On Interactions of Cardinality Constraints, Key and Functional Dependencies

Sven Hartmann

FB Mathematik, Universität Rostock, 18051 Rostock, Germany

August 12, 1999

Abstract. Cardinality constraints as well as key dependencies and functional dependencies are among the most popular classes of constraints in database models. While the formal properties of each of the constraint classes are now well understood, little is known about their interaction. The objective of this paper is to discuss how constraints from these classes go together. We propose methods for reasoning about a set of cardinality constraints, key and certain functional dependencies. Moreover, we construct Armstrong databases for these constraints, which are of special interest for example-based deduction in database design.

1 Introduction

It is widely accepted now that database systems are best designed first at a conceptual level. The result of this process is a conceptual schema which describes the requirements that the desired database system must achieve, and serves as the basis for the following development phases. In conceptual design great attention is devoted to the modeling of semantics. Integrity constraints specify the way by that the elements of a database are associated to each other. Defining and enforcing integrity constraints helps to guarantee system correctness.

During the last few decades, the area of integrity constraints has attracted considerable research interest. A large amount of different constraint classes has been discussed in the literature and actually used in database design. However, their mathematical rigour often entails complex semantics, and consequently handling them in a proper manner requires a good knowledge about their backgrounds. For an overview on semantics in databases, see e.g. [15, 17, 18].

The entity-relationship approach to conceptual database design, first introduced by Chen [4], provides a simple way of representing data in the form of entities and relationships among entities. In this approach, cardinality constraints are among the most popular classes of integrity constraints. They impose lower and upper bounds on the number of relationships an entity of a given type may be involved in. For cardinality constraints and generalizations, see [14, 20].

Once declared, a conceptual schema can be mapped in an automatic way to a variety of implementation targets such as a relational database. The relational database model, which goes back to Codd [5], has a strong theoretical background which can be reused in most other database models. Extended entity-relationship models support all the major concepts of the relational model. For details and further discussion, we refer to [19].

Codd [5] also introduced functional dependencies, which are to be considered as one of the major concepts of the relational model. In particular, normalization theory up to BCNF is based on the idea of functional dependency. Among functional dependencies, key dependencies are of utmost interest. They are used to identify the tuples in a relation.

Several authors extended functional dependencies to other database models such as object-based databases or temporal databases, see [19, 22, 23]. It is reasonable to ask how the extensions affect other constructs within these models. As pointed out in [19], results obtained from relational database theory can also be used in the entity-relationship model. This is of interest since the entity-relationship model is not only popular as an intuitive tool for conceptual design, but it becomes more and more prevalent as a mathematical data model upon which real database management systems are built.

While each constraint class mentioned above is now quite well understood, we know little about their interaction. However, in practice the designer of a database will be confronted with the presence of constraints of either class. In order to ensure the correctness of a schema, the designer has (among others) to check the satisfiability of the specified constraint set. Unfortunately, efficient algorithms for consistency and implication checking are still missing for this situation. It should be noted that reasoning about a constraint set containing constraints from different classes happens to be significantly harder than handling the classes separately. It is well-known that constraints from the classes under consideration interact with each other. For example, key dependencies can be used to express certain cardinality constraints, and vice versa. These interactions may cause a database schema to exhibit undesirable properties.

The objective of this paper is to discuss possible interactions between cardinality constraints and key or functional dependencies. In particular, we present methods for reasoning about a set of cardinality constraints, key dependencies and certain functional dependencies, namely non-unary functional dependencies.

This paper is organized as follows. In Section 2, we briefly describe the data model to be used. All our considerations are carried out in the higher-order entity relationship model (HERM, see [19]) which extends the original approach by Chen. In Section 3, we give a formal definition of the constraints mentioned above. Moreover, the consistency and the implication problem for constraint classes are addressed. Section 4 collects some known results about cardinality constraints to be used in further investigations. Section 5 is devoted to the consistency problem, Section 6 to the implication problem. In Section 7 we discuss some problems concerning unary functional dependencies. Due to the lack of space, proofs are omitted or sketched only. The proofs of the major theorems are given in the final section.

2 The data model

This section briefly describes the basic concepts of the entity-relationship approach. We concentrate ourselves on a characteristic subset of design constructs to be used below.

Let $\underline{E}$ be a non-empty, finite set. In the context of our approach, the elements of $\underline{E}$ are called *entity types*. With each entity type $\underline{e}$ we associate a finite set $\underline{e}^t$ called the domain or population of the type $\underline{e}$ (at moment t). The members of $\underline{e}^t$ are entity instances or entities, for short. Intuitively, entities can be seen as real-world objects which are of interest for some application or purpose. By classifying them and specifying their significant properties (attributes), we obtain entity types which are frequently used to model the objects in their domains.

A *relationship type* $\underline{r}$ is a set (or sequence) $\{\underline{e}_1, \ldots, \underline{e}_k\}$ of elements from $\underline{E}$. Relationship types are used to model associations between real-world objects, i.e. entities. A relationship or instance of type $\underline{r}$ is a tuple from the cartesian product $\underline{e}_1^t \times \ldots \times \underline{e}_k^t$ where $\underline{e}_1^t, \ldots, \underline{e}_k^t$ are the domains of $\underline{e}_1, \ldots, \underline{e}_k$, respectively, at moment t. A finite set $\underline{r}^t$ of such relationships forms the population of $\underline{r}$ at moment t.

The relationship types considered so far are often called relationship types of order 1 (cf. [19]). Analogously, relationship types of higher order may be defined hierarchically. Suppose now, we are given entity and/or relationship types of order less then $i > 0$. A set $\{\underline{u}_1, \ldots, \underline{u}_k\}$ of them forms a *relationship type* $\underline{r}$ of *order* i. As above, we define relationships of type $\underline{r}$ as tuples from the cartesian product $\underline{u}_1^t \times \ldots \times \underline{u}_k^t$ for a moment t.

In a relationship type $\underline{r} = \{\underline{u}_1, \ldots, \underline{u}_k\}$ each of the entity or relationship types $\underline{u}_1, \ldots, \underline{u}_k$ is said to be a *component type* of $\underline{r}$. Additionally, each of the pairs $(\underline{r}, \underline{u}_j)$ is called a *link*.

Let $\underline{S} = \{\underline{r}_1, \ldots, \underline{r}_n\}$ be a set of entity and relationship types where with each relationship type $\underline{r}$, all its component types belong to $\underline{S}$, too. Then $\underline{S}$ is called a *database scheme*. Replacing each type $\underline{r}$ in $\underline{S}$ by its population $\underline{r}^t$ at moment t, we obtain a database or instance $\underline{S}^t$ of $\underline{S}$.

A database $\underline{S}^t$ is finite if each of the populations $\underline{r}^t$ is finite. In this paper, we are only interested in finite databases. Furthermore, a database $\underline{S}^t$ is *fully-populated* iff none of the populations $\underline{r}^t$ is empty.

From the graph-theoretical point of view, a database scheme $\underline{S}$ can be considered as a finite digraph $\mathfrak{ERD} = (\underline{S}, \mathcal{L})$ with vertex set $\underline{S}$ and a collection $\mathcal{L}$ of arcs. In $\mathfrak{ERD}$ there shall be an arc from $\underline{r}$ to $\underline{u}$ whenever $\underline{u}$ is a component type of the relationship type $\underline{r}$. Hence, the arcs in $\mathfrak{ERD}$ are just the links in the database scheme. The digraph $\mathfrak{ERD}$ is known as the *entity-relationship diagram* of $\underline{S}$. Usually, the entity types are represented graphically by rectangles, the relationship types by diamonds. Using the diagram technique, even complex schemes can be understood and handled.

3 Integrity constraints

In this section, we give formal definitions of the constraints to be studied in the sequel. Assume, we are given a database schema $\underline{S}$ together with an instance $\underline{S}^t$. Let $\underline{r}$ be a relationship type in $\underline{S}$ with population $\underline{r}^t$. If $X \subseteq \underline{r}$ and if $r \in \underline{r}^t$, then we denote the restriction of r to X by $r[X]$.

A *min-cardinality constraint* is a statement $cmin(\underline{r}, \underline{u}) = a$ where $\underline{u}$ is a component type of $\underline{r}$. The population $\underline{r}^t$ satisfies this constraint if every instance u of $\underline{u}$ occurs at least a times in relationships $r \in \underline{r}^t$.

Analogously, a *max-cardinality constraint* is a statement $cmax(\underline{r}, \underline{u}) = b$ where $\underline{u}$ is a component type of $\underline{u}$. The constraint holds in $\underline{r}^t$ iff every instance u of type $\underline{u}$ appears at most b times in relationships of type $\underline{r}$.

Cardinality constraints are usually reflected graphically in the entity-relationship diagram as $(cmin, cmax)$-labels associated to the corresponding links. If no min- or max-cardinality constraint is given for a link $(\underline{r}, \underline{u})$, we may assume $cmin(\underline{r}, \underline{u}) = 0$ and $cmax(\underline{r}, \underline{u}) = \infty$, respectively. It is easy to see that this does not represent a real constraint, but is more or less a technical agreement.

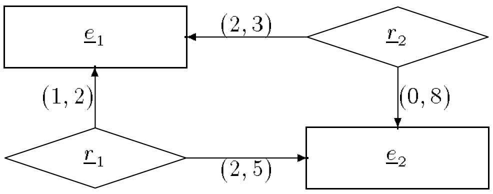

Fig. 1. An entity-relationship diagram with labels for the cardinality constraints.

A subset $X \subseteq \underline{r}$ is a *key* for $\underline{r}$ iff the restrictions $r[X]$, $r \in \underline{r}^t$, are pairwise distinct. If X is a key for $\underline{r}$, then the population $\underline{r}^t$ satisfies the *key dependency* $\underline{r} : X \to \underline{r}$.

Finally, a *functional dependency* is a statement $\underline{r} : X \to Y$ where both, X and Y are subsets of $\underline{r}$. The population $\underline{r}^t$ satisfies the functional dependency $\underline{r} : X \to Y$ iff we have $r_1[Y] = r_2[Y]$ whenever $r_1[X] = r_2[X]$ holds for any two relationships r_1 and r_2 in $\underline{r}^t$. A functional dependency is said to be *unary* if X contains a single component type from $\underline{r}$, and *non-unary* otherwise.

Example. Below we give a small example to provide some motivation for the issues we are going to tackle. Consider a part of a university database schema involving the entity types Professor, Assistant, Course, Building, Lecture Hall and Date. On this set we define a relationship type Lecture reflecting assignments of professors and their assistants to courses in combination with the dates and locations they take place at. Figure 2 shows the corresponding diagram.

In this schema, constraints have to be defined to meet the requirements of the university schedule. Let us consider some examples. The cardinality constraints $cmin(\text{Lecture}, \text{Professor}) = 3$ and $cmax(\text{Lecture}, \text{Professor}) = 6$ state that every professor will give 3 to 6 lectures per term. If every professor has only one assistant per course, then the non-unary functional dependency Lecture:{Professor, Course} $\rightarrow$ {Assistant} holds. Since every lecture hall is situated in only one building, we also have the unary functional dependency Lecture:{Lecture Hall} $\rightarrow$ {Building}. Moreover, the key dependency Lecture:{Course, Date} $\rightarrow$Lecture tells us there is only one lecture for every course per day. Other constraints may be defined due to the schedule of the university or deduced from the ones explicitly stated. The questions we deal with are whether such a set of constraints is conflict-free, and which additional constraints may be deduced without being explicitly stated.

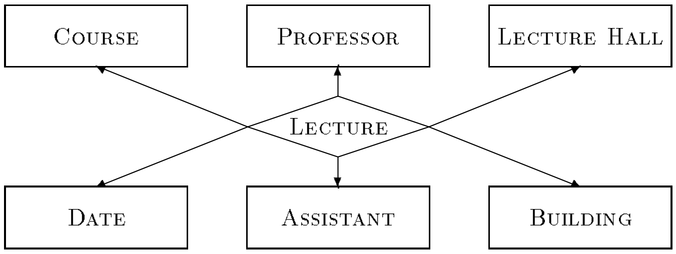

Fig. 2. Entity-relationship diagram for our example.

Let $\mathcal{C}, \mathcal{K}$ and $\mathcal{F}$ denote the classes of all possible cardinality constraints, key dependencies or functional dependencies, respectively. In the sequel, if we refer to a constraint, we always mean one from these classes.

A database instance $\underline{S}^t$ satisfies a given constraint σ iff σ holds for the corresponding population $\underline{r}^t$ in $\underline{S}^t$. Moreover, $\underline{r}^t$ and $\underline{S}^t$ satisfy a set Σ of constraints iff $\underline{r}^t$ and $\underline{S}^t$, respectively, satisfy each of the constraints in Σ.

Let Σ be a constraint set defined on the database schema $\underline{S}$. Every database instance $\underline{S}^t$ satisfying all the constraints in Σ is said to be *legal* for Σ. Consequently, Σ is said to be *satisfiable* if it admits at least one legal database instance. Again it should be noted that we only pay attention to finite databases.

As pointed out in [13], it is usually not enough to consider satisfiability. Obviously, the empty database is legal for every set of cardinality constraints. However it may happen that there is a type in $\underline{S}$ without instances in any legal database. Examples are given in [13]. For practical reasons, we ask for fully-populated legal databases. A constraint set Σ admitting such a database is said to be *consistent*. A fully-populated legal database for Σ will also be called a *sample database*. Sample databases, especially the later described Armstrong databases, can be used for database mining and design-by-example.

The *consistency problem* for a constraint class $\mathcal{Z}$ is to decide whether a given set $\Sigma \subseteq \mathcal{Z}$ is consistent or not. For cardinality constraints, the consistency problem has been considered by several authors, e.g. [10, 13, 20]. It is well-known, that there exist inconsistent sets of cardinality constraints. On the other hand, sets of key or functional dependencies are always consistent. However, the situation may change dramatically if they come together with cardinality constraints.

Example. Consider the schema given in Figure 1. The specified set of cardinality constraints is consistent. Now add a key dependency $\underline{r}_2 : \{\underline{e}_2\} \to \underline{r}_2$. It is easy to check, that the resulting constraint set happens to be inconsistent, and the empty database is the only legal one. A further example for this observation is given in Section 7.

If Σ is a constraint set and σ a single constraint, then Σ implies σ (denoted by $\Sigma \models \sigma$) if every legal database for Σ satisfies σ, too. Conversely, $\Sigma \not\models \sigma$ holds, if there is a legal database (a *certificate*) for Σ violating σ.

The *implication problem* for a constraint class $\mathcal{Z}$ is to decide whether a given set $\Sigma \subseteq \mathcal{Z}$ implies $\sigma \in \mathcal{Z}$ or not. A constraint set Σ is semantically *closed* in $\mathcal{Z}$ if $\sigma \in \Sigma$ whenever $\Sigma \models \sigma$ holds. We denote the closure of a constraint set Σ by $C_{\mathcal{Z}}(\Sigma)$. The determination of the closed sets in a constraint class is of special interest. Clearly, $\Sigma \models \sigma$ holds iff σ is in the closure of Σ. Hence the detection of the closed sets in a constraint class $\mathcal{Z}$ completely solves the implication problem for $\mathcal{Z}$. Moreover, it enables us to decide whether constraint sets are equivalent. Database designers usually look for equivalent constraint sets which are better for control and maintenance in real life databases.

It is often argued, that the consistency problem and the implication problem are two sides of the same coin (e.g. [21]). Checking implication can be done by adding the negation of σ to Σ and testing the resulting set. Conversely, checking inconsistency can be done by testing implication of an unsatisfiable constraint. However, this is only half of the truth. In general, the negation of a constraint from a class $\mathcal{Z}$ is not in $\mathcal{Z}$ again. For example, the negation of a cardinality constraint is an existence constraint, the negation of a functional dependency is an afunctional constraint.

Usually, it is easier to solve the consistency problem than the implication problem, at least for real life constraints. It the remaining sections we shall always start tackling consistency before discussing implication.

Finally, we shall introduce the notions of Armstrong populations and Armstrong databases. Informally, Armstrong databases are sample databases capturing all implications of a constraint set Σ in $\mathcal{Z}$. Recall, that if $\Sigma \not\models \sigma$ there is a legal database for Σ violating σ. We call it a *certificate* for σ. However, there is no a priori guarantee that there exists a single database serving as a certificate for *all* constraints $\sigma \in \mathcal{Z}$ not implied by Σ.

An *Armstrong population* for Σ (with respect to a constraint class $\mathcal{Z}$) is a population $\underline{r}^t$ of $\underline{r}$ satisfying all constraints from Σ, but violating every constraint $\sigma \in \mathcal{Z}$ defined on $\underline{r}$ with $\Sigma \not\models \sigma$. Analogously, an *Armstrong database* for Σ (with respect to $\mathcal{Z}$) is a legal database $\underline{S}^t$ such that $\sigma \in \mathcal{Z}$ holds iff $\Sigma \models \sigma$. Obviously, in an Armstrong database all populations are Armstrong populations.

In order to test whether a constraint follows from an initial set Σ we may test whether it holds in the Armstrong database and decide accordingly. Thus, the implication problem reduces to verification in a single example. The concept of Armstrong databases is closely connected to database mining, i.e. the use of given databases for the extraction of constraints. This is an important tool for design-by-example as suggested by Mannila and Räihä [16].

In Section 6, we shall consider Armstrong databases for sets of cardinality constraints, key and functional dependencies. We shall use them in particular, to prove certain constraint sets to be semantically closed.

4 Reasoning about cardinality constraints

In this section, we shall assemble some terminology and known results concerning cardinality constraints which will be required in the sequel. Throughout, let $\underline{S}$ be a database schema and Σ_C a set of cardinality constraints defined on $\underline{S}$. We start with the following trivial observation.

Lemma 1 *For every link* $(\underline{r}, \underline{u})$ *in* $\underline{S}$ *we have*

$$cmin(\underline{r}, \underline{u}) = 0 \text{ and } cmax(\underline{r}, \underline{u}) = \infty, \tag{1}$$

$$cmin(\underline{r}, \underline{u}) = a \text{ implies } cmin(\underline{r}, \underline{u}) = a' \text{ for all } a' \leq a, \tag{2}$$

$$cmax(\underline{r}, \underline{u}) = b \text{ implies } cmax(\underline{r}, \underline{u}) = b' \text{ for all } b' \geq b. \tag{3}$$

The constraints given by (1) are said to be trivial. Put

$$\alpha(\underline{r}, \underline{u}) = \max\{a : (cmin(\underline{r}, \underline{u}) = a) \in \Sigma\}$$
$$\beta(\underline{r}, \underline{u}) = \min\{b : (cmax(\underline{r}, \underline{u}) = b) \in \Sigma\}.$$

Hence, the sharpest constraints explicitly stated in Σ_C for a link $(\underline{r}, \underline{u})$ are given by $\alpha(\underline{r}, \underline{u})$ and $\beta(\underline{r}, \underline{u})$. The following claim gives a first result concerning the consistency of Σ_C.

Lemma 2 Σ_C *is consistent iff there is a function* $g : \underline{S} \to \mathbb{N} = \{1, 2, \dots\}$ *such that*

$$\alpha(\underline{r}, \underline{u}) \leq \frac{g(\underline{r})}{g(\underline{u})} \leq \beta(\underline{r}, \underline{u}) \tag{4}$$

holds for every link $(\underline{r}, \underline{u})$.

Given such a function g, it is possible to construct a sample database $\underline{S}^t$ for Σ_C where the population $\underline{r}^t$ has size $g(\underline{r})$ for every type $\underline{r}$ in $\underline{S}$. The main problem is to find a suitable function g. In [10], we used shortest-path methods in certain digraphs for this purpose.

Let $\mathfrak{D} = (\underline{S}, \mathcal{L} \cup \mathcal{L}^{-1})$ be the symmetric digraph which we obtain from the entity-relationship diagram $\mathfrak{ERD} = (\underline{S}, \mathcal{L})$ by adding to each link $L = (\underline{r}, \underline{u})$ the reverse arc $L^{-1} = (\underline{u}, \underline{r})$. On the arcs of $\mathfrak{D}$ we define a weight function $w : \mathcal{L} \cup \mathcal{L}^{-1} \to \mathbb{Q} \cup \{\infty\}$ by

$$w(L) = \begin{cases} \infty & \text{if } \alpha(\underline{r}, \underline{u}) = 0, \\ 1/\alpha(\underline{r}, \underline{u}) & \text{otherwise,} \end{cases} \quad \text{and} \quad w(L^{-1}) = \beta(\underline{r}, \underline{u}), \tag{5}$$

where L is the original link $(\underline{r}, \underline{u}) \in \mathcal{L}$ and L^{-1} is its reverse.

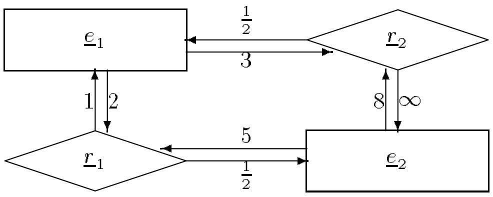

Fig. 3. The digraph $\mathfrak{D}$ corresponding to the diagram in Figure 1.

This weight function can be extended to subdigraphs $\mathfrak{D}'$ of $\mathfrak{D}$ by $w(\mathfrak{D}') = \prod_{L \in \mathcal{A}'} w(L)$, where $\mathcal{A}' \subseteq L \cup L^{-1}$ is the collection of arcs in $\mathfrak{D}'$.

Special interest is devoted to directed cycles in $\mathfrak{D}$. A directed cycle $\mathfrak{Z}$ is said to be *critical* if its weight $w(\mathfrak{Z})$ is less than 1, and *subcritical* if its weight equals 1.

The following result was proved by Lenzerini and Nobili [13] and Thalheim [19]. In [10] a polynomial-time algorithm is proposed to construct sample databases or to detect critical cycles using shortest-path methods (namely a variation of the well-known Bellman-Ford algorithm). Hence, the question whether a set Σ_C of cardinality constraints is consistent or not can be decided in polynomial time.

Lemma 3 *Σ_C is consistent iff the digraph $\mathfrak{D}$ has no critical cycles.*

The implication problem for cardinality constraints was solved in [11]. The main observation is the following.

Lemma 4 *Let Σ_C be consistent, and $(\underline{r}, \underline{u})$ be a link in $\underline{S}$. Then Σ_C implies $cmax(\underline{r}, \underline{u}) = \alpha(\underline{r}, \underline{u})$ iff the link $(\underline{r}, \underline{u})$ lies on a subcritical cycle in $\mathfrak{D}$, and $cmin(\underline{r}, \underline{u}) = \beta(\underline{r}, \underline{u})$ iff the reverse arc $(\underline{u}, \underline{r})$ lies on a subcritical cycle in $\mathfrak{D}$.*

In [11], we called Σ_C *strongly consistent* iff it is consistent and $\alpha(\underline{r}, \underline{u}) = \beta(\underline{r}, \underline{u})$ holds whenever link $(\underline{r}, \underline{u})$ or its reverse arc $(\underline{u}, \underline{r})$ lie on a subcritical cycle. Moreover, we proved the following characterization of closed sets of cardinality constraints.

Lemma 5 *Let Σ_C be consistent. Then Σ_C is closed in $\mathcal{C}$ iff it is strongly consistent and contains all constraints obtained by Lemma 1.*

This solves the implication problem for consistent sets in $\mathcal{C}$. Of course, one can also consider inconsistent sets. However, this seems to be of academic use only, and will be omitted here due to space limitations. In [11], we also provided polynomial-time algorithms to detect arcs on subcritical cycles, such that the implication problem for $\mathcal{C}$ can be solved efficiently.

In order to prove a set Σ_C in $\mathcal{C}$ to be closed, we constructed Armstrong databases.

Lemma 6 *A consistent set Σ_C admits an Armstrong database (with respect to $\mathcal{C}$) iff it implies a non-trivial max-cardinality constraint for every link in $\underline{S}$.*

Assume, there is a link $(\underline{r}, \underline{u})$ such that $cmax(\underline{r}, \underline{u}) = \infty$ is the only max-cardinality constraint for this link obtained from Σ_C. Then, for every integer b, there must be an instance u of type $\underline{u}$ participating in more than b relationships in an Armstrong population of $\underline{r}$. Hence, the resulting database would be infinite. Since we restricted ourselves to finite databases, there will not be an Armstrong database for this case.

However, if we fix b, then there exists a sample database with an instance u involved in more than b relationships of type $\underline{r}$ for every link $(\underline{r}, \underline{u})$ without a nontrivial max-cardinality constraint. Taking such a database for every b, we obtain a sequence of sample databases which we shall call an *Armstrong sequence*. Due to [11], every consistent set Σ_C admits at least an Armstrong sequence of sample databases (with respect to $\mathcal{C}$).

5 The consistency problem

Sets of key and functional dependencies are always consistent. But as seen in the example of Figure 1, this is not longer true if we are given cardinality constraints, too. The reason for this is the following simple observation (see also [20]).

Lemma 7 *The key dependency $\underline{r} : \{\underline{u}\} \to \underline{r}$ implies $cmax(\underline{r}, \underline{u}) = 1$, and vice versa.*

Throughout, let $\underline{S}$ be a database schema, and Σ_C, Σ_K and Σ_F denote sets of cardinality constraints, key dependencies and functional dependencies, respectively, defined on $\underline{S}$.

It is noteworthy, that Lemma 7 contains the only nontrivial interaction between cardinality constraints and key dependencies. Let $\Sigma_K^C = \{cmax(\underline{r}, \underline{u}) = 1 : (\underline{r} : \{\underline{u}\} \to \underline{r}) \in \Sigma_K\}$ contain all cardinality constraints obtained from key dependencies in Σ_K via Lemma 7.

Theorem 8 *Let $\Sigma = \Sigma_C \cup \Sigma_K$. Then Σ is consistent iff $\Sigma_C \cup \Sigma_K^C$ is consistent.*

The proof of this claim uses structures from combinatorial design theory, namely *resolvable transversal designs*. For details on transversal designs, we refer to [12]. The proof of the theorem is given in Section 9.

Example. Theorem 8 shows why the cardinality constraints given in Figure 1 together with the key dependency $\underline{r}_2 : \{\underline{e}_2\} \to \underline{r}_2$ are not longer consistent. Here Σ_K^C contains the additional max-cardinality constraint $cmax(\underline{r}_2, \underline{e}_2) = 1$. This leads to a modification of the weight function w defined by (5) as shown in Figure 4. It is easy to check that the cycle $(\underline{e}_2, \underline{r}_2, \underline{e}_1, \underline{r}_1)$ has weight $1/2$, i.e. is critical.

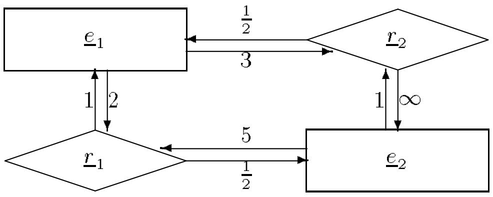

Fig. 4. The digraph representing an additional key dependency.

The consistency of $\Sigma_C \cup \Sigma_K^C$ may be checked in polynomial time as proposed in Section 4. In particular, a consistent set of cardinality constraints together with a set of non-unary key dependencies will always be consistent. The same observation holds for functional dependencies. We will state this in a slightly stronger version as follows.

Theorem 9 *Let $\Sigma = \Sigma_C \cup \Sigma_K \cup \Sigma_F$, and assume that all functional dependencies in Σ_F are trivial, non-unary or implied by key dependencies in Σ_K. Then Σ is consistent iff $\Sigma_C \cup \Sigma_K^C$ is consistent.*

Unfortunately, this claim does not hold for arbitrary sets of functional dependencies as shown in Section 7.

6 The implication problem

We start with some known results on the implication problem for key and functional dependencies, and then use those to obtain generalizations in the presence of cardinality constraints. Delobel and Casey [6] gave a set of inference rules for functional dependencies, which Armstrong [1] proved to be complete. We record these rules for future reference.

Lemma 10 *Let $\underline{r}$ be a relationship type, and X, Y, Z be subsets of its component types. Then*

$$\underline{r} : X \to Y \text{ holds whenever } Y \subseteq X, \tag{6}$$

$$\underline{r} : X \to Y \text{ and } \underline{r} : Y \to Z \text{ imply } \underline{r} : X \to Z, \tag{7}$$

$$\underline{r} : X \to Y \text{ implies } \underline{r} : X \cup Z \to Y \cup Z. \tag{8}$$

For key dependencies, we may conclude the following rules.

Lemma 11 *Let $\underline{r}$ be a relationship type, and X, Y be subsets of its component types. Then we have $\underline{r} : \underline{r} \to \underline{r}$ and*

$$\underline{r} : X \to \underline{r} \textit{ implies } \underline{r} : Y \to \underline{r} \textit{ whenever } X \subseteq Y. \tag{9}$$

According to Armstrong [1], the rules given in Lemma 10 and 11 are correct and complete. Hence, Σ_F (Σ_K) is closed in $\mathcal{F}$ ($\mathcal{K}$) iff it contains all constraints obtained by applying Lemma 10 (11, respectively).

Constraints obtained by (6) are said to be trivial. The Armstrong rules gave rise to a considerable number of papers concerning applications, consequences and alternate rules. For a discussion, see [18]. In [1, 2], one finds methods for constructing Armstrong databases for functional dependencies. Attention to combinatorial properties of functional dependencies has been drawn e.g. in [7, 8].

As pointed out by Demetrovics and Thi [8], functional dependencies can be represented by closure operations. For a relationship type $\underline{r}$ and any subset $X \subseteq \underline{r}$, we put

$$cl(X) = \{\underline{u} \in \underline{r} : \Sigma_F \models (X \to \{\underline{u}\})\}.$$

This induces a closure operation on the subsets of $\underline{r}$. We call X *functionally closed* (with respect to Σ_F) if $cl(X) = X$ holds. Conversely, for every closure operation cl on $\underline{r}$, the set

$$\Sigma_F = \{(\underline{r} : X \to Y) : Y \subseteq cl(X)\}$$

is semantically closed in the class $\mathcal{F}$ of functional dependencies.

To continue with, we shall introduce some graph-theoretical notions which turn out to be useful in the future. Assume, we are given a database instance $\underline{S}^t$ of $\underline{S}$. For every link $(\underline{r}, \underline{u})$ we consider a graph $\mathfrak{G}^t(\underline{r}, \underline{u})$ whose vertices are the relationships in $\underline{r}^t$. Two distinct relationships r_1 and r_2 are connected by an edge in $\mathfrak{G}^t(\underline{r}, \underline{u})$ iff $r_1[\underline{u}] = r_2[\underline{u}]$ holds.

Obviously, $\mathfrak{G}^t(\underline{r}, \underline{u})$ is a collection of vertex-disjoint, complete subgraphs (each of them corresponding to exactly one instance u of type $\underline{u}$). We call $\mathfrak{G}^t(\underline{r}, \underline{u})$ the *representation graph* of the link $(\underline{r}, \underline{u})$ (with respect to the given database instance $\underline{S}^t$). Moreover, let $\mathfrak{G}^t(\underline{r}, X)$ denote the intersection of the representation graphs $\mathfrak{G}^t(\underline{r}, \underline{u})$, $\underline{u} \in X$, for any subset X of $\underline{r}$.

Representation graphs give us useful information on the constraints hold in $\underline{S}^t$.

Lemma 12 *A database instance $\underline{S}^t$ satisfies*

$$\begin{aligned}
cmin(\underline{r}, \underline{u}) = a &\textit{ iff all components of } \mathfrak{G}^t(\underline{r}, \underline{u}) \textit{ have at least } a \textit{ vertices,}\\
cmax(\underline{r}, \underline{u}) = b &\textit{ iff all components of } \mathfrak{G}^t(\underline{r}, \underline{u}) \textit{ have at most } b \textit{ vertices,}\\
\underline{r} : X \to \underline{r} &\textit{ iff } \mathfrak{G}^t(\underline{r}, X) \textit{ is edgeless,}\\
\underline{r} : X \to Y &\textit{ iff } \mathfrak{G}^t(\underline{r}, X) \textit{ is a subgraph of } \mathfrak{G}^t(\underline{r}, Y).
\end{aligned}$$

Here *components* mean maximal connected subgraphs, which should not be confused with the component types of relationship types. With the help of representation graphs, we obtain

Theorem 13 *Let Σ_C be consistent. $\Sigma_C \cup \Sigma_K$ is closed in $\mathcal{C} \cup \mathcal{K}$ iff Σ_C is closed in $\mathcal{C}$, Σ_K is closed in $\mathcal{K}$ and we have*

$$\{(\underline{r}, \underline{u}) : (cmax(\underline{r}, \underline{u}) = 1) \in \Sigma_C\} = \{(\underline{r}, \underline{u}) : (\underline{r} : \underline{u} \to \underline{r}) \in \Sigma_K\}. \qquad (10)$$

Condition (10) ensures that all implications obtained by Lemma 7 are considered. A key dependency $\underline{r} : X \to \underline{r}$ forces X to be a key for $\underline{r}$. Conversely, X is a *non-key*, if it is not a key. A non-key X is said to be an *antikey*, if every proper superset of X is a key.

To prove Theorem 13 we have, among others, to construct representation graphs $\mathfrak{G}^t(\underline{r}, \underline{u})$, such that the intersection graph $\mathfrak{G}^t(\underline{r}, X)$ contains at least one edge for every non-key $X \subseteq \underline{r}$. This proves $\Sigma \not\models (\underline{r} : X \to \underline{r})$. Actually, it is sufficient to consider antikeys, as every non-key X admits an antikey Y such that $X \subseteq Y$, i.e. $\mathfrak{G}^t(\underline{r}, Y)$ is a subgraph of $\mathfrak{G}^t(\underline{r}, X)$.

As a result, we obtain Armstrong populations for Σ with respect to $\mathcal{K}$. Mixing them with an appropriate Armstrong database (Armstrong sequence, respectively) for Σ with respect to $\mathcal{C}$, we obtain the same with respect to $\mathcal{C} \cup \mathcal{K}$. The main tool of our proof is the following theorem due to Hajnal and Szemeredi [9].

Theorem 14 *Let $\mathfrak{H}$ be a graph with $m = ts$ vertices and maximum valency less than t, then there exists a vertex decomposition of $\mathfrak{H}$ into t sets of s pairwise non-adjacent vertices.*

Details on our proof will be given in Section 9. Theorem 13 gives us an efficient way to decide implication in $\mathcal{C} \cup \mathcal{K}$.

Algorithm 15 *Given Σ_C and Σ_K, proceed as follows:*

1. *Determine Σ_K^C by applying Lemma 7 to Σ_K.*
2. *Determine the closure of $\Sigma_C \cup \Sigma_K^C$ in $\mathcal{C}$ as proposed in Section 4.*
3. *Determine Σ_C^K by applying Lemma 7 to the closure $C_{\mathcal{C}}(\Sigma_C \cup \Sigma_K^C)$.*
4. *Determine the closure of $\Sigma_K \cup \Sigma_C^K$ in $\mathcal{K}$ as proposed by Lemma 11.*
5. *$C_{\mathcal{C}}(\Sigma_C \cup \Sigma_K^C) \cup C_{\mathcal{K}}(\Sigma_K \cup \Sigma_C^K)$ gives the closure of $\Sigma_C \cup \Sigma_K$ in $\mathcal{C} \cup \mathcal{K}$.*

Using similar methods, we are able to obtain results on cardinality constraints and functional dependencies.

Theorem 16 *Let Σ_C be consistent and Σ_F contain non-unary functional dependencies only. $\Sigma_C \cup \Sigma_F$ is closed in $\mathcal{C} \cup \mathcal{F}$ iff Σ_C is closed in $\mathcal{C}$ and Σ_F is closed in $\mathcal{F}$.*

This time we have to construct representation graphs $\mathfrak{G}^t(\underline{r}, \underline{u})$ such that for every functionally closed set $X \subseteq \underline{r}$, there is a pair of relationships $r_{1,X}$ and $r_{2,X}$ of type $\underline{r}$ connected by an edge in $\mathfrak{G}^t(\underline{r}, X)$, but not in any of the representation graphs $\mathfrak{G}^t(\underline{r}, \underline{u})$, $\underline{u} \notin X$.

Finally, as a corollary, we obtain the next result.

Corollary 17 *Let Σ_C be consistent and Σ_F contain non-unary functional dependencies only. $\Sigma_C \cup \Sigma_K \cup \Sigma_F$ is closed in $\mathcal{C} \cup \mathcal{K} \cup \mathcal{F}$ iff $\Sigma_C \cup \Sigma_K$ is closed in $\mathcal{C} \cup \mathcal{K}$ and $\Sigma_K \cup \Sigma_F$ is closed in $\mathcal{K} \cup \mathcal{F}$.*

7 Unary functional dependencies

Until now, we usually excluded unary functional dependencies. In this section we shall give a brief impression why these dependencies play a special role in the presence of cardinality constraints. However, it is noteworthy that a set Σ_C of cardinality constraints will never imply a functional dependency different from those obtained by Lemma 7.

Lemma 18 *Let $\underline{r}$ be a relationship type, and $\underline{u}, \underline{v}$ be component types of $\underline{r}$. Σ_C implies a functional dependency $\underline{r} : \{\underline{u}\} \to \{\underline{v}\}$ iff it implies $cmax(\underline{r}, \underline{u}) = 1$.*

Conversely, if we are given cardinality constraints *and* functional dependencies, then one may conclude further implications.

Example. Consider the schemata in Figure 5. The specified sets of cardinality constraints are both consistent. Assume, we are given the functional dependency $\underline{r} : \{\underline{u}\} \to \{\underline{v}\}$, too. In every legal database instance, the representation graph $\mathfrak{G}^t(\underline{r}, \underline{u})$ has to be a subgraph of $\mathfrak{G}^t(\underline{r}, \underline{u})$.

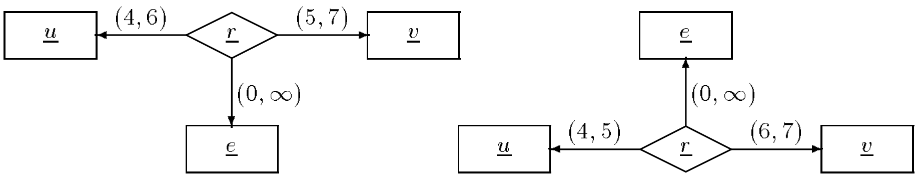

Fig. 5. Examples for interactions with a functional dependency $\underline{r} : \{\underline{u}\} \to \{\underline{v}\}$.

For the first schema, the components of $\mathfrak{G}^t(\underline{r}, \underline{u})$ are complete graphs on 4 to 6 vertices, while the components of $\mathfrak{G}^t(\underline{r}, \underline{v})$ are complete graphs on 5 to 7 vertices. The only possible decomposition of components of $\mathfrak{G}^t(\underline{r}, \underline{v})$ into those of $\mathfrak{G}^t(\underline{r}, \underline{u})$ is the trivial one. Hence, both representation graphs are isomorphic. This gives us the new constraints $\underline{r} : \{\underline{v}\} \to \{\underline{u}\}$ as well as $cmin(\underline{r}, \underline{u}) = 5$ and $cmax(\underline{r}, \underline{v}) = 6$.

For the second schema in Figure 5, the situation is even worse. A similar argumentation shows the specified set of constraints to be inconsistent.

The following result summarizes the observations indicated by the examples. A rigorous exploitation of the argumentation gives further results, which we skip out here due to space limitations.

Lemma 19 *Let $\underline{r}$ be a relationship type, and $\underline{u}, \underline{v}$ be component types of $\underline{r}$. Then*

$$
\begin{aligned}
cmax(\underline{r}, \underline{v}) = b \text{ and } \underline{r} : \{\underline{u}\} \to \{\underline{v}\} &\text{ imply } cmax(\underline{r}, \underline{u}) = b,\\
cmin(\underline{r}, \underline{u}) = a \text{ and } \underline{r} : \{\underline{u}\} \to \{\underline{v}\} &\text{ imply } cmin(\underline{r}, \underline{v}) = a,\\
cmax(\underline{r}, \underline{v}) = b, cmin(\underline{r}, \underline{u}) = a \text{ and } \underline{r} : \{\underline{u}\} \to \{\underline{v}\} &\text{ imply } \underline{r} : \{\underline{v}\} \to \{\underline{u}\}\\
&\text{ whenever } b < 2a.
\end{aligned}
$$

An interesting idea to handle unary functional dependencies was given by Biskup et al. [3]. They use these constraints to decompose relationship types by pivoting. In our example from Section 3, the unary functional dependency Lecture:{Lecture Hall} → {Building} may be used for this approach. As a result we would obtain the diagram in Figure 6. Here this functional dependency is naturally enforced by the structure of the schema.

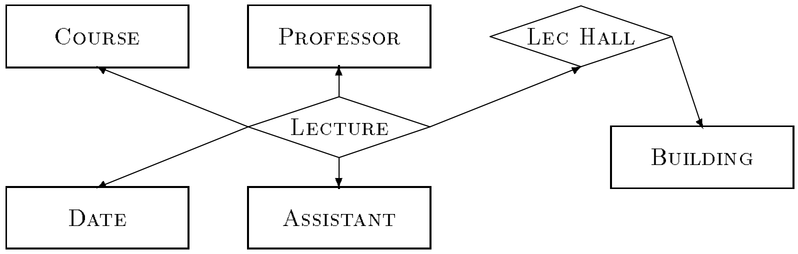

Fig. 6. Entity-relationship diagram after decomposition by pivoting.

Decomposition by pivoting is closely related to decomposition into BCNF. A relationship type $\underline{r}$ is in *Boyce-Codd Normal Form* (BCNF) with respect to Σ_F if every functional dependency $\underline{r} : X \to Y$ in Σ_F is trivial (i.e. $Y \subseteq X$), or X is a key for $\underline{r}$. Applying Theorem 9 and Corollary 17, we may conclude as follows.

Theorem 20 *Assume every relationship type $\underline{r}$ is in BCNF. Then $\Sigma = \Sigma_C \cup \Sigma_K \cup \Sigma_F$ is consistent iff $\Sigma_C \cup \Sigma_K^C$ is consistent. In addition, Σ is closed in $\mathcal{C} \cup \mathcal{K} \cup \mathcal{F}$ iff $\Sigma_C \cup \Sigma_K$ is closed in $\mathcal{C} \cup \mathcal{K}$ and $\Sigma_K \cup \Sigma_F$ is closed in $\mathcal{K} \cup \mathcal{F}$.*

Recall, that closures in $\mathcal{C} \cup \mathcal{K}$ and in $\mathcal{K} \cup \mathcal{F}$ can be determined applying Algorithm 15 and the Armstrong rules, respectively. This gives us an efficient procedure to check implication under the assumption of BCNF.

8 Conclusions

In this paper, we provided an efficient method for reasoning about a constraint set comprising cardinality constraints as well as key and functional dependencies. Reasoning means checking consistency and logical implication. Today, cardinality constraints are embedded in most CASE tools, which are usually based on the entity-relationship model. Although these tools include basic reasoning methods, they do not offer intelligent consistency checking routines for constraint set containing not only functional dependencies, but also cardinality constraints. Applying the ideas presented in this paper, it would be possible to derive interesting properties of schemata, or to detect conflicts among the constraints under discussion.

There are at least two problems which should be investigated in future. In the last section, we gave examples for interactions between cardinality constraints and unary functional dependencies. It would be nice to have a characterization of semantically closed sets in the presence of unary dependencies without additional assumptions such as BCNF.

The second problem concerns Armstrong databases. The Armstrong databases constructed to prove Theorem 13 are extremely large. In order to use Armstrong databases for database mining, they have to be of reasonable size. Thus we are looking for small databases as done in [7] for functional dependencies only. For cardinality constraints, a similar problem is considered in [11]. It would be interesting to decide, whether there exists a sample database where the size of every population is bounded by a given integer. Unfortunately, it is still unknown whether this problem is NP-complete or not. A solution of this question would give us new information on the size of Armstrong databases.

9 Proofs of the main results

In this section, we shall give proofs for the Theorems 8 and 13.

To begin with, we introduce some notation. A *transversal design* $TD(k,q)$ is a set of q^2 tuples from $\{1,\dots,k\}^q$ such that any two tuples agree in at most one entry. A subset of q pairwise disjoint tuples is said to be a *resolution class*. The transversal design is *resolvable* if it is decomposable into q resolution classes. Resolvable transversal designs are known to exist for fixed k and sufficiently large q. In particular, they exist for every prime power $q \geq k$.

Proof (of Theorem 8). The necessity is trivial, so we continue with the sufficiency. Since $\Sigma_C \cup \Sigma_K^C$ is consistent, there exists a sample database $\underline{S}^t$. Denote the number of instances of every type $\underline{r}$ by $g(\underline{r})$. We choose a prime power q larger than each of the integers $g(\underline{r})$, and larger than n, the number of types in $\underline{S}$. From $\underline{S}^t$ we shall construct a new database $\underline{S}^{t*}$, where every type $\underline{r}$ has $qg(\underline{r})$ instances.

Let $\underline{e}$ be an entity type. For every old instance e, we introduce q new instances $(e,1),\ldots,(e,q)$. Next, let $\underline{r}$ be a relationship type with k component types. Due to our choice of q, there is a resolvable transversal design $TD(k,q)$. For every old instance r of $\underline{r}$, fix an own resolution class in the transversal design. Let $r=(u_1,\ldots,u_k)$ be an old instance of $\underline{r}$, and $(i_1,\ldots,i_k)$ a tuple in the associated resolution class. Then we introduce a new instance $((u_1,i_1),\ldots,(u_k,i_k))$. Doing the same for every old relationship and every tuple in the corresponding resolution class, we obtain a new population $\underline{r}^{t*}$ for $\underline{r}$ with exactly $qg(\underline{r})$ instances. We apply the same procedure hierarchically to every relationship type in $\underline{S}$.

Obviously, every new instance (u,i) of a component type from $\underline{r}$ participates in the same number of new relationships as the old instance u did in old relationships. Hence, the new database $\underline{S}^{t*}$ is again a sample database for $\Sigma_C \cup \Sigma_K^C$. It is easy to check, that any two new relationships of a certain type $\underline{r}$ share at most one entry. Therefore $\underline{S}^{t*}$ satisfies every key dependency in Σ_K, too. □

The same construction also indicates a proof for Theorem 9.

We now turn to Theorem 13. Its proof requires some preliminary observations. Let Σ_C be a consistent and closed set of cardinality constraints. According to Lemma 5, Σ_C is strongly consistent. In [11], we gave the following characterization, which is similar to Lemma 2.

Lemma 21 *Let Σ_C be consistent. Then Σ_C is strongly consistent iff there exists a function $g:\underline{S}\to\mathbb{N}$ such that*

$$\frac{g(\underline{r})}{g(\underline{u})} = \beta(\underline{r},\underline{u}), \quad \text{if } \alpha(\underline{r},\underline{u}) = \beta(\underline{r},\underline{u}), \tag{11}$$

$$\alpha(\underline{r},\underline{u}) < \frac{g(\underline{r})}{g(\underline{u})} < \beta(\underline{r},\underline{u}), \quad \text{otherwise}, \tag{12}$$

holds for every link $(\underline{r},\underline{u})$.

To prove this, we constructed Armstrong databases (Armstrong sequences, respectively) for Σ_C where the population $\underline{r}^s$ has size $\lambda g(\underline{r})$ for every type $\underline{r}$ in $\underline{S}$. Here λ denotes any sufficiently large integer.

Let Σ_K be a closed set of key dependencies, and assume Condition (10) holds. In the sequel, let $\underline{r}$ be a relationship type in $\underline{S}$.

Lemma 22 *There is an Armstrong population $\underline{r}^t$ of $\underline{r}$ for Σ_K (with respect to $\mathcal{K}$), satisfying all constraints in Σ_C defined on $\underline{r}$.*

Proof. For every component type $\underline{u}$ of $\underline{r}$, put

$$s(\underline{u}) = \begin{cases} \alpha(\underline{r},\underline{u}) & \text{if } \alpha(\underline{r},\underline{u}) \geq 2, \\ 1 & \text{if } \beta(\underline{r},\underline{u}) \leq 1, \\ 2 & \text{otherwise.} \end{cases}$$

Put $m = (\gamma+2)\prod_{\underline{u}\in\underline{r}} s(\underline{u})$, where γ denotes the number of antikeys of $\underline{r}$ (given by Σ_K). Here, m will be the size of the population $\underline{r}^t$, and every component type $\underline{u}$ of $\underline{r}$ will have exactly $m/s(\underline{u})$ instances. Moreover, each instance of type $\underline{u}$ will participate in exactly $s(\underline{u})$ relationships. Since $\alpha(\underline{r},\underline{u}) \leq s(\underline{u}) \leq \beta(\underline{r},\underline{u})$, this ensures that the claimed cardinality constraints in Σ_C hold.

Put $\underline{r}^t = \{r_1,\ldots,r_m\}$. On the vertex set $\underline{r}^t$, we shall construct representation graphs $\mathfrak{G}^t(\underline{r},\underline{u})$. In the beginning, let all vertices in $\underline{r}^t$ be unmarked. For every antikey $X \subseteq \underline{r}$, choose two unmarked vertices $r_{1,X}, r_{2,X}$, and mark them. Next, connect $r_{1,X}$ and $r_{2,X}$ by an edge in every graph $\mathfrak{G}^t(\underline{r},\underline{u})$ with $\underline{u} \in X$.

Fix a component type $\underline{u}$. For every edge in $\mathfrak{G}^t(\underline{r},\underline{u})$, choose $s(\underline{u})-2$ unmarked vertices from $\underline{r}^t$ and mark them. In addition, let the complete graph on these $s(\underline{u})$ vertices be a component of the graph $\mathfrak{G}^t(\underline{r},\underline{u})$. In the same way, we proceed for all component types $\underline{u}$.

Let $\mathfrak{H}$ denote the union of the graphs $\mathfrak{G}^t(\underline{r},\underline{u})$ constructed so far. Again, we fix a component type $\underline{u}$. Note, that at the moment, $\mathfrak{G}^t(\underline{r},\underline{u})$ consists of a certain number of components of size $s(\underline{u})$ together with a set $I(\underline{u})$ of isolated vertices. Consider the subgraph $\mathfrak{H}'$ of $\mathfrak{H}$ induced by $I(\underline{u})$. Here every vertex has valency at most $\sum_{\underline{v}\neq\underline{u}}(s(\underline{v})-1)$. Due to our choice of m, we have $\sum_{\underline{v}\neq\underline{u}}(s(\underline{v})-1) < m/s(\underline{u})$ and thus maximum valency less than $m/s(\underline{u})$. Applying the Theorem of Hajnal and Szemeredi (Theorem 14) to $\mathfrak{H}'$, its vertex set $I(\underline{u})$ may be decomposed into subsets of size $s(\underline{u})$, such that the vertices within a subset are pairwise non-adjacent. We add the complete graph on each of these subsets as a component to $\mathfrak{G}^t(\underline{r},\underline{u})$. The resulting graph $\mathfrak{G}^t(\underline{r},\underline{u})$ will be the representation graph of $\underline{u}$. We apply the same procedure to each of the component types $\underline{u}$ of $\underline{r}$.

It remains to show that the constructed population $\underline{r}^t$ has the desired properties. Let $cmin(\underline{r},\underline{u}) = a$ and $cmax(\underline{r},\underline{u}) = b$ be cardinality constraints in Σ_C. The representation graph $\mathfrak{G}^t(\underline{r},\underline{u})$ consists of a collection of vertex-disjoint complete graphs on $s(\underline{u})$ vertices each (the components of $\mathfrak{G}^t(\underline{r},\underline{u})$). Each component corresponds to an instance of type $\underline{u}$ participating in exactly $s(\underline{u})$ relationships. Since $a \leq \alpha(\underline{r},\underline{u}) \leq s(\underline{u}) \leq \beta(\underline{r},\underline{u}) \leq b$, the claimed constraints hold. This proves $\underline{r}^t$ to satisfy Σ_C.

Let $\underline{r} : Y \to \underline{r}$ be a key dependency in Σ_K. Assume, the intersection graph $\mathfrak{G}^t(\underline{r},Y)$ contains an edge. Due to our construction, there must be an antikey X with $Y \subseteq X$. This forces Y to be a non-key which gives a contradiction. Hence, $\mathfrak{G}^t(\underline{r},Y)$ is edgeless and the key dependency holds. This proves $\underline{r}^t$ to satisfy Σ_K.

Finally, let $\underline{r} : Y \to \underline{r}$ be a key dependency not in Σ_K. Since Y is a non-key, there exists an antikey X with $Y \subseteq X$. The intersection graph $\mathfrak{G}^t(\underline{r},X)$ contains an edge connecting $r_{1,X}$ and $r_{2,X}$. Since $\mathfrak{G}^t(\underline{r},X)$ is a subgraph of $\mathfrak{G}^t(\underline{r},Y)$, this edge is also in the later one. Thus the key dependency $\underline{r} : Y \to \underline{r}$ is violated. Therefore, $\underline{r}^t$ is an Armstrong population for Σ_K with respect to $\mathcal{K}$. $\square$

For further reference, put $g_r(\underline{r}) = m$ and $g_r(\underline{u}) = m/s(\underline{u})$ for every component type of $\underline{r}$. These values denote the number of instances of $\underline{r}$ and $\underline{u}$ in the population just constructed.

Let $g : \underline{S} \to \mathbb{N}$ be a function satisfying (11,12), let q be a sufficiently large prime power, and let λ and μ be sufficiently large integers such that

$$\begin{aligned}\mu >& g_r(\underline{r}) - \alpha(\underline{r},\underline{u})g_r(\underline{u}),\\ \mu >& \beta(\underline{r},\underline{u})g_r(\underline{u}) - g_r(\underline{r}), \text{ if } \beta(\underline{r},\underline{u}) \text{ is finite,}\end{aligned}$$

hold for every link $(\underline{r},\underline{u})$ in the schema $\underline{S}$.

Again, let $\underline{r}$ be a fixed relationship type of $\underline{S}$. It must be emphasized, that the population $\underline{r}^t$ constructed above usually satisfies cardinality constraints not in Σ_C. However, it can be applied to obtain the following stronger result. Suppose that Σ_C contains a non-trivial max-cardinality constraint for every link $(\underline{r},\underline{u})$.

Lemma 23 *There is an Armstrong population $\underline{r}^z$ of $\underline{r}$ for Σ_K (with respect to $\mathcal{K}$), which is also an Armstrong population for Σ_C (with respect to $\mathcal{C}$).*

Proof. To begin with, put $h_r(\underline{r}) = \mu g(\underline{r}) - g_r(\underline{r})$ and $h_r(\underline{u}) = \mu g(\underline{u}) - g_r(\underline{u})$ for every $\underline{u} \in \underline{r}$. Let us check, that h_r satisfies (11,12). If $\alpha(\underline{r},\underline{u}) = \beta(\underline{r},\underline{u})$, we have $g(\underline{r})/g(\underline{u}) = \beta(\underline{r},\underline{u})$ and $g_r(\underline{r})/g_r(\underline{u}) = s(\underline{u}) = \beta(\underline{r},\underline{u})$. This implies (11). If $\alpha(\underline{r},\underline{u}) < \beta(\underline{r},\underline{u})$, we have

$$\begin{aligned} 1 \le & \; g(\underline{r}) - \alpha(\underline{r},\underline{u})g(\underline{u})\\ g_r(\underline{r}) - \alpha(\underline{r},\underline{u})g_r(\underline{r},\underline{u}) < \mu \le & \; \mu(g(\underline{r}) - \alpha(\underline{r},\underline{u})g(\underline{u}))\\ \alpha(\underline{r},\underline{u})\mu g(\underline{u}) - \alpha(\underline{r},\underline{u})g_r(\underline{u}) < & \; \mu g(\underline{r}) - g_r(\underline{r}),\end{aligned}$$

which implies the left side of (12). The right side holds analogously.

By Lemma 21, there is an Armstrong population $\underline{r}^s$ for Σ_C (with respect to $\mathcal{C}$) such that $\underline{r}^s$ is of size $\lambda h_r(\underline{r})$ and every component type $\underline{u}$ has $\lambda h_r(\underline{u})$ instances. Again, λ denotes a sufficiently large integer. Applying the construction presented in the proof of Theorem 8, we obtain an Armstrong population $\underline{r}^{s*}$ for Σ_C (with respect to $\mathcal{C}$) such that any two relationships have at most one entry in common. $\underline{r}^{s*}$ is of size $q\lambda h_r(\underline{r})$ and every component type $\underline{u}$ has $q\lambda h_r(\underline{u})$ instances.

Next, we take $q\lambda$ copies of the population $\underline{r}^t$ obtained in Lemma 23, and join them with the population $\underline{r}^{s*}$ just constructed. Let $\underline{r}^z$ denote the resultant population. Due to our construction, $\underline{r}^z$ satisfies every constraint in $\Sigma_K \cup \Sigma_C$. Since $\underline{r}^{s*}$ is a subpopulation, no cardinality constraint not in Σ_C holds in $\underline{r}^z$. Moreover, since $\underline{r}^t$ from Lemma 23 is a subpopulation, $\underline{r}^z$ satisfies no key dependency not in Σ_K. This proves the claim. □

The population $\underline{r}^z$ obtained above is of size $h(\underline{r}) = q\lambda\mu g(\underline{r}) = q\lambda g_r(\underline{r}) + q\lambda h_r(\underline{r})$. Analogously, every component type $\underline{u}$ of $\underline{r}$ has $h(\underline{u}) = q\lambda\mu g(\underline{u})$ instances.

Now, we are ready to prove Theorem 13.

Proof (of Theorem 13). Condition (10) is an easy consequence of Lemma 7. This shows the necessity. We shall proceed with the sufficiency.

Assume, Σ_C contains a non-trivial max-cardinality constraint for every relationship type in $\underline{S}$. To prove $\Sigma_C \cup \Sigma_K$ to be closed in $\mathcal{C} \cup \mathcal{K}$, we shall construct a database $\underline{S}^z$ satisfying all constraints in $\Sigma_C \cup \Sigma_K$, but no constraint not in $\Sigma_C \cup \Sigma_K$.

The idea is to comprise the populations constructed above. For every entity type $\underline{e}$ in the database schema $\underline{S}$, we take $h(\underline{e}) = q\lambda\mu g(\underline{e})$ instances. Afterwards, we add the populations $\underline{r}^z$ hierarchically to the database. Finally, we obtain a database $\underline{S}^z$. As shown above, $\underline{S}^z$ satisfies all constraints in $\Sigma_C \cup \Sigma_K$, but no other constraint from $\mathcal{C} \cup \mathcal{K}$. Hence, $\Sigma_C \cup \Sigma_K$ is closed in the class of cardinality constraints and key dependencies. Moreover, $\underline{S}^z$ is an Armstrong database for $\Sigma_C \cup \Sigma_K$ (with respect to $\mathcal{C} \cup \mathcal{K}$).

If Σ_C does not contain a non-trivial max-cardinality constraint for every relationship type in $\underline{S}$, we may conclude in a similar way. However, this time, we do not obtain an Armstrong database, but an Armstrong sequence of sample databases. □

References

1. W.W. Armstrong, Dependency structures of database relationship, Information Processing 74 (1974) 580-583.
2. C. Beeri, M. Dowd, R. Fagin and R. Statman, On the structure of Armstrong relations for functional dependencies, J. ACM 31 (1984) 30-46.
3. J. Biskup, R. Menzel, T. Polle and Y. Sagiv, Decomposition of relationships through pivoting, in: B. Thalheim (ed.), Conceptual Modeling (Springer, Berlin, 1996) 28-41.
4. P.P. Chen, The Entity-Relationship Model: Towards a unified view of data, ACM Trans. Database Syst. 1 (1984) 9-36.
5. E.F. Codd, A relation model of data for large shared data banks, Commun. ACM 13 (1970) 377-387.
6. C. Delobel and R.G. Casey, Decompositions of a database and the theory of Boolean switching functions, IBM J. Res. Dev. 17 (1973) 374-386.
7. J. Demetrovics, Z. Füredi and G.O.H. Katona, Minimum matrix representation of closure operations, Discrete Appl. Math. 11 (1985) 115-128.
8. J. Demetrovics and V.D. Thi, Some results about functional dependencies, Acta Cybern. 8 (1988) 273-278.
9. A. Hajnal and E. Szemeredi, Proof of a conjecture of Erdős, in: P. Erdős, A. Renyi and V.T. Sos (eds.), Combinatorial theory and its applications, Colloq. Math. Soc. János Bolyai 4 (North-Holland, Amsterdam, 1970) 601-623.
10. S. Hartmann, Graph-theoretic methods to construct entity-relationship databases, in: M. Nagl (ed.), Graph-theoretic concepts in computer science, LNCS 1017 (Springer, Berlin, 1995) 131-145.
11. S. Hartmann, On the consistency of int-cardinality constraints, in: T.W. Ling, S. Ram and M.L. Lee (eds.), Conceptual Modeling, LNCS 1507 (Springer, Berlin, 1998) 150-163.
12. D. Jungnickel, T. Beth and H. Lenz, Design Theory (BI, Mannheim, 1985).
13. M. Lenzerini and P. Nobili, On the satisfiability of dependency constraints in Entity-Relationship schemata, Inf. Syst. 15 (1990) 453-461.
14. S.W. Liddle, D.W. Embley and S.N. Woodfield, Cardinality constraints in semantic data models, Data Knowl. Eng. 11 (1993) 235-270.
15. D. Maier, The theory of relational databases (Computer Science Press, Rockville, 1983).
16. H. Mannila and K. Räihä, Design by example: an application of Armstrong relations, J. Comput. Syst. Sci. 33 (1986) 126-141.

17. J. Paredaens, P. De Bra, M. Gyssens and D. Van Gucht, The structure of the relational database model (Springer, Berlin, 1989).
18. B. Thalheim, Dependencies in Relational Databases (Teubner, Stuttgart, 1991).
19. B. Thalheim, Foundations of Entity-Relationship Modeling, Ann. Math. Artif. Intell. 6 (1992) 197-256.
20. B. Thalheim, Fundamentals of cardinality constraints, in: G. Pernul and A.M. Tjoa (eds.), Entity-relationship approach, LNCS 645 (Springer, Berlin, 1992) 7-23.
21. D. Theodorates, Deductive object oriented schemas, in: B. Thalheim (ed.), Conceptual modeling, LNCS 1157 (Springer, Berlin, 1996) 58-72.
22. G.E. Weddell, Reasoning about functional dependencies generalized for semantic data models, ACM Trans. Database Syst. 17 (1992) 32-64.
23. J. Wijsen, J. Vandenbulcke and H. Olivie, Functional dependencies generalized for temporal databases, in: R. Elmasri, V. Kouramajian and B. Thalheim (eds.), Entity-Relationship approach, LNCS 823 (Springer, Berlin, 1994) 99-109.