_N)$ holds, and $\lambda \in \mathcal{X}$. For $\lambda \neq W \in Sub(N)$, say $W = L[W']$, we obtain

$$\pi^N_W(t_N) = [\pi^{N'}_{W'}(t_{N'})] = [\pi^{N'}_{W'}(t'_{N'})] = \pi^N_W(t'_N) \quad \text{iff} \quad W' \in \mathcal{Y} \quad \text{iff} \quad W \in \mathcal{X}.$$

The remaining cases of set- and multiset-valued attributes are covered by Lemma 5.10 and Lemma 5.13, respectively. □

The Main Theorem. As before, the key idea is now to take any $\mathcal{X} \to \mathcal{Y} \notin \Sigma^+$ and to construct an instance which satisfies all dependencies in Σ, but does not satisfy $\mathcal{X} \to \mathcal{Y}$. The proof is based on the same idea that was used in the case of the RDM, but makes also use of the fact that $\mathcal{X}^+$ (the closure of $\mathcal{X}$ under derivation of FDs from Σ) is a non-empty ideal that is closed under the join of reconcilable attributes.

Theorem 5.15. *The generalised Armstrong axioms are sound and complete for the implication of FDs in the presence of records, lists, sets and multisets.*

Proof. Soundness has been established in Proposition 5.5. We show the completeness. Let $N \in \mathcal{NA}$ and Σ be a set of FDs on N. Let $\mathcal{X} \to \mathcal{Y}$ be an FD on N with $\mathcal{X} \to \mathcal{Y} \notin \Sigma^+$. Define $\mathcal{X}^+ = \{Z : \mathcal{X} \to \{Z\} \in \Sigma^+\}$. Then $\lambda \in \mathcal{X}^+$ according to the λ-axiom. The derivability of the union rule implies that $\mathcal{X} \to \mathcal{X}^+ \in \Sigma^+$ holds. If $\mathcal{Y}$ was a subset of $\mathcal{X}^+$, the subset rule would imply that $\mathcal{X} \to \mathcal{Y} \in \Sigma^+$, a contradiction to our assumption. Hence, $\mathcal{Y} \not\subseteq \mathcal{X}^+$, i.e., there is some $Z \in \mathcal{Y}$ with $Z \notin \mathcal{X}^+$. According to the subattribute rule $\mathcal{X}^+$ is an ideal with respect to $\leq$. Moreover, if $U, V \in \mathcal{X}^+$ are reconcilable, then the

restricted join axiom implies that $U \sqcup V \in \mathcal{X}^+$, too. Therefore, using Lemma 5.14 we define $r = \{t_1, t_2\} \subseteq dom(N)$ by

$$\pi_W^N(t_1) = \pi_W^N(t_2) \quad \text{if and only if} \quad W \in \mathcal{X}^+ \tag{12}$$

holds. It is immediate that $\not\models_r \mathcal{X} \to \{Z\}$, and this implies $\not\models_r \mathcal{X} \to \mathcal{Y}$ by definition. It remains to show that $\models_r \Sigma$. Therefore, take any $\mathcal{U} \to \mathcal{V} \in \Sigma$.

- If $\mathcal{U} \not\subseteq \mathcal{X}^+$, then $\pi_U^N(t_1) \neq \pi_U^N(t_2)$ for some $U \in \mathcal{U}$ by (12). Obviously, $\models_r \mathcal{U} \to \mathcal{V}$.
- If $\mathcal{U} \subseteq \mathcal{X}^+$, then $\pi_U^N(t_1) = \pi_U^N(t_2)$ for all $U \in \mathcal{U}$ by (12). Since $\mathcal{X} \to \mathcal{X}^+ \in \Sigma^+$ it follows from the subset rule that also $\mathcal{X} \twoheadrightarrow \mathcal{U} \in \Sigma^+$ holds. Applying the transitivity rule again results in $\mathcal{X} \to \mathcal{V} \in \Sigma^+$. The subset rule guarantees that $\mathcal{V} \subseteq \mathcal{X}^+$. We conclude by (12) that $\pi_V^N(t_1) = \pi_V^N(t_2)$ holds for all $V \in \mathcal{V}$. This shows $\models_r \mathcal{U} \to \mathcal{V}$.

As $\Sigma^* = \{\mathcal{X} \to \mathcal{Y} \mid \Sigma \models \mathcal{X} \to \mathcal{Y}\}$, it follows that $\models_r \Sigma^*$. Therefore, $\mathcal{X} \to \mathcal{Y} \notin \Sigma^*$. This proves the completeness. □

5.1.5 A Note on Reconcilability

We demonstrate that reconcilability of X and Y is an exact condition for the soundness of the restricted join axiom $\frac{}{\{X, Y\} \to \{X \sqcup_N Y\}}$. This means that one cannot find a weaker sufficient condition for that rule to hold. Proposition 5.5 already implies that reconcilability is a sufficient condition. If X and Y are not reconcilable, then we show that there is some instance r with $\not\models_r \{X, Y\} \to \{X \sqcup Y\}$. It is then sufficient to find an ideal $\mathcal{Y}$ satisfying the properties of Lemma 5.14 and where $X, Y \in \mathcal{Y}$, but $X \sqcup Y \notin \mathcal{Y}$. This guarantees the existence of t_N, t'_N with $\pi_W^N(t_N) = \pi_W^N(t'_N)$ if and only if $W \in \mathcal{Y}$. The desired r is then $\{t_N, t'_N\}$.

Lemma 5.16. *Let $N \in \mathcal{NA}$ and $X, Y \in Sub(N)$. Then $\mathcal{Y} = \{U \sqcup V : U \leq X, V \leq Y, U$ and V are reconcilable$\}$ is a non-empty ideal with respect to $\leq$ and for all $S, T \in \mathcal{Y}$ that are reconcilable follows $S \sqcup T \in \mathcal{Y}$.*

Proof. $\mathcal{Y}$ is non-empty as $\lambda \in \mathcal{Y}$ holds. We show that $\mathcal{Y}$ is an ideal with respect to $\leq$. Let $S \in \mathcal{Y}$, i.e., $S = U \sqcup V$ with $U \leq X, V \leq Y$ and U, V are reconcilable. Let $T \leq S$. Then $T = S \sqcap T = (U \sqcup V) \sqcap T = (U \sqcap T) \sqcup (V \sqcap T)$ where $U \sqcap T \leq U \leq X$ and $V \sqcap T \leq V \leq Y$ holds. We show that $U \sqcap T, V \sqcap T$ are reconcilable, and conclude that $T \in \mathcal{Y}$, too. We proceed by induction on reconcilable nested attributes. If $U \leq V$, then $U \sqcap T \leq V \sqcap T$. Similarly, if $V \leq U$, then $V \sqcap T \leq U \sqcap T$. If $T = \lambda$, then $U \sqcap T = V \sqcap T$. Let $N = L(N_1, \ldots, N_k), U = L(U_1, \ldots, U_k), V = L(V_1, \ldots, V_k)$ and $T = L(T_1, \ldots, T_k)$. Since U, V are reconcilable it follows that U_i, V_i are reconcilable for all $i = 1, \ldots, k$. Consequently, $U_i \sqcap V_i$ and $V_i \sqcap T_i$ are also reconcilable for $i = 1, \ldots, k$. The reconcilability of $U \sqcap T$ and $V \sqcap T$ follows from the fact that $U \sqcap T = L(U_1 \sqcap T_1, \ldots, U_k \sqcap T_k)$ and $V \sqcap T = L(V_1 \sqcap T_1, \ldots, V_k \sqcap T_k)$. Let $N = L[N'], U = L[U'], V = L[V']$ and $T = L[T']$. Then U', V' are reconcilable by definition, and $U' \sqcap T', V' \sqcap T'$ are reconcilable as well. Since $U \sqcap T = L[U' \sqcap T']$ and $V \sqcap T = L[V' \sqcap T']$ it is proven that $U \sqcap T$ and $V \sqcap T$ are indeed reconcilable.

It remains to show that $\mathcal{Y}$ is closed under the join of reconcilable elements. Let $S, T \in \mathcal{Y}$ be reconcilable. We proceed again by induction on the definition of reconcilable nested attributes in order to show that $S \sqcup T \in \mathcal{Y}$ holds as well. Note that this is true, if $X = \lambda$ or $Y = \lambda$. If $S \leq T$, then $S \sqcup T = T \in \mathcal{Y}$, and if $T \leq S$, then $S \sqcup T = S \in \mathcal{Y}$. Let $N = L(N_1, \ldots, N_k), X = L(X_1, \ldots, X_k), Y = L(Y_1, \ldots, Y_k)$. It follows that $\mathcal{Y} = \{L(M_1, \ldots, M_k) : M_i \in \mathcal{Y}_i\}$ where $\mathcal{Y}_i = \{U_i \sqcup V_i : U_i \leq X_i, V_i \leq Y_i$ and U_i, V_i are reconcilable$\}$ is a non-empty ideal for every $i = 1, \ldots, k$. Let $S, T \in \mathcal{Y}$ be reconcilable. Then $S = L(S_1, \ldots, S_k), T = L(T_1, \ldots, T_k)$ with $S_i, T_i \in \mathcal{Y}_i$ for $i = 1, \ldots, k$. Furthermore, S_i, T_i are reconcilable. We know that $S_i \sqcup T_i \in \mathcal{Y}_i$ holds for every $i = 1, \ldots, k$, and therefore $S \sqcup T = L(S_1, \ldots, S_k) \sqcup L(T_1, \ldots, T_k) = L(S_1 \sqcup T_1, \ldots, S_k \sqcup T_k) \in \mathcal{Y}$ which proves this case.

Let $N = L[N'], X = L[X'], Y = L[Y']$. It follows that $\mathcal{Y} = \{L[M] : M \in \mathcal{Y}'\} \cup \{\lambda\}$ where $\mathcal{Y}' = \{U' \sqcup V' : U' \leq X', V' \leq Y'$ and U', V' are reconcilable$\}$ is a non-empty ideal. If $\mathcal{Y}' = \emptyset$, then $\mathcal{Y} = \{\lambda\}$ and $S \sqcup T = \lambda \in \mathcal{Y}$. Let $S, T \in \mathcal{Y}$ be reconcilable, say $S = L[S']$ and $T = L[T']$. Consequently, $S', T' \in \mathcal{Y}'$, and the reconcilability of S', T' follows from the reconcilability of S, T. We know that $S' \sqcup T' \in \mathcal{Y}'$ which means that $S \sqcup T = L[S'] \sqcup L[T'] = L[S' \sqcup T'] \in \mathcal{Y}$ holds. □

5.2 Minimality

We will investigate whether the generalised Armstrong axioms form a minimal, sound and complete set of inference rules for the implication of FDs in the sense of Definition 3.7.

Lemma 5.17. *The reflexivity axiom is independent from* $\mathfrak{R} =$ {*subattribute axiom, extension rule, restricted join axiom, transitivity rule*}.

Proof. Let $N = L\{A\}, \Sigma = \emptyset$ and $\sigma = \{\lambda, L\{\lambda\}, L\{A\}\} \to \{\lambda\}$. We present $\Sigma^+_{\mathfrak{R}}$ by the following table where the row names denote the left-hand side $\mathcal{X}$, and the column names denote the right-hand side $\mathcal{Y}$ of an FD $\mathcal{X} \to \mathcal{Y}$. An FD $\mathcal{X} \to \mathcal{Y}$ belongs to $\Sigma^+_{\mathfrak{R}}$ if and only if the entry at row $\mathcal{X}$ and column $\mathcal{Y}$ is a cross $\times$.

	$\{\lambda\}$	$\{L\{\lambda\}\}$	$\{L\{A\}\}$	$\{\lambda, L\{\lambda\}\}$	$\{\lambda, L\{A\}\}$	$\{L\{\lambda\}, L\{A\}\}$	$\{\lambda, L\{\lambda\}, L\{A\}\}$
$\{\lambda\}$	×						
$\{L\{\lambda\}\}$	×	×		×			
$\{L\{A\}\}$	×	×	×	×	×	×	×
$\{\lambda, L\{\lambda\}\}$	×	×		×			
$\{\lambda, L\{A\}\}$	×	×	×	×	×	×	×
$\{L\{\lambda\}, L\{A\}\}$	×	×	×	×	×	×	×
$\{\lambda, L\{\lambda\}, L\{A\}\}$							

We can see that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, as $\{\lambda\} \subseteq \{\lambda, L\{\lambda\}, L\{A\}\}$ we conclude that σ can be inferred from Σ using the reflexivity axiom. □

Lemma 5.18. *The subattribute axiom is independent from* $\mathfrak{R} =$ {*reflexivity axiom, extension rule, restricted join axiom, transitivity rule*}.

Proof. Let $N = L(A), \Sigma = \emptyset$ and $\sigma = \{L(A)\} \rightarrow \{\lambda\}$. The following table represents $\Sigma^+_{\mathfrak{R}}$.

	$\{\lambda\}$	$\{L(A)\}$	$\{\lambda, L(A)\}$
$\{\lambda\}$	×		
$\{L(A)\}$		×	
$\{\lambda, L(A)\}$	×	×	×

We can see that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, as $\lambda \leq L(A)$ we conclude that σ can be inferred from Σ using the subattribute axiom. □

Lemma 5.19. *The extension rule is independent from* $\mathfrak{R}$ =*{reflexivity axiom, subattribute axiom, restricted join axiom, transitivity rule}.*

Proof. Let $N = L(A), \Sigma = \emptyset$ and $\sigma = \{L(A)\} \rightarrow \{\lambda, L(A)\}$. The following table represents $\Sigma^+_{\mathfrak{R}}$.

	$\{\lambda\}$	$\{L(A)\}$	$\{\lambda, L(A)\}$
$\{\lambda\}$	×		
$\{L(A)\}$	×	×	
$\{\lambda, L(A)\}$	×	×	×

We can see that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, as $\{L(A)\} \rightarrow \{\lambda\} \in \Sigma^+_{\mathfrak{R}}$ we conclude that σ can be inferred from Σ using the extension rule and $\mathfrak{R}$. □

Lemma 5.20. *The restricted join axiom is independent from* $\mathfrak{R}$ =*{reflexivity axiom, subattribute axiom, extension rule, transitivity rule}.*

Proof. Let $N = L(A, B), \Sigma = \emptyset$ and $\sigma = \{L(A), L(B)\} \rightarrow \{L(A, B)\}$. We compute $\Sigma^+_{\mathfrak{R}}$ by the tables

	$\{\lambda\}$	$\{L(A)\}$	$\{L(B)\}$	$\{L(A,B)\}$	$\{\lambda, L(A)\}$	$\{\lambda, L(B)\}$	$\{\lambda, L(A,B)\}$	$\{L(A), L(B)\}$
$\{\lambda\}$	×							
$\{L(A)\}$	×	×			×			
$\{L(B)\}$	×		×			×		
$\{L(A,B)\}$	×	×	×	×	×	×	×	×
$\{\lambda, L(A)\}$	×	×			×			
$\{\lambda, L(B)\}$	×		×			×		
$\{\lambda, L(A,B)\}$	×	×	×	×	×	×	×	×
$\{L(A), L(B)\}$	×	×	×		×	×		×
$\{L(A), L(A,B)\}$	×	×	×	×	×	×	×	×
$\{L(B), L(A,B)\}$	×	×	×	×	×	×	×	×
$\{\lambda, L(A), L(B)\}$	×	×	×		×	×		×
$\{\lambda, L(A), L(A,B)\}$	×	×	×	×	×	×	×	×
$\{\lambda, L(B), L(A,B)\}$	×	×	×	×	×	×	×	×
$\{L(A), L(B), L(A,B)\}$	×	×	×	×	×	×	×	×
$\{\lambda, L(A), L(B), L(A,B)\}$	×	×	×	×	×	×	×	×

and

	$\{L(A), L(A,B)\}$	$\{L(B), L(A,B)\}$	$\{\lambda, L(A), L(B)\}$	$\{\lambda, L(A), L(A,B)\}$
$\{\lambda\}$				
$\{L(A)\}$				
$\{L(B)\}$				
$\{L(A,B)\}$	×	×	×	×
$\{\lambda, L(A)\}$				
$\{\lambda, L(B)\}$				
$\{\lambda, L(A,B)\}$	×	×	×	×
$\{L(A), L(B)\}$			×	
$\{L(A), L(A,B)\}$	×	×	×	×
$\{L(B), L(A,B)\}$	×	×	×	×
$\{\lambda, L(A), L(B)\}$			×	
$\{\lambda, L(A), L(A,B)\}$	×	×	×	×
$\{\lambda, L(B), L(A,B)\}$	×	×	×	×
$\{L(A), L(B), L(A,B)\}$	×	×	×	×
$\{\lambda, L(A), L(B), L(A,B)\}$	×	×	×	×

and

	$\{\lambda, L(B), L(A,B)\}$	$\{L(A), L(B), L(A,B)\}$	$\{\lambda, L(A), L(B), L(A,B)\}$
$\{\lambda\}$			
$\{L(A)\}$			
$\{L(B)\}$			
$\{L(A,B)\}$	×	×	×
$\{\lambda, L(A)\}$			
$\{\lambda, L(B)\}$			
$\{\lambda, L(A,B)\}$	×	×	×
$\{L(A), L(B)\}$			
$\{L(A), L(A,B)\}$	×	×	×
$\{L(B), L(A,B)\}$	×	×	×
$\{\lambda, L(A), L(B)\}$			
$\{\lambda, L(A), L(A,B)\}$	×	×	×
$\{\lambda, L(B), L(A,B)\}$	×	×	×
$\{L(A), L(B), L(A,B)\}$	×	×	×
$\{\lambda, L(A), L(B), L(A,B)\}$	×	×	×

We can see that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, as $L(A)$ and $L(B)$ are reconcilable, we conclude that σ can be inferred from Σ using the restricted join axiom. □

Lemma 5.21. *The transitivity rule is independent from* $\mathfrak{R}$ *={reflexivity axiom, subattribute axiom, extension rule, restricted join axiom}.*

Proof. Let $N = L\{A\}, \Sigma = \emptyset$ and $\sigma = \{L\{\lambda\}, L\{A\}\} \to \{\lambda\}$. The following table represents $\Sigma^+_{\mathfrak{R}}$.

	$\{\lambda\}$	$\{L\{\lambda\}\}$	$\{L\{A\}\}$	$\{\lambda, L\{\lambda\}\}$	$\{\lambda, L\{A\}\}$	$\{L\{\lambda\}, L\{A\}\}$	$\{\lambda, L\{\lambda\}, L\{A\}\}$
$\{\lambda\}$	×						
$\{L\{\lambda\}\}$	×	×		×			
$\{L\{A\}\}$	×	×	×		×	×	
$\{\lambda, L\{\lambda\}\}$	×	×		×			
$\{\lambda, L\{A\}\}$	×		×		×		
$\{L\{\lambda\}, L\{A\}\}$		×	×			×	
$\{\lambda, L\{\lambda\}, L\{A\}\}$	×	×	×	×	×	×	×

We can see that $\sigma \notin \Sigma^+_{\mathfrak{R}}$. However, $\{L\{\lambda\}, L\{A\}\} \to \{\lambda\}$ can be inferred from $\{L\{\lambda\}, L\{A\}\} \to \{L\{\lambda\}\}, \{L\{\lambda\}\} \to \{\lambda\} \in \Sigma^+_{\mathfrak{R}}$ by the transitivity rule. We conclude that σ can be inferred from Σ using the transitivity rule and $\mathfrak{R}$. □

It is interesting to note that in every of the previous lemmata trivial FDs have been identified as witnesses for the independence of the respective inference rule, i.e., FDs that follow from the empty set of FDs specified.

The previous lemmata prove the following main result. It shows that there is no proper subset of the generalised Armstrong axioms which forms also a complete set of inference rules for the implication of FDs.

Theorem 5.22. *The generalised Armstrong axioms form a minimal, sound and complete set of inference rules for the implication of FDs in the presence of records, lists, sets and multisets.* □

5.3 Minimal Axiomatisations for all Combinations

Theorem 5.22 captures the implication of FDs in the presence of all types considered in this section. It is now interesting to ask what the minimal axiomatisations for all subsets of the set of all four types are. The extended abstract [139] presented an axiomatisation of FDs in the presence of records and sets. The generalised Armstrong axioms from Definition 5.4 are in fact already all needed to capture implication in the presence of these two types. The proofs in Section 5.2 show now that this axiomatisation is also minimal.

Multisets behave similar to sets in the sense that elements of a multiset are not ordered. Values on the join of two subattributes are therefore not determined by the values on the individual subattributes. An axiomatisation of FDs in the presence of records and multisets is again given by the generalised Armstrong axioms. Moreover, the proofs in Section 5.2 are completely analogous, if set-valued attributes are replaced by multiset-valued attributes. Therefore, the axiomatisation is even minimal.

The situation becomes easier if only records and lists are considered. Here the join axiom is valid in unrestricted form due to the fact that elements of a list are totally ordered. This means that it is sufficient to consider FDs of the form $X \to Y$ where X and Y are subattributes of some nested attribute N. Sets of subattributes are no longer required as all elements of the set can be joined without changing the semantics. As shown

in Chapter 3 the implication of FDs can be captured by a generalisation of Armstrong's original axioms. We can therefore summarise the results on the axiomatisability of FDs in the presence of various type constructors in the following theorem.

Theorem 5.23. *The Armstrong axioms, i.e.,*

$$\frac{}{X \to Y}\, Y \leq X, \quad \frac{X \to Y}{X \to X \sqcup_N Y}, \quad \frac{X \to Y,\ Y \to Z}{X \to Z}$$

form a minimal, sound and complete set of inference rules for the implication of FDs in the presence of records, and in the presence of records and lists.

Let $\mathcal{T}$ be any non-empty subset of {lists, sets, multisets} apart from {lists}. The generalised Armstrong axioms, i.e.,

$$\frac{}{\mathcal{X} \to \mathcal{Y}}\, \mathcal{Y} \subseteq \mathcal{X}, \quad \frac{}{\{X\} \to \{Y\}}\, Y \leq X, \quad \frac{\mathcal{X} \to \mathcal{Y}}{\mathcal{X} \to \mathcal{X} \cup \mathcal{Y}},$$

$$\frac{}{\{X, Y\} \to \{X \sqcup_N Y\}}\, X, Y \text{ reconcilable}, \quad \frac{\mathcal{X} \to \mathcal{Y},\ \mathcal{Y} \to \mathcal{Z}}{\mathcal{X} \to \mathcal{Z}},$$

form a minimal, sound and complete set of inference rules for the implication of FDs in the presence of records and $\mathcal{T}$. □

In terms of Figure 1.1, Theorem 5.23 extends the knowledge on the class of FDs and the problem of axiomatisability along the data type dimension covering all combinations of record-, list-, set- and multiset-valued attributes. We would like to do the same extension for the implication problem of FDs.

5.4 Implication Problem

In view of Theorem 5.15, $\Sigma \models \mathcal{X} \to \mathcal{Y}$ holds if and only if $\Sigma \vdash_{\mathfrak{R}} \mathcal{X} \to \mathcal{Y}$ holds where $\mathfrak{R}$ are the generalised Armstrong axioms from Definition 5.4. Given some set Σ one can enumerate all FDs derivable from it. However, the enumeration algorithm is time consuming and therefore impractical. We will now present a provably-correct membership algorithm and prove that it works efficiently, i.e., in polynomial time in the number of subattributes of the underlying nested attribute and the number of FDs given.

5.4.1 The Closure

Similar to the RDM [29] and similar to the case of list-valued attributes in Chapter 3 we introduce the notion of a closure for a set of nested attributes with respect to a given set of FDs. Please note that this notion already played an important role in proving Theorem 5.15. The closure is defined with respect to the set $\mathfrak{R}$ of the generalised Armstrong axioms from Definition 5.4.

Definition 5.24. Let $N \in \mathcal{NA}$, $\mathcal{X} \subseteq Sub(N)$ a set of subattributes of N, and Σ a set of FDs on N. The *closure* $\mathcal{X}^+ \subseteq Sub(N)$ of $\mathcal{X}$ with respect to Σ is $\mathcal{X}^+ = \{Z : \mathcal{X} \to \{Z\} \in \Sigma^+\}$. □

According to Theorem 5.15 the closure $\mathcal{X}^+$ of $\mathcal{X}$ is therefore the set of all nested attributes which are functionally determined by $\mathcal{X}$ with respect to a given set Σ of FDs. The computation of $\mathcal{X}^+$ is sufficient for deciding whether $\Sigma \models \mathcal{X} \to \mathcal{Y}$ holds.

Lemma 5.25. *Let $N \in \mathcal{NA}$, and Σ a set of FDs on N. Then*

$$\mathcal{X} \to \mathcal{Y} \in \Sigma^+ \quad \textit{if and only if} \quad \mathcal{Y} \subseteq \mathcal{X}^+ \quad .$$

Proof. If $\mathcal{X} \to \mathcal{Y} \in \Sigma^+$, then $\mathcal{X} \to \{Y\} \in \Sigma^+$ for all $Y \in \mathcal{Y}$ by the subset rule. This means all $Y \in \mathcal{Y}$ are elements of $\mathcal{X}^+$, i.e., $\mathcal{Y} \subseteq \mathcal{X}^+$.

Assuming that every $Y \in \mathcal{Y}$ also satisfies $Y \in \mathcal{X}^+$ implies that $\mathcal{X} \to \{Y\} \in \Sigma^+$ for all $Y \in \mathcal{Y}$. We infer that $\mathcal{X} \to \mathcal{Y} \in \Sigma^+$ by the derivability of the union rule. □

Let $\mathcal{X}, \mathcal{Y} \subseteq Sub(N)$. We call $\mathcal{X}$ a *generalised subset* of $\mathcal{Y}$, denoted by $\mathcal{X} \subseteq_{\text{gen}} \mathcal{Y}$, if and only if for every $X \in \mathcal{X}$ there is some $Y \in \mathcal{Y}$ with $X \leq Y$ (Hoare-ordering). Note that $\subseteq_{\text{gen}}$ is a pre-order (reflexive, transitive) on the powerset $\mathcal{P}(Sub(N))$ of $Sub(N)$. The distinct sets $\mathcal{X} = \{L[A], L[\lambda]\}$ and $\mathcal{Y} = \{L[A]\}$ are generalised subsets of one another, i.e., $\subseteq_{\text{gen}}$ is not symmetric.

The projection of any tuple on a superattribute always determines the projection of this tuple on each of its subattributes. It is therefore sufficient to consider only maximal subattributes with respect to $\leq$. The set of all maximal elements of some $\leq$-ideal $\mathcal{X} \subseteq Sub(N)$ is formally defined as $\mathcal{X}_{\max} = \{X \in \mathcal{X} : \forall Z \in \mathcal{X}. X \leq Z \text{ implies } X = Z\}$. It is an immediate consequence of Lemma 5.26 that $\mathcal{Y} \subseteq \mathcal{X}^+$ if and only if $\mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}^+_{\max}$.

Lemma 5.26. *Let $N \in \mathcal{NA}$, and $\mathcal{X} \subseteq Sub(N)$ an ideal with respect to $\leq$. For all $\mathcal{Y} \subseteq Sub(N)$ we have*

$$\mathcal{Y} \subseteq \mathcal{X} \quad \textit{if and only if} \quad \mathcal{Y} \subseteq_{gen} \mathcal{X}_{max} \quad .$$

Proof. We show the *only if* part first. Let $Y \in \mathcal{Y}$ be arbitrary. From $\mathcal{Y} \subseteq \mathcal{X}$ follows $Y \in \mathcal{X}$. Consequently there is some $Z \in \mathcal{X}_{\max}$ with $Y \leq Z$. This shows that $\mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}_{\max}$.

It remains to consider the *if* part. Let $Y \in \mathcal{Y}$ be arbitrary again. Since $\mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}_{\max}$, there is some $Z \in \mathcal{X}_{\max}$ with $Y \leq Z$. Since $\mathcal{X}_{\max} \subseteq \mathcal{X}$ we have $Z \in \mathcal{X}$. However, $\mathcal{X}$ is an ideal with respect to $\leq$, i.e., $Y \in \mathcal{X}$ holds as well. This shows $\mathcal{Y} \subseteq \mathcal{X}$. □

5.4.2 Units of Nested Attributes

In order to solve the implication problem for FDs on some nested attribute N we will split N into mutually reconcilable subattributes N_i and solve the projected implication problems on the N_i simultaneously. The idea is to choose the units N_i of N such that for all subattributes $U, V \in Sub(N)$ we have that U and V are reconcilable if and only if for all units N_i of N the subattributes $U \sqcap N_i$ and $V \sqcap N_i$ are comparable with respect to $\leq$. This motivates the following definition.

Definition 5.27. Let $N \in \mathcal{NA}$. A nested attribute $N_i \in \mathcal{NA}$ is a *unit of* N if and only if

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

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

The property that two subattributes $U, V \in Sub(N)$ are reconcilable is not transitive: if $N = L(K\{M(A,B)\}, C)$ and $U = L(K\{M(A,\lambda)\},\lambda)$, $W = L(\lambda, C)$ and $V = L(K\{M(\lambda,B)\},\lambda)$, then U and W are reconcilable, W and V are reconcilable, but U and V are not reconcilable. In fact, $U, V \in Sub(L(K\{M(A,B)\},\lambda))$, but they are incomparable with respect to $\leq$.

EXAMPLE 5.5. Let $N = L_1(L_2\langle L_3(A,B)\rangle, L_4[L_5(C, L_6\langle D\rangle)], L_7(E, L_8\{L_9(F,G,H)\}))$. The units of N are

- $L_1(L_2\langle L_3(A,B)\rangle, \lambda, L_7(\lambda,\lambda))$,
- $L_1(\lambda, L_4[L_5(C,\lambda)], L_7(\lambda,\lambda))$,
- $L_1(\lambda, L_4[L_5(\lambda, L_6\langle D\rangle)], L_7(\lambda,\lambda))$,
- $L_1(\lambda,\lambda, L_7(E,\lambda))$ and
- $L_1(\lambda,\lambda, L_7(\lambda, L_8\{L_9(F,G,H)\}))$.

Clearly $L_1(\lambda,\lambda, L_7(\lambda, L_8\{L_9(\lambda,G,H)\}))$ also has properties 1. and 2. of Definition 5.27, but is not maximal with respect $\leq$. □

Next we give an inductive characterisation of units.

Lemma 5.28. *Let* $N \in \mathcal{NA}$. *Then* $\mathcal{U}(N) = \bigcup_{i=1}^{k} \{L(\lambda_{N_1}, \ldots, M, \ldots, \lambda_{N_k}) : M \in \mathcal{U}(N_i)$ *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.*

Proof. We prove the equivalence of this definition and Definition 5.27 by induction on the structure of the nested attribute N.

If $N = \lambda$ or $N = A$ is a flat attribute, then $X \leq Y$ or $Y \leq X$ for all $X, Y \in Sub(N)$, i.e., N is its only unit.

If $N = L\langle M\rangle$ or $N = L\{M\}$, then $X \leq Y$ or $Y \leq X$ for all reconcilable $X, Y \in Sub(N)$. This follows directly from the definition of reconcilable attributes. Consequently, N is again its only unit.

Let $N = L[M]$. We show that $L[M'] \in \mathcal{U}(N)$, if $M' \in \mathcal{U}(M)$. Clearly, $L[M'] \leq N$ as $M' \leq M$. Let $X, Y \leq L[M']$. If $X = \lambda$ or $Y = \lambda$, then $X \leq Y$ or $Y \leq X$. If $X = L[X']$ and $Y = L[Y']$ are reconcilable, then X' and Y' are reconcilable as well. Consequently, $X' \leq Y'$ or $Y' \leq X'$, and therefore also $X \leq Y$ or $Y \leq X$. The maximality of $L[M']$ follows from the maximality of M'. If $N = L(N_1, \ldots, N_k)$ and $N_i = \lambda_{N_i}$ for every $i = 1, \ldots, k$, then $\mathcal{U}(N) = \{N\}$ as well.

It remains to consider the case where $N = L(N_1, \ldots, N_k)$ and $N \neq \lambda_N$. We show that $L(M) \in \mathcal{U}(N)$, if $M \in \mathcal{U}(N_i)$ and $N_i \neq \lambda_{N_i}$. We know that $M \neq \lambda_{N_i}$ since $N \neq \lambda_N$. First $L(M) \leq L(N_i) \leq N$ since $M \leq N_i$ holds. Suppose now there are reconcilable $X, Y \leq L(M)$. Then $X = L(X'), Y = L(Y')$ with reconcilable $X', Y' \leq M$. It follows that $X' \leq Y'$ or $Y' \leq X'$ holds. Consequently, $X \leq Y$ or $Y \leq X$ holds as well. It remains to show the maximality of $L(M)$. M itself is maximal, i.e., all $L(M')$ with $M \leq M' \leq N_i$ and $M \neq M'$ do not satisfy the second property in Definition 5.27. Suppose some $L(M, K) \leq N$ with $K \neq \lambda_{N_j}$ and $K \leq N_j$ for $i \neq j$. Clearly, $L(M), L(K) \leq L(M, K)$ are reconcilable, but they are $\leq$-incomparable as $L(M) \neq \lambda_N$ and $L(K) \neq \lambda_N$. It follows that $L(M)$ is indeed $\leq$-maximal with the first two properties. □

Every nested attribute is the join of its mutually reconcilable units.

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.*

Proof. The proof is done in both cases by induction on N using Lemma 5.28. We show first that $N = \bigsqcup\{M \mid M \in \mathcal{U}(N)\}$. In the cases where $N = \lambda, N = A$ is a flat attribute, $N = L\{M\}$, $N = L\langle M\rangle$ and $N = L(N_1, \ldots, N_k)$ with $N_i = \lambda_{N_i}$ for $i = 1, \ldots, k$, we have that $\mathcal{U}(N)$ is a singleton containing N. Therefore the statement is obvious. Let $N = L(N_1, \ldots, N_k)$ where $N_i \neq \lambda_{N_i}$ for some i holds. The hypothesis is that $N_i = \bigsqcup\{M \mid M \in \mathcal{U}(N_i)\}$ holds for all $i = 1, \ldots, k$. This implies

$$\begin{aligned} N = L(N_1, \ldots, N_k) &= \bigsqcup_{i=1}^{k} L(\lambda_{N_1}, \ldots, N_i, \ldots, \lambda_{N_k}) \\ &= \bigsqcup_{i=1}^{k} L(\lambda_{N_1}, \ldots, \bigsqcup\{M \mid M \in \mathcal{U}(N_i)\}, \ldots, \lambda_{N_k}) \\ &= \bigsqcup_{i=1}^{k} \bigsqcup\{L(\lambda_{N_1}, \ldots, M, \ldots, \lambda_{N_k}) \mid M \in \mathcal{U}(N_i)\} \\ &= \bigsqcup\{L(\lambda_{N_1}, \ldots, M, \ldots, \lambda_{N_k}) \mid L(\lambda_{N_1}, \ldots, M, \ldots, \lambda_{N_k}) \in \mathcal{U}(N)\}. \end{aligned}$$

If $N = L[M]$, then $M = \bigsqcup\{M' \mid M' \in \mathcal{U}(M)\}$ and therefore

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

and this concludes the proof for the first statement.

For the second statement there is nothing to show when $N = \lambda, N = A$ is a flat attribute, $N = L\{M\}$, $N = L\langle M\rangle$ or $N = L(N_1, \ldots, N_k)$ with $N_i = \lambda_{N_i}$ for $i = 1, \ldots, k$. The statement is trivial if $U = \lambda$ or $V = \lambda$. Let $N = L[M]$, $N_1 = L[N_1']$, $N_2 = L[N_2']$ with $N_1, N_2 \in \mathcal{U}(N)$ and $N_1 \neq N_2$. For $U = L[U']$ and $V = L[V']$ with $U' \leq N_1'$ and $V' \leq N_2'$ the reconcilability of U' and V' follows. Consequently, U and V are reconcilable too. Let $N = L(N_1, \ldots, N_k)$ with $N_i \neq \lambda_{N_i}$ for some i. Let $M_1 = L(M), M_2 = L(M') \in \mathcal{U}(N)$ be distinct with $M \in \mathcal{U}(N_i)$ and $M' \in \mathcal{U}(N_j)$. Moreover, let $U = L(U'), V = L(V')$ with $U' \leq M$ and $V' \leq M'$. If $i \neq j$, then U and V are reconcilable since M_1 and M_2 are reconcilable. If $i = j$, then $M \neq M'$ since $M_1 \neq M_2$, and $M, M' \in \mathcal{U}(N_i)$. This implies the reconcilability of U' and V', and therefore also the reconcilability of U and V. This shows the second statement. □

It follows that two subattributes U and V of N are reconcilable precisely if for all units N_i of N the subattributes $U \sqcap N_i$ and $V \sqcap N_i$ are comparable with respect to $\leq$.

5.4.3 Computing the Closure

We are finally prepared to present the algorithm. Lemma 5.25 and Lemma 5.26 reduced the implication problem $\Sigma \models \mathcal{X} \to \mathcal{Y}$ to the problem of computing $\mathcal{X}^{+}_{\max}$, the set of maximal subattributes which are functionally determined by $\mathcal{X}$. Given some set $\mathcal{X} \subseteq Sub(N)$ of nested attributes, the function $\max(\mathcal{X})$ returns all maximal elements of $\mathcal{X}$ with respect to $\leq$. Moreover, if $\mathcal{U}(N) = \{N_1, \ldots, N_k\}$, then $\mathcal{X}_i = \{X \sqcap N_i : X \in \mathcal{X}\}$ for $i = 1, \ldots, k$.

Algorithm 5.4.1 (Nested Attribute Closure)

Input: $\boldsymbol{N \in \mathcal{N}\mathrm{A}}$, $\boldsymbol{\mathcal{X} \subseteq Sub(N)}$, set $\boldsymbol{\Sigma}$ of FDs on $\boldsymbol{N}$

Output: $\boldsymbol{\mathcal{X}^{\mathrm{alg}}_{\max}}$

Method:

VAR $\mathcal{X}_i^{\mathrm{new}}, \mathcal{X}_i^{\mathrm{old}} \subseteq Sub(N)$ for $i = 1, \ldots, k$, $N_1, \ldots, N_k \in Sub(N)$;

(1) Compute $\mathcal{U}(N) = \{N_1, \ldots, N_k\}$;
(2) **FOR** $i = 1$ **TO** k **DO** $\mathcal{X}_i^{\mathrm{new}} := \max(\{X \sqcap N_i : X \in \mathcal{X}\})$;
(3) **REPEAT**
(4) **FOR** $i = 1$ **TO** k **DO** $\mathcal{X}_i^{\mathrm{old}} := \mathcal{X}_i^{\mathrm{new}}$;
(5) **FOR each** $\mathcal{U} \to \mathcal{V} \in \Sigma$ **DO**
(6) **IF** $\mathcal{U}_i \subseteq_{\mathrm{gen}} \mathcal{X}_i^{\mathrm{new}}$ for $i = 1, \ldots, k$ **THEN**
(7) **FOR** $i = 1$ **TO** k **DO** $\mathcal{X}_i^{\mathrm{new}} := \max(\mathcal{X}_i^{\mathrm{new}} \cup \mathcal{V}_i)$;
(8) **ENDIF**;
(9) **ENDDO**;
(10) **UNTIL** $\mathcal{X}_i^{\mathrm{new}} = \mathcal{X}_i^{\mathrm{old}}$ for $i = 1, \ldots, k$;
(11) $\mathcal{X}^{\mathrm{alg}}_{\max} := \{X_1 \sqcup \ldots \sqcup X_k : X_i \in \mathcal{X}_i^{\mathrm{new}}\}$;
(12) **RETURN**$(\mathcal{X}^{\mathrm{alg}}_{\max})$;

□

In order to illustrate Algorithm 5.4.1 we turn now to an example.

EXAMPLE 5.6. Suppose we are given the following input instance for Algorithm 5.4.1:

- $N = L_1(L_2\langle L_3(A, B)\rangle, L_4[L_5(C, L_6\langle D\rangle)], L_7(E, L_8\{L_9(F, G, H)\}))$ with the units from Example 5.5,
- Σ is given by
  - FD_1: $\{L_1(L_2\langle L_3(A)\rangle), L_1(L_2\langle L_3(B)\rangle, L_7(L_8\{L_9(F, H)\}))\} \to \{L_1(L_7(L_8\{L_9(F, G)\}))\}$,
  - FD_2: $\{L_1(L_2\langle L_3(B)\rangle, L_4[L_5(C)], L_7(E))\} \to \{L_1(L_4[L_5(L_6\langle\lambda\rangle)], L_7(L_8\{L_9(F, H)\}))\}$,
  - FD_3: $\{\lambda\} \to \{L_1(L_2\langle L_3(B)\rangle)\}$
- and $\mathcal{X} = \{L_1(L_2\langle L_3(A)\rangle, L_4[L_5(C)], L_7(E))\}$.

For the sets $\mathcal{X}_i^{\mathrm{new}}$ we compute $\mathcal{X}_1^{\mathrm{new}} = \{L_1(L_2\langle L_3(A)\rangle)\}$, $\mathcal{X}_2^{\mathrm{new}} = \{L_1(L_4[L_5(C)])\}$, and $\mathcal{X}_4^{\mathrm{new}} = \{L_1(L_7(E))\}$ and $\mathcal{X}_3^{\mathrm{new}} = \mathcal{X}_5^{\mathrm{new}} = \{\lambda_N\}$.

Consider the first run through the REPEAT loop (line (3) to line (10)) of Algorithm 5.4.1. For FD_1 and FD_2 there are no changes. Due to FD_3 we obtain

$$\mathcal{X}_1^{\mathrm{new}} = \{L_1(L_2\langle L_3(A)\rangle), L_1(L_2\langle L_3(B)\rangle)\}.$$

In the second run through the loop FD_2 gives

$$\mathcal{X}_3^{\text{new}} = \{L_1(L_4[L_5(L_6\langle\lambda\rangle)])\} \text{ and } \mathcal{X}_5^{\text{new}} = \{L_1(L_7(L_8\{L_9(F,H)\}))\}.$$

In the third run, FD_1 causes the update

$$\mathcal{X}_5^{\text{new}} = \{L_1(L_7(L_8\{L_9(F,G)\})), L_1(L_7(L_8\{L_9(F,H)\}))\}.$$

There are no further changes in the fourth run through the loop, i.e., the final values are

$$\begin{array}{ll} \mathcal{X}_1^{\text{new}} = \{L_1(L_2\langle L_3(A)\rangle), L_1(L_2\langle L_3(B)\rangle)\}, & \mathcal{X}_2^{\text{new}} = \{L_1(L_4[L_5(C)])\}, \\ \mathcal{X}_3^{\text{new}} = \{L_1(L_4[L_5(L_6\langle\lambda\rangle)])\}, & \mathcal{X}_4^{\text{new}} = \{L_1(L_7(E))\}, \\ \mathcal{X}_5^{\text{new}} = \{L_1(L_7(L_8\{L_9(F,G)\})), L_1(L_7(L_8\{L_9(F,H)\}))\}. & \end{array}$$

It remains to compute $\mathcal{X}_{\max}^{\text{alg}}$:

$$\begin{array}{l} L_1(L_2\langle L_3(A,\lambda)\rangle, L_4[L_5(C, L_6\langle\lambda\rangle)], L_7(E, L_8\{L_9(F,G,\lambda)\})), \\ L_1(L_2\langle L_3(\lambda,B)\rangle, L_4[L_5(C, L_6\langle\lambda\rangle)], L_7(E, L_8\{L_9(F,G,\lambda)\})), \\ L_1(L_2\langle L_3(A,\lambda)\rangle, L_4[L_5(C, L_6\langle\lambda\rangle)], L_7(E, L_8\{L_9(F,\lambda,H)\})), \\ L_1(L_2\langle L_3(\lambda,B)\rangle, L_4[L_5(C, L_6\langle\lambda\rangle)], L_7(E, L_8\{L_9(F,\lambda,H)\})). \end{array}$$

which is the output of Algorithm 5.4.1. □

5.4.4 Correctness

We will now prove the correctness of Algorithm 5.4.1, i.e., $\mathcal{X}_{\max}^{\text{alg}} = \mathcal{X}_{\max}^{+}$. Recall that a set $\mathcal{X} \subseteq Sub(N)$ is called an *anti-chain* with respect to $\leq$ if and only if every two different elements of $\mathcal{X}$ are incomparable, i.e., for all $X, Y \in \mathcal{X}$ with $X \neq Y$ follows $X \not\leq Y$ and $Y \not\leq X$.

Lemma 5.30. *$\mathcal{X}_{max}^{+}$ and $\mathcal{X}_{max}^{alg}$ are both anti-chains with respect to $\leq$.*

Proof. It is obvious that the set $\mathcal{X}_{\max}$ of all $\leq$-maximal elements of any $\mathcal{X} \subseteq Sub(N)$ is an anti-chain with respect to $\leq$. It follows that $\mathcal{X}_{\max}^{+}$ is an anti-chain with respect to $\leq$, and it remains to show that $\mathcal{X}_{\max}^{\text{alg}}$ is an anti-chain, too. We proceed by induction on the number j of runs through the REPEAT loop of Algorithm 5.4.1 to show that $\mathcal{X}_i^{\text{new}}$ is an anti-chain for every $i = 1, \ldots, k$. If $j = 0$, then $\mathcal{X}_i^{\text{new}} = \max(\mathcal{X}_i)$ contains only the maximal elements of $\mathcal{X}_i$ and is therefore an anti-chain. Suppose that $\mathcal{X}_i^{\text{new}}$ is an anti-chain after the jth run through the REPEAT loop. Whenever the value of $\mathcal{X}_i^{\text{new}}$ is changed within the $j+1$-st run, then it is changed to $\max(\mathcal{X}_i^{\text{new}} \cup \mathcal{V}_i)$ consisting again of $\leq$-maximal elements only. Therefore, after the REPEAT loop has been aborted (before executing line (11) that is) $\mathcal{X}_i^{\text{new}}$ is an anti-chain for every i. It follows that $\mathcal{X}_{\max}^{\text{alg}} = \{X_1 \sqcup \ldots \sqcup X_k : X_i \in \mathcal{X}_i^{\text{new}}\}$ also forms an anti-chain. □

Moreover, if $\mathcal{X}, \mathcal{Y} \subseteq Sub(N)$ are two anti-chains, $\mathcal{X} \subseteq_{\text{gen}} \mathcal{Y}$ and $\mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}$, then $\mathcal{X} = \mathcal{Y}$.

Lemma 5.31. *Let $N \in \mathcal{NA}$ and $\mathcal{X}, \mathcal{Y} \subseteq Sub(N)$ be two anti-chains with respect to $\leq$. If $\mathcal{X} \subseteq_{gen} \mathcal{Y}$ and $\mathcal{Y} \subseteq_{gen} \mathcal{X}$, then $\mathcal{X} = \mathcal{Y}$.*

Proof. Let $Y \in \mathcal{Y}$. We show that $Y \in \mathcal{X}$ holds. Since $\mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}$, there is some $X \in \mathcal{X}$ with $Y \leq X$. Since $\mathcal{X} \subseteq_{\text{gen}} \mathcal{Y}$ holds as well, we find some $Y' \in \mathcal{Y}$ with $X \leq Y'$. We conclude by transitivity of $\subseteq_{\text{gen}}$ that $Y \leq Y'$ for $Y, Y' \in \mathcal{Y}$. As $\mathcal{Y}$ is an anti-chain with respect to $\leq$ we have to conclude that $Y = Y'$ and therefore also $Y = X$. It follows that $Y \in \mathcal{X}$ and, therefore, $\mathcal{Y} \subseteq \mathcal{X}$ holds. The inclusion $\mathcal{X} \subseteq \mathcal{Y}$ follows similar using that $\mathcal{X}$ is a $\leq$-anti-chain. It follows $\mathcal{X} = \mathcal{Y}$. □

Theorem 5.32. *Algorithm 5.4.1 is correct, i.e.,* $\mathcal{X}_{max}^{alg} = \mathcal{X}_{max}^{+}$.

Proof. According to Lemma 5.30 and Lemma 5.31 it is sufficient to show $\mathcal{X}_{\text{max}}^{\text{alg}} \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{+}$ and $\mathcal{X}_{\text{max}}^{+} \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{\text{alg}}$. We will first show that $\mathcal{X}_{\text{max}}^{\text{alg}} \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{+}$ holds.

We infer from the soundness of the reflexivity rule that $\mathcal{X} \subseteq \mathcal{X}^{+} = \{Z : \mathcal{X} \to \{Z\} \in \Sigma^{+}\}$ holds. Consequently, $\mathcal{X} \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{+}$ by Lemma 5.26. Since $X' \in \mathcal{X}_i$ implies $X' = X \sqcap N_i$ for some $X \in \mathcal{X}$ we have $X' \leq X$. This implies $\mathcal{X}_i \subseteq_{\text{gen}} \mathcal{X}$. We conclude $\mathcal{X}_i \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{+}$ by transitivity of $\subseteq_{\text{gen}}$. In particular, $\max(\mathcal{X}_i) \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{+}$. This implies $\mathcal{X}_i^{\text{new}} \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{+}$ for all i after line (2) of Algorithm 5.4.1 has been executed.

Suppose now that $\mathcal{X}_i^{\text{new}} \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{+}$ holds for all i within some j-th run. Furthermore suppose that $\mathcal{U} \to \mathcal{V} \in \Sigma$ and $\mathcal{U}_i \subseteq_{\text{gen}} \mathcal{X}_i^{\text{new}}$ for all i (otherwise nothing changes). It follows that $\mathcal{U}_i \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{+}$ for all i, and therefore $\mathcal{X} \to \mathcal{U}_i \in \Sigma^{+}$ for all i by Lemma 5.25 and Lemma 5.26. Lemma 5.29 shows that $U \in \mathcal{U}_i$ and $U' \in \mathcal{U}_j$ with $i \neq j$ are reconcilable. The following inference schema is therefore sound:

$$\dfrac{\dfrac{\dfrac{\mathcal{X} \to \mathcal{U}_i \;\; \mathcal{U}_i \to \{U\}}{\mathcal{X} \to \{U\}} \quad \dfrac{\mathcal{X} \to \mathcal{U}_j \;\; \mathcal{U}_j \to \{U'\}}{\mathcal{X} \to \{U'\}}}{\mathcal{X} \to \{U, U'\}} \quad \dfrac{}{\{U, U'\} \to \{U \sqcup U'\}}\, U,U' \text{are reconcilable}}{\mathcal{X} \to \{U \sqcup U'\}}$$

and we obtain $\mathcal{X} \to \overline{\mathcal{U}} \in \Sigma^{+}$ where $\overline{\mathcal{U}} = \{U_1 \sqcup \ldots \sqcup U_k : U_i \in \mathcal{U}_i\}$. We know further that $\mathcal{U} \subseteq \overline{\mathcal{U}}$ holds as $U \in \mathcal{U}$ can be represented as $U = U \sqcap N = U \sqcap (N_1 \sqcup \ldots \sqcup N_k) = U_1 \sqcup \ldots \sqcup U_k$ with $U_i \in \mathcal{U}_i$ using Lemma 5.29. The soundness of the subset rule implies therefore $\mathcal{X} \to \mathcal{U} \in \Sigma^{+}$. We assumed further that $\mathcal{U} \to \mathcal{V} \in \Sigma$ which leads to $\mathcal{X} \to \mathcal{V} \in \Sigma^{+}$ applying the transitivity rule. Moreover, $\mathcal{V}_i \subseteq_{\text{gen}} \mathcal{V}$ and we infer by the soundness of the following schema

$$\dfrac{\dfrac{}{\mathcal{X} \to \{X\}}\, \{X\} \subseteq \mathcal{X} \quad \dfrac{}{\{X\} \to \{Y\}}\, Y \leq X}{\mathcal{X} \to \{Y\}}$$

and the soundness of the union rule that $\mathcal{X} \to \mathcal{V}_i \in \Sigma^{+}$ holds for all i. Using Lemma 5.25 and Lemma 5.26 results in $\mathcal{V}_i \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{+}$. Since also $\mathcal{X}_i^{\text{new}} \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{+}$ holds, we have $\mathcal{X}_i^{\text{new}} \cup \mathcal{V}_i \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{+}$ and, in particular, $\max(\mathcal{X}_i^{\text{new}} \cup \mathcal{V}_i) \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{+}$. This induction shows that $\mathcal{X}_i^{\text{new}} \subseteq_{\text{gen}} \mathcal{X}_{\text{max}}^{+}$ holds for $i = 1, \ldots, k$ before executing line (11). Again, applying Lemma 5.25 and Lemma 5.26 gives $\mathcal{X} \to \mathcal{X}_i^{\text{new}} \in \Sigma^{+}$ for $i = 1, \ldots, k$. As before, $X \in \mathcal{X}_i^{\text{new}}$ and $Y \in \mathcal{X}_j^{\text{new}}$ are reconcilable. Repeatedly applying the inference schema

$$\dfrac{\dfrac{\dfrac{\mathcal{X} \to \mathcal{X}_i^{\text{new}} \;\; \mathcal{X}_i^{\text{new}} \to \{X\}}{\mathcal{X} \to \{X\}} \quad \dfrac{\mathcal{X} \to \mathcal{X}_j^{\text{new}} \;\; \mathcal{X}_j^{\text{new}} \to \{Y\}}{\mathcal{X} \to \{Y\}}}{\mathcal{X} \to \{X, Y\}} \quad \dfrac{}{\{X, Y\} \to \{X \sqcup Y\}}\, X,Y \text{are reconcilable}}{\mathcal{X} \to \{X \sqcup Y\}}$$

results in $\mathcal{X} \to \mathcal{X}^{\text{alg}}_{\text{max}} \in \Sigma^+$ with $\mathcal{X}^{\text{alg}}_{\text{max}} = \{X_1 \sqcup \ldots \sqcup X_k : X_i \in \mathcal{X}^{\text{new}}_i\}$. Lemmata 5.25 and 5.26 give $\mathcal{X}^{\text{alg}}_{\text{max}} \subseteq_{\text{gen}} \mathcal{X}^{+}_{\text{max}}$.

We will now show that $\mathcal{X}^{+}_{\text{max}} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$ holds as well. The definition of $\mathcal{X}^{+}_{\text{max}}$ depends on Σ. Consider therefore the chain

$$\Sigma = \Sigma_0 \subset \Sigma_1 \subset \cdots \subset \Sigma_s = \Sigma^+$$

where Σ_{j+1} results from Σ_j by application of exactly one of the inference rules among the generalised Armstrong axioms from Theorem 5.15. We will use induction on j to show the following:

$$\text{if } \mathcal{Y} \to \mathcal{Z} \in \Sigma_j \text{ and } \mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}, \text{ then } \mathcal{Z} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}. \tag{13}$$

We can then conclude for $j = s$ that $\mathcal{Z} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$ follows from $\mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$ and $\mathcal{Y} \to \mathcal{Z} \in \Sigma^+$. Using $\mathcal{Y} = \mathcal{X}$ and $\mathcal{Z} = \mathcal{X}^{+}_{\text{max}}$ gives then $\mathcal{X}^{+}_{\text{max}} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$ because $\mathcal{X} \to \mathcal{X}^{+}_{\text{max}} \in \Sigma^+$ and $\mathcal{X} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$ hold.

It remains to show (13). Let $j = 0, \mathcal{Y} \to \mathcal{Z} \in \Sigma$ and $\mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$. This implies that $\mathcal{Y}_i \subseteq_{\text{gen}} \left(\mathcal{X}^{\text{alg}}_{\text{max}}\right)_i = \mathcal{X}^{\text{new}}_i$ for all i after executing line (11) of Algorithm 5.4.1. Since $\mathcal{Y} \to \mathcal{Z} \in \Sigma$ and $\mathcal{Y}_i \subseteq_{\text{gen}} \mathcal{X}^{\text{new}}_i$ for all i we have $\mathcal{Z}_i \subseteq_{\text{gen}} \mathcal{X}^{\text{new}}_i$ for all i due to line (7) of Algorithm 5.4.1. This implies $\mathcal{Z} \subseteq \overline{\mathcal{Z}} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$.

Let now be $j > 0$. Then $\Sigma_j - \Sigma_{j-1}$ contains exactly one $\mathcal{Y} \to \mathcal{Z}$ which has been inferred by using one of the generalised Armstrong axioms from Theorem 5.15. We distinguish therefore between five different cases.

- $\mathcal{Y} \to \mathcal{Z}$ has been inferred using the reflexivity axiom. Then $\mathcal{Z} \subseteq \mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$ holds.
- $\mathcal{Y} \to \mathcal{Z}$ has been inferred using the subattribute axiom. This leaves us with $\mathcal{Y} = \{Y\}, \mathcal{Z} = \{Z\}$ and $Z \leq Y$. It follows again that $\mathcal{Z} \subseteq_{\text{gen}} \mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$ holds.
- $\mathcal{Y} \to \mathcal{Z}$ has been inferred using the extension rule. In this case $\mathcal{Z} = \mathcal{Y} \cup \mathcal{W}$ with $\mathcal{Y} \to \mathcal{W} \in \Sigma_{j-1}$. From $\mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$ follows $\mathcal{W} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$ by hypothesis and therefore $\mathcal{Z} = \mathcal{Y} \cup \mathcal{W} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$.
- $\mathcal{Y} \to \mathcal{Z}$ has been inferred using the transitivity rule. Then there are $\mathcal{Y} \to \mathcal{W}, \mathcal{W} \to \mathcal{Z} \in \Sigma_{j-1}$. From $\mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$ follows $\mathcal{W} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$ by hypothesis ($\mathcal{Y} \to \mathcal{W} \in \Sigma_{j-1}$) and subsequently $\mathcal{Z} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$ by hypothesis again ($\mathcal{W} \to \mathcal{Z} \in \Sigma_{j-1}$).
- $\mathcal{Y} \to \mathcal{Z}$ has been inferred using the restricted join axiom. This leaves us with $\mathcal{Y} = \{Y, W\}, \mathcal{Z} = \{Y \sqcup W\}$ where Y and W are reconcilable. From $\mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$ follows $Y \sqcap N_i \leq Z_i$ and $W \sqcap N_i \leq Z'_i$ for some $Z_i, Z'_i \in \mathcal{X}^{\text{new}}_i$ for $i = 1, \ldots, k$. Since Y and W are reconcilable it follows that $Y \sqcap N_i$ and $W \sqcap N_i$ are reconcilable for $i = 1, \ldots, k$. Consequently, $Y \sqcap N_i \leq W \sqcap N_i$ or $W \sqcap N_i \leq Y \sqcap N_i$. Moreover, $(Y \sqcup W) \sqcap N_i = W \sqcap N_i \leq Z'_i$ or $(Y \sqcup W) \sqcap N_i = Y \sqcap N_i \leq Z_i$ for $i = 1, \ldots, k$. Using Lemma 5.29 we have

$$\begin{aligned} Y \sqcup W &= (Y \sqcup W) \sqcap N \\ &= (Y \sqcup W) \sqcap (N_1 \sqcup \ldots \sqcup N_k) \\ &= ((Y \sqcup W) \sqcap N_1) \sqcup \ldots \sqcup ((Y \sqcup W) \sqcap N_k) \\ &\leq X_1 \sqcup \ldots \sqcup X_k \in \mathcal{X}^{\text{alg}}_{\text{max}} \end{aligned}$$

  for some $X_i \in \mathcal{X}^{\text{new}}_i$. This means $Y \sqcup W \leq X$ for some $X \in \mathcal{X}^{\text{alg}}_{\text{max}} = \{X_1 \sqcup \ldots \sqcup X_k : X_i \in \mathcal{X}^{\text{new}}_i\}$. We conclude that $\mathcal{Z} \subseteq_{\text{gen}} \mathcal{X}^{\text{alg}}_{\text{max}}$.

This concludes the proof. □

EXAMPLE 5.7. We continue Example 5.6. According to Theorem 5.32 we know that

- $\Sigma \not\models \mathcal{X} \rightarrow \{Z = L_1(L_2\langle L_3(A, \lambda)\rangle, \lambda, L_7(\lambda, L_8\{L_9(F, G, H)\}))\}$ since Z does not have any superattribute in $\mathcal{X}^{\text{alg}}_{\text{max}}$, but
- $\Sigma \models \mathcal{X} \rightarrow \{L_1(L_2\langle L_3(A, \lambda)\rangle, \lambda, L_7(\lambda, L_8\{L_9(F, G, \lambda)\})); L_1(L_2\langle L_3(A, \lambda)\rangle, \lambda, L_7(\lambda, L_8\{L_9(F, \lambda, H)\}))\}$. □

EXAMPLE 5.8. We continue Example 5.2 of the retailer. Suppose N and Σ are given as in Example 5.2 and we want to find the closure of the subattribute Sales(List[Order(Cart⟨Article(Price)⟩)]) with respect to Σ. Using Algorithm 5.4.1 we obtain the following units of N:

- N_1 = Sales(Day),
- N_2 = Sales(List[Order(Cart⟨Article(Title,Description,Price)⟩)]),
- N_3 = Sales(List[Order(Customer(Name))]),
- N_4 = Sales(List[Order(Customer(Address))]),
- N_5 = Sales(List[Order(Customer(Payment))]),
- N_6 = Sales(List[Order(SubTotal)]),
- N_7 = Sales(Sold{Product(Item,CustName)}),
- N_8 = Sales(Total),
- N_9 = Sales(NOrd),
- N_{10} = Sales(NProd),
- N_{11} = Sales(NShip).

In this example all $\mathcal{X}_i^{\text{new}}$ are singletons and therefore written as subattributes. Initially we have $\mathcal{X}_2^{\text{new}}$ = Sales(List[Order(Cart⟨Article(Price)⟩)]) and $\mathcal{X}_1^{\text{new}} = \mathcal{X}_3^{\text{new}} = \cdots = \mathcal{X}_{11}^{\text{new}} = \lambda$. The first run through the REPEAT loop has the following sequence of updates (considering that the FDs in Σ are selected in the order they were presented in Example 5.2):

$$\mathcal{X}_6^{\text{new}} = N_6,\ \mathcal{X}_8^{\text{new}} = N_8,\ \mathcal{X}_9^{\text{new}} = N_9, \text{ and } \mathcal{X}_{10}^{\text{new}} = N_{10}.$$

The join of these subattributes and $\mathcal{X}_2^{\text{new}}$ is

Sales(List[Order(Cart⟨Article(Price)⟩,SubTotal)],Total,NOrd,NProd).

This shows that given the list of multisets of individual prices, one can determine the list of total values of the orders, the total value of sales, the total number of orders and the total number of products ordered. □

5.4.5 Complexity

We will now study the complexity of Algorithm 5.4.1 in terms of the size of the input. The input consists of some nested attribute N, some set Σ of FDs on N and a set $\mathcal{X}$ of subattributes of N. We define the size n of N as the number of subattributes of N, i.e., $n = \mid Sub(N) \mid$. This is a reasonable measure since we consider sets of subattributes in

general. The size s of Σ is simply defined as the number of its elements, i.e., $s = |\ \Sigma\ |$. The size of $\mathcal{X}$ is defined as $|\ \mathcal{X}\ |$. Note that we have $|\ \mathcal{X}\ | \leq n$ for all $\mathcal{X} \subseteq Sub(N)$. Lemma 5.28 suggests a strategy to compute the maximal units of N. This can obviously be done in time $\mathcal{O}(n)$. Algebraic operations such as meet, join and union on sets of subattributes of N can all be performed in time $\mathcal{O}(n)$ as well. The same holds for checking the subattribute relationship between two nested attributes.

Theorem 5.33. *In the presence of record-, list-, set- and multiset-valued attributes, the implication problem $\Sigma \models \mathcal{X} \to \mathcal{Y}$ for FDs on a nested attribute N can be solved in time $\mathcal{O}(n^4 \cdot s \cdot \min\{s, n\})$ where $n = |\ Sub(N)\ |$ and $s = |\ \Sigma\ |$.*

Proof. The termination of Algorithm 5.4.1 follows from the complexity analysis of the REPEAT loop below. As already noted, $\mathcal{U}(N) = \{N_1, \ldots, N_k\}$ is computed in time $\mathcal{O}(n)$. Let $n_i = |\ Sub(N_i)\ |$ for $i = 1, \ldots, k$. The sets $\mathcal{X}_i = \{X \sqcap N_i : X \in \mathcal{X}\}$ can be computed in time $\mathcal{O}(|\ \mathcal{X}\ | \cdot n_i)$ for every $i = 1, \ldots, k$. Since $\sum_{i=1}^{k} n_i \leq k \cdot \max_{i=1}^{k} n_i \leq k \cdot n \leq n^2$ and $|\ \mathcal{X}\ | \leq n$ it takes $\mathcal{O}(n^3)$ operations for this. In order to compute the maximal elements in every set $\mathcal{X}_i$ we need to check whether $X \leq Y$ holds for every pair $X, Y \in \mathcal{X}_i$. Doing this for every $i = 1, \ldots, k$ takes

$$\sum_{i=1}^{k} n_i^3 \leq k \cdot \left(\max_{i=1}^{k} n_i\right)^3 \leq k \cdot n^3 \leq n^4$$

steps. The initialisation (line (1) and (2)) takes therefore $\mathcal{O}(n^4)$ operations.

Since every dependency in Σ can contribute to a change of any $\mathcal{X}_i^{\text{new}}$ at most once, there are at most $s + 1$ runs through the REPEAT loop (line (3) to (10)). Every $\mathcal{X}_i^{\text{new}}$ can contain at most as many elements as a maximal anti-chain of $(Sub(N_i), \leq)$. If w_{N_i} denotes the cardinality of a maximal anti-chain of $(Sub(N_i), \leq)$, then $|\ \mathcal{X}_i^{\text{new}}\ | \leq w_{N_i}$. For all $i = 1, \ldots, k$ one can find maximal anti-chains $\mathcal{A}_{N_i}$ of $(Sub(N_i), \leq)$ which are pairwise disjoint, i.e., $\mathcal{A}_{N_i} \cap \mathcal{A}_{N_j} = \emptyset$ whenever $i \neq j$. Therefore, the sum over all w_{N_i} is at most w_N. Thus, there are also at most

$$\sum_{i=1}^{k} |\ \mathcal{X}_i^{\text{new}}\ | + 1 \leq \sum_{i=1}^{k} w_{N_i} + 1 \leq w_N + 1 \leq n + 1$$

runs through the REPEAT loop. We conclude that there are at most $\min\{s, n\} + 1$ passes through the loop. Inside the inner FOR loop (line (6)), the condition $\mathcal{U}_i \subseteq_{\text{gen}} \mathcal{X}_i^{\text{new}}$ has to be tested. It is satisfied if and only if for every element U in $\mathcal{U}_i$ there is some element X in $\mathcal{X}_i^{\text{new}}$ with $U \leq X$. This takes therefore at most $|\ \mathcal{U}_i\ | \cdot |\ \mathcal{X}_i^{\text{new}}\ | \cdot n_i$ steps for every i. The constraint $\mathcal{U}_i \subseteq_{\text{gen}} \mathcal{X}_i^{\text{new}}$ for all $i = 1, \ldots, k$ is therefore verified in at most

$$\sum_{i=1}^{k} (|\ \mathcal{U}_i\ | \cdot |\ \mathcal{X}_i^{\text{new}}\ | \cdot n_i) \leq \max_{i=1}^{k}\{|\ \mathcal{U}_i\ |\} \cdot \max_{i=1}^{k}\{|\ \mathcal{X}_i^{\text{new}}\ |\} \cdot \sum_{i=1}^{k} n_i \leq n^2 \cdot k \cdot \max_{i=1}^{k} n_i \leq n^4$$

steps, i.e., in time $\mathcal{O}(n^4)$. With arguments we have used before it is easy to see that the FOR loop in line (7) takes at most $\mathcal{O}(n^4)$ operations. Every run through the REPEAT loop takes therefore time in $\mathcal{O}(n^4 \cdot s)$ which leaves us with $\mathcal{O}(n^4 \cdot s \cdot \min\{s, n\})$ steps for the entire loop.

Computing all elements $X_1 \sqcup \ldots \sqcup X_k$ of $\mathcal{X}_{\max}^{\text{alg}}$, where $X_i \in \mathcal{X}_i^{\text{new}}$ for every $i = 1, \ldots, k$, needs at most $\prod_{i=1}^{k} \mid \mathcal{X}_i^{\text{new}} \mid$ operations (line (11)). As every $\mathcal{X}_i^{\text{new}}$ contains at most w_{N_i} elements the definition of a unit implies that $\prod_{i=1}^{k} \mid \mathcal{X}_i^{\text{new}} \mid \leq \prod_{i=1}^{k} w_{N_i} \leq n$ holds. The time complexity of Algorithm 5.4.1 is therefore indeed $\mathcal{O}(n^4 \cdot s \cdot \min\{s, n\})$.

In order to decide whether $\Sigma \models \mathcal{X} \rightarrow \mathcal{Y}$ holds it is sufficient to decide whether $\mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}_{\max}^{\text{alg}}$ (Lemma 5.25, Lemma 5.26 and Theorem 5.15). It takes $\mathcal{O}(n^4 \cdot s \cdot \min\{s, n\})$ steps to compute $\mathcal{X}_{\max}^{\text{alg}}$. Subsequently, $\mathcal{Y} \subseteq_{\text{gen}} \mathcal{X}_{\max}^{\text{alg}}$ can be verified in time $\mathcal{O}(n^3)$. □

Corollary 5.34. *In any case of the following type combinations: {records, sets}, {records, multisets}, {records, sets, multisets}, the implication problem $\Sigma \models \mathcal{X} \rightarrow \mathcal{Y}$ for FDs on a nested attribute N can be solved in time $\mathcal{O}(n^3 \cdot s \cdot \min\{s, n\})$ where $n = \mid Sub(N) \mid$ and $s = \mid \Sigma \mid$.*

Proof. In any of those cases we have $\prod_{i=1}^{k} n_i = n$, i.e., the number of subattributes of N is equal to the product of the number of subattributes of its units N_i. In that case, the meet of any two different units is λ_N. This results in smaller upper bounds in the proof of Theorem 5.33. □

5.4.6 Some Applications

Algorithm 5.4.1 can again be applied to compute non-redundant covers and decide whether an arbitrary set of subattributes is a superkey with respect to a set of FDs.

Theorem 5.35. *In the presence of record-, list-, set- and multiset-valued attributes, non-redundant covers for a set Σ of FDs on some nested attribute N can be computed in time $\mathcal{O}(n^4 \cdot s^2 \cdot \min\{n, s\})$ where $n = \mid Sub(N) \mid$ and $s = \mid \Sigma \mid$.* □

A set $\mathcal{X} \subseteq Sub(N)$ of subattributes of some nested attribute N is called a *superkey* for N with respect to a given set Σ of FDs on N if and only if $\Sigma \models \mathcal{X} \rightarrow N$ holds. This means that $\mathcal{X}$ is a superkey for N if and only if $N \in \mathcal{X}^+$.

Algorithm 5.4.2 (Superkey)

Input: $\boldsymbol{N} \in \boldsymbol{\mathcal{N}}$A, set $\boldsymbol{\Sigma}$ of FDs on $\boldsymbol{N}$, $\boldsymbol{\mathcal{X}} \subseteq \boldsymbol{Sub(N)}$

Output: $\begin{cases} \text{yes} & \text{, if } \boldsymbol{\mathcal{X}} \text{ is a superkey for } \boldsymbol{N} \text{ with respect to } \boldsymbol{\Sigma} \\ \text{no} & \text{, else} \end{cases}$

Method:

(1) Compute $\mathcal{X}^{alg}_{max}$ using Algorithm 5.4.1 with input $(N, \Sigma, \mathcal{X})$;
(2) **IF** $N \in \mathcal{X}^{alg}_{max}$ **THEN RETURN**(yes)
(3) **ELSE RETURN**(No);

□

Theorem 5.36. *In the presence of record-, list-, set- and multiset-valued attributes, Algorithm 5.4.2 decides in time* $\mathcal{O}(n^4 \cdot s \cdot \min\{n, s\})$ *whether* $\mathcal{X} \in Sub(N)$ *is a superkey for* N *with respect to a set* Σ *of FDs defined on* N, *where* $n = \mid Sub(N) \mid$ *and* $s = \mid \Sigma \mid$. □

5.5 The Implication Problem for all Combinations

Figure 5.7 shows upper complexity bounds for the implication problem of FDs in the presence of various types. If the input parameters are the nested attribute N and Σ a set of FDs defined on N, then $n = \mid SubB(N) \mid$, $m = \mid Sub(N) \mid$ and $s = \mid \Sigma \mid$.

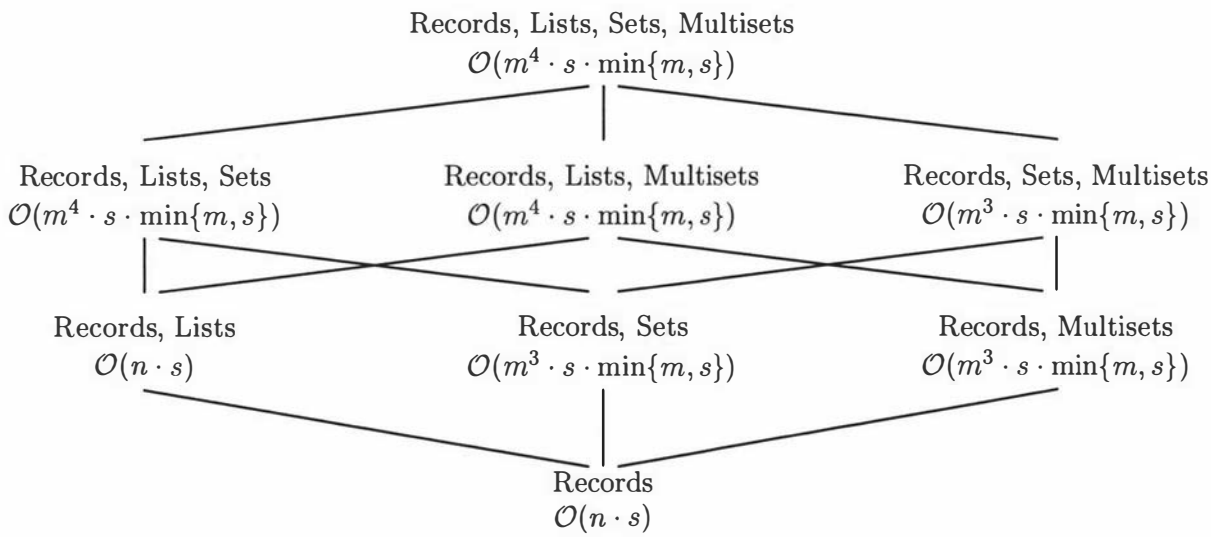

Fig. 5.7. Upper Complexity Bounds for the Implication Problem in the Presence of various Types

5.6 Related Work

Dependency theory is a well-studied area of research in the context of the RDM. Excellent surveys are provided in [109, 264, 278]. The RDM is completely captured by the presence of record-valued attributes.

The nested relational data model [179] has also attracted research on dependency theory, especially on the issue of normalisation [207, 215]. The FDs studied in those papers

arise from a relational representation of the data assuming a complete unnesting. Take for instance the nested schema {Course,(Student-ID, Name)*} in which for each course the set of participating students is stored, i.e., their student identification number together with their name. A typical FD would be

$$\text{Student-ID} \rightarrow \text{Name},$$

i.e., the student identification number uniquely determines the student's name over all courses. FDs in which a set of objects is determined by some object or in which a set of objects determines an object are not considered. An example of such an FD would be

$$\text{Course} \rightarrow (\text{Student-ID})^*$$

where the course determines the set of the identification numbers of its participants. This, however, can be done using record- and set-valued attributes. Consider the nested attribute Enrolment(Course,Participant{Student(ID,Name)}). The FD above is then specified by

$$\text{Enrolment(Course)} \rightarrow \text{Enrolment(Participant\{Student(ID)\})}.$$

On the other hand, FDs in which inside a set-valued attribute L{N} some subattributes of N determine another subattribute of N can be expressed by the previous approaches but are not yet covered by our approach. The previous example suggests for instance to consider the structure of embedded nested attributes such as Student(ID,Name). Then the FD

$$\text{Student(ID)} \rightarrow \text{Student(Name)}$$

does reflect the FD above. The nested relational data model is covered by the presence of record- and set-valued attributes.

Next we consider two approaches which have studied functional dependencies in the presence of finite sets. In [137] FDs are defined as *well-defined path expressions* in the presence of records and finite sets. An axiomatisation for the implication of those FDs is provided. However, the FDs do not allow arbitrary nesting, and most importantly, the right-hand side of every FD is always a single path. As the results in this thesis point out the case where the right-hand side is the union of paths is particularly interesting in the presence of sets (the join axiom is only valid in restricted form). FDs of the form

$$\{S\{L(A)\}, S\{L(B)\}\} \rightarrow S\{L(A,B)\}$$

cannot be expressed by the approach in [137] as this FD is different from the two trivial FDs

$$\{S\{L(A)\}, S\{L(B)\}\} \rightarrow S\{L(A)\} \text{ and } \{S\{L(A)\}, S\{L(B)\}\} \rightarrow S\{L(B)\}.$$

There are still differences even if we consider only single paths in the right-hand side. Consider for instance the nested attribute $N(L\{K(A,B,C)\}, D)$ together with the FD

$$N(L\{K(A,B)\}) \rightarrow N(D)$$

where the set of value pairs on A, B determines the value on D. FDs which are expressible by the approach in [137] are

$$N : [L \rightarrow D] \quad \text{and} \quad N : [L : A, L : B \rightarrow D]$$

assuming that the labels identify the (embedded) nested attributes. These, however, are both different from

$$N(L\{K(A, B)\}) \rightarrow N(D).$$

The first FD corresponds to

$$N(L\{K(A, B, C)\}) \rightarrow N(D)$$

and the second corresponds to

$$\{N(L\{K(A)\}), N(L\{K(B)\})\} \rightarrow N(D),$$

respectively. On the other hand, in order to express the FD $N : L[A \rightarrow B]$ in our context, we need to consider the embedded nested attributes $K(A, B, C)$ where the FD $K(A) \rightarrow K(B)$ could be defined. Moreover, attributes in which λ occurs are not covered in [137]. In summary, the approach in [137] uses partly the expressiveness of the set constructor, but does not take care of the fact that the extension rule is not valid in the presence of sets.

A further approach to defining FDs in the context of the nested relational data model is provided in [180]. So-called *null extended FDs* are defined to admit null values and study the relationship between multi-valued dependencies $X \twoheadrightarrow Y$ and FDs $X \rightarrow Y^*$ (here Y refers to the complete unnesting of the relation-valued attribute Y^*), i.e., the interaction of different dependency classes in the context of nesting and unnesting. Null extended FDs are again defined on the basis of paths. FDs from the RDM cannot be expressed. Furthermore, relation-valued attributes can only occur on the right-hand side of null extended FDs. Consider the nested attribute $N = L(A, K\{M(B, S\{C\})\})$ which would be expressed as $A(B(C)^*)^*$ in a slightly simplified nested relational data model. Examples for null extended FDs are

$$A \rightarrow (B(C)^*)^* \text{ or } AB \rightarrow (C)^*.$$

The last of these is not covered yet by our data model. In order to express the last null extended FD in our context we need to consider combinations of embedded nested attributes, i.e., $L(A, M(B, S\{C\}))$ in this case. Conversely, the FD $L(A, K\{M(B)\}) \rightarrow L(K\{M(S\{C\})\})$ is again not expressible as a null extended FD. The expressiveness of null extended FDs and FDs in the presence of null, flat, record- and set-valued attributes is different.

Most recently, the major research interest is on the model of semi-structured data and XML [1, 53]. Work on integrity constraints in the context of XML and object-oriented databases can be found in [14, 51, 58, 110, 111, 178, 262, 285, 286, 287, 295]. The approaches in [14, 51, 178, 262, 285, 295] are again based on a relational representation of the data, thus resulting again in a different expressiveness from our approach. FDs in [14] are not

axiomatisable at all. In order to illustrate the difference to our data model a bit more we look at some examples.

Consider the XML data tree in Figure 5.8 containing data on courses organised by the dancing club of the local high school.

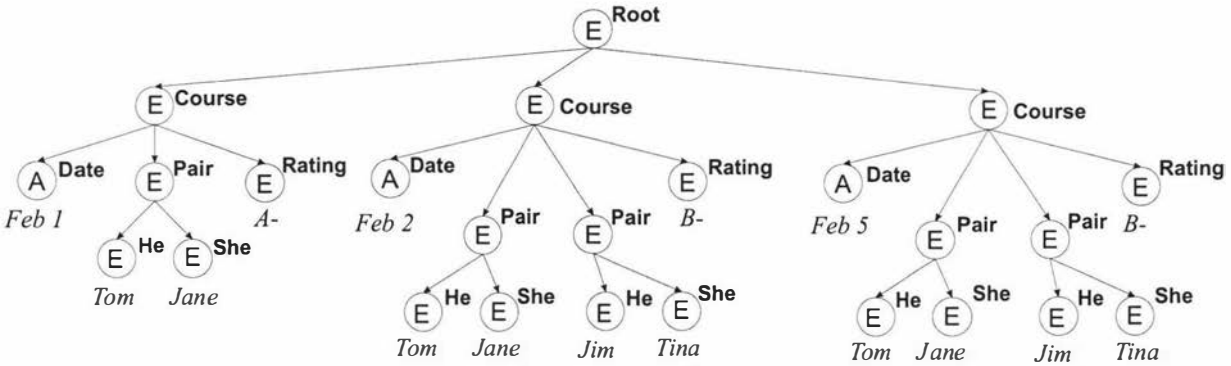

Fig. 5.8. An XML data tree carrying some functional dependency.

The XML document corresponding to this XML data tree is shown in Figure 5.9.

```
01<Root>
02  <Course Date="Feb 1">
03    <Pair>
04      <He>Tom</He>
05      <She>Jane</She>
06    </Pair>
07    <Ranking>A-</Ranking>
08  </Course>
09  <Course Date="Feb 2">
10    <Pair>
11      <He>Tom</He>
12      <She>Jane</She>
13    </Pair>
14    <Pair>
15      <He>Jim</He>
16      <She>Tina</She>
17    </Pair>
18    <Ranking>B-</Ranking>
19  </Course>
20  <Course Date="Feb 5">
21    <Pair>
22      <He>Tom</He>
23      <She>Jane</She>
24    </Pair>
25    <Pair>
26      <He>Jim</He>
27      <She>Tina</She>
28    </Pair>
29    <Ranking>B-</Ranking>
30  </Course>
31</Root>
```

Fig. 5.9. An XML document corresponding to the XML data tree in Fig. 5.8.

It happens that neither gentlemen nor ladies change their dance partners. That is, for every pair in the XML data tree He determines She, and vice versa. Both observations are likely to be called functional dependencies.

Now consider the XML data tree in Figure 5.10. It is obvious that the observed functional dependencies do no longer hold. Nevertheless the data stored in this tree is not independent from each other: whenever two courses coincide in all their pairs then they coincide in their rating, too. That is, in every course the set of Pairs determines the Rating. The reason for this might be straightforward. Suppose, during every course each pair is asked whether they enjoyed dancing with each other (and suppose that the answer will not change over time). Afterwards, the average rating is calculated for the course and stored within the XML document. This, in fact, leads to the functional dependency observed in Figure 5.10.

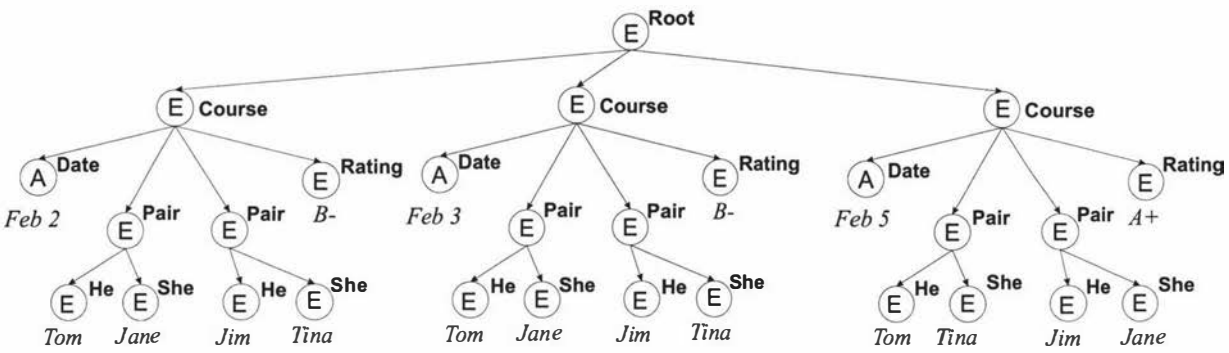

Fig. 5.10. Another XML data tree still carrying some functional dependency.

Surprisingly, [14, 178, 285] all introduced the first kind of functional dependencies for XML while the second kind has been neglected so far in the literature on XML. The reason for this is the path-based approach towards functional dependencies used in all three papers. The second kind, however, represents functional dependencies that can be captured using nested attributes. Suppose we have the nested attribute

$$\text{Course(Date,Pair\{Partner(He,She)\},Rating)},$$

then the functional dependency above reads as

$$\text{Course(Pair\{Partner(He,She)\})} \rightarrow \text{Course(Rating)}.$$

In order to capture the first kind of functional dependencies via nested attributes one needs to consider the embedded nested attribute Partner(He,She). In this case the FDs read as Partner(He) $\rightarrow$ Partner(She) and Partner(She) $\rightarrow$ Partner(He). For a graph-oriented approach towards functional dependencies in XML that is based on homomorphisms between subgraphs see [138] and [144].

In order to capture the full expressiveness of XML one will need to consider the union and reference type. Thus, a Kleene-star element definition $\langle !ELEMENT\ X(Y)^* \rangle$ can be represented by the list-valued nested attribute $X[Y]$, a sequence element definition

$\langle !ELEMENT\ X(Y_1, \ldots, Y_n)\rangle$ by the record-valued attribute $X(Y_1, \ldots, Y_n)$, and an alternative element definition $\langle !ELEMENT\ X(Y_1 \mid \cdots \mid Y_n)\rangle$ by $X(Y_1 \oplus \cdots \oplus Y_n)$. Furthermore, as the plus-operator in regular expressions can be expressed by the Kleene-star, an element definition $\langle !ELEMENT\ X(Y)^+\rangle$ can be represented by the record-valued attribute $X(Y, X'[Y])$ with a new label X'. Similarly, optional elements can be expressed by alternatives with empty elements, thus an element definition $\langle !ELEMENT\ X(Y?)\rangle$ will be represented by the union-valued attribute $L(X(Y) \oplus X'(\lambda))$. In order to capture the reference structures in XML documents we may need to consider rational tree attributes. See [76] for fundamental properties of infinite trees. In this case, the subattribute lattice may become infinite.

In summary, our approach based on explicit subattributes deviates significantly from previous approaches in the nested relational data model, object-oriented data models and XML, yielding a complementary expressiveness. In particular, the algebraic approach based on a Brouwerian algebra of subattributes is original. To the author's best knowledge there is not any other work which deals specifically with lists and multisets in the context of FDs.

Chapter 6

Summary

This chapter shall summarise the main results and contributions of this thesis and comment on future research by listing some open problems.

6.1 Main Results

The major contribution of this thesis is the provision of a mathematically well-founded framework that allows the study of different classes of dependencies in the presence of various combinations of data types. Data models are classified according to the data types that they support. The approach is therefore independent of any specific data model. Although it is not claimed that this is the ultimate unifying framework to investigate problems of dependency theory in complex-value data models, the presence of specific types in any data model do motivate the study of those kinds of problems investigated in this thesis. The examples used throughout the thesis illustrate that dependencies naturally occur among complex objects. The extension of the relational theory of various dependency classes to the presence of complex data types allows to specify more real-world constraints and increases therefore the number of application domains. Moreover, a formal foundation for automated reasoning about these constraints is provided and the start to a complex-value database design theory made.

It has been demonstrated that the presence of such complex objects like records, lists, sets and multisets leads to the algebraic structure of a Brouwerian algebra. The relational data model is based on the Boolean powerset algebra $(\mathcal{P}(R), \subseteq, \cup, \cap, (\cdot)^{\mathcal{C}}, \emptyset, R)$. From a purely algebraic point of view, the gain in expressiveness due to the introduction of complex objects results therefore in the loss of the involutional character of the complement operation. Throughout the thesis it is shown that Brouwerian algebras are sufficiently powerful to generalise and extend well-known results from relational databases.

In the presence of lists it is sufficient to consider the subattributes of a nested attribute in order to define functional and multi-valued dependencies while in the presence of sets or multisets, sets of subattributes need to be introduced. In that sense lists are simpler than sets and multisets. This is not really surprising as the list type possesses both features: the elements of a list are totally ordered and multiple occurrences of the same element are

allowed. Since multisets also allow the occurrence of duplicates, it must be the total order on the elements of a list that guarantees the soundness of the join axiom, i.e., that the values of the projections on subattributes determine the value of the projection on their join. While list and multiset type allow the reasoning about the number of their elements, the set type is only capable of distinguishing between empty and non-empty sets. The fact that elements of a list or multiset may occur more than once is decisive to that regard.

The dependency classes that have been studied add a complementary expressiveness to those that have previously been studied in the literature. MVDs have not been studied at all in the presence of lists. FDs have not been studied at all in the presence of lists and multisets.

Regarding the problem of axiomatising the class of FDs, Theorem 5.23 captures the main result. Once the framework of nested attributes is given, it is straightforward to obtain a generalisation of Armstrong's axioms for the class of FDs in the presence of records and lists. The introduction of set- and multiset-valued attributes calls for a more sophisticated definition of FDs. Left- and right-hand side are now sets of subattributes instead of single subattributes. Besides Armstrong's original axioms two more axioms are required to capture the class of FDs in the presence of sets or multisets. The completeness proof, which still remains constructive, uses rather deep arguments in case of set- and multiset-valued attributes. Theorem 5.23 provides minimal axiomatisations for the class of FDs in the presence of all combinations of records, lists, sets and multisets that at least include records.

Figure 5.7 summarises the upper complexity bounds for the implication problem of FDs in the presence of all previous type combinations. In the context of records and lists, a provably-correct and linear-time algorithm is proposed for computing the closure of a nested attribute with respect to a given set of FDs. The size of the input is defined in the number of join-irreducible subattributes and the number of FDs given. The representation theorem for Brouwerian algebras suggests a different, topological view of FDs. This alternative perspective is even more similar to the framework of relational databases, in the sense that operations are performed on (closed) sets. In the presence of sets or multisets, provably-correct and polynomial-time algorithms are proposed for computing the closure of a set of nested attributes with respect to a given set of FDs. The size of the input, however, is now defined as the number of all subattributes and the number of FDs given. This is justified by the fact that a set of subattributes is semantically different from the join of these subattributes.

Theorem 4.3 shows that MVDs are still equivalent to binary join dependencies, even in the presence of records and lists. Theorem 4.13, Theorem 4.28 and Theorem 4.31 provide (minimal) axiomatisations for the class of FDs and MVDs, Theorem 4.43 and Theorem 4.44 propose minimal axiomatisations for the class of MVDs. An interesting fact is that the MVD $X \twoheadrightarrow Y$ implies the non-trivial FD $X \rightarrow Y \sqcap Y^C$ which gives the set of inference rules a distinctive Brouwerian flavour. Recall that MVDs do not imply any non-trivial FDs in the context of the RDM. Further differences to the RDM are given by the minimal sets of inference rules. This is due to the fact that non-maximal join-irreducible subattributes cannot be represented as the Brouwerian complement of any set of subattributes.

The provably-correct and polynomial-time Algorithm 4.4.1 computes dependency basis and nested attribute closure for a given subattribute and a given set of FDs and MVDs. It naturally generalises the well-known membership algorithm for FDs and MVDs in relational databases. This shows that the implication problem for FDs and MVDs is efficiently decidable in the presence of records and lists.

The applicability of efficiently solving the various implication problems is demonstrated by proposing efficient algorithms for computing non-redundant covers of sets of dependencies and deciding whether a (set of) nested attribute(s) is a superkey with respect to a given set of dependencies.

Database design theory in terms of FDs is extended to the presence of records and lists. Formal definitions of design criteria such as the absence of redundancies and the absence of abnormal update behavior are generalised and adapted to this framework. The Nested List Normal Form (NLNF) is proposed as a normal form that syntactically describes well-designed nested attributes. NLNF is strictly weaker than a straightforward extension of Boyce-Codd Normal Form. The proposal is semantically justified by formally showing the equivalence to the absence of redundancy, strong insertion anomalies, and strong type-1 and strong type-2 replacement anomalies. Furthermore, strong type-3 replacement anomalies cannot occur for nested attributes in NLNF. In order to verify that an instance of a nested attribute in NLNF satisfies all FDs given, it is sufficient to verify that this instance satisfies all key dependencies and all inevitable FDs. Finally, a provably-correct algorithm is proposed which decomposes an arbitrary nested attribute with respect to a given set of FDs into subattributes that are all in NLNF with respect to the set of projected FDs. This decomposition is lossless in the sense that every instance, satisfying all the FDs given, is the generalised natural join of its projections on the decomposed subattributes. Some problems with the algorithm are pointed out. The algorithm may execute in time exponential in the size of the given nested attribute and set of FDs, the cardinality of the decomposition may be exponential in the size of the given nested attribute. Moreover, deciding whether an arbitrary subattribute is in NLNF with respect to the projected set of FDs is *coNP*-complete. Finally, some of the FDs that have been specified may be lost during the decomposition process. The results obtained for NLNF as well as the problems with the decomposition algorithm generalise well-known results from the RDM.

6.2 Open Problems

Figure 1.1 gives an indication of opportunities for future research. Although an axiomatisation for the class of FDs has been achieved in all combinations of records, lists, sets and multisets, the expressiveness for the class of FDs can be increased. We have seen examples which suggest to study the *interaction* of FDs defined on embedded nested attributes. Consider for instance the nested attribute

$$L[N(A,B,C)] \text{ together with its embedded attribute } N(A,B,C).$$

Suppose the functional dependency $N(A) \to N(B)$ has been specified on $N(A,B,C)$ and $r \subseteq dom(N)$ satisfies this FD. If $r' \subseteq \{[t_1,\ldots,t_n] \mid t_i \in r\}$, then $r' \subseteq dom(L[N(A,B,C)])$

satisfies the FD $L[N(A)] \rightarrow L[N(B)]$. Considering embedded attributes of a nested attribute therefore leads to the soundness of further inference rules which need to be studied in order to achieve an axiomatisation for this new class of FDs. The expressiveness can be even more increased if not only embedded attributes, but also combinations of embedded attributes are studied. This was suggested by the example of null-extended FDs and the FDs previously studied in XML. In the same spirit, one may try to increase the expressiveness of MVDs by studying the interaction between MVDs on embedded attributes.

Another approach to increasing the expressiveness is to *extend the number of subattributes* for a fixed nested attribute. This can be done by relating the information content of different data types to one another. The list-valued attribute $L[A]$ for instance may have the subattribute $L\langle A\rangle$ which itself has the subattribute $L\{A\}$. In the first step we drop the information on the order of the elements, in the second step we drop the possibility of multiple occurrences of the same element. It is then interesting to study to which extent this approach still results in a sufficiently powerful structure in which dependencies can be investigated.

A more general treatment of data dependencies in complex-value databases may have a successful turnout as in the RDM [109, 264, 278]. The problem is that of finding a suitable logic (if there is one) such that dependencies can be associated with formulae in that logic. The first-order theories of lists, sets and multisets established in [99] seem promising.

What changes if *lists, sets or multisets are allowed to have an infinite number of elements*?

It is desirable to improve the running time of Algorithm 4.4.1 for deciding the implication of FDs and MVDs. Substantial research on that subject has again been done for relational databases and the papers [98, 118, 135, 152, 173, 223, 239, 277] may give some more information.

It seems interesting to study *multi-valued dependencies in the presence of the set and multiset constructor*. For relational databases, the flat relation r satisfies the MVD $A \twoheadrightarrow B$ if and only if the relation r^*, that results from a NEST operation over attribute B, satisfies the FD $A \rightarrow (B)^*$, see [113]. This observation will have a direct impact on the interaction of FDs and MVDs on different combinations of embedded attributes.

The minimality results in thesis have been achieved with respect to Definition 3.7. As mentioned before the notion of minimality can be improved. Strictly speaking, *minimality* would also refer to the fact that the constraints for every inference rule cannot be weakened without losing completeness. This stronger notion of minimality should be studied in the future. Moreover, it would be interesting to find other (all?) minimal sets of inference rules for the various axiomatisations.

Although the context-dependent Brouwerian complement rule could be replaced by the much weaker context-dependent N-axiom, it is still interesting to ask how mixed meet rule and auto-complement rule, respectively, can be weakened.

Another interesting line of research is *data mining*. It would be interesting to develop algorithms that determine all FDs and MVDs on a nested attribute N that are satisfied by a particular instance $r \subseteq dom(N)$. For relational databases, [167, 189, 198, 199] and [242, 300] have developed algorithms for FDs and MVDs, respectively.

There is a polynomial-time algorithm for obtaining a lossless BCNF decomposition for relation schemata [270]. The idea from that paper may give hints how to obtain a *polynomial-time algorithm for achieving a lossless NLNF decomposition.*

An open problem is to *extend the NLNF proposal to the class of MVDs, and the class of FDs and MVDs*, to semantically justify the proposal in terms of absence of redundancies and abnormal update behavior, and to generalise the decomposition approach. Relevant papers that address these problems in the context of relational databases are [280, 281, 283, 289, 290] and [133]. The key to solving these problems is an appropriate definition of inevitable MVDs. Let N be a nested attribute and Σ a set of FDs and MVDs on N. The conditions when an FD from Σ^+ is in Σ_{inev} are exactly as before. An MVD $X \twoheadrightarrow Y \in \Sigma^+$ is in Σ_{inev} if and only if $Y \leq X$ or $Y \in NMaxB(N)$ or $X \sqcup Y = N$. The set Σ^+_{inev} of inevitable FDs and MVDs on N with respect to Σ is then the closure of Σ_{inev} under the complete set of inference rules for FDs and MVDs in the presence of lists from Theorem 4.28. It follows for an MVD $X \twoheadrightarrow Y \in \Sigma^+$ that $X \twoheadrightarrow Y \in \Sigma^+_{\text{inev}}$ if and only if $Y^{cc} \leq X$ or $Y^c \leq X$ holds. A nested attribute N is said to be in Nested List Normal Form with respect to the set Σ of FDs and MVDs on N if and only if every $X \twoheadrightarrow Y \in \Sigma^*$ is inevitable on N with respect to Σ or X is a superkey for N with respect to Σ. It is conjectured that the results on the 4NF in relational databases from [280] carry over to the Nested List Normal Form. As it was the case with BCNF and NLNF, a simple extension of 4NF implies NLNF, but not vice versa. Along these investigations it might prove useful to generalise such notions as *reduced MVDs* and *minimal covers* of sets of MVDs [216], *pure set of FDs and MVDs* [154], *envelope set* [301, 302] and *conflict-free MVDs* [247] from relational databases to the context of complex object types.

A further desirable goal is to propose normal forms for nested attributes in the presence of more type constructors. The axiomatisations of FDs in the context of records, lists, sets and multisets suggest to continue along those lines. The decisive notion in a *Complex-value Normal Form* proposal may be that of a unit of a nested attribute, taking over the role of maximal basis attributes in the proposal of the NLNF.

Normalisation is nothing but an *optimisation.* Considering the example of the prime factorisation Factor(Integer,Prime[Number],Exponent[Number]), one may ask whether the list constructor is really appropriate here. Instead of storing the list of prime factors and the list of exponents, one may store the set of prime factor/exponent pairs. It would be interesting to see whether the specification of inevitable FDs such as

$$\text{Factor(Prime}[\lambda]) \rightarrow \text{Factor(Exponent}[\lambda]) \text{ and Factor(Exponent}[\lambda] \rightarrow \text{Factor(Prime}[\lambda])$$

suggest that the data type is inappropriate. A further observation is the following. Consider the list-valued attribute $L[M]$ and suppose the FD $\lambda \rightarrow L[\lambda]$ has been specified. It says informally that all tuples coincide on $L[\lambda]$, i.e., the length of the list $L[M]$ is constant, say k. In this case, it is certainly more appropriate to use the record-valued attribute $L(M_1, \ldots, M_k)$ with $dom(M_i) = dom(M)$ for $i = 1, \ldots, k$.

The notions of redundancy and update anomalies that were used in this thesis are not the only notions that appear in the literature. Vincent has introduced the concept of *value-redundancy* in [283]. Given a relation schema R, an attribute A in R, a set Σ of

FDs and MVDs on R, a relation r over R and a tuple t in r, the data value occurrence $t[A]$ is redundant with respect to Σ iff for every replacement of $t[A]$ by a value a' such that $t[A] \neq a'$ and resulting in a new relation r', then $\not\models_{r'} \Sigma$. (R, Σ) is defined to be in redundancy free normal form if and only if there does not exist an r over R with $\models_r \Sigma$ which contains a data value occurrence that is redundant with respect to Σ. Vincent shows that (R, Σ) is in redundancy free normal form if and only if R is in 4NF with respect to Σ. Update anomalies, as defined in this thesis, are called key-based update anomalies in [280]. So-called *fact-based update anomalies* are also introduced in [280], and the relationship between their absence and BCNF and 4NF are examined. The extensions of these notions to the framework of nested attributes and their relationship to the Nested List Normal Form are further directions of future research. Arenas and Libkin [13] use techniques from *information theory* to define a measure of information content of elements in a database with respect to a set of constraints. This provides a set of tools for testing when a normal form proposal corresponds to a good design. The results give information-theoretic justifications for normal forms such as BCNF, 4NF, project-join normal form (PJ/NF), fifth normal form (5NF), domain-key normal form (DK/NF) and the XML normal form XNF proposed in [14], as well as information-theoretic criteria for justifying normalisation algorithms. It would be interesting to test the measure with respect to the Nested List Normal Form proposal, and later on the Complex-value Normal Form proposal.

Normalisation is a well-studied area in the context of relational databases. Besides BCNF and 4NF, there are many other normal form proposals. An extension of third normal form (3NF) [70, 304], PJ/NF [104], 5NF [282] and DK/NF [105] to nested attributes seem desirable.

The *(disjoint) union type* is well-worth investigating as it can be used to represent alternatives. It is very important for the higher-order entity-relationship model [265] and XML [53]. In order to give a small illustration of the difficulty of the union type we look at the following example. Figure 6.1 shows the structure of the union-valued attribute $L(A \oplus B)$.

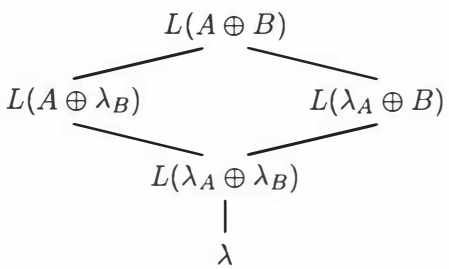

Fig. 6.1. Subattribute Lattice of a Union-valued Attribute

Note that the subattribute $L(\lambda_A \oplus \lambda_B)$ indicates from which domain a value stems. If the projection on $L(\lambda_A \oplus \lambda_B)$ is ok_A, then the value comes from $dom(A)$, and if the projection is ok_B, then the value comes from $dom(B)$. Suppose that one needs to find two different elements $t_1, t_2 \in dom(L(A \oplus B))$ with $\pi^N_W(t_1) = \pi^N_W(t_2)$ iff $W \leq L(\lambda_A \oplus \lambda_B)$. Consequently, both of the values must be ok_A or both of the values must be ok_B on $L(\lambda_A \oplus \lambda_B)$. That means we have either $t_1, t_2 \in dom(A)$ or $t_1, t_2 \in dom(B)$. However, in this case we also have $\pi^N_{L(\lambda_A \oplus B)}(t_1) = \pi^N_{L(\lambda_A \oplus B)}(t_2)$ or $\pi^N_{L(A \oplus \lambda_B)}(t_1) = \pi^N_{L(A \oplus \lambda_B)}(t_2)$. That shows that one cannot find any two elements of $dom(L(A \oplus B))$ with this property, which indicates that there is an FD

$$L(\lambda_A \oplus \lambda_B) \rightarrow L(A \oplus \lambda_B) \text{ or } L(\lambda_A \oplus B)$$

where the right-hand side is a disjunction of subattributes. Such FDs are relevant, if an axiomatisation of FDs in the presence of unions is pursued.

Another challenge is the inclusion of the *reference type* into the types of interest. It is particularly important for object-oriented databases and XML. A possible approach to investigate the reference type is to represent nested attributes as labelled trees where the labels of a non-leaf node are used to define embedded attributes and leaf nodes are either null or flat attributes or referencing labels to other nodes. This leads to rational trees which are infinite, but in which the number of different subtrees is still finite.

More classes of relational dependencies will be the subject of future studies in the presence of various combinations of data types. The book [264] identifies more than 90 different constraint classes for relational databases. The class of join dependencies is more general than the class of MVDs. Interestingly, there does not exist a finite Hilbert-style axiomatisation for this class [229], however, a sound and complete set of Gentzen-style inference rules is proposed in [39]. A different and important class are inclusion dependencies which are not uni-relational, i.e., refer to more than one relation schema.

Finally, the proposed concepts and algorithms should be *implemented.* The research report [252] contains a C++ implementation for computing dependency basis and nested attribute closure of a given subattribute with respect to a given set of functional and multi-valued dependencies in the presence of records and lists.

Bibliography

1. Abiteboul, S., P. Buneman and D. Suciu, "Data on the Web: From Relations to Semistructured Data and XML," Morgan Kaufmann Publishers, 2000.
2. Abiteboul, S., S. Cluet, T. Milo, P. Mogilevsky, J. Siméon and S. Zohar, *Tools for data translation and integration*, Data Engineering Bulletin **22** (1999), pp. 3–8.
3. Abiteboul, S., P. C. Fischer and H.-J. Schek, editors, "Nested Relations and Complex Objects, Papers from the Workshop "Theory and Applications of Nested Relations and Complex Objects", Darmstadt, Germany, April 6-8, 1987," Number 361 in Lecture Notes in Computer Science, Springer, 1989.
4. Abiteboul, S. and R. Hull, *IFO: A formal semantic database model*, Transactions on Database Systems (TODS) **12** (1987), pp. 525–565.
5. Abiteboul, S., R. Hull and V. Vianu, "Foundations of Databases," Addison-Wesley, 1995.
6. Abiteboul, S. and P. C. Kanellakis, *Object identity as a query language primitive*, in: *Proceedings of the International Conference on Management of Data (SIGMOD)*, ACM, 1989, pp. 159–173.
7. Aho, A. V., C. Beeri and J. D. Ullman, *The theory of joins in relational databases*, Transactions on Database Systems (TODS) **4** (1979), pp. 297–314.
8. Amos, M., G. Paun, G. Rozenberg and A. Salomaa, *Topics in the theory of DNA computing*, Theoretical Computer Science **287** (2002), pp. 3–38.
9. Anderson, I., "Combinatorics of finite sets," Oxford Science Publications, The Clarendon Press Oxford University Press, New York, 1987.
10. Arapis, C., *Temporal specifications of object behavior*, in: *Proceedings of the 3rd Symposium on Mathematical Fundamentals of Database and Knowledge Base Systems (MFDBS)*, number 495 in Lecture Notes in Computer Science (1991), pp. 308–324.
11. Arenas, M. and L. Libkin, *On verifying consistency of XML specifications*, in: *Principles of Database Systems (PODS)* (2002), pp. 259–270.
12. Arenas, M. and L. Libkin, *What's hard about XML schema constraints?*, in: *Database and Expert Systems Applications (DEXA)*, 2002, pp. 269–278.
13. Arenas, M. and L. Libkin, *An information-theoretic approach to normal forms for relational and XML data*, in: *Principles of Database Systems (PODS)* (2003), pp. 15–26.
14. Arenas, M. and L. Libkin, *A normal form for XML documents*, Transactions on Database Systems (TODS) **29** (2004), pp. 195–232.

15. Armstrong, W. W., *Dependency structures of database relationships*, Information Processing (1974), pp. 580–583.
16. Armstrong, W. W., Y. Nakamura and P. Rudnicki, *Armstrong's axioms*, Journal of formalized Mathematics **14** (2002).
17. Arora, A. K. and C. R. Carlson, *The information preserving properties of relational database transformations*, in: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1978, pp. 352–359.
18. Asano, T., N. Katoh, K. Obokata and T. Tokuyama, *Matrix rounding under the l_p-discrepancy measure and its application to digital halftoning*, SIAM Journal on Computing **32** (2003), pp. 1423–1435.
19. Asirelli, P., P. Inverardi and G. Plagenza, *Integrity constraints in deductive databases: an overview*, Technical Report 1996-B4-12-03, Istituto di Elaborazione dell'Informazione, Pisa, Italy (1996).
20. Atkinson, M., F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier and S. Zdonik, *The object-oriented database system manifesto*, in: *Proceedings of the International Conference on Deductive and Object-Oriented Databases*, 1989, pp. 40–57.
21. Banatre, J. and D. Le Metayer, *Programming by multiset transformation*, Communications of the ACM **36** (1993), pp. 98–111.
22. Bancilhon, F., C. Delobel and P. Kanellakis, editors, "Building an Object-Oriented Database Sytem: The Story of O_2," Morgan Kaufmann, 1992.
23. Barendregt, H. P., editor, "The Lambda Calculus: Its syntax and semantics," North Holland, Amsterdam, 1984.
24. Basten, T., *Parsing partially ordered multisets*, International Journal of Computer Science **8** (1997), pp. 379–407.
25. Batini, C., S. Ceri and S. B. Navathe, "Conceptual Database Design: An Entity-Relationship Approach," Benjamin Cummings, 1992.
26. Beeri, C., *On the role of data dependencies in the construction of relational database schemes*, Technical Report TR-43, The Hebrew University of Jerusalem (1979).
27. Beeri, C., *On the membership problem for functional and multivalued dependencies in relational databases*, Transactions on Database Systems (TODS) **5** (1980), pp. 241–259.
28. Beeri, C., *A formal approach to object-oriented databases*, Data and Knowledge Engineering **5** (1990), pp. 353–382.
29. Beeri, C. and P. A. Bernstein, *Computational problems related to the design of normal form relational schemata*, Transactions on Database Systems (TODS) (1979), pp. 30–59.
30. Beeri, C., P. A. Bernstein and N. Goodman, *A sophisticate's introduction to database normalization theory*, in: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1978, pp. 113–124.
31. Beeri, C., M. Dowd, R. Fagin and R. Statman, *On the structure of armstrong relations for functional dependencies*, Journal of the ACM **31** (1984), pp. 30–46.
32. Beeri, C., R. Fagin and J. H. Howard, *A complete axiomatization for functional and*

multivalued dependencies in database relations, in: *International Conference on Management of Data (SIGMOD)*, ACM, 1977, pp. 47–61.
33. Beeri, C. and P. Honeyman, *Preserving functional dependencies*, SIAM Journal on Computing **10** (1981), pp. 647–656.
34. Beeri, C. and M. Kifer, *An integrated approach to logical design of relational database schemes*, Transactions on Database Systems (TODS) **11** (1986), pp. 134–158.
35. Beeri, C., A. O. Mendelzon, Y. Sagiv and J. D. Ullman, *Equivalence of relational database schemes*, SIAM Journal on Computing **10** (1981), pp. 352–370.
36. Beeri, C. and T. Milo, *Schemas for integration and translation of structured and semi-structured data*, in: *Proceedings of the International Conference on Database Theory (ICDT), Jerusalem, Israel*, 1999, pp. 296–313.
37. Beeri, C. and J. Rissanen, *Faithful representation of relational database schemes*, Technical Report RJ2722, IBM Research Laboratory (1980).
38. Beeri, C. and M. Y. Vardi, *The implication problem for data dependencies*, in: *Proceedings of the International Conference on Algorithms, Languages and Programming*, number 115 in Lecture Notes in Computer Science (1981), pp. 73–85.
39. Beeri, C. and M. Y. Vardi, *Formal systems for join dependencies*, Theoretical Computer Science **38** (1985), pp. 99–116.
40. Bernstein, P. A., *Normalisation and functional dependencies in the relational data base model*, Technical Report CSRG-60, University of Toronto (1975).
41. Bernstein, P. A., *Synthesizing third normal form relations from functional dependencies*, Transactions on Database Systems (TODS) **1** (1976), pp. 277–298.
42. Bernstein, P. A. and N. Goodman, *What does Boyce-Codd normal form do?*, in: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1980, pp. 245–259.
43. Bernstein, P. A., J. R. Swenson and D. C. Tzichritzis, *A unified approach to functional dependencies and relations*, in: *Proceedings of the International Conference on the Management of Data (SIGMOD)*, ACM, 1975, pp. 237–245.
44. Berry, G. and G. Boudol, *The chemical abstract machine*, Theoretical Computer Science **96** (1992), pp. 217–248.
45. Birkhoff, G., "Lattice Theory," American Mathematical Society, 1940.
46. Biskup, J., *On the complementation rule for multivalued dependencies in database relations*, Acta Informatica **10** (1978), pp. 297–305.
47. Biskup, J., *Database schema design theory: achievements and challenges*, in: *Information Systems and Data Management*, number 1066 in Lecture Notes in Computer Science (1995), pp. 14–44.
48. Biskup, J., *Achievements of relational database schema design theory revisited*, in: *Semantics in databases*, number 1358 in Lecture Notes in Computer Science (1998), pp. 29–54.
49. Biskup, J., U. Dayal and P. A. Bernstein, *Synthesizing independent database schemas*, in: *Proceedings of the International Conference on the Management of Data (SIGMOD)*, ACM, 1979, pp. 143–153.

50. Bobrow, D. G., K. M. Kahn, G. Kiczales, L. Masinter, M. Stefik and F. Zdybel, *Commonloops: Merging Lisp and object-oriented programming*, in: *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications*, ACM, 1986, pp. 17–29.
51. Bommel, M. F. and G. E. Weddell, *Reasoning about equations and functional dependencies on complex objects*, Transactions on Knowledge and Data Engineering **6** (1994), pp. 455–469.
52. Börger, E., E. Grädel and Y. Gurevich, "The Classical Decision Problem," Perspectives in Mathematical Logic, Springer, 1997.
53. Bray, T., J. Paoli, C. M. Sperberg-McQueen, E. Maler and F. Yergeau, *Extensible markup language (XML) 1.0 (third edition) W3C recommendation 04 february 2004*, http://www.w3.org/TR/2004/REC-xml-20040204/ (2004).
54. Brodie, M. L. and D. Ridjanovic, *On the design and specification of database transactions*, in: *On Conceptual Modelling* (1984), pp. 277–306.
55. Bry, F. and P. Kröger, *A computational biology database digest: data, data analysis, and data management*, Distributed and Parallel Databases **13** (2003), pp. 7–42.
56. Buneman, P., S. Davidson, W. Fan, C. Hara and W. Tan, *Keys for XML*, Computer Networks **39** (2002), pp. 473–487.
57. Buneman, P., S. Davidson, W. Fan, C. Hara and W. Tan, *Reasoning about keys for XML*, Information Systems **28** (2003), pp. 1037–1063.
58. Buneman, P., W. Fan, J. Siméon and S. Weinstein, *Constraints for semi-structured data and XML*, SIGMOD Record **30** (2001), pp. 47–54.
59. Buneman, P., W. Fan and S. Weinstein, *Path constraints in semistructured databases*, Journal of Computer and System Sciences **61** (2000), pp. 146–193.
60. Calude, C., G. Paun, G. Rozenberg and A. Salomaa, editors, "Multiset Processing, Mathematical, Computer Science, and Molecular Computing Points of View [Workshop on Multiset Processing, WMP 2000, Curtea de Arges, Romania, August 21-25, 2000]," Springer, 2001.
61. Calvanese, D., G. DeGiacomo and M. Lenzerini, *Representing and reasoning on XML documents: a description logic approach*, Journal of Logic and Computation **9** (1999), pp. 295–318.
62. Cardelli, L. and P. Wegener, *On understanding types, data abstraction and polymorphism*, ACM Computing Surveys **17** (1985), pp. 471–522.
63. Chen, P. P., *The entity-relationship model: Towards a unified view of data*, Transactions on Database Systems (TODS) **1** (1976), pp. 9–36.
64. Chen, P. P., *English sentence structure and entity-relationship diagrams*, Information Science **29** (1983), pp. 127–149.
65. Chomicki, J., *Efficient checking of temporal integrity constraints using bounded history encoding*, Transactions on Database Systems (TODS) **20** (1995), pp. 149–186.
66. Chomicki, J. and D. Niwinski, *On the feasibility of checking temporal integrity constraints*, Journal of Computer and System Sciences **51** (1995), pp. 523–535.
67. Cluet, S., C. Delobel, J. Siméon and K. Smaga, *Your mediators need data conversion*,

in: *Proceedings of International Conference on Management of Data (SIGMOD)*, ACM, 1998, pp. 177–188.
68. Codd, E. F., *A relational model of data for large shared data banks*, Communications of the ACM **13** (1970), pp. 377–387.
69. Codd, E. F., *Normalized database structure: A brief tutorial*, in: *Workshop on Data Description, Access and Control*, ACM, 1971, pp. 1–17.
70. Codd, E. F., *Further normalization of the database relational model*, in: *Courant Computer Science Symposia 6: Data Base Systems* (1972), pp. 33–64.
71. Codd, E. F., *Relational completeness of database sublanguages*, in: *Courant Computer Science Symposia 6: Data Base Systems* (1972), pp. 65–98.
72. Codd, E. F., *Recent investigations in relational database system*, in: *Proceedings of the IFIP Conference*, 1974, pp. 1017–1021.
73. Colmerauer, A., *Equations and inequations on finite and infinite trees*, in: *Proceedings of the International Conference on Fifth Generation Computer Systems 1984, Tokyo, Japan*, 1984, pp. 85–99.
74. Cosmadakis, S. S. and P. C. Kanellakis, *Equational theories and database constraints*, in: *Proceedings of the International Symposium on the Theory of Computing*, ACM, 1985, pp. 73–284.
75. Cosmadakis, S. S., P. C. Kanellakis and S. Spyratos, *Partition semantics for relations*, Journal of Computer and System Sciences **32** (1986), pp. 203–233.
76. Courcelle, B., *Fundamental properties of infinite trees*, Theoretical Computer Science **25** (1983), pp. 95–169.
77. Crolard, T., *Subtractive logic*, Theoretical Computer Science **254** (2001), pp. 151–185.
78. Curry, H. B., "Foundations of Mathematical Logic," Dover, 1976.
79. Czermak, J., *A remark on gentzen's calculus of sequents*, Notre Dame Journal of Formal Logic **18** (1977), pp. 471–474.
80. Dantsin, E. and A. Voronkov, *Complexity of query answering in logic databases with complex values*, in: *Logical Foundations of Computer Science, 4th International Symposium, LFCS'97, Yaroslavl, Russia, July 6-12, 1997*, number 1234 in Lecture Notes in Computer Science (1997), pp. 56–66.
81. Date, C. J., "An Introduction to Database Systems," Addison-Wesley, 1986.
82. Date, C. J. and H. Darwen, *Relation-valued attributes or will the real first normal form please stand up?*, in: *Relational Database Writings 1989-1991*, 1992, pp. 75–98.
83. Date, C. J. and H. Darwen, "A Guide to the SQL Standard," Addison-Wesley, 1993.
84. Davidson, A., M. Fuchs, M. Hedin, M. Jain, J. Koistinen, C. Lloyd, M. Maloney and K. Schwarzhof, *Schema for object-oriented XML 2.0, W3C note 30 july 1999*, http://www.w3.org/TR/NOTE-SOX/.
85. DeBra, P. and J. Paredaens, *Horizontal decompositions for handling exceptions to functional dependencies*, in: *Advances in Data Base Theory, vol.2*, 1984, pp. 123–144.
86. Delobel, C., *Normalisation and hierarchical dependencies in the relational data model*, Transactions on Database Systems (TODS) **3** (1978), pp. 201–222.

87. Delobel, C. and M. Adiba, "Relational database systems," North Holland, 1985.
88. Delobel, C. and R. C. Casey, *Decomposition of a database and the theory of boolean switching functions*, IBM Journal of Research and Development **17** (1972), pp. 370–386.
89. Demetrovics, J., *On the number of candidate keys*, Information Processing Letters **7** (1978), pp. 266–269.
90. Demetrovics, J., *On the equivalence of candidate keys with sperner sets*, Acta Cybernetica **4** (1979), pp. 247–252.
91. Demetrovics, J., *Candidate keys and antichains*, SIAM Journal on Algebraic and Discrete Methods **1** (1980), p. 92.
92. Demetrovics, J., Z. Füredi and G. O. H. Katona, *Minimum matrix representations of closure operations.*, Discrete Applied Mathematics **11** (1985), pp. 115–128.
93. Demetrovics, J. and G. Gyepesi, *On functional dependency and some generalisations of it*, Acta Cybernetica **5** (1981), pp. 295–305.
94. Demetrovics, J. and G. O. H. Katona, *Combinatorial problems of database models*, in: *Colloquia Mathematica Societatis Janos Bolyai 42, Algebra, Combinatorics and Logic in Computer Science*, 1983, pp. 331–352.
95. Demetrovics, J., L. Libkin and I. B. Muchnik, *Functional dependencies and the semilattice of closed classes*, in: *MFDBS 89, 2nd Symposium on Mathematical Fundamentals of Database Systems, Visegrád, Hungary, June 26-30, 1989*, number 364 in Lecture Notes in Computer Science (1989), pp. 135–147.
96. Demetrovics, J. and V. D. Thi, *Some results about functional dependencies*, Acta Cybernetica **8** (1988), pp. 273–278.
97. Deutsch, A., L. Popa and V. Tannen, *Physical data independence, constraints, and optimization with universal plans*, in: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1999, pp. 459–470.
98. Diederich, J. and J. Milton, *New methods and fast algorithms for database normalization*, Transactions on Database Systems (TODS) **13** (1988), pp. 339–365.
99. Dovier, A., A. Policriti and G. Rossi, *A uniform axiomatic view of lists, multisets, and sets, and the unification algorithm*, Fundamenta Informaticae **36** (1998), pp. 201–234.
100. Dummett, M., "Elements of Intuitionism," Clarendon Press, 2000.
101. Engel, K., "Sperner theory," Cambridge University Press, 1997.
102. Fagin, R., *The decomposition versus synthetic approach to relational database design*, in: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1977, pp. 441–446.
103. Fagin, R., *Multivalued dependencies and a new normal form for relational databases*, Transactions on Database Systems (TODS) **2** (1977), pp. 262–278.
104. Fagin, R., *Normal forms and relational database operators*, in: *Proceedings of the International Conference on the Management of Data (SIGMOD)*, 1979, pp. 153–160.
105. Fagin, R., *A normal form for relational databases that is based on domains and keys*, Transactions on Database Systems (TODS) **6** (1981), pp. 387–415.

106. Fagin, R., *Armstrong databases*, Technical Report RJ3440, IBM Research Laboratory, San Jose, California, USA (1982).
107. Fagin, R., *Horn clauses and data dependencies*, Journal of the ACM **29** (1982), pp. 952–985.
108. Fagin, R., A. O. Mendelzon and J. D. Ullman, *A simplified universal relational assumption and its properties*, Transactions on Database Systems (TODS) **7** (1982), pp. 343–360.
109. Fagin, R. and M. Y. Vardi, *The theory of data dependencies: a survey*, in: *Mathematics of Information Processing: Proceedings of Symposia in Applied Mathematics* (1986), pp. 19–71.
110. Fan, W. and L. Libkin, *On XML integrity constraints in the presence of DTDs*, Journal of the ACM **49** (2002), pp. 368–406.
111. Fan, W. and J. Siméon, *Integrity constraints for XML*, Journal of Computer and System Sciences **66** (2003), pp. 254–291.
112. Fischer, P. C., J. H. Jou and D. M. Tsou, *Succinctness in dependency systems*, Theoretical Computer Science **24** (1983), pp. 323–329.
113. Fischer, P. C., L. V. Saxton, S. J. Thomas and D. van Gucht, *Interactions between dependencies and nested relational structures*, Journal of Computer and System Sciences **31** (1985), pp. 105–129.
114. Fishman, D. H., D. Beech, H. P. Cate, E. F. Chow, T. Connors, J. W. Davis, N. Derret, C. G. Hoch, W. Kent, P. Lyngbæk, B. Mahbod, M.-A. Neimat, T. A. Ryan and M.-C. Shan, *IRIS:an object-oriented database management system*, Transactions on Office Information Systems **5** (1987), pp. 48–69.
115. Fleming, C. C. and B. van Halle, "Handbook of Relational Database Design," Addison-Wesley, 1989.
116. Florescu, D., L. Raschid and P. Valduriez, *A methodology for query reformulation in cis using semantic knowledge*, International Journal of Cooperative Information Systems (IJCIS) **5** (1996), pp. 431–468.
117. Floyd, R. W. and L. Steinberg, *An adaptive algorithm for spatial grey scale*, in: *Proceedings of the Society of Information Display*, 1976, pp. 75–77.
118. Galil, Z., *An almost linear-time algorithm for computing a dependency basis in a relational database*, Journal of the ACM **29** (1982), pp. 96–102.
119. Gallaire, H. and J. Minker, "Logic and Databases," Plenum Press, New York, 1978.
120. Ganascia, J.-G., *Algebraic structure of some learning systems*, in: *Algorithmic Learning Theory, 4th International Workshop, ALT '93, Tokyo, Japan*, number 744 in Lecture Notes in Computer Science (1993), pp. 398–409.
121. Gardarin, G., J.-P. Cheiney, G. Kiernan, D. Pastre and H. Stora, *Managing complex objects in an extensible relational DBMS*, in: *Proceedings of the International Conference on Very Large Data Bases* (1989), pp. 55–65.
122. Garey, M. R. and D. S. Johnson, "Computers and Intractibility: A Guide to the Theory of NP-Completeness," Freeman, San Franciso, 1979.
123. Gertz, M. and U. W. Lipeck, *"Temporal" integrity constraints in temporal databases*, in: *Recent Advances in Temporal Databases, Proceedings of the International Work-*

shop on Temporal Databases, Zürich, Switzerland, 17-18 September 1995, 1995, pp. 77–92.

124. Ginsburg, S. and R. Hull, *Order dependency in the relational model*, Theoretical Computer Science **26** (1983), pp. 149–195.
125. Ginsburg, S. and S. M. Zaiddan, *Properties of functional dependency families*, Journal of the ACM **29** (1982), pp. 678–698.
126. Gischer, J. L., *The equational theory of pomsets*, Theoretical Computer Science **61** (1988), pp. 199–224.
127. Gödel, K., *Die Vollständigkeit der Axiome des logischen Funktionenkalküls*, Monatshefte der Mathematischen Physik (1930), pp. 349–360.
128. Goldberg, A. and D. Robson, "Smalltalk-80: The Language and Its Implementation," Addison-Wesley, 1980.
129. Goodman, N. D., *The logic of contradiction*, Zeitschrift für Mathematische Logik und Grundlagen der Mathematik **27** (1981), pp. 119–126.
130. Gore, R., *Dual intuitionistic logic revisited*, in: *TABLEAUX00: Automated Reasoning with Analytic Tableaux and Related Methods*, number 1847 in Lecture Notes in Artificial Intelligence (2000), pp. 252–267.
131. Gorn, S., *Explicit definitions and linguistic dominoes*, in: *Systems and Computer Science*, 1967, pp. 77–115.
132. Gottlob, G., *Computing covers for embedded functional dependencies*, in: *Principles of Database Systems (PODS)*, 1987, pp. 58–69.
133. Grahne, G. and K. P. Räihä, *Database decomposition into 4NF*, in: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1983, pp. 186–196.
134. Gyssens, M., J. Paredaens and D. Van Gucht, *A graph-orienteed object database model*, in: *Principles of Database Systems (PODS)*, 1990, pp. 417–424.
135. Hagihara, K., M. Ito, K. Taniguchi and T. Kasami, *Decision problems for multivalued dependencies in relational databases*, SIAM Journal on Computing **8** (1979), pp. 247–264.
136. Hammer, M. and D. McLeod, *Database description with SDM: A semantic database model*, Transactions on Database Systems (TODS) **6** (1981), pp. 351–386.
137. Hara, C. S. and S. B. Davidson, *Reasoning about nested functional dependencies*, in: *Principles of Database Systems (PODS)* (1999), pp. 91–100.
138. Hartmann, S. and S. Link, *More functional dependencies for XML*, in: *Advances in Databases and Information Systems, 7th East European Conference, ADBIS 2003, Dresden, Germany, September 3-6*, number 2798 in Lecture Notes in Computer Science (2003), pp. 355–369.
139. Hartmann, S. and S. Link, *On functional dependencies in advanced data models*, Electronic Notes in Theoretical Computer Science (ENTCS) **84** (2003).
140. Hartmann, S. and S. Link, *The implication problem of functional dependencies in complex-value databases*, accepted for Electronic Notes in Theoretical Computer Science (ENTCS) (2004).
141. Hartmann, S. and S. Link, *A membership algorithm for functional and multi-valued*

dependencies in the presence of lists, Electronic Notes in Theoretical Computer Science (ENTCS) **91** (2004), pp. 171–194.

142. Hartmann, S. and S. Link, *Multi-valued dependencies in the presence of lists*, in: *Proceedings of the 23rd International Conference on Principles of Database Systems (PODS)*, ACM, 2004, pp. 330–341.
143. Hartmann, S. and S. Link, *Normalisation in the presence of lists*, in: *15th Australasian Database Conference (ADC)*, Conferences in Research and Practice in Information Technology **27** (2004), pp. 53–64.
144. Hartmann, S., S. Link and M. Kirchberg, *A subgraph-based approach towards functional dependencies for XML*, in: *Proceedings of the 7th World-Multiconference on Systemics, Cybernetics and Informatics (SCI), Volume IX, Computer Science and Engineering II, Orlando, Florida, USA, July 27 - 30*, 2003, pp. 200–205.
145. Hartmann, S., S. Link and K.-D. Schewe, *Functional dependencies in the presence of records, lists, sets and multisets*, Technical Report 1, Department of Information Systems, Massey University, Palmerston North, New Zeland (2004).
146. Hartmann, S., S. Link and K.-D. Schewe, *Reasoning about functional and multi-valued dependencies in the presence of lists*, in: *3rd International Symposium on Foundations of Information and Knowledge Systems (FoIKS)*, number 2942 in Lecture Notes in Computer Science (2004), pp. 134–154.
147. Hull, R., *Finitely specifiable implicational dependency families*, Journal of the ACM **31** (1984), pp. 210–226.
148. Hull, R., *A survey of theoretic research on typed complex database objects*, in: *Databases* (1987), pp. 193–256.
149. Hull, R. and R. King, *Semantic database modeling: Survey, applications and research issues*, ACM Computing Surveys **19** (1987).
150. Hull, R., K. Tanaka and M. Yoshikawa, *Behavior analysis of object-oriented databases: Method structure, execution trees and reachability*, in: *Proceedings 3rd International Conference on Foundations of Data Organization and Algorithms*, 1989, pp. 372–388.
151. Hull, R. and C. K. Yap, *The Format model: A theory of data organization*, Journal of the ACM **31** (1984), pp. 518–537.
152. Ito, M., M. Iwasaki, K. Taniguchi and K. Kasami, *Membership problems for data dependencies in relational expressions*, Theoretical Computer Science **34** (1984), pp. 315–335.
153. Ito, M. and G. E. Weddell, *Implication problems for functional constraints on databases supporting complex objects*, Journal of Computer and System Sciences **50** (1995), pp. 165–187.
154. Jajoda, S., *Recognizing multivalued dependencies in relation schemas*, The Computer Journal **29** (1986), pp. 458–459.
155. Jarvis, J. F., C. N. Judice and W. H. Ninke, *A survey of techniques for the display of continuous-tone pictures on bilevel display*, Computer Graphics Image Processing **5** (1976), pp. 13–40.

156. Jäschke, G. and H.-J. Schek, *Remarks on the algebra of non first normal form relations*, in: *Principles of Database Systems (PODS)*, 1982, pp. 124–138.
157. Kandzia, P. and M. Mangelmann, *On covering Boyce-Codd normal form*, Information Processing Letters **11** (1980), pp. 218–223.
158. Kanellakis, P. C., "Elements of Relational Database Theory," Elsevier, 1990 pp. 1074–1156.
159. Kari, L., G. Paun, G. Rozenberg, A. Salomaa and S. Yu, *DNA computing, sticker systems, and universality*, Acta Informatica **35** (1998), pp. 401–420.
160. Katsavounidis, I. and C.-C. J. Kuo, *A multiscale error diffusion technique for digital halftoning*, Transactions on Image Processing **6** (1997), pp. 483–490.
161. Kent, W., "Data and Reality," North Holland, 1978.
162. Kent, W., *Limitations of record-based information models*, Transactions on Database Systems (TODS) **4** (1979), pp. 107–131.
163. Kent, W., *The many forms of a single fact*, in: *Proceedings of the IEEE Compcon Conference*, 1989.
164. Kerschberg, L. and J. E. S. Pacheco, *A functional data base model*, Technical report, Pontificia Univ. Catolica do Rio de Janeiro, Rio de Janeiro, Brasil (1976).
165. Kifer, M., G. Lausen and J. Wu, *Logical foundations of object-oriented and frame-based languages*, Technical Report 93/06, Computer Science Department, SUNY at Stony Brook, New York (1993).
166. Kim, W. and F. Lochovski, editors, "Object-Oriented Concepts, Databases and Applications," Addison-Wesley, 1989.
167. Kivinen, J. and H. Mannila, *Approximate inference of functional dependencies from relations*, Theoretical Computer Science **149** (1995), pp. 129–149.
168. Kobayashi, I., *Databases and conceptual schemata: A formal framework*, in: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1986, pp. 3–23.
169. Koubarakis, M., *Databases and temporal constraints: Semantics and complexity*, in: *Recent Advances in Temporal Databases, Proceedings of the International Workshop on Temporal Databases, Zürich, Switzerland, 17-18 September 1995*, 1995, pp. 93–109.
170. Kuper, G., *On the expressive power of logic programming languages with sets*, in: *Principles of Database Systems (PODS)*, 1988, pp. 10–14.
171. Kuper, G., *Logic programming with sets*, Journal of Computer and System Sciences **41** (1990), pp. 44–64.
172. Kuper, G. and M. Y. Vardi, *A new approach to database logic*, in: *Principles of Database Systems (PODS)*, 1984, pp. 86–96.
173. Lakshmanan, V. S. and C. E. VeniMadhavan, *An algebraic theory of functional and multivalued dependencies in relational databases*, Theoretical Computer Science **54** (1987), pp. 103–128.
174. Lamperti, G., M. Melchiori and M. Zanella, *On multisets in database systems*, in: *Workshop on Multiset Processing (WMP)*, number 2235 in Lecture Notes in Computer Science (2000), pp. 147–216.

175. Lawvere, F. W., *Introduction*, in: *Categories in Continuum Physics*, Lecture Notes in Mathematics **1174** (1986).
176. Lawvere, F. W., *Intrinsic co-heyting boundaries and the leibniz rule in certain toposes*, in: *Category Theory*, Lecture Notes in Mathematics **1488** (1991), pp. 279–297.
177. Layman, A., E. Jung, E. Maler, H. S. Thompson, J. Paoli, J. Tigue, N. H. Mikula and S. De Rose, *XML-data, W3C note 05 jan 1998*, http://www.w3.org/TR/1998/NOTE-XML-data/.
178. Lee, M., T. Ling and W. Low, *Designing functional dependencies for XML*, in: *Advances in Database Technology - EDBT 2002, 8th International Conference on Extending Database Technology, Prague, Czech Republic, March 25-27*, number 2287 in Lecture Notes in Computer Science (2002), pp. 124–141.
179. Levene, M., "The Nested Universal Relation Database Model," Springer, 1992.
180. Levene, M. and G. Loizou, *Semantics for null extended nested relations*, Transactions on Database Systems (TODS) **18** (1993), pp. 414–459.
181. Levene, M. and G. Loizou, "A Guided Tour of relational databases and beyond," Springer, 1999.
182. Li, B., *Fuzzy bags and applications*, Fuzzy sets and systems **34** (1990), pp. 61–71.
183. Li, J., S. Ng and L. Wong, *Bioinformatics adventures in database research*, in: *Proceedings of the International Conference on Database Theory (ICDT)*, number 2572 in Lecture Notes in Computer Science (2002), pp. 31–46.
184. Lien, Y. E., *On the semantics of the entity-relationship model*, in: *Entity-Relationship Approach to Systems Analysis and Design*, 1980, pp. 155–167.
185. Lien, Y. E., *On the equivalence of data models*, Journal of the ACM **29** (1982), pp. 333–363.
186. Ling, T., F. Tompa and T. Kameda, *An improved third normal form for relational databases*, Transactions on Database Systems (TODS) **6** (1981), pp. 326–346.
187. Ling, T. W. and L. L. Yan, *NF-NR: A practical normal form for nested relations*, Journal of Systems Integration **4** (1994), pp. 309–340.
188. Lipeck, U. W. and G. Saake, *Monitoring dynamic integrity constraints based on temporal logic*, Information Systems **12** (1987), pp. 255–269.
189. Lopes, S., J.-M. Petit and L. Lakhal, *Efficient discovery of functional dependencies and armstrong relations*, in: *Proceedings of the 7th International Conference on Extending Database Technology: Advances in Database Technology*, number 1777 in Lecture Notes In Computer Science (2000), pp. 350–364.
190. Lucchesi, C. L. and S. L. Osborne, *Candidate keys for relations*, Journal of Computer and System Sciences **17** (1978), pp. 270–279.
191. Maier, D., *Minimum covers in relational database model*, Journal of the ACM **27** (1980), pp. 664–674.
192. Maier, D., "The Theory of Relational Databases," Computer Science Press, 1983.
193. Maier, D., A. O. Mendelzon, F. Sadri and J. D. Ullman, *Adequacy of decompositions of relational databases*, Journal of Computer and System Sciences **21** (1980), pp. 368–379.

194. Majid, S., *On the nature of physics*, http://www.maths.qmw.ac.uk/ majid/pessay.html.
195. Makinouchi, A., *A consideration of normal form of not-necessarily-normalized relations in the relational database model*, in: *Proceedings of International Conference on Very Large Data Bases (VLDB)*, 1977, pp. 447–453.
196. Mannila, H. and K. P. Räihä, *Small armstrong relations for database design*, in: *Principles of Database Systems (PODS)*, 1985, pp. 245–50.
197. Mannila, H. and K. P. Räihä, "The Design of Relational Databases," Addison-Wesley, 1992.
198. Mannila, H. and K. P. Räihä, *On the complexity of inferring functional dependencies*, Discrete Applied Mathematics **40** (1992), pp. 237–243.
199. Mannila, H. and K. P. Räihä, *Algorithms for inferring functional dependencies from relations*, Data & Knowledge Engineering **12** (1994), pp. 83–99.
200. Manola, F. and E. Miller, *Resource description framework primer, W3C recommendation 10 february 2004*, http://www.w3.org/TR/2004/REC-rdf-primer-20040210/.
201. McKinsey, J. C. C. and A. Tarski, *The algebra of topology*, Annals of Mathematics **45** (1944), pp. 141–191.
202. McKinsey, J. C. C. and A. Tarski, *On closed elements in closure algebras*, Annals of Mathematics **47** (1946), pp. 122–146.
203. Melton, J., *An SQL3 snapshot*, in: *Proceedings of the IEEE Conference on Data Engineering*, 1996, pp. 666–672.
204. Mendelzon, A. O., *On axiomatising multivalued dependencies in relational databases*, Journal of the ACM **26** (1979), pp. 37–44.
205. Mitsa, T. and K. Parker, *Digital halftoning using a blue-noise mask*, in: *Image Processing Algorithms and Techniques II*, The International Society for Optical Engineering (SPIE) **1452**, 1991, pp. 47–56.
206. Mok, W. Y., *A comparative study of various nested normal forms*, IEEE Transactions on Knowledge & Data Engineering **14** (2002), pp. 369–385.
207. Mok, W. Y., Y. K. Ng and D. W. Embley, *A normal form for precisely charachterizing redundancy in nested relations*, Transactions on Database Systems (TODS) **21** (1996), pp. 77–106.
208. Naqvi, S. and S. Tsur, "A logical language for data and knowledge bases," Computer Science Press, 1989.
209. National Centre for Biotechnology Information, *Genbank overview*, http://www.ncbi.nih.gov/Genbank/GenbankOverview.html.
210. National Centre for Biotechnology Information, *Sample genbank record and file format description*, http://www.ncbi.nih.gov/Sitemap/samplerecord.html.
211. Ng, W., *Ordered functional dependencies in relational databases*, Information Systems **27** (1999), pp. 535–554.
212. Nicolas, J.-M., *First order logic formalization for functional, multivalued and mutual dependencies*, in: *Proceedings of the International Conference on Management of Data (SIGMOD)*, ACM, 1978, pp. 40–46.
213. Nicolas, J.-M., *Logic for improving integrity checking in relational databases*, Acta Informatica **18** (1982), pp. 227–253.

214. Osborne, S. L., *Testing for the existence of a covering Boyce-Codd normal form*, Information Processing Letters **8** (1979), pp. 11–14.
215. Özsoyoglu, Z. M. and L. Y. Yuan, *A new normal form for nested relations*, Transactions on Database Systems (TODS) **12** (1987), pp. 111–136.
216. Özsoyoglu, Z. M. and L. Y. Yuan, *Reduced MVDs and minimal covers*, Transactions on Database Systems (TODS) **12** (1987), pp. 377–394.
217. Özsoyoglu, Z. M. and L. Y. Yuan, *On the normalisation in nested relational databases*, in: *Nested Relations and Complex Objects in Databases*, number 361 in Lecture Notes in Computer Science (1989), pp. 243–271.
218. Pagliani, P., *Intrinsic co-heyting boundaries and information incompleteness in rough set analysis*, in: *Rough Sets and Current Trends in Computing*, number 1424 in Lecture Notes in Computer Science, 1998, pp. 123–130.
219. Pappas, T. and D. L. Neuhoff, *Least-squares model-based halftoning*, in: *Human Vision, Visual Processing and Digital Display III*, International Society for Optical Engineering (SPIE) **1666**, 1992, pp. 165–176.
220. Paredaens, J., P. De Bra, M. Gyssens and D. Van Gucht, "The Structure of the Relational Database Model," Springer-Verlag, 1989.
221. Paredaens, J. and D. Janssens, *Decompositions of relations: a comprehensive approach*, in: *Advances in Data Base Theory, vol. 1* (1981), pp. 73–100.
222. Paredaens, J. and D. van Gucht, *Possibilities and limitations of using flat operators in nested algebra expressions*, in: *Principles of Database Systems (PODS)*, 1988, pp. 29–38.
223. Parker, D. S. and C. Delobel, *Algorithmic applications for a new result on multivalued dependencies*, in: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1979, pp. 67–74.
224. Paun, G., *DNA computing: Distributed splicing systems*, in: *Structures in Logic and Computer Science*, number 1261 in Lecture Notes in Computer Science (1997), pp. 353–370.
225. Paun, G., *Computing with membranes*, Journal of Computer and System Sciences **61** (2000), pp. 108–143.
226. Paun, G. and G. Rozenberg, *A guide to membrane computing*, Theoretical Computer Science **287** (2002), pp. 73–100.
227. Pearl, J., "Probabilistic Reasoning in Intelligent Systems," Morgan Kaufman, 1988.
228. Pearl, J. and T. Verma, *The logic of representing dependencies by directed graphs*, in: *Proceedings AAAI Conference 1987*, 1988, pp. 374–379.
229. Petrov, S. V., *Finite axiomatization of languages for representation of system properties: Axiomatization of dependencies*, Information Sciences **47** (1989), pp. 339–372.
230. Rauszer, C., *Semi-boolean algebras and their application to intuitionistic logic with dual operations*, Fundamenta Mathematicae **LXXIII** (1974), pp. 219–249.
231. Rauszer, C., *An algebraic and kripke-style approach to a certain extension of intuitionistic logic*, Technical report, Institute of Mathematics, Polish Academy of Sciences (1980).

232. Reps, T. W., *Algebraic properties of program integration*, Science of Computer Programming **17** (1991), pp. 139–215.
233. Richardson, J., *Supporting lists in a datamodel*, in: *Proceeding of the International Conference on Very Large Data Bases (VLDB)*, 1992, pp. 127–192.
234. Rissanen, J., *Independent components of relations*, Transactions on Database Systems (TODS) **2** (1977), pp. 317–325.
235. Rissanen, J., *Theory of relations for databases - a tutorial survey*, in: *Proceedings of 7th Symposium on Mathematical Foundations of Computer Science*, number 64 in Lecture Notes in Computer Science, 1978, pp. 536–551.
236. Rissanen, J. and C. Delobel, *Decomposition of files, a basis for data storage and retrieval*, Technical Report RJ2220, IBM Reseach Lab (1975).
237. Roth, M. A. and H. F. Korth, *The design of ¬1nf relational databases into nested normal form*, in: *Proceedings of the International Conference on Management of Data (SIGMOD)*, 1987, pp. 143–159.
238. Roth, M. A., H. F. Korth and A. Silberschatz, *Theory of non-first-normal form relational databases*, Technical Report TR-84-36, University of Texas, Austin, Texas, USA (1986).
239. Sagiv, Y., *An algorithm for inferring multivalued dependencies with an application to propositional logic*, Journal of the ACM **27** (1980), pp. 250–262.
240. Sagiv, Y., C. Delobel, D. S. Parker Jr. and R. Fagin, *An equivalence between relational database dependencies and a fragment of propositional logic*, Journal of the ACM **28** (1981), pp. 435–453.
241. Sanchez, E., *Solutions in composite fuzzy relation equations: application to medical diagnosis in Brouwerian logic*, in: *Fuzzy Automata and Decision Processes*, 1977, pp. 221–234.
242. Savnik, I. and P. A. Flach, *Discovery of multivalued dependencies from relations*, Intelligent Data Analysis **4** (2000), pp. 195–211.
243. Schewe, K.-D. and B. Thalheim, *Fundamental concepts of object oriented databases*, Acta Cybernetica **11** (1993), pp. 49–85.
244. Scholl, M. H. and H.-J. Schek, *A relational object model*, in: *Proceedings of International Conference on Database Theory (ICDT)* (1990), pp. 89–105.
245. Schreiber, W. F., "Fundamentals of Electronic Imaging Systems: Some Aspects of Image Processing," Springer, 1986.
246. Schulze, M. and T. Pappas, *Blue noise and model-based halftoning*, in: *Human Vision, Visual Processing and Digital Display V*, International Society for Optical Engineering (SPIE) **2179**, 1994, pp. 181–194.
247. Sciore, E., *Real-world MVDs*, in: *Proceedings of the International Conference on Management of Data (SIGMOD)*, 1981, pp. 121–132.
248. Seshadri, P., M. Livny and R. Ramakrishnan, *The design and implementation of sequence database system*, in: *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 1996, pp. 99–110.
249. Shapiro, V., *Maintenance of geometric representations through space decomposi-*

tions, International Journal of Computational Geometry and Applications **6** (1997), pp. 383–418.
250. Shipman, D., *The functional data model and the data language DAPLEX*, Transactions on Database Systems (TODS) **6** (1981), pp. 140–173.
251. Silva, A. M. and A. Melkanoff, *A method for helping discover the dependencies of a relation*, in: *Advances in Database Theory* (1981), pp. 115–133.
252. Speer, J., *Implementing a membership algorithm for functional and multi-valued dependencies in the presence of lists*, Technical report, Dept of Information Systems, Massey University, Palmerston North, New Zealand (2004).
253. Stell, J. G. and M. F. Worboys, *The algebraic structure of sets of regions*, in: *Proceedings Spatial Information Theory (COSIT)*, 1997, pp. 163–174.
254. Stevenson, R. L. and G. R. Arce, *Binary display of hexagonally sampled continuous-tone images*, Journal of the Optical Society of America A: Optics, Image Science, and Vision **2** (1985), pp. 1009–1013.
255. Stone, M. H., *Topological representations of distributive lattices and Brouwerian logics*, Casopis pro pestovani matematiky a fysiky **67** (1937-1938), pp. 1–25.
256. Stonebraker, M., P. Brown and D. Moore, "Object-relational DBMSs: Tracking the Next Great Wave 2e," Morgan Kaufman, 1999.
257. Stroustrup, B., "The C++ Programming Language, 2d ed." Addison-Wesley, 1991.
258. Stucki, P., *Mecca—a multiple-error correcting computation algorithm for bilevel image hardcopy reproduction*, Technical Report RZ1060, IBM Research Lab (1981).
259. Su, S. Y. W., *SAM*: A semantic association model for corporate and scientific statistical databases*, Information Sciences **29** (1983), pp. 151–199.
260. Suzuki, Y., Y. Fujiwara, J. Takabayashi and H. Tanaka, *Artificial life applications of a class of P systems: Abstract rewriting systems on multisets*, in: *Workshop on Multiset Processing (WMP)*, number 2235 in Lecture Notes in Computer Science (2000), pp. 299–346.
261. Suzuki, Y. and H. Tanaka, *A new moleculer computing model, artifical cell systems*, in: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), Las Vegas, Nevada, USA* (2000), pp. 833–840.
262. Tari, Z., J. Stokes and S. Spaccapietra, *Object normal forms and dependency constraints for object-oriented schemata*, Transactions on Database Systems (TODS) **22** (1997), pp. 513–569.
263. Tarski, A., *Der Aussagenkalkül und die Topologie*, Fundamenta Mathematicae **25** (1938), pp. 103–134.
264. Thalheim, B., "Dependencies in Relational Databases," Teubner-Verlag, 1991.
265. Thalheim, B., "Entity-Relationship Modeling: Foundations of Database Technology," Springer-Verlag, 2000.
266. Thalheim, B., *Conceptual treatment of multivalued dependencies*, in: *Proceedings of the International Conference on Conceptual Modeling (ER)*, number 2813 in Lecture Notes in Computer Science (2003), pp. 363–375.
267. Thomas, S. J. and P. C. Fischer, *Nested relational structures*, Advances in Computing Research **3** (1986), pp. 269–307.

268. Thompson, H. S., D. Beech, M. Maloney and N. Mendelsohn, *XML schema, W3C recommendation, may 2001*, http://www.w3.org/XML/Schema.
269. Tjoa, A. M. and L. Berger, *Transformation of requirement specifications expressed in natural language into an EER model*, in: *Entity-Relationship Approach*, number 823 in Lecture Notes in Computer Science (1993), pp. 206–217.
270. Tsou, D.-M. and P. C. Fischer, *Decomposition of a relation scheme into Boyce-Codd normal form*, SIGACT News **14** (1982), pp. 23–29.
271. Ulichney, R. A., "Digital Halftoning," Cambridge, MA: MIT Press, 1987.
272. Ulichney, R. A., *The void-and-cluster method for dither array generation*, in: *Human Vision, Visual Processing and Digital Display IV*, International Society for Optical Engineering (SPIE) **1913**, 1993, pp. 332–343.
273. Ullman, J. D., "Principles of Database Systems," Computer Science Press, Potomac, Maryland, 1979.
274. Ullman, J. D., "Principles of Database and Knowledge Base Systems, vol. I," Computer Science Press, Rockville, 1988.
275. Urbas, I., *Dual intuitionistic logic*, Notre Dame Journal of Formal Logic **37** (1996), pp. 440–451.
276. Vardi, M. Y., *On decomposition of relational databases*, in: *Conference on Foundations of Computer Science* (1982), pp. 176–185.
277. Vardi, M. Y., *Inferring multivalued dependencies from functional and join dependencies*, Acta Informatica **19** (1983), pp. 305–324.
278. Vardi, M. Y., *Fundamentals of dependency theory*, in: E. Börger, editor, *Trends in Theoretical Computer Science* (1987), pp. 171–224.
279. Vincent, M., *Modification anomalies and Boyce-Codd normal form*, Research and Practical Issues in Databases (1992), pp. 251–264.
280. Vincent, M., "The semantic justification for normal forms in relational database design," Ph.D. thesis, Monash University, Melbourne, Australia (1994).
281. Vincent, M., *Insertion anomalies and the justification for 4NF in relational databases*, Australian Computer Science Communications **17** (1995), pp. 540–545.
282. Vincent, M., *A corrected 5NF definition for relational database design*, Theoretical Computer Science **185** (1997), pp. 379–391.
283. Vincent, M., *Semantic foundation of 4NF in relational database design*, Acta Informatica **36** (1999), pp. 1–41.
284. Vincent, M. and M. Levene, *Restructuring partitioned normal form relations without information loss*, SIAM Journal on Computing **29** (2000), pp. 1550–1567.
285. Vincent, M. and J. Liu, *Functional dependencies for XML*, in: M. O. X. Zhou, Y. Zhang, editor, *Web Technologies and Applications: 5th Asia-Pacific Web Conference, APWeb 2003, Xian, China, April 23-25, 2003. Proceedings*, number 2642 in Lecture Notes in Computer Science (2003), pp. 22–34.
286. Vincent, M. and J. Liu, *Multivalued dependencies in XML*, in: *British National Conference on Databases*, number 2712 in Lecture Notes in Computer Science (2003), pp. 4–18.

287. Vincent, M., J. Liu and C. Liu, *A redundancy free 4NF for XML*, in: *Proceedings of the XML Database Symposium*, number 2824 in Lecture Notes in Computer Science (2003), pp. 254–266.
288. Vincent, M. and B. Srinivasan, *Armstrong relations for functional and multivalued dependencies in relational databases*, in: *Advances in Database Research* (1993), pp. 317–328.
289. Vincent, M. and B. Srinivasan, *Redundancy and the justification of fourth normal form in relational databases*, International Journal of Foundations of Computer Science **4** (1993), pp. 355–365.
290. Vincent, M. and B. Srinivasan, *Update anomalies and the justification of fourth normal form in relational databases*, Information Sciences **81** (1994), pp. 87–102.
291. Vorobyov, S. G. and A. Voronkov, *Complexity of nonrecursive logic programs with complex values*, in: *Principles of Database Systems (PODS)*, 1998, pp. 244–253.
292. Vossen, G., "Data Models, Database Languages and Database Management Systems," Addison-Wesley, 1991.
293. Wang, C. P. and H. H. Wedekind, *Segment synthesis in logical data base design*, IBM Journal on Research and Development **19** (1975), pp. 71–77.
294. Wang, X., C. Bettini, A. Brodsky and S. Jajodia, *Logical design for temporal databases with multiple granularities*, Transaction on Database Systems **22** (1997), pp. 115–170.
295. Weddell, G. E., *Reasoning about functional dependencies generalized for semantic data models*, Transactions on Database Systems (TODS) **17** (1992), pp. 32–64.
296. Wijsen, J., *Design of temporal relational databases based on dynamic and temporal functional dependencies*, in: *Recent Advances in Temporal Databases, Proceedings of the International Workshop on Temporal Databases, Zürich, Switzerland, 17-18 September 1995*, 1995, pp. 61–76.
297. Wijsen, J., *Temporal FDs on complex objects*, Transactions on Database Systems (TODS) **24** (1999), pp. 127–176.
298. Wolter, F., *On logics with coimplication*, Journal of Philosophical Logic **27** (1998), pp. 353–387.
299. Yager, R. R., *On the theory of bags*, International Journal of General Systems **13** (1986), pp. 23–37.
300. Yan, M. H. and A. W.-C. Fu, *Algorithm for discovering multivalued dependencies*, in: *Proceedings of the International Conference on Information and Knowledge Management*, ACM, 2001, pp. 556–558.
301. Yuan, L. Y. and Z. M. Özsoyoglu, *Logical design of relational database schemes*, in: *Principles of Database Systems (PODS)*, ACM, 1987, pp. 38–47.
302. Yuan, L. Y. and Z. M. Özsoyoglu, *Design of desirable database schemes*, Journal of Computer and System Sciences **45** (1992), pp. 435–470.
303. Zaniolo, C., "Analysis and Design of Relational Schemata for Database Systems," Ph.D. thesis, UCLA, Technical Report UCLA-ENG-7769 (1976).
304. Zaniolo, C., *A new normal form for the design of relational database schemata*, Transactions on Database Systems (TODS) **7** (1982), pp. 489–499.

305. Zaniolo, C., *Key constraints and monotonic aggregates in deductive databases*, in: *Computational Logic: Logic Programming and Beyond*, 2002, pp. 109–134.
306. Zdonik, S. B. and D. Maier, editors, "Readings in Object-Oriented Database Systems," Morgan Kaufman, 1990.