On the implication problem for cardinality constraints and functional dependencies*

Sven Hartmann

Abstract

In database design, integrity constraints are used to express database semantics. They specify the way by that the elements of a database are associated to each other. The implication problem asks whether a given set of constraints entails further constraints. In this paper, we study the finite implication problem for cardinality constraints. Our main result is a complete characterization of closed sets of cardinality constraints. Similar results are obtained for constraint sets containing cardinality constraints, but also key and functional dependencies. Moreover, we construct Armstrong databases for these constraint sets, which are of special interest for example-based deduction in database design.

Keywords: cardinality constraint, key, functional dependency, implication problem, Armstrong database, transversal design, clique graph

MSC (1991): 68R10, 68P15

*to appear in: *Annals of Mathematics and Artificial Intelligence.*

Address: FB Mathematik, Universität Rostock, 18051 Rostock, Germany
E-mail address: sven.hartmann@mathematik.uni-rostock.de

1 Introduction

During the last few decades, the area of integrity constraints has attracted considerable research interest in database theory. A large amount of different constraint classes has been discussed in the literature and actually used in database design. Integrity constraints are intended to describe the properties that a database must respect to plausibly represent the underlying domain of interest. Specifying and enforcing integrity constraints helps to guarantee the correctness of the data collected in the database.

Many papers in database theory exclusively deal with the implication problem, i.e. the question 'Given a set Σ of constraints and a further constraint σ, does every database instance satisfying Σ also satisfy σ?' We refer the reader to [3, 42, 47], which are major comprehensive sources on database semantics and the emergence of the implication problem in database theory.

In the sequel, we deal with cardinality constraints, which are among the most popular constraint classes in conceptual database design. Cardinality constraints impose restrictions on the number of relationships an object of a given type may be involved in. Thus they limit the possible structure of a database. We refer the reader to the surveys [39, 48] for details and to [13, 35, 38, 43, 49] for recent research papers. The present paper is devoted to the implication problem for cardinality constraints. Our goal is to provide efficient combinatorial methods for reasoning about sets of cardinality constraints.

Cardinality constraints appear in various forms in most of the data models proposed in the last three decades. In order to present our results in the exact context, we shall use the language of the entity-relationship model. Since the appearance of the seminal work of Chen [14], this data model has become the standard approach towards conceptual database modeling. 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 must achieve, and serves as the basis for the following development phases. In conceptual design great attention is devoted to the modeling of semantics, i.e. to the specification of suitable integrity constraints. Cardinality constraints, in particular, have already been present in the original paper of Chen [14]. Comprehensive text books on entity-relationship modeling are [4, 15, 50].

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 data model, which goes back to Codd [17], has a strong theoretical background which can be reused in most other data models. Extended entity-relationship models support all the major concepts of the relational model. For details and further discussion, we refer to [50].

Codd [17] 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 may be used to identify the tuples in a relation.

Due to the their importance, functional dependencies are also available in extended entity-relationship models. As pointed out in [50], results obtained from relational database theory concerning functional dependencies can easily be reused. This is of interest since the entity-relationship model is not only popular as an intuitive tool for conceptual design, but may serve as a mathematical data model upon which real database management systems are built.

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 whether the specified constraint set is satisfiable. Unfortunately, efficient algorithms for consistency and implication checking are still missing for this situation. But 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.

A second objective of the present paper is to discuss the interplay between cardinality constraints and key or functional dependencies. In particular, we provide methods for reasoning about a set of cardinality constraints, key dependencies and certain functional dependencies, namely non-unary functional dependencies. This enables us to check whether a given constraint set implies further constraints of any of the three classes.

This paper is organized as follows. Section 2 provides some preliminary notations. In Section 3, we briefly describe the data model to be used. In Section 4, we give a formal definition of the constraints mentioned above. Moreover, the consistency and the implication problem for constraint classes are addressed. Sections 5 to 10 are devoted to the investigation of cardinality constraints. In Sections 11 and 12, we study interactions of cardinality constraints and key or functional dependencies. In Section 13, we discuss some problems concerning unary functional dependencies.

2 Preliminaries

2.1 Digraphs

In this section, we assemble basic terminology and known results to be used later on. We shall start with some notions from graph theory. Let $\vec{D} = (V, A)$ be a *digraph* with vertex set V and arc set A. When $a = (v, w)$ is an arc in A, then v is its initial vertex and w is its terminal vertex.

A *diwalk* $\vec{P}$ from a vertex $v = v_0$ to a vertex $w = v_k$ is a sequence of arcs (v_0, v_1), $(v_1, v_2), \dots, (v_{k-1}, v_k)$ in A. Here, k denotes the length of the diwalk. Again, v is the initial vertex and w the terminal vertex of $\vec{P}$. A diwalk is said to be *empty* if v equals w and $k = 0$, while it is said to be *closed* if v equals w and $k > 0$. A diwalk is a *dipath* if the vertices $v_0, v_1, \dots, v_k$ are mutually distinct. In particular, every empty diwalk is a dipath. Analogously, a closed diwalk is a *dicycle* if the vertices $v_1, \dots, v_k$ are mutually distinct.

2.2 Dioids and Farkas' lemma

To continue with, we collect some results from discrete optimization. Within this paper, most calculations are performed using integers or rationals. By $\mathbb{N}_n$ and $\mathbb{Q}_n$ we denote the set of all integers and the set of all rationals, respectively, larger or equal to n. Sometimes, it will be necessary to adjoin a new element ∞. We declare $z + \infty = \infty + z = \infty$, $z\infty = \infty z = \infty$ and $z < \infty$ for every $z \in \mathbb{Q}$.

Given two binary operations $\oplus$ and $\odot$ on a set M, the triple $(M, \oplus, \odot)$ is a commutative *dioid* (or *path algebra*) if M forms a commutative monoid with either of these operations, and $\oplus$ is distributive with regard to $\odot$. Clearly, $(\mathbb{Q}_0 \cup \{\infty\}, \min, \cdot)$ is an example of a commutative dioid. For further details on dioids, we refer to [30, 37].

A fundamental result on linear equations and inequations in $\mathbb{Q}_0$ is given by the well-known lemma of Farkas, which we record here for further reference.

Theorem 1 (Farkas). *Given a rational m-by-n-matrix A and a rational m-vector b, there is a rational n-vector $x \geq o$ satisfying $Ax = b$ if and only if we have $b^T y \geq 0$ for every rational m-vector y with $A^T y \geq o$.*

Farkas' lemma also yields a characterization for strict inequalities, which is usually attributed to Motzkin, cf. [45].

Theorem 2 (Motzkin). *Given a rational m-by-n-matrix A and a rational m-vector b, there is a rational n-vector $x \geq o$ with positive component x_d, and satisfying $Ax = b$, if and only if we have $b^T y \geq 0$ for every rational m-vector y with $A^T y \geq o$, such that $b^T y > 0$ holds whenever the component $(A^T y)_d$ is positive.*

2.3 Transversal designs

We conclude with some notions from combinatorial design theory. For a mapping $\pi : X \to Z$ and a subset Y of X, let $\pi[Y]$ denote the restriction of π to the domain Y. Two mappings π and π' with domain X agree in a component $x \in X$, when we have $\pi(x) = \pi'(x)$. Two mappings with domain X are disjoint, if they do not agree in any element of X. They are said to be different, if they disagree in at least one element of X.

Given positive integers n and q, let T be a collection of q^2 mappings from $\{1, \dots, n\}$ to $\{1, \dots, q\}$. We call T a *transversal design* $TD(n, q)$ if any two of the mappings agree in at most one component. The members of T are also known as the *blocks* of the transversal design. Transversal designs have been widely studied in literature. We refer the interested reader to [8, 18, 26], which are major comprehensive sources for combinatorial design theory. Standard results are due to MacNeish [40], and Chowla, Erdős and Straus [16].

Theorem 3 (MacNeish). *Given a positive integer n, there exists a transversal design $TD(n, q)$ for every prime power q with $q \geq n - 1$.*

Theorem 4 (Chowla, Erdős, Straus). *Given a positive integer n, there exists a transversal design $TD(n, q)$ for almost every positive integer q.*

A set of q mutually disjoint blocks in a transversal design T is said to be a *resolution class*. T is *resolvable* if it is decomposable into q resolution classes. It is noteworthy, that every transversal design $TD(n + 1, q)$ yields a resolvable transversal design $TD(n, q)$, cf. [8].

3 The data model to be used

The entity-relationship approach to conceptual design, first introduced by Chen [14], considers the target of a database as consisting of entities and relationships. In this section, we shall briefly review basic concepts of this approach. We restrict ourselves to a characteristic subset of design primitives that happen to be essential for our further investigation. For excellent surveys on entity-relationship modeling, we refer to [42, 50].

Entities and relationships are objects that are stored in a database. Intuitively, entities may be seen as basic objects in the domain of interest, whereas relationships are derived objects representing connections between other objects. Usually, a database contains lots of objects with common properties. By classifying them and pointing out their significant properties, we obtain *object types* that are used to model the objects under discussion. All objects modeled by an object type $\underline{v}$ form an *object set* $\underline{v}^t$. Its members are said to be *instances of type* $\underline{v}$. Entities are instances of entity types, while relationships are instances of relationship types.

3.1 Database schemas

All object types declared for some application, collectively, form a database schema. Throughout this paper, a *database schema* is a finite digraph $\vec{S} = (V, L)$ without dicycles. Its vertices are the *object types* of the schema. Its arcs are said to be *links*. Let $(\underline{r}, \underline{c})$ be a link in the schema. Then $\underline{c}$ is said to be a *component* of $\underline{r}$. For every object type $\underline{r}$, let $\mathrm{Co}(\underline{r})$ denote the set of all its components.

A vertex $\underline{e}$ is an *entity type* if it does not admit any component, i.e. if $\mathrm{Co}(\underline{e})$ is empty. Designing a database usually starts with specifying entity types to model the basic real-world objects in the target of the database. All remaining vertices are *relationship types*. Roughly speaking, each relationship type $\underline{r}$ reflects real-world connections between objects of the types in $\mathrm{Co}(\underline{r})$.

The *arity* of an object type $\underline{v}$ is the number of its components. The *order* of $\underline{v}$ is the maximum length of a dipath with initial vertex $\underline{v}$. Obviously, entity types have arity 0 and order 0. Due to its definition, a database schema does not contain dicycles. Thus for every relationship type $\underline{r}$ of order k, its components are of order $k-1$ or smaller. This property is known as the *hierarchical structure* of database schemas: Every relationship type $\underline{r}$ is declared on the basis of types already specified, i.e. types of smaller order.

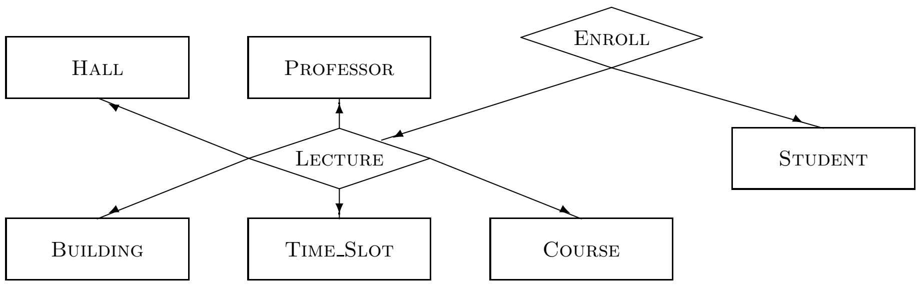

Figure 1: The database schema for our example.

Example. Consider a small university database schema involving the entity types Professor, Course, Hall, Building, Time_Slot and Student. On these object types we define a relationship type Lecture reflecting assignments of professors to courses in combination with the dates and locations they take place at. The component set of Lecture is Co(Lecture) = {Professor, Course, Hall, Building, Time_Slot}. A second relationship type Enroll with component set Co(Enroll) = {Student, Lecture} represents enrollments of students to offered lectures.

Figure 1 shows the corresponding database schema. As usual, entity types are drawn as rectangles, and relationship types as diamonds. This representation is known as the entity-relationship diagram of a database schema.

3.2 Populations and database instances

A *population* $\underline{v}^t$ over an object type $\underline{v}$ is a finite set of objects represented by that type. Since real-world databases are finite, we restrict ourselves to finite object sets throughout this paper.

Let $\underline{r}$ be a relationship type and $\mathrm{Co}(\underline{r}) = \{\underline{c}_1, \ldots, \underline{c}_n\}$ its component set. In the entity-relationship model, relationship types are regarded as aggregations of their components. In this vein, every relationship r of type $\underline{r}$ yields a mapping which assigns an object c_i of type $\underline{c}_i$ to the component $\underline{c}_i$. We also write $r(\underline{c}_i) = c_i$. The objects $r(\underline{c}_i)$ are said to *be involved* in r or to *participate* in r. For the sake of convenience, we use the notation $r = r(c_1, \ldots, c_n)$ to refer to a relationship r with $r(\underline{c}_1) = c_1, \ldots, r(\underline{c}_n) = c_n$.

Example. By this convention, Ma007(Mullin, Cryptography, Big_Hall, Main_Building, Tue7-9) is a relationship of type Lecture, and En1704(Ron, Ma007) is a relationship of type Enroll.

A population $\underline{r}^t$ over $\underline{r}$ is a finite set of relationships of type $\underline{r}$. Usually, we suppose the objects $r(\underline{c}_i)$ for every relationship $r \in \underline{r}^t$ to be chosen from a prespecified object set $\underline{c}_i^t$ over $\underline{c}_i$. Thus a population $\underline{r}^t$ gives rise to a set of named mappings from $\mathrm{Co}(\underline{r})$ to the cartesian product $\underline{c}_1^t \times \ldots \times \underline{c}_n^t$. In addition, we call $\underline{c}_i^t$ a *codomain* of the population $\underline{r}^t$.

A *database instance* or *database*, for short, $\vec{S}^t$ contains a population $\underline{v}^t$ for each object type $\underline{v}$ in the database schema $\vec{S}$. It suggests itself that in a database instance $\vec{S}^t$, the codomains of a relationship type $\underline{r}$ are just the populations $\underline{c}_i^t$ over its components $\underline{c}_i \in \mathrm{Co}(\underline{r})$.

3.3 Further constituents of the data model

Above we gave an informal introduction to database schemas in the entity-relationship model. Usually, an entity type is characterized by *attributes* representing the properties of the instances of this type. The same applies to relationship types: In addition to their components they may possess attributes to describe relationship instances in greater detail. In this paper, however, the explicit specification or absence of attributes has no impact on the results. If desired, attributes may be considered like components. We may even specify cardinality constraints for links between object types and their attributes.

For some applications it is useful to allow parallel links in the database schema. Then an object type occurs several times as a component of the same relationship type. To avoid confusion, *roles* are associated with the different occurrences. As an example consider a relationship type Descendent whose components are both of type Person. The roles of the two components could be Parent and Child. Then the database schema is a digraph with multiple arcs. Though not always explicitly set out, all our considerations do also apply to this case. In particular, the complexity of the algorithms determined below takes into account the possibility of multiple arcs.

4 Integrity constraints

In this section, we give formal definitions of the constraints to be studied. Throughout, let $\underline{r}$ be a relationship type, and $\underline{r}^t$ be a population with codomains $\underline{c}^t$ for the components $\underline{c} \in \mathrm{Co}(\underline{r})$.

4.1 Cardinality constraints

For every object c in the codomain $\underline{c}^t$, let $\deg(\underline{r}^t, c)$ denote the number of relationships r in $\underline{r}^t$ with $r(\underline{c}) = c$. We call $\deg(\underline{r}^t, c)$ the *degree* of c with respect to the population $\underline{r}^t$.

A *cardinality constraint* on $\underline{r}$ is a statement $card(\underline{r}, \underline{c}) = M$, where M is a set of non-negative integers. The population $\underline{r}^t$ satisfies this constraint if, for every object c in the codomain $\underline{c}^t$, the degree $\deg(\underline{r}^t, c)$ lies in M.

Cardinality constraints are often reflected graphically in the entity-relationship diagram: Given a cardinality constraint $card(\underline{r}, \underline{c}) = M$, the corresponding link $(\underline{r}, \underline{c})$ is labeled with the set M. If no cardinality constraint is specified for a link $(\underline{r}, \underline{c})$, we may assume the *trivial* cardinality constraint $card(\underline{r}, \underline{c}) = \mathbb{N}_0$. It is easy to see that this does not represent a real constraint, but is just a technical agreement.

A cardinality constraint is said to be *finite* if the set M is finite, and *infinite* otherwise. In practice, the sets M are often intervals, i.e. of the form $\{\alpha, \alpha + 1, \ldots, \beta\}$ or $\{\alpha, \alpha + 1, \ldots\}$.

Example. A travel agency is going to organize sight-seeing tours through Europe. Each of the offered tours visits a number of Europe's most popular cities. The itinerary of the tours for every week is planned according to the database schema in Figure 2. It contains three entity types Tour, City and Guide and two relationship types Start and Visit. Relationships of type Start determine in which city a certain tour starts, relationships of type Visit specify that a certain city is visited during a given tour.

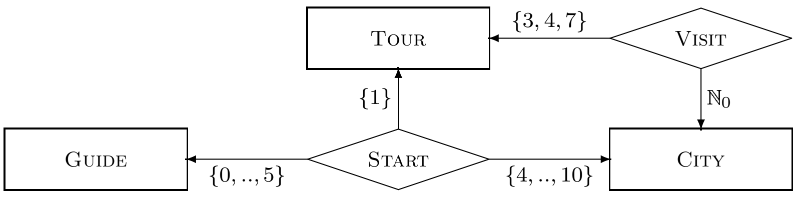

Figure 2: A database schema with labels for the cardinality constraints.

Obviously, every tour has exactly one starting point. This motivates the cardinality constraint $card$(Start, Tour) $= \{1\}$. Further the management decided that every tour visits 3 or 4 cities. But, from time to time they offer special tours visiting 7 cities to attract new clients. Together, this is expressed by the cardinality constraint

$card(\textsc{Visit}, \textsc{Tour}) = \{3, 4, 7\}$. Figure 2 shows all cardinality constraints specified by the travel agency.

4.2 Key dependencies

A *key dependency* on $\underline{r}$ is a statement $\underline{r} : X \to \underline{r}$ where X is a non-empty subset of $\mathrm{Co}(\underline{r})$. The population $\underline{r}^t$ satisfies this key dependency if the restrictions $r[X]$ are mutually distinct for all relationships r in $\underline{r}^t$. In this case, the subset X is called a *key* for the population $\underline{r}^t$. A key dependency is said to be *unary* if X consists of a single component only, and *non-unary* otherwise.

4.3 Functional dependencies

Finally, a *functional dependency* on $\underline{r}$ is a statement $\underline{r} : X \to Y$ where both, X and Y are non-empty subsets of $\mathrm{Co}(\underline{r})$. The population $\underline{r}^t$ satisfies this functional dependency if 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 $\underline{r}^t$. Again, a functional dependency is said to be *unary* if X consists of a single component only, and *non-unary* else. Furthermore, we call a functional dependency *trivial* if Y is a subset of X.

A population $\underline{r}^t$ satisfying the key dependency $\underline{r} : X \to \underline{r}$ also satisfies the functional dependency $\underline{r} : X \to \mathrm{Co}(\underline{r})$. In fact, most functional dependencies occurring in practice arise from the specification of key dependencies.

Example. Below we give a small example to provide some motivation for the issues we are going to tackle. In the university schema, constraints have to be declared in order to represent the politics of the university and to meet the requirements of the schedule. Let us give some examples. The cardinality constraint $card(\textsc{Lecture}, \textsc{Professor}) = \{3, \ldots, 6\}$ states that every professor gives 3 to 6 lectures per term. If every professor always uses the same hall per course, then the non-unary functional dependency $\textsc{Lecture}{:}\{\textsc{Professor}, \textsc{Course}\} \to \{\textsc{Hall}\}$ holds. Since every lecture hall is situated in a single building, we also have the unary functional dependency $\textsc{Lecture}{:}\{\textsc{Hall}\} \to \{\textsc{Building}\}$. Moreover, knowing the hall and the time_slot uniquely determines the given lecture. This may be expressed by the key dependency $\textsc{Lecture}{:}\{\textsc{Hall}, \textsc{Time\_Slot}\} \to \textsc{Lecture}$.

Other constraints may be declared 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.

Let $\mathcal{C}, \mathcal{K}$ and $\mathcal{F}$ denote the classes of all cardinality constraints, key dependencies or functional dependencies definable on the relationship types in a given database schema $\vec{S}$. Below, if we refer to a constraint, we always mean one from these classes. A database $\vec{S}^t$ satisfies a certain constraint σ, if σ is satisfied by the corresponding population $\underline{r}^t$ in $\vec{S}^t$.

4.4 Satisfiability, consistency and implication

The typical situation in database design is that we are given a database schema $\vec{S}$ together with an application-dependent constraint set Σ declared on it. A database instance $\vec{S}^t$ is only meaningful for the application under discussion, if it satisfies all constraints in Σ. We call such a database instance *legal for* Σ. Consequently, Σ is *satisfiable* if it admits at least one legal database instance.

At this point, a brief remark is called for. As mentioned earlier, we only pay attention to finite databases. When we speak of satisfiability, consistency or implication, we always mean *finite* satisfiability, *finite* consistency and *finite* implication.

The *satisfiability problem* for a constraint class $\mathcal{Z}$ is to decide whether a given set $\Sigma \subseteq \mathcal{Z}$ is satisfiable or not. For the constraint classes studied in the present paper, this problem is fairly easy. Obviously, the empty database is legal for every set of cardinality constraints, key and functional dependencies. But, as pointed out in [38], it is usually not enough to consider satisfiability: It may happen that there is an object type in $\vec{S}$ with empty population in every legal database instance. Such an object type is said to be *redundant for* Σ. Examples for cardinality constraints are given in [35, 38].

For practical reasons, we are interested in efficient methods to detect redundant object types. With this, we may assist the designer of a database in order to delete redundant parts of the schema or to repair the specified constraint set. For some applications, one explicitly asks for fully-populated legal database instances, i.e. legal databases where every object type has non-empty population. A constraint set Σ admitting such a database instance is said to be *consistent*.

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 special subclasses of cardinality constraints, the consistency problem has been considered e.g. in [34, 38, 49]. A general result was presented in [35]. 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 2. The specified set of cardinality constraints is consistent. Now add the key dependency $\textsc{Visit}{:}\{\textsc{City}\} \to \textsc{Visit}$. The resulting constraint set turns out to be inconsistent, and $\textsc{Guide}$ becomes the only non-redundant object type. We discuss this observation in Section 11.

If Σ is a constraint set and σ a single constraint, then Σ semantically *implies* σ (denoted by $\Sigma \models \sigma$) if every legal database instance for Σ satisfies σ, too. Conversely, $\Sigma \not\models \sigma$ denotes that there is a legal database instance (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 *$\mathcal{Z}$-closed* if $\sigma \in \Sigma$ holds whenever $\Sigma \models \sigma$ holds for every constraint $\sigma \in \mathcal{Z}$.

The determination of semantically closed sets is of particular interest. Clearly, $\Sigma \models \sigma$ holds if and only if 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 semantically *equivalent*, i.e. whether we have $\Sigma_1 \models \Sigma_2$ as well as $\Sigma_2 \models \Sigma_1$. Database designers usually look for some equivalent constraint set which is better for control and maintenance in real-life databases.

It is sometimes argued, that the consistency problem and the implication problem are two sides of the same coin, cf. [51]. 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 a functional independency. For a discussion of this issue, we refer to [9, 50].

4.5 Armstrong databases

Informally, Armstrong databases are legal databases capturing all implications of a constraint set Σ in $\mathcal{Z}$. Recall that, if $\Sigma \not\models \sigma$ holds, there is a legal database (a certificate) for Σ violating 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.

Let $\mathcal{Z}$ be a constraint class and Σ be some constraint set. A database $\vec{S}^t$ is *$\mathcal{Z}$-Armstrong for* Σ if it is legal for Σ and satisfies a constraint $\sigma \in \mathcal{Z}$ if and only if $\Sigma \models \sigma$.

Assume, we have found an Armstrong database for an initial constraint set Σ. In order to test whether a constraint follows from Σ we may test whether it holds in this 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 important for design-by-example as suggested by Mannila and Räihä [41]. In Section 10, we shall consider Armstrong databases for sets of cardinality constraints.

A useful tool to establish Armstrong databases are disjoint unions, that is, unions of mutually disjoint database instances over the same schema. It is not difficult to see that the union satisfies a given cardinality constraint (key dependency, functional dependency) if and only each of the participating database instances does so. This is a valuable property of the constraint classes under discussion.

5 Reasoning about cardinality constraints

5.1 The main result

In this section, we start investigating the implication problem for cardinality constraints. Suppose we are given a set Σ_C of cardinality constraints specified on a database schema $\vec{S}$. In the subsequent sections we are going to prove the correctness of the following implications for every link $(\underline{r}, \underline{c})$ in $\vec{S}$:

(C1) $\Sigma_C \models \big(card(\underline{r}, \underline{c}) = \mathbb{N}_0\big)$.

(C2) If Σ_C contains $card(\underline{r}, \underline{c}) = M$, then $\Sigma_C \models \big(card(\underline{r}, \underline{c}) = M'\big)$ for every superset $M' \supseteq M$.

(C3) If Σ_C contains $card(\underline{r}, \underline{c}) = M$ for all $M \in \mathcal{M}$, then $\Sigma_C \models \big(card(\underline{r}, \underline{c}) = \bigcap_{M \in \mathcal{M}} M\big)$.

(C4) If $\underline{c}$ is redundant for Σ_C, then $\Sigma_C \models \big(card(\underline{r}, \underline{c}) = \emptyset\big)$.

(C5) If $\underline{r}$ is redundant for Σ_C, then $\Sigma_C \models \big(card(\underline{r}, \underline{c}) = \{0\}\big)$.

(C6) If the link $(\underline{r}, \underline{c})$ lies on a subcritical dicycle for Σ_C, then $\Sigma_C \models \big(card(\underline{r}, \underline{c}) = \{\alpha(\underline{r}, \underline{c})\}\big)$.

(C7) If the reverse arc of the link $(\underline{r}, \underline{c})$ lies on a subcritical dicycle for Σ_C, then $\Sigma_C \models \big(card(\underline{r}, \underline{c}) = \{\beta(\underline{r}, \underline{c})\}\big)$.

Details on these rules as well as missing notions will be given later on. The correctness of (C1) to (C3) is evident. As an example consider a population $\underline{r}^t$ satisfying both $card(\underline{r}, \underline{c}) = \{3, \ldots, 7\}$ and $card(\underline{r}, \underline{c}) = \{d \in \mathbb{N}_0 : \ d \text{ even}\}$. Every object c in the codomain $\underline{c}^t$ has degree $\deg(\underline{r}^t, c)$ which is even and between 3 and 7, i.e. is either 4 or 6. Hence, we may conclude that $\underline{r}^t$ satisfies $card(\underline{r}, \underline{c}) = \{4, 6\}$. This simple implication is formalized in (C3). We record this observation for further reference.

Lemma 5. *(C1), (C2) and (C3) are correct.*

The major objective of the present paper is to prove the following characterization for closed sets of cardinality constraints.

Theorem 6. *A set Σ_C of cardinality constraints specified on a database schema $\vec{S}$ is $\mathcal{C}$-closed if and only if it contains all constraints implied by Σ_C due to the rules (C1) to (C7).*

5.2 Candidate degrees

Throughout, let $\vec{S} = (V, L)$ be a database schema, and Σ_C be a set of cardinality constraints declared on $\vec{S}$. $\mathrm{CanDeg}(\underline{r}, \underline{c})$ denotes the intersection of all sets M such that $card(\underline{r}, \underline{c}) = M$ belongs to Σ_C. If no cardinality constraint is specified for the link $(\underline{r}, \underline{c})$, we put $\mathrm{CanDeg}(\underline{r}, \underline{c}) = \mathbb{N}_0$. By virtue of (C3) or (C1), we have $\Sigma_C \models \big(card(\underline{r}, \underline{c}) = \mathrm{CanDeg}(\underline{r}, \underline{c})\big)$. This constraint is the 'sharpest' possible obtained from Σ_C via the rules (C1) to (C3) only. Every original constraint may be reconstructed from this one by applying (C2). The values in $\mathrm{CanDeg}(\underline{r}, \underline{c})$ will be called the *candidate degrees* for the objects in the codomain $\underline{c}^t$, with respect to Σ_C.

Lemma 7. *Σ_C and the set $\Sigma_C^{Can} = \{card(\underline{r}, \underline{c}) = CanDeg(\underline{r}, \underline{c}) : (\underline{r}, \underline{c}) \in L\}$ of cardinality constraints are semantically equivalent.*

Proof. $\Sigma_C \models \Sigma_C^{Can}$ follows immediately from Lemma 5. Moreover, in each database instance $\vec{S}^t$ satisfying Σ_C^{Can}, we have $\deg(\underline{r}^t, c) \in \mathrm{CanDeg}(\underline{r}, \underline{c}) \subseteq M$ for every set M with $card(\underline{r}, \underline{c}) = M$ given in Σ_C. This verifies $\Sigma_C^{Can} \models \Sigma_C$. $\square$

Note that Σ_C^{Can} contains exactly one constraint per link. In practice, it usually suffices to specify only one cardinality constraint per link. Then the generation of Σ_C^{Can} turns out to be completely trivial.

Finally, we introduce some additional notations. For every link $(\underline{r}, \underline{c})$, let $\alpha(\underline{r}, \underline{c})$ and $\beta(\underline{r}, \underline{c})$ denote the infimum and the supremum of $\mathrm{CanDeg}(\underline{r}, \underline{c})$, respectively. If $\mathrm{CanDeg}(\underline{r}, \underline{c})$ is infinite, we have $\beta(\underline{r}, \underline{c}) = \infty$. If $\mathrm{CanDeg}(\underline{r}, \underline{c})$ is empty, we have $\alpha(\underline{r}, \underline{c}) = \infty$ and $\beta(\underline{r}, \underline{c}) = 0$.

The subsequent sections devoted to the proof of our main result are organized as follows. In Section 6, we introduce representation graphs and admissible functions, which turn out to be major tools to construct legal database instances. In Section 7, we present a procedure to detect object types with empty population in all legal database instances. These object types are redundant and give rise to the rules (C4) and (C5).

In Section 8, we continue the study of admissible functions. In Section 9, we point out the importance of subcritical dicycles. Their investigation results in the rather involved rules (C6) and (C7). Finally, in Section 10, we show the implication rules collected so far to be sufficient for the characterization of closed sets of cardinality constraints. This characterization yields the desired solution to the finite implication problem.

6 Representation graphs and admissible functions

The empty database is legal for every set of cardinality constraints. However, empty databases are of minor practical interest. In this section, we study the existence of legal database instances with prespecified population sizes. Most of the results here happen to be rather technical, but form the basis of what follows. In particular, Theorem 11 is of major interest for future investigation.

6.1 Representation graphs

We start with some graph-theoretical notions which turn out to be useful in the future. A *clique graph* is a graph where every connected component is a complete graph. Component means here a maximal connected subgraph, which should not be confused with the components of relationship types. To avoid irritation, we shall use the notion *clique* to refer to the connected components of a clique graph.

Suppose now, we are given a population $\underline{r}^t$. For every component $\underline{c} \in \mathrm{Co}(\underline{r})$ we consider a graph $G^t(\underline{r}, \underline{c})$ whose vertices are the relationships in $\underline{r}^t$. Two relationships r_1 and r_2 are connected by an edge in $G^t(\underline{r}, \underline{c})$ if and only if $r_1(\underline{c}) = r_2(\underline{c})$ holds. Obviously, $G^t(\underline{r}, \underline{c})$ is a clique graph, where each clique corresponds to exactly one object c in the codomain $\underline{c}^t$. We call $G^t(\underline{r}, \underline{c})$ the *representation graph* of the population $\underline{r}^t$ for the link $(\underline{r}, \underline{c})$. Given some clique graph G, it is easy to construct a population $\underline{r}^t$ whose representation graph $G^t(\underline{r}, \underline{c})$ is isomorphic to G.

Representation graphs may be used to reflect cardinality constraints:

Lemma 8. *Let $\underline{r}^t$ be a population with codomain $\underline{c}^t$ for the component $\underline{c} \in Co(\underline{r})$.*

(i) If $\underline{r}^t$ satisfies the cardinality constraint $card(\underline{r}, \underline{c}) = M$, then the size of every clique in $G^t(\underline{r}, \underline{c})$ lies in M.

(ii) Conversely, if the size of every clique in $G^t(\underline{r}, \underline{c})$ belongs to M, then $\underline{r}^t$ satisfies the cardinality constraint $card(\underline{r}, \underline{c}) = M \cup \{0\}$. If, in addition, the number of cliques in $G^t(\underline{r}, \underline{c})$ equals the size of the codomain $\underline{c}^t$, then $\underline{r}^t$ even satisfies $card(\underline{r}, \underline{c}) = M$.

Proof. In $G^t(\underline{r}, \underline{c})$, every clique corresponds to exactly one object c in the codomain $\underline{c}^t$. The vertices of this clique are just the relationships in which c participates in. The clique is of size $\deg(\underline{r}^t, c)$. If $card(\underline{r}, \underline{c}) = M$ is satisfied, we have $\deg(\underline{r}^t, c) \in M$. Thus the clique size lies in M. On the other hand, $G^t(\underline{r}, \underline{c})$ contains a clique for every object in the codomain $\underline{c}^t$, but those which do not participate in any relationship in $\underline{r}^t$. This proves the second observation. If the number of cliques in $G^t(\underline{r}, \underline{c})$ equals the size of $\underline{c}^t$, then every object c in $\underline{c}^t$ participates in at least one relationship in $\underline{r}$. □

Among others, Lemma 8 points out a peculiarity of the candidate degree 0. Given a population $\underline{r}^t$ it is easy to decide a cardinality constraint $card(\underline{r}, \underline{c}) = M$, supposed that M contains 0. But if 0 is not in M we need to know the codomain $\underline{c}^t$, or to be more precise, the size $g(\underline{c})$ of this codomain.

6.2 Databases with prespecified population sizes

Henceforth, let Σ_C be a set of cardinality constraints declared on a database schema $\vec{S} = (V, L)$.

Lemma 9. *Suppose we are given an object set $\underline{c}^t$ of size $g(\underline{c})$ for every component $\underline{c} \in Co(\underline{r})$ of a relationship type $\underline{r}$. Then there is a population $\underline{r}^t$ with codomains $\underline{c}^t$, such that $\underline{r}^t$ is of size $g(\underline{r})$ and satisfies Σ_C, if and only if for every link $(\underline{r}, \underline{c})$ there are non-negative integers x_d, with d running over the set $CanDeg(\underline{r}, \underline{c})$ of candidate degrees, such that*

$$\sum_{d \in CanDeg(\underline{r},\underline{c})} x_d = g(\underline{c}), \tag{1}$$

$$\sum_{d \in CanDeg(\underline{r},\underline{c})} d x_d = g(\underline{r}) \tag{2}$$

hold.

Proof. (*Necessity.*) Suppose we are given a population $\underline{r}^t$ of size $g(\underline{r})$ which satisfies Σ_C. Consider a link $(\underline{r}, \underline{c})$. Let x_d count the number of all objects c in the codomain $\underline{c}^t$ with degree $\deg(\underline{r}^t, c) = d$. Of course, the degree of every object c belongs to $\mathrm{CanDeg}(\underline{r}, \underline{c})$. This implies (1). Clearly, dx_d gives us the number of those relationships r in $\underline{r}^t$, whose component $c = r(\underline{c})$ is of degree $\deg(\underline{r}^t, c) = d$. Thus we obtain (2).

(*Sufficiency.*) Suppose, for every link $(\underline{r}, \underline{c})$, we are given non-negative integers x_d satisfying (1) and (2). We aim at constructing a population $\underline{r}^t$ with codomains $\underline{c}^t$ containing relationships $r_1, r_2, \dots, r_{g(\underline{r})}$.

For every link $(\underline{r}, \underline{c})$, consider a clique graph G with $g(\underline{r})$ vertices and with exactly x_d cliques of size d, for every $d > 0$ in $\mathrm{CanDeg}(\underline{r}, \underline{c})$. Now we assign the vertices of G to the relationships $r_1, \dots, r_{g(\underline{r})}$, and the cliques of G to the objects in the codomain $\underline{c}^t$. Thus G becomes the representation graph $G^t(\underline{r}, \underline{c})$ for the link $(\underline{r}, \underline{c})$. Due to (1), the number of cliques in G is at most $g(\underline{c})$. It equals $g(\underline{c})$ exactly when 0 is not a candidate degree.

By Lemma 8, $\underline{r}^t$ satisfies the cardinality constraint $card(\underline{r}, \underline{c}) = \mathrm{CanDeg}(\underline{r}, \underline{c})$. Consequently the same holds for every cardinality constraint in Σ_C specified for the link $(\underline{r}, \underline{c})$. □

As seen in the proof of Lemma 9, the question whether there is a suitable population $\underline{r}^t$ with given codomains $\underline{c}^t$ merely depends on the sizes $g(\underline{c})$ of the codomains. This motivates a two-step approach towards the construction of legal database instances. In a first step, we specify the population size $g(\underline{v})$ for every object type $\underline{v}$. In a second step, we associate the relationships in each population $\underline{r}^t$ to the objects in the codomains $\underline{c}^t$ as described above. The following result summarizes the demands on the population sizes in order to facilitate the construction of a legal database instance.

Theorem 10. *Given a function $g : V \to \mathbb{N}_0$, there is a legal database instance $\vec{S}^t$ for Σ_C with populations $\underline{v}^t$ of size $g(\underline{v})$, $\underline{v} \in V$, if and only if for every link $(\underline{r}, \underline{c})$ in $\vec{S}$ there are non-negative integers x_d satisfying the equations (1) and (2).*

Proof. In a database instance $\vec{S}^t$, the codomains $\underline{c}^t$ of a population $\underline{r}^t$ over a relationship type $\underline{r}$ are just the populations over the object types $\underline{c} \in \mathrm{Co}(\underline{r})$. Thus the claim is an easy consequence of Lemma 9. □

Unfortunately, it seems to be rather difficult to check whether a function g meets the conditions of Theorem 10. However, these conditions may be relaxed when we are satisfied with legal database instances whose population sizes are prespecified up to a common multiple. Though some information is thereby sacrificed, the result is sufficiently rich for our purposes. We obtain the following consequence of Theorem 10 by applying the lemma of Farkas.

Theorem 11. *Given a function $g : V \to \mathbb{Q}_0$, there is a positive integer λ and a legal database instance $\vec{S}^t$ for Σ_C with populations $\underline{v}^t$ of size $\lambda g(\underline{v})$, $v \in V$, if and only if for every link $(\underline{r}, \underline{c})$ in $\vec{S}$ we have*

$$g(\underline{r}) = 0 \qquad \text{if } g(\underline{c}) = 0, \tag{3}$$

$$\alpha(\underline{r}, \underline{c}) \leq \frac{g(\underline{r})}{g(\underline{c})} \leq \beta(\underline{r}, \underline{c}) \qquad \text{if } g(\underline{c}) > 0. \tag{4}$$

Proof. (*Necessity.*) Suppose there is a legal database instance $\vec{S}^t$ and some positive integer λ such that the populations in $\vec{S}^t$ are of size $\lambda g(\underline{v})$, respectively. Both (3) and (4) are consequences of (1) and (2). Note that for empty sets $\mathrm{CanDeg}(\underline{r}, \underline{c})$ of candidate degrees, (1) and (2) immediately imply $g(\underline{c}) = g(\underline{r}) = 0$.

(*Sufficiency.*) Suppose g satisfies the conditions (3) and (4) for every link. The idea is to apply Theorem 10.

For our purposes here, it happens to be inconvenient that some of the sets $\mathrm{CanDeg}(\underline{r}, \underline{c})$ might be infinite. Within this proof we restrict ourselves to finite subsets: If $\mathrm{CanDeg}(\underline{r}, \underline{c})$ is infinite, we pick some integer $\beta'(\underline{r}, \underline{c}) \in \mathrm{CanDeg}(\underline{r}, \underline{c})$ such that

$$g(\underline{r})/g(\underline{c}) < \beta'(\underline{r}, \underline{c})$$

holds whenever $g(\underline{c}) > 0$. Define

$$\mathrm{CanDeg}'(\underline{r}, \underline{c}) = \{d \in \mathrm{CanDeg}(\underline{r}, \underline{c}) : d \leq \beta'(\underline{r}, \underline{c})\}.$$

For the sake of simplicity, we also put $\mathrm{CanDeg}'(\underline{r}, \underline{c}) = \mathrm{CanDeg}(\underline{r}, \underline{c})$ and $\beta'(\underline{r}, \underline{c}) = \beta(\underline{r}, \underline{c})$ for all finite sets $\mathrm{CanDeg}(\underline{r}, \underline{c})$ of candidate degrees.

Consider a link $(\underline{r}, \underline{c})$. By the lemma of Farkas there exist non-negative rationals x_d satisfying the equations

$$\sum_{d \in \mathrm{CanDeg}'(\underline{r},\underline{c})} x_d = g(\underline{c}),$$

$$\sum_{d \in \mathrm{CanDeg}'(\underline{r},\underline{c})} d x_d = g(\underline{r}),$$

if and only if the inequality

$$pg(\underline{c}) + qg(\underline{r}) \geq 0 \tag{5}$$

holds for all rationals p, q satisfying $p + dq \geq 0$ for every candidate degree $d \in \mathrm{CanDeg}'(\underline{r}, \underline{c})$.

It remains to show that (3), (4) and $p + dq \geq 0$ for each $d \in \mathrm{CanDeg}'(\underline{r}, \underline{c})$ give us (5). If $g(\underline{c}) = 0$ then (3) yields $g(\underline{r}) = 0$, and condition (5) trivially holds. Otherwise suppose $g(\underline{c}) > 0$. We distinguish two cases. If $q \geq 0$, we obtain

$$p + \frac{g(\underline{r})}{g(\underline{c})} q \geq p + \alpha(\underline{r}, \underline{c}) q \geq 0,$$

since $\alpha(\underline{r}, \underline{c})$ is in $\mathrm{CanDeg}'(\underline{r}, \underline{c})$. If $q \leq 0$, we obtain

$$p + \frac{g(\underline{r})}{g(\underline{c})} q \geq p + \beta'(\underline{r}, \underline{c}) q \geq 0,$$

since $\beta'(\underline{r}, \underline{c})$ lies in $\mathrm{CanDeg}'(\underline{r}, \underline{c})$. In both cases we derive (5). Hence we may apply the lemma of Farkas. For the remaining candidate degrees d not in $\mathrm{CanDeg}'(\underline{r}, \underline{c})$, we simply put $x_d = 0$. This gives us (1) and (2).

However, the function g is only a rational one. Moreover, the values x_d ensured by the lemma of Farkas are rationals. We need integers to use Theorem 10. Therefore we have to find is a suitable integer $\lambda > 0$ such that all the values λx_d as well as all the values $\lambda g(\underline{v})$ are integers. The idea is to choose λ as a common multiple of all these rationals. Applying Theorem 10 to the integral function λg concludes the proof. $\square$

We call a function $g : V \to \mathbb{Q}_0$ *admissible for* Σ_C if it meets (3) and (4) for every link $(\underline{r}, \underline{c})$. By Theorem 11, g is admissible for Σ_C if and only if we can find a positive integer λ and a database instance $\vec{S}^t$ with population sizes $\lambda g(\underline{v})$ such that $\vec{S}^t$ is legal for Σ_C.

Finally, we record two simple observations, which are consequences of Theorem 11.

Lemma 12. *Let g be an admissible function for Σ_C. Then g satisfies $\alpha(\underline{r}, \underline{c}) g(\underline{c}) \leq g(\underline{r}) \leq \beta(\underline{r}, \underline{c}) g(\underline{c})$ for a link $(\underline{r}, \underline{c})$ whenever CanDeg$(\underline{r}, \underline{c})$ is non-empty.*

Proof. The claim follows by (4) when $g(\underline{c}) > 0$, and by (3) when $g(\underline{c}) = 0$ and $\alpha(\underline{r}, \underline{c}) < \infty$. But, $\alpha(\underline{r}, \underline{c})$ is finite exactly when $\mathrm{CanDeg}(\underline{r}, \underline{c}) \neq \emptyset$. This concludes the proof. $\square$

Lemma 13. *Let g_1 and g_2 be admissible functions for Σ_C, and μ_1 and μ_2 be non-negative rationals. Then $g = \mu_1 g_1 + \mu_2 g_2$ is admissible for Σ_C, too.*

Proof. The claim can be verified by the help of Theorem 11. However, it is easier to argue as follows. Since g_i $(i = 1, 2)$ is admissible, we find an integer λ_i and a legal database instance $\vec{S}^{t_i}$ for Σ_C with population sizes $\lambda_i g_i(\underline{v})$. The disjoint union of μ_i copies of $\vec{S}^{t_i}$ is again legal for Σ_C. Hence $\mu_i g_i$ is again admissible. Similarly, the disjoint union of $\lambda_2\mu_1$ copies of $\vec{S}^{t_1}$ and $\lambda_1\mu_2$ copies of $\vec{S}^{t_2}$ is again legal for Σ_C and has population sizes $\lambda_1\lambda_2(\mu_1 g_1(\underline{v}) + \mu_2 g_2(\underline{v}))$. Consequently, $\mu_1 g_1 + \mu_2 g_2$ is admissible. $\square$

7 Redundant object types

Throughout this section, let Σ_C be a set of cardinality constraints specified on a database schema $\vec{S} = (V, L)$. An object type $\underline{v}$ is said to be *redundant for* Σ_C if its population $\underline{v}^t$ is empty in every database instance $\vec{S}^t$ which is legal for Σ_C. This corresponds to the identity $g(\underline{v}) = 0$ which has to be met by every admissible function g for Σ_C. Lemma 13 immediately provides the following observation.

Lemma 14. *There is an admissible function g for Σ_C satisfying $g(\underline{v}) > 0$ for every non-redundant object type $\underline{v}$.*

In Section 5 we claimed the following implications for every link $(\underline{r}, \underline{c})$, which make use of redundant object types:

(C4) If $\underline{c}$ is redundant for Σ_C, then $\Sigma_C \models \big(card(\underline{r}, \underline{c}) = \emptyset\big)$.

(C5) If $\underline{r}$ is redundant for Σ_C, then $\Sigma_C \models \big(card(\underline{r}, \underline{c}) = \{0\}\big)$.

Lemma 15. *(C4) and (C5) are correct.*

Proof. Let $\vec{S}^t$ be a database instance which is legal for Σ_C. If the population $\underline{c}^t$ is empty, then the set of degrees $\deg(\underline{r}^t, c)$ happens to be empty when c runs over the (empty) set $\underline{c}^t$. This provides (C4). If the population $\underline{r}^t$ is empty, then no object in $\underline{c}^t$ can participate in a relationship in $\underline{r}^t$. This gives us (C5). $\square$

7.1 Ingredients from combinatorial optimization

In the sequel, we shall use methods from combinatorial optimization to determine all redundant object types for the given constraint set Σ_C. Consider the database schema $\vec{S} = (V, L)$. For every link $\ell = (\underline{r}, \underline{c})$, we define its *reverse arc* $\ell^{-1} = (\underline{c}, \underline{r})$. Let L^{-1} denote the set of all reverse arcs, and put $A = L \cup L^{-1}$. We now turn our attention to the resultant digraph $\vec{D} = (V, A)$, the *symmetric digraph* of $\vec{S}$. On its arc set A we define a weight function $\omega : A \to \mathbb{Q}_0 \cup \{\infty\}$ by

$$\begin{aligned} \omega(\ell) = & \begin{cases} \infty & \text{if } \alpha(\underline{r}, \underline{c}) = 0, \\ 0 & \text{if } \alpha(\underline{r}, \underline{c}) = \infty, \\ \frac{1}{\alpha(\underline{r}, \underline{c})} & \text{otherwise,} \end{cases} \\ \omega(\ell^{-1}) = & \ \beta(\underline{r}, \underline{c}), \end{aligned} \tag{6}$$

for the link $\ell = (\underline{r}, \underline{c})$. This weight function may easily be extended to diwalks by setting

$$\omega(\vec{P}) = \prod_{i=1}^{k} \omega((\underline{v}_{i-1}, \underline{v}_i))$$

for the diwalk $\vec{P}$ with arcs $(\underline{v}_{i-1}, \underline{v}_i)$. If $\vec{P}$ is an empty diwalk, we put $\omega(\vec{P}) = 1$.

Example. Figure 3 shows a database schema $\vec{S}$ together with its symmetric digraph $\vec{D}$. We labeled every arc a in $\vec{D}$ by its weight $\omega(a)$.

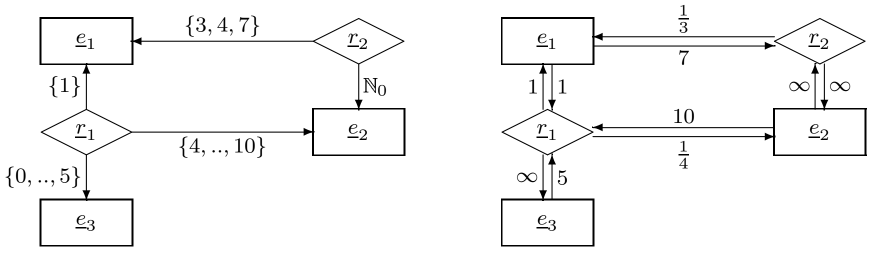

Figure 3: A database schema $\vec{S}$ with cardinality constraints and its symmetric digraph $\vec{D}$.

For any two object types $\underline{v}$ and $\underline{w}$, we now define the *distance* from $\underline{v}$ to $\underline{w}$ by

$$dist(\underline{v}, \underline{w}) = \inf\{\omega(\vec{P}) : \vec{P} \text{ is a diwalk from } \underline{v} \text{ to } \underline{w} \text{ in } \vec{D}\}.$$

Lemma 16. *A function $g : V \to \mathbb{Q}_0$ is admissible for Σ_C if and only if it satisfies (3) and*

$$g(\underline{w}) \leq g(\underline{v}) dist(\underline{v}, \underline{w}) \tag{7}$$

for any two object types $\underline{v}$ and $\underline{w}$.

Proof. (*Necessity.*) Let g be admissible. We have (3) and (4) for every link $\ell = (\underline{r}, \underline{c})$, and want to conclude the inequalities

$$g(\underline{c}) \leq g(\underline{r})\omega(\ell), \tag{8}$$
$$g(\underline{c}) \leq g(\underline{r})\omega(\ell^{-1}). \tag{9}$$

If $g(\underline{c}) = 0$, both inequalities trivially hold. If $g(\underline{c}) > 0$, the right hand side of (4) yields (9). Similarly, the left hand side of (4) gives us (8) when $0 < \alpha(\underline{r}, \underline{c}) < \infty$.

Moreover, $\alpha(\underline{r},\underline{c}) = \infty$ may be excluded due to $g(\underline{c}) > 0$, whereas (8) is trivial when $\alpha(\underline{r},\underline{c}) = 0$. By induction, we obtain $g(\underline{w}) \leq g(\underline{v})\omega(\vec{P})$ for every diwalk $\vec{P}$ from a vertex $\underline{v}$ to a vertex $\underline{w}$. This gives us (7).

(*Sufficiency.*) Suppose (3) and (7) are true. Applying inequality (7) to a link $\ell = (\underline{r},\underline{c})$ immediately yields (8) and (9). It remains to show (4). Therefore, we suppose $g(\underline{c}) > 0$. The right hand side of (4) is a consequence of (9), while the left hand side follows from (8) when $0 < \alpha(\underline{r},\underline{c}) < \infty$. Again $\alpha(\underline{r},\underline{c}) = \infty$ may be excluded due to $g(\underline{c}) > 0$, whereas the claim becomes trivial for $\alpha(\underline{r},\underline{c}) = 0$. This proves g to be admissible. □

If we restrict ourselves to dipaths, we may define the *elementary distance* from $\underline{v}$ to $\underline{w}$ by

$$edist(\underline{v},\underline{w}) = \inf\{\omega(\vec{P}) : \vec{P} \text{ is a dipath from } \underline{v} \text{ to } \underline{w} \text{ in } \vec{D}\}.$$

In particular we have $edist(\underline{v},\underline{v}) = 1$ for every vertex $\underline{v}$, since the empty diwalk from $\underline{v}$ to $\underline{v}$ is the only dipath from $\underline{v}$ to $\underline{v}$. The interplay between both definitions of distances is illustrated by the following result. A dicycle is said to be *absorbing* or *critical for* Σ_C if its weight is less than 1.

Lemma 17. *Let $\underline{v}$ and $\underline{w}$ be object types in $\vec{D}$.*

(i) If there is a diwalk from $\underline{v}$ to $\underline{w}$ in $\vec{D}$ containing an absorbing dicycle, we have $dist(\underline{v},\underline{w}) = 0$.

(ii) Conversely, if none of the diwalks from $\underline{v}$ to $\underline{w}$ in $\vec{D}$ contains an absorbing dicycle, then $dist(\underline{v},\underline{w}) = edist(\underline{v},\underline{w})$ holds.

Proof. (i) Suppose there is a diwalk $\vec{P}$ from $\underline{v}$ to $\underline{w}$ containing an absorbing dicycle $\vec{C}$. Then $\vec{C}$ has weight $\omega(\vec{C}) < 1$. On traversing $\vec{C}$ not only once but arbitrary often, we obtain diwalks of arbitrary small (non-negative) weight. This proves $dist(\underline{v},\underline{w}) = 0$ as claimed.

(ii) By definition, $dist(\underline{v},\underline{w}) \leq edist(\underline{v},\underline{w})$ holds. Assume we have strict inequality. Then there is at least one diwalk from $\underline{v}$ to $\underline{w}$, whose weight is smaller than $edist(\underline{v},\underline{w})$. Among all these diwalks, let $\vec{P}$ be one with a minimum number of arcs. Due to the assumption, $\vec{P}$ is not a dipath but contains some dicycle $\vec{C}$. Consider the diwalk $\vec{P}'$ obtained from $\vec{P}$ by deleting all arcs in $\vec{C}$. We have

$$\omega(\vec{P}')\omega(\vec{C}) = \omega(\vec{P}) < edist(\underline{v},\underline{w}) \leq \omega(\vec{P}'),$$

since $\vec{P}'$ has fewer arcs than $\vec{P}$. Thus the dicycle $\vec{C}$ has weight $\omega(\vec{C}) < 1$ and is absorbing which contradicts the presumption of (ii). Consequently, $dist(\underline{v},\underline{w}) = edist(\underline{v},\underline{w})$ is true. □

This lemma, in addition, shows that all distances in $\vec{D}$ belong to $\mathbb{Q}_0 \cup \{\infty\}$. As mentioned in Section 2, the triple $(\mathbb{Q}_0 \cup \{\infty\}, min, \cdot)$ is a commutative dioid. In the case of weight functions mapping the arc set to a dioid, the emergence of absorbing dicycles has been thoroughly discussed in combinatorial optimization, cf. [30]. Therein, the reader will also find a couple of results similar to Lemma 17.

7.2 Determination of redundant object types

We now present a characterization of all redundant object types for the constraint set Σ_C.

Theorem 18. *An object type $\underline{w}$ is redundant for Σ_C if and only if one of the following three conditions holds:*

(i) There is an object type $\underline{v}$ with $dist(\underline{v}, \underline{w}) = 0$.

(ii) There is a redundant object type $\underline{u}$ and a link from $\underline{w}$ to $\underline{u}$.

(iii) There is a redundant object type $\underline{v}$ with $dist(\underline{v}, \underline{w}) < \infty$.

Proof. (*Necessity.*) Suppose an object type $\underline{w}$ is redundant. Then every admissible function g satisfies $g(\underline{w}) = 0$. Our objective is to construct an admissible function g such that $g(\underline{w}) = 0$ holds only if one of the conditions (i) to (iii) applies.

Let Red_0 be the set of all vertices in $\vec{D}$ satisfying (i). From Red_0, we obtain Red by applying (ii) and (iii) as long as possible. Since $\vec{S}$ is finite, this procedure terminates after a finite number of steps. For all vertices $\underline{w}$ in Red, we put $g(\underline{w}) = 0$. For every other vertex $\underline{v}$, we define

$$g(\underline{v}) = \min\{dist(\underline{u}, \underline{v}) : \underline{u} \in V\}.$$

Clearly, $g(\underline{v}) > 0$ holds for all vertices $\underline{v}$ not in Red. It remains to show that g is admissible by checking (3) and (7).

Let $\ell = (\underline{r}, \underline{c})$ be any link with $g(\underline{c}) = 0$. Then $\underline{c}$ is in Red, and by (ii) we also have $\underline{r}$ in Red. This gives us $g(\underline{r}) = 0$ and thus (3).

Next we turn our attention to (7). If $\underline{w} \in Red$, we have $g(\underline{w}) = 0$ and the inequality trivially holds. If $\underline{v} \in Red$ and $dist(\underline{v}, \underline{w}) < \infty$, then $\underline{w}$ belongs to Red due to (iii) and (7) is true. Otherwise, if $\underline{v} \in Red$ but $dist(\underline{v}, \underline{w}) = \infty$, the claimed inequality is trivial again. Let us now assume that neither $\underline{v}$ nor $\underline{w}$ is in Red. For every vertex $\underline{u} \in V$, we have

$$\begin{aligned} dist(\underline{u}, \underline{w}) &\leq dist(\underline{u}, \underline{v}) dist(\underline{v}, \underline{w}) \\ \min\{dist(\underline{u}, \underline{w}) : \underline{u} \in V\} &\leq \min\{dist(\underline{u}, \underline{v}) : \underline{u} \in V\} dist(\underline{v}, \underline{w}) \\ g(\underline{w}) &\leq g(\underline{v}) dist(\underline{v}, \underline{w}). \end{aligned}$$

Hence condition (7) holds. This shows g to be admissible. Since $g(\underline{w}) = 0$ holds only for the vertices in Red, every redundant object type belongs to Red as claimed.

(*Sufficiency.*) We have to prove that each of the three conditions forces $\underline{w}$ to be redundant. For that, we show $g(\underline{w}) = 0$ for every admissible function g. If (i) holds, then Lemma 16 provides $g(\underline{w}) = 0$ due to (7). If (ii) holds, we apply (3) to derive $g(\underline{w}) = 0$. Finally, let (iii) be true. Since $g(\underline{v}) = 0$, Lemma 16 again implies $g(\underline{w}) = 0$ due to (7). This proves $\underline{w}$ to be redundant in all three cases. □

It is noteworthy, that the admissible function g constructed to verify Theorem 18 is an example for the claim of Lemma 14.

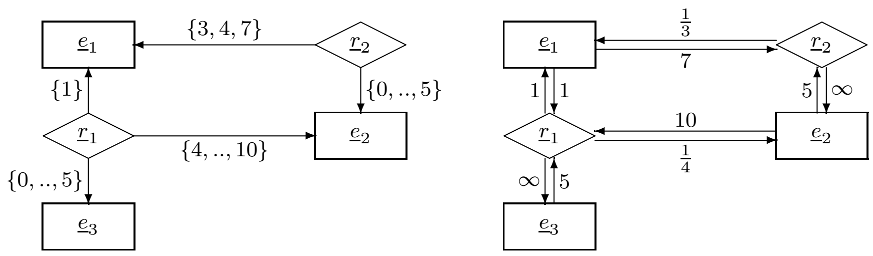

Figure 4: A database schema $\vec{S}$ containing redundant object types for the specified cardinality constraints.

Example. Consider the set Σ_C of cardinality constraints specified on the database schema in Figure 4. In the symmetric digraph $\vec{D}$ we have $edist(\underline{r}_2, \underline{e}_1) = 1/3$, but $dist(\underline{r}_2, \underline{e}_1) = 0$. This is caused by the absorbing dicycle $\vec{C}$ from $\underline{r}_2$ via $\underline{e}_1$, $\underline{r}_1$ and $\underline{e}_2$ back to $\underline{r}_2$. Its weight is $\omega(\vec{C}) = 5/12$. Consequently, $\underline{e}_1$ is redundant for Σ_C. Similarly all the object types in $\vec{C}$ turn out to be redundant. On the other hand, $\underline{e}_3$ is not redundant.

To apply Theorem 18 in practice, we need an efficient algorithm to calculate the distances in the digraph $\vec{D}$. This question has been widely studied in combinatorial optimization, and is also known as the all-pairs-shortest-distance problem. For an overview, we refer to [30, 37]. The difficulty in computing the distance between two vertices is that the number of diwalks connecting them might be infinite. In many cases, however, it suffices to consider dipaths. For our purposes here, Lemma 17 proposes an appropriate way of computing distances which is based on the inspection of absorbing dicycles.

In the literature, a plenty of algorithms has been suggested to compute the distances in a digraph. A major source book on this topic is again [30]. Recent results including new subcubic algorithms for this problem are to be found in [46]. We present here a variation of an algorithm by Noltemeier [44], which was originally suggested for the dioid $(\mathbb{R} \cup \{\infty\}, min, +)$, but may easily be formulated for the dioid $(\mathbb{Q}_0 \cup \{\infty\}, min, \cdot)$.

Algorithm 19 (cf. [44]). *Suppose we are given the digraph $\vec{D} = (V, A)$ and the weight function $\omega : A \to \mathbb{Q}_0 \cup \{\infty\}$. The algorithm determines the distance $dist(\underline{v}, \underline{w})$ for every pair of vertices in $\vec{D}$:*

```
1. for $\underline{v} \in V$ do for $\underline{w} \in V$ do $\delta(\underline{v},\underline{w}) \leftarrow \infty$; od; od;
2. for $\underline{u} \in V$ do $\delta(\underline{u},\underline{u}) \leftarrow 1$; od;
3. for $a = (\underline{v},\underline{w}) \in A$ do $\delta(\underline{v},\underline{w}) \leftarrow \min\{\delta(\underline{v},\underline{w}), \omega(a)\}$; od;
4. for $\underline{u} \in V$ do for $\underline{v} \in V$ do for $\underline{w} \in V$ do
      $\delta(\underline{v},\underline{w}) \leftarrow \min\{\delta(\underline{v},\underline{w}), \delta(\underline{v},\underline{u})\delta(\underline{u},\underline{w})\}$;
   od; od; od;
5. for $\underline{u} \in V$ with $\delta(\underline{u},\underline{u}) < 1$ do
      for $\underline{v} \in V$ do
         if $\delta(\underline{u},\underline{v}) < \infty$ then $\delta(\underline{u},\underline{v}) \leftarrow 0$; fi;
         if $\delta(\underline{v},\underline{u}) < \infty$ then $\delta(\underline{v},\underline{u}) \leftarrow 0$; fi;
         for $\underline{w} \in V$ do
            if $\delta(\underline{v},\underline{u})\delta(\underline{u},\underline{w}) < \infty$ then $\delta(\underline{v},\underline{w}) \leftarrow 0$; fi;
         od;
      od;
   od;
6. for $\underline{v} \in V$ do for $\underline{w} \in V$ do $dist(\underline{v},\underline{w}) \leftarrow \delta(\underline{v},\underline{w})$; od; od.
```

Note, that steps 1–4 correspond to the well-known algorithm of Floyd-Warshall, cf. [30, 37]. After step 4, we already have the distance for all vertex pairs satisfying $dist(\underline{v},\underline{w}) = edist(\underline{v},\underline{w})$. Furthermore, all vertices on some absorbing dicycle satisfy $\delta(\underline{u},\underline{u}) < 1$. This motivates step 5. For a detailed discussion of the algorithm, we refer to [44]. Its complexity is $O(|V|^3 + |A|)$.

Once we succeeded in computing the distances, we may easily proceed in determining all redundant object types.

Algorithm 20. *Consider the constraint set Σ_C declared on $\vec{S} = (V, L)$. For every vertex pair $(\underline{v},\underline{w})$ let $dist(\underline{v},\underline{w})$ be its distance in the symmetric digraph $\vec{D} = (V, A)$. For every vertex $\underline{w}$, let $A^-(\underline{w})$ be the set of all arcs with initial vertex $\underline{w}$ in $\vec{D}$, and $A^+(\underline{w})$ be the set of all arcs with terminal vertex $\underline{w}$ in $\vec{D}$.*

The algorithm determines the set Red of redundant object types for Σ_C:

```
1. $Red_0 \leftarrow \emptyset$;
   for $\underline{v} \in V$ do for $\underline{w} \in V$ do
      if $dist(\underline{v},\underline{w}) = 0$ then add $\underline{w}$ to $Red_0$; fi;
   od;
2. $Red \leftarrow Red_0$;
   $A_{NR} \leftarrow \emptyset$; $A_{RN} \leftarrow \emptyset$; $A_{NN} \leftarrow \emptyset$;
   for $a = (\underline{v},\underline{w}) \in A$ do
      if $\underline{v} \in Red$ and $\underline{w} \notin Red$ then add $a$ to $A_{NR}$; fi;
      if $\underline{v} \notin Red$ and $\underline{w} \in Red$ then add $a$ to $A_{RN}$; fi;
      if $\underline{v} \notin Red$ and $\underline{w} \notin Red$ then add $a$ to $A_{NN}$; fi;
   od;
```

```
3. while $A_{NR} \cup A_{RN} \neq \emptyset$ do
       choose $a \in A_{NR} \cup A_{RN}$;
       if $a \in A_{NR}$ then delete $a$ from $A_{NR}$ else delete $a$ from $A_{RN}$; fi;
       if ($a = (\underline{w}, \underline{u}) \in A_{NR}$ and $a \in L$) or ($a = (\underline{v}, \underline{w}) \in A_{RN}$ and $dist(\underline{v}, \underline{w}) < \infty$)
   then
           add $\underline{w}$ to Red;
           $A_{NR} \leftarrow A_{NR} \backslash (A^-(\underline{w}) \cap A_{NR}) \cup (A^+(\underline{w}) \cap A_{NN})$;
           $A_{RN} \leftarrow A_{RN} \cup (A^-(\underline{w}) \cap A_{NN}) \backslash (A^+(\underline{w}) \cap A_{RN})$;
           $A_{NN} \leftarrow A_{NN} \backslash (A^-(\underline{w}) \cup A^+(\underline{w}))$;
       fi;
   od.
```

Proof. The algorithm is based on Theorem 18. In step 1, we determine the set Red_0 of all object types which are redundant due to (i) in Theorem 18. In step 2, we introduce three subsets A_{NR}, A_{RN} and A_{NN} of the arc set. A_{NR} consists of all arcs which enter the set *Red* of all redundant object types recognized so far, A_{RN} consists of all arcs leaving this set and A_{NN} consists of all arcs whose vertices both are not in *Red*. In step 3, we use these sets to find further redundant object types due to (ii) and (iii) in Theorem 18.

Note that the condition (iii) in Theorem 18 is equivalent to:

(iii') There is a redundant object type $\underline{v}$ with an arc from $\underline{v}$ to $\underline{w}$ and $dist(\underline{v}, \underline{w}) < \infty$.

In the algorithm, we use (iii') instead of (iii) to keep the complexity small. Evidently, (iii) yields (iii'). But also the converse is true. Suppose (iii') holds. Since $dist(\underline{v}, \underline{w}) < \infty$, there is a diwalk $\vec{P}$ from $\underline{v}$ to $\underline{w}$ with finite weight. Consequently, any two vertices on this diwalk have finite distance. On applying (iii') successively to the arcs on $\vec{P}$, we conclude that all vertices on $\vec{P}$ are redundant. This holds in particular for $\underline{w}$ and proves (iii).

In step 3, we now check the conditions (ii) and (iii') by studying all arcs entering or leaving the set *Red* of those redundant object types which have already been recognized. Whenever we find a new redundant object type $\underline{w}$, we add it to the set *Red* and update the sets A_{NR}, A_{RN} and A_{NN} accordingly. Thus we find all redundant object types by virtue of Theorem 18.

In every run of the while-loop in step 3, the size of $A_{NR} \cup A_{RN} \cup A_{NN}$ decreases by at least one. Hence, the algorithm terminates after at most $|A|$ runs. □

Given all distances in $\vec{D}$, this algorithm determines the set *Red* with complexity $O(|V|^2 + |A|) = O(|V|^2 + |L|)$. Thus we can determine the set of all redundant object types with complexity $O(|V|^3 + |L|)$ on the basis of Theorem 18.

7.3 Consistency of cardinality constraints

Theorem 18 also provides a nice graph-theoretic characterization of those sets of cardinality constraints that do not cause redundant types, i.e. of consistent sets of cardinality constraints. For special kinds of cardinality constraints, similar results were given by Lenzerini and Nobili [38] and by Thalheim [49]. The importance of absorbing dicycles was first pointed out in [38]. The consistency problem for arbitrary sets of cardinality constraints was tackled in [35].

Corollary 21. Σ_C *is consistent if and only if* $\vec{D}$ *contains neither absorbing dicycles nor arcs of weight 0.*

Proof. (*Necessity.*) Let Σ_C be consistent. Assume there are absorbing dicycles or arcs of weight 0. In both cases, we trivially have vertices $\underline{v}, \underline{w}$ with distance $dist(\underline{v}, \underline{w}) = 0$. This yields the existence of redundant object types by Theorem 18, which is a contradiction to Σ_C being consistent.

(*Sufficiency.*) Suppose there are neither absorbing dicycles nor arcs of weight 0. By Theorem 18 the existence of redundant object types merely depends on the existence of a vertex pair $\underline{v}, \underline{w}$ with $dist(\underline{v}, \underline{w}) = 0$. Assume there is such a vertex pair $\underline{v}, \underline{w}$ in $\vec{D}$. Since all diwalks from $\underline{v}$ to $\underline{w}$ are without absorbing dicycles, we have $edist(\underline{v}, \underline{w}) = dist(\underline{v}, \underline{w}) = 0$ by Lemma 17. Consequently, there is a dipath $\vec{P}$ from $\underline{v}$ to $\underline{w}$ with weight $\omega(\vec{P}) = 0$. But this forces at least one of its arcs to be of weight 0, which contradicts the presumption. Hence there are no redundant object types. □

8 More on admissible functions

In Section 5 we introduced the set $\text{CanDeg}(\underline{r}, \underline{c})$ of candidate degrees for every link $(\underline{r}, \underline{c})$. By virtue of Lemma 7, Σ_C always implies $card(\underline{r}, \underline{c}) = \text{CanDeg}(\underline{r}, \underline{c})$. However, this does not mean that every candidate degree really occurs as a degree in some legal database instance. The question arises, which candidate degrees may be excluded due to the semantic power of Σ_C.

As before, let Σ_C be declared on a database schema $\vec{S}$. Fix a link $(\underline{r}, \underline{c})$ and a candidate degree $d^* \in \text{CanDeg}(\underline{r}, \underline{c})$. Consider a population $\underline{r}^t$ which satisfies Σ_C. This population *supplies evidence of* $(\underline{r}, \underline{c}, d^*)$ if its codomain $\underline{c}^t$ contains an object c with degree $\deg(\underline{r}^t, c) = d^*$. Similarly, a legal database instance $\vec{S}^t$ *supplies evidence of* $(\underline{r}, \underline{c}, d^*)$ if its population $\underline{c}^t$ contains an object c whose degree $\deg(\underline{r}^t, c)$ equals d^*.

Σ_C implies $card(\underline{r}, \underline{c}) = \text{CanDeg}(\underline{r}, \underline{c}) \backslash \{d^*\}$ if and only if there is no legal database instance $\vec{S}^t$ supplying evidence of $(\underline{r}, \underline{c}, d^*)$. The investigation of evidence supplying databases is motivated by the following result on closed sets in the class $\mathcal{C}$ of cardinality constraints.

Theorem 22. *Σ_C is $\mathcal{C}$-closed if and only if it contains all constraints implied by Σ_C via (C1) to (C3) and, for every $(\underline{r}, \underline{c}, d^*)$, there exists a database instance which is legal for Σ_C and supplies evidence of $(\underline{r}, \underline{c}, d^*)$.*

Proof. Note that for every cardinality constraint $card(\underline{r}, \underline{c}) = M$ contained in Σ_C the set M is a superset of $\mathrm{CanDeg}(\underline{r}, \underline{c})$. This simple observation is crucial for the following proof.

(*Necessity.*) Let Σ_C be $\mathcal{C}$-closed. Lemma 5 verifies the correctness of (C1) to (C3). Assume now, there is some $(\underline{r}, \underline{c}, d^*)$ without a legal database instance supplying evidence of it. Then Σ_C implies the cardinality constraint $card(\underline{r}, \underline{c}) = \mathrm{CanDeg}(\underline{r}, \underline{c}) \backslash \{d^*\}$. But $\mathrm{CanDeg}(\underline{r}, \underline{c}) \backslash \{d^*\}$ is a proper subset of $\mathrm{CanDeg}(\underline{r}, \underline{c})$. Therefore, this cardinality constraint is not in Σ_C, which contradicts Σ_C to be $\mathcal{C}$-closed. Consequently, for every $(\underline{r}, \underline{c}, d^*)$ there must be a legal database instance supplying evidence.

(*Sufficiency.*) We have to show that Σ_C is $\mathcal{C}$-closed under the given conditions. Let $card(\underline{r}, \underline{c}) = M$ be some cardinality constraint implied by Σ_C. For every candidate degree d^* in $\mathrm{CanDeg}(\underline{r}, \underline{c})$, we have a legal database instance supplying evidence. Therefore, M must be a superset of $\mathrm{CanDeg}(\underline{r}, \underline{c})$. By (C2) this forces the cardinality constraint under discussion to be in Σ_C. Hence Σ_C is $\mathcal{C}$-closed. □

In this section, we again use admissible functions to study the existence of evidence supplying database instances. Actually, the results here are obtained by strengthening the results from Section 6. Throughout, consider a link $\ell = (\underline{r}, \underline{c})$ in the database schema $\vec{S}$, and let d^* be a fixed candidate degree in $\mathrm{CanDeg}(\underline{r}, \underline{c})$.

A population $\underline{r}^t$ supplies evidence of $(\underline{r}, \underline{c}, d^*)$ with $d^* > 0$ when the representation graph $G^t(\underline{r}, \underline{c})$ has a clique of size d^*, whereas it supplies evidence of $(\underline{r}, \underline{c}, 0)$ when the size of the codomain $\underline{c}^t$ exceeds the number of cliques in $G^t(\underline{r}, \underline{c})$. Together with Lemma 9 and Theorem 10, this observation yields the following two results.

Lemma 23. *Suppose we are given an object set $\underline{c}_i^t$ of size $g(\underline{c}_i)$ for every component $\underline{c}_i \in Co(\underline{r})$. Then there is a population $\underline{r}^t$ with codomains $\underline{c}_i^t$, such that $\underline{r}^t$ is of size $g(\underline{r})$, satisfies Σ_C and supplies evidence of $(\underline{r}, \underline{c}, d^*)$, if and only the condition of Lemma 9 holds under the additional presumption $x_{d^*} > 0$ for the fixed candidate degree $d^* \in CanDeg(\underline{r}, \underline{c})$.*

Proof. (*Necessity.*) The observation is an immediate consequence of Lemma 9. Let $\underline{r}^t$ be the claimed population supplying evidence of $(\underline{r}, \underline{c}, d^*)$. As in the proof of Lemma 9, let x_d denote the number of objects c in the codomain $\underline{c}^t$ with degree $\deg(\underline{r}^t, c) = d$. Since $\underline{r}^t$ supplies evidence of $(\underline{r}, \underline{c}, d^*)$, we have at least one object c in $\underline{c}^t$ with $\deg(\underline{r}^t, c) = d^*$. Clearly, this gives us $x_{d^*} \geq 1$.

(*Sufficiency.*) Suppose $x_{d^*} > 0$ holds. In the proof of Lemma 9, we constructed a population $\underline{r}^t$ which satisfies Σ_C. Its codomain $\underline{c}^t$ contains exactly x_{d^*} objects c with

$\deg(\underline{r}^t, c) = d^*$. Since x_{d^*} is positive, the population $\underline{r}^t$ in fact supplies evidence of $(\underline{r}, \underline{c}, d^*)$. □

Theorem 24. *Let $g : V \to \mathbb{N}_0$ be a given function. There is a database instance $\vec{S}^t$ with population sizes $g(\underline{v})$, such that $\vec{S}^t$ is legal for Σ_C and supplies evidence of $(\underline{r}, \underline{c}, d^*)$, if and only if the condition of Theorem 10 holds under the additional presumption $x_{d^*} > 0$ for the fixed candidate degree $d^* \in CanDeg(\underline{r}, \underline{c})$.*

Proof. We proceed as in proof of Theorem 10, but use Lemma 23 instead of Lemma 9 for the relationship type $\underline{r}$ under discussion. □

As pointed out in Section 6, the population sizes in a legal database instance yield an admissible function g. The following result describes which admissible functions correspond to database instances supplying evidence of $(\underline{r}, \underline{c}, d^*)$.

Theorem 25. *Let $g : V \to \mathbb{Q}_0$ be a given function. There is a positive integer λ and a database instance $\vec{S}^t$ with population sizes $\lambda g(\underline{v})$, such that $\vec{S}^t$ is legal for Σ_C and supplies evidence of $(\underline{r}, \underline{c}, d^*)$, if and only if g is admissible for Σ_C with $g(\underline{c}) > 0$ and satisfies at least one of the relations*

$$\alpha(\underline{r}, \underline{c}) g(\underline{c}) < g(\underline{r}) < \beta(\underline{r}, \underline{c}) g(\underline{c}) \tag{10}$$

or

$$g(\underline{r}) = d^* g(\underline{c}). \tag{11}$$

for the fixed link $\ell = (\underline{r}, \underline{c})$.

Proof. (*Necessity.*) Consider a legal database instance $\vec{S}^t$ with population sizes $\lambda g(\underline{v}^t)$ for some positive integer λ. By Theorem 11, the existence of $\vec{S}^t$ shows g to be admissible, and we have (3) and (4). Suppose $\vec{S}^t$ supplies evidence of $(\underline{r}, \underline{c}, d^*)$. Then $\vec{S}^t$ contains at least one object of type $\underline{c}$. Therefore, the value of $g(\underline{c})$ is positive.

Now we discuss the additional statements (10) and (11). By (4), we have

$$\alpha(\underline{r}, \underline{c}) g(\underline{c}) \leq g(\underline{r}) \leq \beta(\underline{r}, \underline{c}) g(\underline{c}).$$

Here $g(\underline{r})$ either satisfies (10) or takes one of the bounds of that interval. Assume $g(\underline{r})$ equals the lower bound $\alpha(\underline{r}, \underline{c}) g(\underline{c})$. Applying Theorem 24, we observe

$$g(\underline{r}) = \sum_{d \in \mathrm{CanDeg}(\underline{r}, \underline{c})} d x_d \geq \alpha(\underline{r}, \underline{c}) g(\underline{c}) + (d^* - \alpha(\underline{r}, \underline{c})) x_{d^*} \geq \alpha(\underline{r}, \underline{c}) g(\underline{c}) = g(\underline{r}),$$

since $\alpha(\underline{r}, \underline{c})$ is the minimum candidate degree and $\sum_{d \in \mathrm{CanDeg}(\underline{r}, \underline{c})} x_d = g(\underline{c})$ holds. Herein, $x_{d^*} > 0$ gives us $d^* = \alpha(\underline{r}, \underline{c})$ which yields (11) as claimed. The case $g(\underline{r}) = \beta(\underline{r}, \underline{c}) g(\underline{c})$ is treated in the same way.

(*Sufficiency.*) Suppose g is admissible and we have $g(\underline{c}) > 0$ as well as one of (10) or (11). Our goal is to apply Theorem 24: Since g is admissible, the condition of Theorem 10 is satisfied for every link in $\vec{S}$.

We still have to verify $x_{d^*} > 0$ for the fixed link $\ell = (\underline{r}, \underline{c})$. Recall the proof of Theorem 11. Again we restrict ourselves to a finite subset $\mathrm{CanDeg}'(\underline{r}, \underline{c})$ of candidate degrees. We choose this subset $\mathrm{CanDeg}'(\underline{r}, \underline{c})$ as done in the proof of Theorem 11, but require $\beta'(\underline{r}, \underline{c})$ to be larger or equal to the fixed candidate degree d^*. Thus d^* lies in $\mathrm{CanDeg}'(\underline{r}, \underline{c})$.

We want to use the theorem of Motzkin: There exist non-negative rationals x_d with $x_{d^*} > 0$ and satisfying the equations

$$\sum_{d \in \mathrm{CanDeg}'(\underline{r}, \underline{c})} x_d = g(\underline{c}),$$

$$\sum_{d \in \mathrm{CanDeg}'(\underline{r}, \underline{c})} d x_d = g(\underline{r})$$

if and only if the inequalities

$$pg(\underline{c}) + qg(\underline{r}) \geq 0 \quad , \tag{12}$$

$$pg(\underline{c}) + qg(\underline{r}) > 0 \qquad \text{if } p + d^*q > 0 \tag{13}$$

hold for all rationals p, q with $p+dq \geq 0$ for every candidate degree $d \in \mathrm{CanDeg}'(\underline{r}, \underline{c})$.

The inequalities (4), $g(\underline{c}) > 0$ and $p + dq \geq 0$ for each $d \in \mathrm{CanDeg}'(\underline{r}, \underline{c})$ imply (12) as shown in Theorem 11.

It remains to discuss (13) when $p + d^*q > 0$. First, suppose that (11) holds. Since $g(\underline{c}) > 0$, we have $pg(\underline{c}) + qg(\underline{r}) = (p + d^*q)g(\underline{c}) > 0$ which yields (13). Otherwise, suppose that (10) holds. Due to our choice of $\beta'(\underline{r}, \underline{c})$ we have

$$\alpha(\underline{r}, \underline{c})g(\underline{c}) < g(\underline{r}) < \beta'(\underline{r}, \underline{c})g(\underline{c}).$$

We distinguish three cases. If $q > 0$, we obtain

$$pg(\underline{c}) + qg(\underline{r}) > (p + q\alpha(\underline{r}, \underline{c}))g(\underline{c}) \geq 0,$$

since $\alpha(\underline{r}, \underline{c})$ is in $\mathrm{CanDeg}'(\underline{r}, \underline{c})$. If $q < 0$, we obtain

$$pg(\underline{c}) + qg(\underline{r}) > (p + q\beta'(\underline{r}, \underline{c}))g(\underline{c}) \geq 0,$$

since $\beta'(\underline{r}, \underline{c})$ lies in $\mathrm{CanDeg}'(\underline{r}, \underline{c})$. Finally, if $q = 0$, we have

$$pg(\underline{c}) + qg(\underline{r}) = pg(\underline{c}) + 0 = (p + d^*q)g(\underline{c}) > 0,$$

since both $p + d^*q$ and $g(\underline{c})$ are positive. In all three cases we derive (13). Hence we may apply the theorem of Motzkin. For the remaining candidate degrees d not in $\mathrm{CanDeg}'(\underline{r}, \underline{c})$, we put $x_d = 0$. This implies the condition of Theorem 10 for the fixed link $\ell = (\underline{r}, \underline{c})$ with the additional property $x_{d^*} > 0$.

We may now apply Theorem 24 to the integral function λg, where λ is a suitable positive integer. This concludes the proof. □

The final result in this section continues our study of closed sets of cardinality constraints.

Theorem 26. *Σ_C is $\mathcal{C}$-closed if and only if it contains all constraints implied by Σ_C via (C1) to (C4) and there exists an admissible function g for Σ_C, such that $g(\underline{v}) > 0$ holds for every non-redundant object type $\underline{v}$ and g satisfies either (10) or*

$$\alpha(\underline{r},\underline{c})g(\underline{c}) = g(\underline{r}) = \beta(\underline{r},\underline{c})g(\underline{c}) \tag{14}$$

for every link $(\underline{r},\underline{c})$ with non-redundant $\underline{c}$.

Proof. (*Necessity.*) Let Σ_C be $\mathcal{C}$-closed. The correctness of (C1) to (C4) follows from Lemmas 5 and 15. It remains to construct a suitable function g. By Lemma 14 there is an admissible function g_0 with $g_0(\underline{v}) > 0$ for every non-redundant object type $\underline{v}$. We shall discuss whether g_0 has all the desired properties. For that, consider a link $(\underline{r},\underline{c})$ with non-redundant $\underline{c}$.

Of course, CanDeg$(\underline{r},\underline{c})$ is not empty. Suppose CanDeg$(\underline{r},\underline{c})$ contains only one candidate degree $d^* = \alpha(\underline{r},\underline{c}) = \beta(\underline{r},\underline{c})$. By Lemma 12, we have $d^*g_0(\underline{c}) = \alpha(\underline{r},\underline{c})g_0(\underline{c}) \leq g_0(\underline{r}) \leq \beta(\underline{r},\underline{c})g_0(\underline{c}) = d^*g_0(\underline{c})$, which implies (14).

Otherwise, suppose CanDeg$(\underline{r},\underline{c})$ is of size 2 or larger. Lemma 22 provides legal database instances $\vec{S}^t$ supplying evidence of every $(\underline{r},\underline{c},d^*)$ with $d^* \in$ CanDeg$(\underline{r},\underline{c})$. Theorem 25 gives us an admissible function g satisfying $g(\underline{c}) > 0$ and at least one of the relations (10) or (11). Consider two candidate degrees $d_1 < d_2$ in CanDeg$(\underline{r},\underline{c})$ with corresponding admissible functions g_1 and g_2. Put $g_\ell = g_1 + g_2$, which is again admissible. Clearly, g_ℓ satisfies (10).

Unfortunately, g_0 itself does not necessarily satisfy one of (10) or (14) for a given link $\ell = (\underline{r},\underline{c})$ with two or more candidate degrees in CanDeg$(\underline{r},\underline{c})$. For this reason, we add g_ℓ to g_0 for each of these links. The resultant function g is still admissible and has all the desired properties.

(*Sufficiency.*) Our objective is to apply Theorem 22. For every link $(\underline{r},\underline{c})$ and every candidate degree $d^* \in$ CanDeg$(\underline{r},\underline{c})$ we need a legal database instance supplying evidence of $(\underline{r},\underline{c},d^*)$. Theorem 25 gives us such database instance whenever there exists a certain admissible function. As we shall see below, the given function g is suitable for this.

Consider a link $(\underline{r},\underline{c})$. Suppose $\underline{c}$ is redundant. Σ_C contains $card(\underline{r},\underline{c}) = \emptyset$ by (C4). Thus CanDeg$(\underline{r},\underline{c})$ is empty, and no candidate degree has to be discussed. Conversely, let $\underline{c}$ be not redundant. Then we have $g(\underline{c}) > 0$. We are ready if g satisfies (10). Otherwise, if g satisfies (14), this yields $\alpha(\underline{r},\underline{c}) = \beta(\underline{r},\underline{c})$. Hence $d^* = \alpha(\underline{r},\underline{c}) = \beta(\underline{r},\underline{c})$ is the only candidate degree for this link, and we immediately derive (11).

Consequently, the given function g allows us to apply Theorem 25. This ensures the existence of legal database instances supplying evidence of every $(\underline{r},\underline{c},d^*)$, and proves Σ_C to be $\mathcal{C}$-closed by Theorem 22. □

9 Subcritical dicycles

In this section, we continue our study of dicycles in the symmetric digraph $\vec{D}$ with weight function ω induced by a given set Σ_C of cardinality constraints. We call a dicycle $\vec{C}$ *subcritical for* Σ_C if its weight satisfies $\omega(\vec{C}) = 1$. Similar to absorbing dicycles, also subcritical dicycles play a crucial role for implications of cardinality constraints.

Lemma 27. *If g is an admissible function for Σ_C, and $a = (\underline{v}, \underline{w})$ is an arc lying on a subcritical dicycle for Σ_C, then $g(\underline{w}) = g(\underline{v})\omega(a)$ holds.*

Proof. Let $\vec{C}$ be a subcritical dicycle containing the arc a. By $\vec{P}$ we denote the dipath from $\underline{w}$ to $\underline{v}$, obtained from $\vec{C}$ after deleting the arc a. Since g is admissible, Lemma 14 implies $g(\underline{w}) \le g(\underline{v})dist(\underline{v}, \underline{w}) \le g(\underline{v})\omega(a)$ as well as

$$\begin{aligned} g(\underline{v}) &\le g(\underline{w})dist(\underline{w}, \underline{v}) \le g(\underline{w})\omega(\vec{P}), \\ g(\underline{v})\omega(a) &\le g(\underline{w})\omega(\vec{P})\omega(a) = g(\underline{w})\omega(\vec{C}) = g(\underline{w}). \end{aligned}$$

Together, these inequalities prove the claimed identity. □

The next observation is a simple consequence of Lemma 27 and the definition of the weight function ω. It should be mentioned, that every arc a on a subcritical dicycle satisfies $0 < \omega(a) < \infty$. For this reason, it suffices to consider $0 < \alpha(\underline{r}, \underline{c}) < \infty$ to verify the first claim.

Corollary 28. *Let g be admissible for Σ_C, and consider a link $\ell = (\underline{r}, \underline{c})$ in $\vec{S}$.*

(i) If $\ell = (\underline{r}, \underline{c})$ lies on a subcritical dicycle for Σ_C, we have $g(\underline{r}) = \alpha(\underline{r}, \underline{c})g(\underline{c})$.

(ii) If $\ell^{-1} = (\underline{c}, \underline{r})$ lies on a subcritical dicycle for Σ_C, we have $g(\underline{r}) = \beta(\underline{r}, \underline{c})g(\underline{c})$.

The preceding result almost immediately gives rise to the following implication rules:

(C6) If the link $(\underline{r}, \underline{c})$ lies on a subcritical dicycle for Σ_C, then $\Sigma_C \models \big(card(\underline{r}, \underline{c}) = \{\alpha(\underline{r}, \underline{c})\}\big)$.

(C7) If the reverse arc of the link $(\underline{r}, \underline{c})$ lies on a subcritical dicycle for Σ_C, then $\Sigma_C \models \big(card(\underline{r}, \underline{c}) = \{\beta(\underline{r}, \underline{c})\}\big)$.

Lemma 29. *(C6) and (C7) are correct.*

Proof. Suppose $\ell = (\underline{r}, \underline{c})$ lies on a subcritical dicycle. Consider any candidate degree $d^* \in \text{CanDeg}(\underline{r}, \underline{c})$ admitting a legal database instance $\vec{S}^t$ which supplies evidence for $(\underline{r}, \underline{c}, d^*)$. For every object type $\underline{v}$, let $g(\underline{v})$ denote the size of the population $\underline{v}^t$ in $\vec{S}^t$. The function g is admissible and satisfies $g(\underline{c}) > 0$ for the component $\underline{c}$. By Theorem 25, we have $\alpha(\underline{r}, \underline{c})g(\underline{c}) < g(\underline{r}) < \beta(\underline{r}, \underline{c})g(\underline{c})$ or $d^*g(\underline{c}) = g(\underline{r})$. The first inequality, however, does not apply here due to Corollary 28. For the same reason, we obtain $d^*g(\underline{c}) = g(\underline{r}) = \alpha(\underline{r}, \underline{c})g(\underline{c})$. This implies $d^* = \alpha(\underline{r}, \underline{c})$ and proves (C6) to be true. A similar argument verifies (C7). □

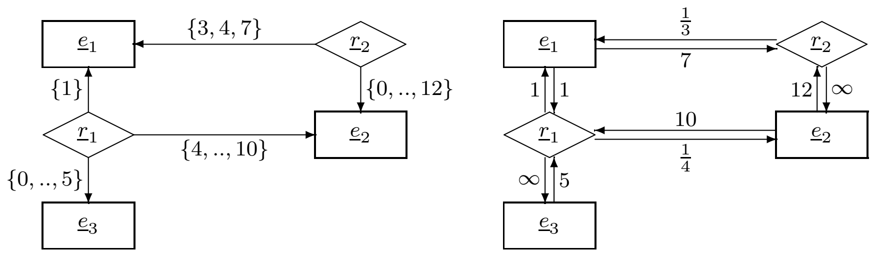

Figure 5: A set of cardinality constraints which gives rise to a subcritical dicycle in $\vec{D}$.

Example. Consider the set Σ_C of cardinality constraints indicated by Figure 5. Take the dicycle $\vec{C}$ from $\underline{r}_2$ via $\underline{e}_1$, $\underline{r}_1$ and $\underline{e}_2$ back to $\underline{r}_2$ in the symmetric digraph $\vec{D}$. Its weight is $\omega(\vec{C}) = 12/12 = 1$. Hence $\vec{C}$ is subcritical. On applying rule (C6) to the link $(\underline{r}_2, \underline{e}_1)$, we derive the cardinality constraint $card(\underline{r}_2, \underline{e}_1) = \{3\}$. That is, the candidate degrees 4 and 7 may be excluded.

Analogously, Σ_C implies $card(\underline{r}_1, \underline{e}_2) = \{4\}$ due to (C6) and $card(\underline{r}_2, \underline{e}_2) = \{12\}$ due to (C7). Of course, we may also apply (C7) to the link $(\underline{r}_1, \underline{e}_1)$, but this will not give us any new information: $\beta(\underline{r}_1, \underline{e}_1) = 1$ has already been the only candidate degree for this link.

Remark 30. *The example above points out that an application of (C6) or (C7) to some link* $(\underline{r}, \underline{c})$ *makes sense only if CanDeg*$(\underline{r}, \underline{c})$ *is of size two or larger, i.e. if we have at least two candidate degrees for this link. In particular, this observation applies in the following case: Suppose we have a link* $(\underline{r}, \underline{c})$ *where* $\underline{c}$ *is redundant. Then we may apply (C4) which excludes all candidate degrees. An application of (C6) or (C7) to this link turns to be useless: There is no candidate degree to be excluded any more.*

It is noteworthy, that for an arc which lies on a subcritical dicycle either both vertices are redundant or both are non-redundant. This is confirmed by the following lemma which assembles observations on the interplay of subcritical dicycles and redundant object types.

Lemma 31.

(i) Let $\vec{C}$ *be a subcritical dicycle for* Σ_C*. Then either all vertices on* $\vec{C}$ *are redundant for* Σ_C *or none of them.*

(ii) Let $\vec{C}$ *be a closed diwalk with weight* $\omega(\vec{C}) = 1$*. Then* $\vec{C}$ *either splits into dicycles which are all subcritical for* Σ_C *or all vertices on* $\vec{C}$ *are redundant for* Σ_C*.*

Proof. (i) Any two vertices $\underline{v}$ and $\underline{w}$ on the dicycle $\vec{C}$ have finite distance $dist(\underline{v}, \underline{w}) < \infty$ since $\omega(\vec{C}) = 1$ holds. Whenever one of the vertices is redundant than all vertices on $\vec{C}$ are redundant by the third condition in Theorem 18.

(ii) Every closed diwalk $\vec{C}$ may be decomposed into a set of dicycles, say $\vec{C}_1, \ldots, \vec{C}_s$. We have $1 = \omega(\vec{C}) = \omega(\vec{C}_1) \cdots \omega(\vec{C}_s)$. Hence these dicycles either have all the weight 1, or at least one of them is absorbing. In the first case, we are ready. In the second case, each vertex on this absorbing dicycle is redundant. Moreover, the distance of any two vertices on the diwalk $\vec{C}$ is finite since $\omega(\vec{C}) = 1$ holds. Theorem 18 again proves all vertices on $\vec{C}$ to be redundant. □

9.1 Determination of arcs on subcritical dicycles

We are interested in an efficient method to detect all arcs on subcritical dicycles. Fortunately, we can restrict ourselves to those arcs that connect two non-redundant vertices as detailed above.

Theorem 32. *Let $a = (\underline{v}, \underline{w})$ be an arc such that neither $\underline{v}$ nor $\underline{w}$ is redundant for Σ_C. Then a lies on a subcritical dicycle for Σ_C if and only if $\omega(a)dist(\underline{w}, \underline{v}) = 1$ holds.*

Proof. (*Necessity.*) Suppose a lies on a subcritical dicycle $\vec{C}$. Let $\vec{P}$ denote the dipath obtained from $\vec{C}$ by deleting the arc a. Since $\underline{v}$ is not redundant, we have $dist(\underline{v}, \underline{v}) = edist(\underline{v}, \underline{v}) = 1$ by the first condition in Theorem 18. This yields

$$1 = dist(\underline{v}, \underline{v}) \leq dist(\underline{v}, \underline{w})dist(\underline{w}, \underline{v}) \leq \omega(a)dist(\underline{w}, \underline{v}) \leq \omega(a)\omega(\vec{P}) = \omega(\vec{C}) = 1,$$

and thus $\omega(a)dist(\underline{w}, \underline{v}) = 1$ as desired.

(*Sufficiency.*) Suppose $\omega(a)dist(\underline{w}, \underline{v}) = 1$ holds. Clearly, we have $0 < dist(\underline{w}, \underline{v}) < \infty$. By Lemma 16, we also have $dist(\underline{w}, \underline{v}) = edist(\underline{w}, \underline{v})$. Hence there is a dipath $\vec{P}$ from $\underline{w}$ to $\underline{v}$ with $\omega(\vec{P}) = edist(\underline{w}, \underline{v})$. When adding the arc a to the dipath $\vec{P}$, we obtain a dicycle $\vec{C}$ of weight

$$\omega(\vec{C}) = \omega(a)\omega(\vec{P}) = \omega(a)edist(\underline{w}, \underline{v}) = \omega(a)dist(\underline{w}, \underline{v}) = 1,$$

that is, a subcritical dicycle containing the arc a. □

By Theorem 32, the set of all arcs lying on subcritical dicycles, but without redundant vertices may be detected with complexity $O(|A|) = O(|L|)$, supposed we are given the distances for all vertex pairs in $\vec{D}$. This enables us to apply the rules (C6) and (C7) efficiently.

9.2 Subcritical dicycles and admissible functions

Before we turn our attention to the characterization of closed sets, we still give a result on the relation of admissible functions introduced in Section 6 and subcritical dicycles.

Theorem 33. *Let $a = (\underline{v}, \underline{w})$ be an arc such that neither $\underline{v}$ nor $\underline{w}$ is redundant for Σ_C. Then a lies on a subcritical dicycle if and only if every admissible function g for Σ_C satisfies $g(\underline{w}) = \omega(a)g(\underline{v})$.*

Proof. For the necessity, see Lemma 27.

(*Sufficiency.*) Suppose $g(\underline{w}) = \omega(a)g(\underline{v})$ holds whenever g is admissible. Assume now, a lies not on a subcritical dicycle. We are going to construct an admissible function g which meets $g(\underline{w}) < \omega(a)g(\underline{v})$ to cause a contradiction.

To begin with, we introduce a slightly modified weight function on the digraph $\vec{D}$. Let π_α be the product of the values $\alpha(\underline{r}, \underline{c})$, where $(\underline{r}, \underline{c})$ runs over the set of all links with $0 < \alpha(\underline{r}, \underline{c}) < \infty$ in the schema. If this set is empty, we put $\pi_\alpha = 1$. For every arc a', its original weight $\omega(a')$ is ∞ or an integral multiple of $1/\pi_\alpha$. For every dicycle $\vec{C}$, its original weight $\omega(\vec{C})$ is ∞ or an integral multiple of $1/\pi_\alpha^{|V|}$.

As announced, we define a new weight function $\omega_a : A \to \mathbb{Q}_0 \cup \{\infty\}$ by

$$\omega_a(a') = \begin{cases} \frac{\pi_\alpha^{|V|}}{1+\pi_\alpha^{|V|}}\,\omega(a) & \text{if } a' = a, \\ \omega(a') & \text{otherwise,} \end{cases}$$

for every arc $a' \in A$. Since $\underline{w}$ is not redundant, we have $\omega(a) > 0$ and thus also $\omega_a(a) > 0$. Again, this weight function may be extended to diwalks in $\vec{D}$. We have

$$\omega_a(\vec{P}) = \begin{cases} \omega(\vec{P})\,\frac{\omega_a(a)}{\omega(a)} & \text{if } \vec{P} \text{ contains } a, \\ \omega(\vec{P}) & \text{otherwise.} \end{cases}$$

It is interesting to observe, that a dicycle $\vec{C}$ is absorbing with respect to the new weight function ω_a exactly when it is absorbing with respect to the original weight function ω. This is trivial when $\vec{C}$ does not contain the arc a. So suppose a lies on $\vec{C}$. If $\vec{C}$ is absorbing with respect to ω, we immediately have $\omega_a(\vec{C}) < \omega(\vec{C}) < 1$. Thus $\vec{C}$ is absorbing with respect to ω_a, too. Conversely, if $\vec{C}$ is absorbing with respect to ω_a, we have

$$\omega(\vec{C}) = \omega_a(\vec{C})\frac{\omega(a)}{\omega_a(a)} < \frac{\omega(a)}{\omega_a(a)} = 1 + \frac{1}{\pi_\alpha^{|V|}}.$$

As mentioned, $\omega(\vec{C})$ is an integral multiple of $1/\pi_\alpha^{|V|}$. This gives us $\omega(\vec{C}) \le 1$, and yields $\omega(\vec{C}) < 1$ since a does not lie on any subcritical dicycle. Hence $\vec{C}$ is absorbing with respect to the original weight function ω.

As in Section 7, we may define the distance $dist_a(\underline{v}', \underline{w}')$ and the elementary distance $edist_a(\underline{v}', \underline{w}')$ for any vertex pair in $\vec{D}$. We always have $dist_a(\underline{v}', \underline{w}') \le dist(\underline{v}', \underline{w}')$. This, in particular, implies $dist_a(\underline{v}', \underline{w}') = 0$ whenever $dist(\underline{v}', \underline{w}') = 0$ holds.

But the converse is also true: Suppose $dist_a(\underline{v}', \underline{w}') = 0$. First, let there be a diwalk from $\underline{v}'$ to $\underline{w}'$ containing an absorbing dicycle with respect to ω. Lemma 17 immediately provides $dist(\underline{v}', \underline{w}') = 0$. Otherwise, let all diwalks from $\underline{v}'$ to $\underline{w}'$ be without absorbing dicycles with respect to ω. As shown above, the same holds for ω_a. Similar to Lemma 17, we may conclude $edist_a(\underline{v}', \underline{w}') = dist_a(\underline{v}', \underline{w}')$, which equals 0 as supposed. Consequently, there is a dipath $\vec{P}$ from $\underline{v}'$ to $\underline{w}'$ with $\omega_a(\vec{P}) = 0$. However, this implies $\omega(\vec{P}) = 0$. Thus the original distance from $\underline{v}'$ to $\underline{w}'$ is $dist(\underline{v}', \underline{w}') = 0$ as claimed.

Now we consider the function $g : V \to \mathbb{Q}_0$ defined by

$$g(\underline{v}') = \begin{cases} 0 & \text{if } \underline{v}' \text{ is redundant,} \\ \min\{dist_a(\underline{u}', \underline{v}') : \underline{u}' \in V\} & \text{otherwise.} \end{cases}$$

First we show that g is admissible. For that we use Lemma 16, that is, we have to verify (3) and (7) for every link $(\underline{r}, \underline{c})$. Suppose $g(\underline{c}) = 0$. Then $\underline{c}$ is redundant, and by Theorem 18 the object type $\underline{r}$ is redundant, too. This implies $g(\underline{r}) = 0$ and proves (3). Furthermore, we have

$$g(\underline{w}') \leq g(\underline{v}')dist_a(\underline{v}', \underline{w}') \leq g(\underline{v}')dist(\underline{v}', \underline{w}'),$$

which proves (7). Thus g is admissible.

We still have to confirm that g is suitable for our purposes here. By the discussion above and Theorem 18, we have $g(\underline{v}') = 0$ if and only if $\underline{v}'$ is redundant. Moreover, $dist_a(\underline{u}', \underline{w}') \leq dist_a(\underline{u}', \underline{v}')dist_a(\underline{v}', \underline{w}')$ trivially holds for any three vertices $\underline{u}', \underline{v}', \underline{w}'$. For our fixed arc $a = (\underline{v}, \underline{w})$, this implies

$$g(\underline{w}) \leq g(\underline{v})dist_a(\underline{v}, \underline{w}) \leq g(\underline{v})\omega_a(a) < g(\underline{v})\omega(a)$$

since $\underline{v}$ is not redundant. Hence g satisfies the desired inequality $g(\underline{w}) < \omega(a)g(\underline{v})$.

The existence of g causes the claimed contradiction. Consequently, a lies on some subcritical dicycle. □

Corollary 34. *Let $\ell = (\underline{r}, \underline{c})$ be a link such that both $\underline{r}$ and $\underline{c}$ are non-redundant for Σ_C. If neither ℓ nor ℓ^{-1} lies on a subcritical dicycle, then there exists an admissible function g for Σ_C satisfying $\alpha(\underline{r}, \underline{c})g(\underline{c}) < g(\underline{r}) < \beta(\underline{r}, \underline{c})g(\underline{c})$.*

Proof. Since $\underline{c}$ is non-redundant, CanDeg$(\underline{r}, \underline{c})$ is not empty. Every admissible function g satisfies the inequality $\alpha(\underline{r}, \underline{c})g(\underline{c}) \leq g(\underline{r}) \leq \beta(\underline{r}, \underline{c})g(\underline{c})$ by Lemma 12. We are now looking for two admissible functions g_1 and g_2 with $g_1(\underline{r}) < \beta(\underline{r}, \underline{c})g_1(\underline{c})$ and $\alpha(\underline{r}, \underline{c})g_2(\underline{c}) < g_2(\underline{r})$, respectively. Their sum $g_1 + g_2$ is again admissible and has the claimed property.

It remains to find g_1 and g_2. By Theorem 33 there is an admissible function g_1 with $g_1(\underline{r}) < \omega(\ell^{-1})g_1(\underline{c}) = \beta(\underline{r}, \underline{c})g_1(\underline{c})$ as desired. Further, Theorem 33 gives us an admissible function g_2 with $g_2(\underline{c}) < \omega(\ell)g_2(\underline{r})$. That is, $\alpha(\underline{r}, \underline{c})g_2(\underline{c}) < g_2(\underline{r})$ when $0 < \alpha(\underline{r}, \underline{c}) < \infty$ holds. The case $\alpha(\underline{r}, \underline{c}) = \infty$ may be excluded because CanDeg$(\underline{r}, \underline{c})$ is not empty. If $\alpha(\underline{r}, \underline{c}) = 0$, we may choose any admissible function g_2 with $g_2(\underline{r}) > 0$. Such a function exists since $\underline{r}$ is not redundant. □

9.3 Updating the weight function

Suppose we derive a new cardinality constraint by applying one of the rules (C4) to (C7) to some link $\ell = (\underline{r}, \underline{c})$. When adding the new constraint to the set Σ_C we obtain a new constraint set Σ'_C, which is semantically equivalent to Σ_C. Usually, this allows us to exclude some of the candidate degrees from $\mathrm{CanDeg}(\underline{r}, \underline{c})$, i.e. $\mathrm{CanDeg}'(\underline{r}, \underline{c})$ will be a proper subset of $\mathrm{CanDeg}(\underline{r}, \underline{c})$. We mention this because we calculated the weight function ω on the basis of $\mathrm{CanDeg}(\underline{r}, \underline{c})$.

The question arises whether there are dicycles which have not been subcritical for Σ_C, but are subcritical for the new constraint set Σ'_C, that is, we ask for dicycles $\vec{C}$ satisfying $\omega(\vec{C}) \neq 1$ and $\omega'(\vec{C}) = 1$. The next results cover this problem.

Lemma 35. *Let Σ'_C be derived from Σ_C by adding a new cardinality constraint implied by Σ_C via (C4) or (C5). A dicycle $\vec{C}$ is subcritical for Σ'_C only if it is subcritical for Σ_C.*

Proof. Suppose the new cardinality constraint is $card(\underline{r}, \underline{c}) = \emptyset$ due to (C4). With respect to Σ', we obtain $\mathrm{CanDeg}'(\underline{r}, \underline{c}) = \emptyset$, $\alpha'(\underline{r}, \underline{c}) = \infty$ and $\beta'(\underline{r}, \underline{c}) = 0$. This gives us the new weights $\omega'(\ell) = \omega'(\ell^{-1}) = 0$ for the link $\ell = (\underline{r}, \underline{c})$ and its reverse arc. Clearly, the weight of every arc a different from ℓ and ℓ^{-1} remains unchanged, that is, $\omega'(a) = \omega(a)$ holds. Consider a dicycle $\vec{C}$ with $\omega'(\vec{C}) \neq \omega(\vec{C})$. Then $\vec{C}$ contains ℓ or ℓ^{-1}. But both have weight 0, and thus $\omega'(\vec{C})$ is either 0 or ∞.

Next suppose the new cardinality constraint is $card(\underline{r}, \underline{c}) = \{0\}$ due to (C5). This time, we obtain $\mathrm{CanDeg}'(\underline{r}, \underline{c}) = \mathrm{CanDeg}(\underline{r}, \underline{c}) \cap \{0\}$, $\alpha'(\underline{r}, \underline{c}) = 0$ or ∞, and $\beta'(\underline{r}, \underline{c}) = 0$. This gives us the new weights $\omega'(\ell) = \infty$ or 0, as well as $\omega'(\ell^{-1}) = 0$ for the link $\ell = (\underline{r}, \underline{c})$ and its reverse arc. Again consider a dicycle $\vec{C}$ with $\omega'(\vec{C}) \neq \omega(\vec{C})$. Then $\vec{C}$ contains ℓ or ℓ^{-1}. Each of them has weight 0 or ∞, and thus $\omega'(\vec{C})$ is either 0 or ∞.

In both cases $\omega'(\vec{C})$ equals $\omega(\vec{C})$ or 0 or ∞. Therefore we have $\omega'(\vec{C}) = 1$ only if $\omega(\vec{C}) = 1$ holds. □

Lemma 36. *Let Σ'_C be derived from Σ_C by adding a new cardinality constraint implied by Σ_C via (C6) or (C7) applied to a link $\ell = (\underline{r}, \underline{c})$. Let a be an arc different from ℓ and ℓ^{-1} such that its vertices are both non-redundant for Σ_C. Then a lies on a subcritical dicycle for Σ'_C only if it lies on a subcritical dicycle for Σ_C.*

Proof. Suppose the new cardinality constraint is $card(\underline{r}, \underline{c}) = \{\alpha(\underline{r}, \underline{c})\}$ due to (C6). We obtain $\mathrm{CanDeg}'(\underline{r}, \underline{c}) = \{\alpha(\underline{r}, \underline{c})\}$, and $\alpha'(\underline{r}, \underline{c}) = \beta'(\underline{r}, \underline{c}) = \alpha(\underline{r}, \underline{c})$. For the link $\ell = (\underline{r}, \underline{c})$ we have $\omega'(\ell) = \omega(\ell)$. For its reverse arc we derive $\omega(\ell^{-1}) = \alpha(\underline{r}, \underline{c})$. The weight of every arc different from ℓ and ℓ^{-1} remains unchanged, too.

Of course, there must be a dicycle $\vec{C}_0$ which contains ℓ and satisfies $\omega(\vec{C}) = 1$. The existence of this dicycle allows us to apply (C6). Let $\vec{P}_0$ be the dipath from $\underline{r}$ to $\underline{c}$

which we obtain from $\vec{C}_0$ after deleting the link ℓ. We have

$$1 = \omega(\vec{C}_0) = \omega(\ell)\omega(\vec{P}_0) = \frac{1}{\alpha(\underline{r}, \underline{c})}\omega(\vec{P}_0),$$

where $0 < \alpha(\underline{r}, \underline{c}) < \infty$ holds as pointed out in the proof of Corollary 28.

Let $\vec{C}$ be some dicycle with $\omega'(\vec{C}) \neq \omega(\vec{C})$. Then $\vec{C}$ contains ℓ^{-1}. Let $\vec{P}$ be the dipath from $\underline{r}$ to $\underline{c}$, which we obtain from $\vec{C}$ by deleting the arc ℓ^{-1}. Here we have

$$\omega'(\vec{C}) = \omega'(\ell^{-1})\omega'(\vec{P}) = \alpha(\underline{r}, \underline{c})\omega(\vec{P}).$$

If $\omega'(\vec{C}) = 1$ holds, too, we conclude

$$1 = \omega'(\vec{C})\omega(\vec{C}_0) = \omega(\vec{P})\omega(\vec{P}_0) = \omega(\vec{P} \cup \vec{P}_0),$$

where $\vec{P} \cup \vec{P}_0$ is the closed diwalk obtained by stringing together $\vec{P}$ and $\vec{P}_0$.

Now consider the arc a under discussion. Assume it lies on a dicycle $\vec{C}$ which is subcritical for Σ'_C, that is, with $\omega'(\vec{C}) = 1$. We are ready if $\omega(\vec{C}) = 1$ holds. Otherwise consider the closed diwalk $\vec{P} \cup \vec{P}_0$ described above. It contains a and thus some non-redundant vertices for Σ_C. By Lemma 31, $\vec{P} \cup \vec{P}_0$ splits into dicycles which are all subcritical for Σ_C. One of them contains a and proves the claim for (C6).

For (C7) we use a similar argument. This concludes our proof. □

It remains to study the link $\ell = (\underline{r}, \underline{c})$ and its reverse arc $\ell^{-1} = (\underline{c}, \underline{r})$ after an application of either (C6) or (C7) to ℓ. Evidently, these two arcs together form a subcritical dicycle for Σ'_C. With respect to Σ'_C, however, there is only one candidate degree left over, i.e. $\mathrm{CanDeg}'(\underline{r}, \underline{c})$ is of size 1. Another application of (C6) or (C7) will not give us any new information as discussed in Remark 30: It suffices to apply (C6) and (C7) to those links which admit two or more candidate degrees. An update of the weight function does not result in any new link of this kind. Hence, for our purposes here, it is not necessary to update the weight function after an application of one of the rules (C4) to (C7).

10 Closed sets of cardinality constraints

10.1 Proof of the main theorem

In the preceding sections, we assembled a number of results to derive new constraints implied by an original set Σ_C of cardinality constraints. Due to their correctness proved above, the implication rules (C1) to (C7) may now be used as inference rules. The question arises whether these rules are complete, i.e. enable us to derive *all* cardinality constraints implied by Σ_C. The answer to this question is positive as claimed in Section 5.

We are now in the position to prove our main result, that is, Theorem 6. Let $\vec{S}$ be a database schema and Σ_C a set of cardinality constraints declared on $\vec{S}$. We claimed that the constraint set Σ_C is $\mathcal{C}$-closed if and only if it contains all constraints implied by Σ_C due to the rules (C1) to (C7).

Proof of Theorem 6. (*Necessity.*) The rules are correct as proved in Lemmas 5, 15 and 29. Since Σ_C is $\mathcal{C}$-closed, all constraints derived by any of these rules belong to Σ_C.

(*Sufficiency.*) Our aim is to apply Theorem 26. For this, it suffices to find an admissible function g such that $g(\underline{v}) > 0$ holds for every non-redundant object type $\underline{v}$ and g satisfies (10) or (14) for every link $(\underline{r}, \underline{c})$ with non-redundant $\underline{c}$.

Consider a link $(\underline{r}, \underline{c})$ where $\underline{c}$ is not redundant. Of course, $\mathrm{CanDeg}(\underline{r}, \underline{c})$ is non-empty. First, suppose $\mathrm{CanDeg}(\underline{r}, \underline{c})$ contains only one candidate degree $d^* = \alpha(\underline{r}, \underline{c}) = \beta(\underline{r}, \underline{c})$. For every admissible function g we have $d^* g(\underline{c}) = \alpha(\underline{r}, \underline{c}) g(\underline{c}) \leq g(\underline{r}) \leq \beta(\underline{r}, \underline{c}) g(\underline{c}) = d^* g(\underline{c})$ by Lemma 12. That is, condition (14) holds.

Otherwise, suppose $\mathrm{CanDeg}(\underline{r}, \underline{c})$ is of size 2 or larger. By (C5), $\underline{r}$ may not be redundant. Due to (C6) and (C7), we have neither $\ell = (\underline{r}, \underline{c})$ nor its reverse arc ℓ^{-1} on a subcritical dicycle. Hence Corollary 34 provides an admissible function g_ℓ satisfying (10) for the link ℓ under consideration.

By Lemma 14 there is an admissible function g_0 with $g_0(\underline{v}) > 0$ for every non-redundant object type $\underline{v}$. Of course, we do not know whether g_0 itself satisfies one of (10) or (14) for every link $\ell = (\underline{r}, \underline{c})$ with non-redundant $\underline{c}$. Therefore, we add g_ℓ to g_0 for each of these links. The resultant function g is still admissible and has all the properties under discussion. This allows us to apply Theorem 26, and Σ_C is $\mathcal{C}$-closed as claimed. □

10.2 How to decide implication

Theorem 6 gives a solution to the implication problem. In order to decide whether $\Sigma_C \models \sigma$ holds for some cardinality constraint σ, it suffices to check whether σ is in the $\mathcal{C}$-closure of Σ_C or not. An efficient way to determine this closure is provided by the following algorithm.

Algorithm 37. *Given a set Σ_C of cardinality constraints declared on a database schema $\vec{S}$, the algorithm determines the $\mathcal{C}$-closure $\overline{\Sigma}_C$ of Σ_C:*

1. *Determine Σ_C^{Can} from Σ_C by applying (C1) and (C3) to every suitable link.*
2. *Determine the set Red of redundant object types for Σ_C.*
3. *Determine a superset Σ'_C of Σ_C^{Can} by applying (C4) to (C7) to every suitable link.*
4. *Determine $\Sigma_C'^{Can}$ from Σ'_C by applying (C3) to every suitable link.*

5. *Determine $\overline{\Sigma}_C$ from $\Sigma_C'^{Can}$ by applying (C2) to every suitable link.*

Proof. Every constraint in $\overline{\Sigma}_C$ is implied by Σ_C. It remains to check whether $\overline{\Sigma}_C$ is $\mathcal{C}$-closed as claimed. For this task, we shall apply Theorem 6. Due to steps 1 and 5, $\overline{\Sigma}_C$ contains the trivial cardinality constraint proposed by (C1) for every link. In steps 4 and 5, we made sure that $\overline{\Sigma}_C$ contains every constraint derivable by (C2) and (C3). In step 2, we determined the set of redundant object types for Σ_C. Since Σ_C and $\overline{\Sigma}_C$ are semantically equivalent, the redundant object types for $\overline{\Sigma}_C$ are exactly the same as for Σ_C. Due to step 3, $\overline{\Sigma}_C$ contains every constraint proposed by (C4) and (C5).

It remains to discuss (C6) and (C7). Due to Remark 30, it is sufficient to apply (C6) and (C7) to those links which admit at least two candidate degrees. Consider a link of this kind such that the link itself or its reverse arc lies on a subcritical dicycle for Σ_C'. Then the link or its reverse arc, respectively, lies on a subcritical dicycle for Σ_C, too, as described in Section 9.3. Hence $\overline{\Sigma}_C$ contains every constraint proposed by (C6) and (C7) due to step 3. By our main Theorem 6, the generated constraint set $\overline{\Sigma}_C$ is $\mathcal{C}$-closed as desired. □

The application of (C4) to (C7) to the set Σ_C^{Can} has been discussed in Sections 7 and 9. This is possible in polynomial time. Now, let σ be a given cardinality constraint, say $card(\underline{r}, \underline{c}) = M$. To decide $\Sigma_C \models \sigma$, we have to consider the unique cardinality constraint $card(\underline{r}, \underline{c}) = M'$ in $\Sigma_C'^{Can} = \overline{\Sigma}_C^{Can}$, which is specified for the link $(\underline{r}, \underline{c})$. Then $\Sigma_C \models \sigma$ holds if and only if M is a superset of M'.

As mentioned earlier, Armstrong databases are a popular way to represent all implications of a given constraint set. The following result tells us which sets of cardinality constraints admit Armstrong databases.

Theorem 38. *There exists a $\mathcal{C}$-Armstrong database for Σ_C if and only if Σ_C implies a finite cardinality constraint for every link in the database schema $\vec{S}$.*

Proof. Without loss of generality, we may assume that Σ_C is $\mathcal{C}$-closed.

(*Necessity.*) Let $\vec{S}^t$ be an $\mathcal{C}$-Armstrong database for Σ_C. Then $\vec{S}^t$ is legal for Σ_C and violates every cardinality constraint not in Σ_C. Since we restricted ourselves to finite databases, every population in $\vec{S}^t$ is finite. Consider a link $(\underline{r}, \underline{c})$. Let M be the set containing all the degrees $\deg(\underline{r}^t, c)$ where c runs over the population $\underline{c}^t$. Since $\underline{c}^t$ is finite, the resultant set M is finite, too. Hence $\vec{S}^t$ satisfies the finite cardinality constraint $card(\underline{r}, \underline{c}) = M$. Consequently, this constraint belongs to Σ_C.

(*Sufficiency.*) Suppose Σ_C implies a finite cardinality constraint for every link. Consider a link $(\underline{r}, \underline{c})$. Since Σ_C is $\mathcal{C}$-closed, it contains the cardinality constraint $card(\underline{r}, \underline{c}) = \mathrm{CanDeg}(\underline{r}, \underline{c})$. Due to its definition and the existence of a finite cardinality constraint for the link $(\underline{r}, \underline{c})$ in Σ_C, the set $\mathrm{CanDeg}(\underline{r}, \underline{c})$ has to be finite.

Let d^* be a candidate degree in $\mathrm{CanDeg}(\underline{r}, \underline{c})$. In the proof of Theorem 6, we constructed a database instance which supplies evidence of $(\underline{r}, \underline{c}, d^*)$. We choose such a database instance for every link $(\underline{r}, \underline{c})$ and every $d^* \in \mathrm{CanDeg}(\underline{r}, \underline{c})$, and denote their disjoint union by $\vec{S}^t$. Since all the sets $\mathrm{CanDeg}(\underline{r}, \underline{c})$ are finite, the resultant database instance $\vec{S}^t$ is finite, too.

Clearly, $\vec{S}^t$ is legal for Σ_C. It is easy to see that $\vec{S}^t$ satisfies a cardinality constraint $card(\underline{r}, \underline{c}) = M$ only if M is a superset of $\mathrm{CanDeg}(\underline{r}, \underline{c})$. But in this case, the cardinality constraint belongs to Σ_C. Hence $\vec{S}^t$ satisfies exactly those cardinality constraints that are in Σ_C, and is $\mathcal{C}$-Armstrong as claimed. □

11 Interactions with key dependencies

In this section, we turn our attention to key dependencies and their interplay with cardinality constraints. A population which is empty or contains only a single relationship obviously satisfies every possible key dependency. Thus, sets of key dependencies are always satisfiable and even consistent. But as seen in the example of Figure 2, this is no longer true if we are given cardinality constraints, too. The reason for this are the following simple observations which hold for every link $(\underline{r}, \underline{c})$ (see also [49]):

(KC) If Σ contains the key dependency $\underline{r} : \{\underline{c}\} \to \underline{r}$, then $\Sigma \models (card(\underline{r}, \underline{c}) = \{0, 1\})$.

(CK) If Σ contains the cardinality constraint $card(\underline{r}, \underline{c}) = \{0, 1\}$, then $\Sigma \models (\underline{r} : \{\underline{c}\} \to \underline{r})$.

Lemma 39. *(KC) and (CK) are correct, that is, the constraints* $\underline{r} : \{\underline{c}\} \to \underline{r}$ *and* $card(\underline{r}, \underline{c}) = \{0, 1\}$ *are semantically equivalent for every link* $(\underline{r}, \underline{c})$.

Proof. We start with (KC). Let $\underline{r}^t$ be a population satisfying $\underline{r} : \{\underline{c}\} \to \underline{r}$. Consider an object c in the codomain $\underline{c}^t$. If c participates in two (or more) relationships r_1 and r_2 in $\underline{r}^t$, then their restrictions $r_1[\{\underline{c}\}]$ and $r_2[\{\underline{c}\}]$ are equal. This gives a contradiction to $\{\underline{c}\}$ being a key for $\underline{r}^t$. Hence the claimed implication holds.

Next, we treat (CK). Let $\underline{r}^t$ satisfy $card(\underline{r}, \underline{c}) = \{0, 1\}$. Again, consider an object c in the codomain $\underline{c}^t$. There is at most one relationship r in $\underline{r}^t$ whose restriction $r[\{\underline{c}\}]$ gives c. Thus $\{\underline{c}\}$ is a key for $\underline{r}^t$ as claimed. □

According to the previous result, certain cardinality constraints may be used to express unary key dependencies, and vice versa. As we shall see below, Lemma 39 describes the only nontrivial interaction between cardinality constraints and key dependencies. Throughout, let $\vec{S}$ be a database schema. Given a constraint set Σ specified on $\vec{S}$, let

$$\Sigma^{Card} = \{(card(\underline{r}, \underline{c}) = \{0, 1\}) : (\underline{r} : \{\underline{c}\} \to \underline{r}) \in \Sigma\}$$

contain all cardinality constraints obtained from key dependencies in Σ via (KC), and

$$\Sigma^{Key} = \{(\underline{r} : \{\underline{c}\} \to \underline{r}) : (card(\underline{r}, \underline{c}) = \{0, 1\}) \in \Sigma\}$$

contain all key dependencies derived from cardinality constraints in Σ via (CK). In this section, we consider sets Σ_C and Σ_K of cardinality constraints and key dependencies, respectively, declared on $\vec{S}$. Moreover, we usually put $\Sigma = \Sigma_C \cup \Sigma_K$.

11.1 Construction of legal databases

Lemma 40. *Let $\underline{r}^t$ be a given population with codomains $\underline{c}^t$, $\underline{c} \in Co(\underline{r})$, such that $\underline{r}^t$ satisfies Σ_K^{Card}. Then there exists a population $\underline{r}^{t'}$ which satisfies the whole set Σ_K and exactly the same cardinality constraints as $\underline{r}^t$ does.*

Proof. Denote the size of $\underline{r}^t$ by $g(\underline{r})$ and the size of any codomain $\underline{c}^t$ by $g(\underline{c})$. Let $\underline{r}$ be of arity n, that is, with n components. We choose a positive integer $q \geq g(\underline{r})$ such that there exists a resolvable transversal design $TD(n, q)$. By the theorem of MacNeish, we may choose q as a prime power satisfying $q \geq \max\{n, g(\underline{r})\}$. From $\underline{r}^t$ we shall construct a new population $\underline{r}^{t'}$ of size $qg(\underline{r})$.

Let $\underline{c}$ be a component of $\underline{r}$. For every old object c in $\underline{c}^t$, we introduce q new objects $(c, 1), \ldots, (c, q)$. Denote the resultant object set by $\underline{c}^{t'}$. The population $\underline{r}^{t'}$ to be constructed will have the codomains $\underline{c}^{t'}$, $\underline{c} \in \mathrm{Co}(\underline{r})$.

Due to our choice of q, there is a resolvable transversal design $TD(n, q)$. For every old relationship r in $\underline{r}^t$, we fix an own resolution class in the transversal design. Let $r = r(c_1, \ldots, c_n)$ be an old relationship in $\underline{r}^t$, and $(i_1, \ldots, i_n)$ be a block in the associated resolution class. Then we generate a new relationship $r' = r'((c_1, i_1), \ldots, (c_n, i_n))$. Doing the same for every old relationship and every block in the corresponding resolution class, we obtain a new population $\underline{r}^{t'}$ over $\underline{r}$ with exactly $qg(\underline{r})$ members.

Every new object (c, i) in a codomain $\underline{c}^{t'}$ participates in the same number of new relationships as the old object $c \in \underline{c}^t$ does in old relationships. Hence, the new population $\underline{r}^{t'}$ satisfies exactly the same cardinality constraints as the old population $\underline{r}^t$ does.

Consequently, $\underline{r}^{t'}$ satisfies Σ_K^{Card}, and thus every unary key dependency in Σ_K due to Lemma 39. Moreover, any two new relationships in $\underline{r}^{t'}$ share at most one entry. Therefore $\underline{r}^{t'}$ satisfies *every* possible non-unary key dependency, particularly those in Σ_K. Together this proves the new population $\underline{r}^{t'}$ to satisfy Σ_K as desired. $\square$

Theorem 41. *Let $\vec{S}^t$ be a given database instance which is legal for Σ_K^{Card}. Then there exists a database instance $\vec{S}^{t'}$ which is legal for the whole set Σ_K and satisfies exactly the same cardinality constraints as $\vec{S}^t$ does.*

Proof. In a database instance $\vec{S}^t$, the codomains $\underline{c}^t$ of a population $\underline{r}^t$ over a relationship type $\underline{r}$ are just the populations over the object types $\underline{c} \in \mathrm{Co}(\underline{r})$. To generate the new populations $\underline{r}^{t'}$ we always use the same integer q. Therefore we choose q such that it exceeds the size of *all* the old populations $\underline{r}^t$, and such that it ensures the existence of a resolvable transversal design $TD(n, q)$ even for the largest arity n of a relationship type in $\vec{S}$. With this, the claim is an easy consequence of Lemma 40. ☐

Theorem 42. *An object type* $\underline{v}$ *is redundant for* $\Sigma = \Sigma_C \cup \Sigma_K$ *if and only if it is redundant for* $\Sigma_C \cup \Sigma_K^{Card}$*.*

Proof. (*Necessity.*) Let $\underline{v}$ be redundant for Σ. Assume $\underline{v}$ is not redundant for $\Sigma_C \cup \Sigma_K^{Card}$. Then there is a database instance $\vec{S}^t$, which is legal for $\Sigma_C \cup \Sigma_K^{Card}$, and whose population $\underline{v}^t$ is non-empty. By virtue of Theorem 41, we also have a database instance $\vec{S}^{t'}$, which is legal for Σ. But due to its construction, the population $\underline{v}^{t'}$ is q times larger than $\underline{v}^t$, where q is some suitable positive integer. Therefore, $\underline{v}^{t'}$ is non-empty, too. But this contradicts our assumption. Thus $\underline{v}$ has to be redundant for $\Sigma_C \cup \Sigma_K^{Card}$.

(*Sufficiency.*) This part of the claim is obvious since Σ semantically implies $\Sigma_C \cup \Sigma_K^{Card}$ by Lemma 39. ☐

Corollary 43. $\Sigma = \Sigma_C \cup \Sigma_K$ *is consistent if and only if* $\Sigma_C \cup \Sigma_K^{Card}$ *is consistent.*

The consistency of $\Sigma_C \cup \Sigma_K^{Card}$ may be checked in polynomial time as proposed above. In particular, a consistent set of cardinality constraints together with a set of non-unary key dependencies is always consistent.

Example. Recall the database schema $\vec{S}$ in Figure 3. The set Σ_C of cardinality constraints declared on it is consistent. However, the situation changes if we add the key dependency $\underline{r}_2 : \{\underline{e}_2\} \to \underline{r}_2$. The resultant constraint set Σ is no longer consistent. Here Σ_K^{Card} contains the additional cardinality constraint $card(\underline{r}_2, \underline{e}_2) = \{0, 1\}$. With respect to $\Sigma_C \cup \Sigma_K^{Card}$, the set of candidate degrees for the link $(\underline{r}_2, \underline{e}_2)$ is $\text{CanDeg}(\underline{r}_2, \underline{e}_2) = \{0, 1\}$. This leads to a modification of the weight function ω defined on the symmetric digraph $\vec{D}$ as shown in Figure 6. The dicycle from $\underline{e}_2$ via $\underline{r}_2$, $\underline{e}_1$ and $\underline{r}_1$ back to $\underline{e}_2$ has weight 1/12, i.e. is absorbing for $\Sigma_C \cup \Sigma_K^{Card}$. Thus all object types on this dicycle are redundant for Σ.

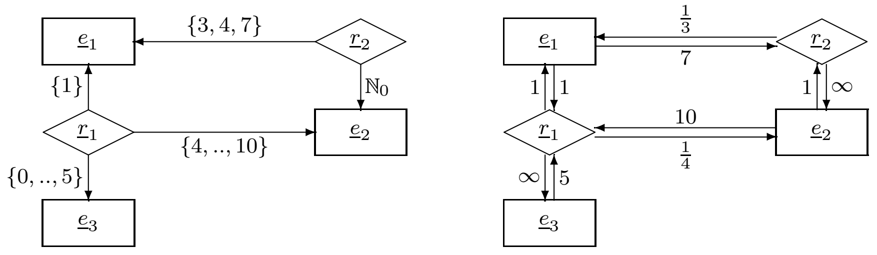

Figure 6: The modified weight function under consideration of an additional key dependency $\underline{r}_2 : \{\underline{e}_2\} \to \underline{r}_2$.

Theorem 44. $\Sigma = \Sigma_C \cup \Sigma_K$ *is* $\mathcal{C}$*-closed if and only if* Σ_C *is* $\mathcal{C}$*-closed and contains every constraint implied by* Σ_K *due to (KC).*

Proof. (*Necessity.*) Let Σ be $\mathcal{C}$-closed. The correctness of (KC) was shown in Lemma 39. This gives us $\Sigma_K^{Card} \subseteq \Sigma_C$. Now consider a cardinality constraint σ not in Σ_C and thus not Σ. Since Σ is $\mathcal{C}$-closed, there exists a database instance $\vec{S}^t$ which is legal for Σ, but violates σ. As Σ_C is a subset of Σ, this proves Σ_C to be $\mathcal{C}$-closed, too.

(*Sufficiency.*) Suppose Σ_C is $\mathcal{C}$-closed and contains Σ_K^{Card} as a subset. Let σ be a cardinality constraint not in Σ and, consequently, not in Σ_C. Since Σ_C is $\mathcal{C}$-closed, we find a database instance $\vec{S}^t$ which is legal for Σ_C, but violates σ. Due to $\Sigma_K^{Card} \subseteq \Sigma_C$, this database instance is in particular legal for Σ_K^{Card}. Now we apply Theorem 41. This gives us a database instance $\vec{S}^{t'}$, which is legal for Σ_K. Moreover $\vec{S}^{t'}$ satisfies the same cardinality constraints as $\vec{S}^t$ does. Therefore, it is legal for Σ_C, but violates σ. This proves Σ to be $\mathcal{C}$-closed. □

11.2 Implications from key dependencies

To continue with, we restate a well-known result on implications of key dependencies. The combinatorial structure of key dependencies was investigated in [2, 20, 24]. These papers, in particular, contain a simple observation which we record here for future reference.

Lemma 45 (cf. [2, 19, 20]). *Let $\underline{r}$ be a relationship type, and X be a non-empty subset of $Co(\underline{r})$. The following rule is correct:*

(K) *If Σ_K contains $\underline{r} : X \to \underline{r}$, then $\Sigma_K \models (\underline{r} : Y \to \underline{r})$ for every $Y \subseteq Co(\underline{r})$ with $X \subseteq Y$.*

As shown by Armstrong [2] and Demetrovics [20], a set Σ_K of key dependencies is $\mathcal{K}$-closed if and only if it contains all constraints obtained from Σ_K by applying Lemma 45.

At this point a brief remark is called for. In a number of papers on database design, the key dependencies $\underline{r} : \mathrm{Co}(\underline{r}) \to \underline{r}$ are assumed to be 'built-in axioms'. This forces the relationships in a population $\underline{r}^t$ to yield mutually distinct mappings from $\mathrm{Co}(\underline{r}) = \{\underline{c}_1, \dots, \underline{c}_n\}$ to the cartesian product $\underline{c}_1^t \times \cdots \times \underline{c}_n^t$. Under this presumption, the relationships may be identified by their components $r(\underline{c}_i)$. For a discussion of identification mechanisms in the entity-relationship model, we refer to [5, 50]. In the definition of database instances in Section 3, we do not explicitly claim the relationships in a population to have this property. However, this can easily be reached if desired: We must simply add these key dependencies as trivial constraints to every set Σ_K, that is, agree on the following rule (KA) for every relationship type $\underline{r}$:

(KA) $\Sigma_K \models (\underline{r} : \mathrm{Co}(\underline{r}) \to \underline{r})$.

A key $X \subseteq \mathrm{Co}(\underline{r})$ for a population $\underline{r}^t$ is said to be *minimal* if no proper subset of X is a key for $\underline{r}^t$ again. By Lemma 45, the set of minimal keys for a population $\underline{r}^t$ completely determines the set of all keys for $\underline{r}^t$. This gives a special motivation for the investigation of minimal keys. Two different minimal keys will never contain each other: Sets of minimal keys correspond to *Sperner families* over the component set $\mathrm{Co}(\underline{r})$, see [20]. In that paper, the interested reader will also find a construction of $\mathcal{K}$-Armstrong populations for a given set of key dependencies. Complexity issues concerning this construction are tackled by Demetrovics and Thi [24].

11.3 Ingredients from graph theory

Our aim is to give a characterization of $(\mathcal{C} \cup \mathcal{K})$-closed sets, where $\mathcal{C} \cup \mathcal{K}$ is the joint class of cardinality constraints and key dependencies. Again we use representation graphs which have been introduced in Section 6. The reader is reminded that, for every link $(\underline{r}, \underline{c})$, we considered a clique graph $G^t(\underline{r}, \underline{c})$ whose vertices are the relationships in $\underline{r}^t$. Two distinct relationships r_1 and r_2 are connected by an edge in $G^t(\underline{r}, \underline{c})$ if $r_1(\underline{c}) = r_2(\underline{c})$ holds. Moreover, let $G^t(\underline{r}, X)$ denote the intersection of the representation graphs $G^t(\underline{r}, \underline{c})$, $\underline{c} \in X$, for any subset X of $\mathrm{Co}(\underline{r})$. It is easy to check, that $G^t(\underline{r}, X)$ contains an edge connecting r_1 and r_2 exactly when we have $r_1[X] = r_2[X]$, and thus is a clique graph again.

As on cardinality constraints, representation graphs give us useful information on the key dependencies satisfied by a population $\underline{r}^t$.

Lemma 46. *The population $\underline{r}^t$ satisfies the key dependency $\underline{r} : X \to \underline{r}$ if and only if $G^t(\underline{r}, X)$ is edgeless.*

Proof. In $G^t(\underline{r}, X)$, two vertices r_1 and r_2 are connected by an edge if and only if the restrictions $r_1[X]$ and $r_2[X]$ are equal. However, X is a key if and only if the restrictions $r[X]$ are mutually distinct for the relationships r in $\underline{r}^t$. This verifies the claimed result. □

We collect a few more notations from graph theory. By K_k we denote the *complete graph* on k vertices. The vertex-disjoint union of μ copies of K_k yields a clique graph which we denote by μK_k. Given a graph G, the *valency* of a vertex specifies the number of edges containing this vertex. The minimum valency over all vertices in G is usually denoted by $\delta(G)$. A major tool for our further investigation is the following theorem of Hajnal and Szemerédi [33]. This result was first conjectured by Erdős [27] and gives a necessary condition on the occurrence of clique graphs as subgraphs in a given graph H. For a detailed discussion, we refer to [11].

Theorem 47 (Hajnal and Szemérdi). *Let H be a graph with $m = \mu k$ vertices and minimum valency $\delta(H) \geq m - \mu$. Then H has a subgraph isomorphic to the clique graph μK_k.*

Using the theorem of Hajnal and Szemerédi we derive the following result which happens to be essential for the solution of the implication problem for cardinality constraints and key dependencies.

Theorem 48. *Suppose we are given positive integers n and $k \geq 2$. Further, let Y be a subset of $\{1, \ldots, n\}$, and e be a fixed edge in the complete graph K_m with $m = k^n$ vertices. Then there is a collection of spanning subgraphs $H_1, \ldots, H_n$ of K_m satisfying the following conditions:*

(i) For every j with $1 \leq j \leq n$, the subgraph H_j is isomorphic to the clique graph μK_k with $\mu = k^{n-1}$.

(ii) For every pair i, j with $1 \leq i < j \leq n$ and $\{i, j\} \nsubseteq Y$, the subgraphs H_i and H_j are edge-disjoint.

(iii) For every pair i, j with $1 \leq i < j \leq n$ and $\{i, j\} \subseteq Y$, the subgraphs H_i and H_j share exactly one edge, namely the fixed edge e.

Before we start our proof, we like to recall that Bernoulli's inequality

$$k^s \geq 1 + s(k-1) \tag{15}$$

holds for every non-negative integer s and every positive integer k.

Proof. We choose a complete subgraph F_y of size k in K_m for every $y \in Y$, such that any two of these subgraphs have exactly the fixed edge e in common. This is possible since

$$m = k^n \geq 2 + n(k-2) \geq 2 + |Y|(k-2)$$

holds due to our choice of m and (15). In the sequel, F_y will always be a first clique in the subgraph H_y to be constructed. This is to ensure condition (iii).

Assume we already have suitable subgraphs H_i for $i < j$, and we are now going to construct H_j. First, suppose j is not in Y. We have to arrange the m vertices in cliques of size k each. For that, however, we may use neither the edges in the subgraphs H_i, $i < j$, nor the edges in the cliques F_y, $y \in Y$. Let G be the spanning subgraph of K_m containing all the remaining, i.e. permitted edges for H_j. It is not difficult to see, that every vertex in G has valency at least

$$\delta(G) \geq \delta(K_m) - (n-1)\delta(K_k) = (m-1) - (n-1)(k-1).$$

By (15) we derive $\mu = m/k = k^{n-1} \geq 1 + (n-1)(k-1)$ and thus $\delta(G) \geq m - \mu$. Hence we may apply the theorem of Hajnal and Szemerédi to G. This gives us a subgraph which is isomorphic to μK_k and will be chosen for H_j.

Otherwise, suppose j is in Y. This time we already have a first clique F_j, and it remains to arrange the $m - k$ vertices not in F_j in cliques of size k each. As above, let G be the spanning subgraph of K_m containing all the permitted edges for

H_j. Furthermore, let G' be the subgraph of G induced by the remaining vertices under discussion. Analogously, we derive $\delta(G') \geq (m-k) - (\mu - 1)$. On applying the theorem of Hajnal and Szemerédi to G', we obtain a subgraph isomorphic to $(\mu - 1)K_k$. Its union with the first clique F_j provides the desired subgraph H_j.

When constructing subgraphs $H_1, \ldots, H_n$ as described above, the validity of (i) to (iii) is easy to check. This concludes the proof. □

Lemma 49. *Let Σ_C be $\mathcal{C}$-closed, Σ_K be $\mathcal{K}$-closed and suppose $\Sigma_K^{Card} \subseteq \Sigma_C$ as well as $\Sigma_C^{Key} \subseteq \Sigma_K$ hold. Consider a relationship type $\underline{r}$ and a non-empty subset $Y \subseteq Co(\underline{r})$. If Σ_K does not contain the key dependency $\underline{r} : Y \to \underline{r}$, then there is a population $\underline{r}^s$ which satisfies $\Sigma = \Sigma_C \cup \Sigma_K$, but violates $\underline{r} : Y \to \underline{r}$.*

Proof. Since Σ_K is $\mathcal{K}$-closed, there is a population $\underline{r}^t$ which satisfies Σ_K and violates $\underline{r} : Y \to \underline{r}$. But unfortunately, $\underline{r}^t$ will usually not satisfy Σ_C. In fact, the difficulty arises from the simultaneous consideration of key dependencies and cardinality constraints.

Suppose now the set Y is of size one, say $Y = \{\underline{c}\}$. Then Σ may not contain the cardinality constraint $card(\underline{r}, \underline{c}) = \{0, 1\}$ which is semantically equivalent to the unary key dependency under discussion. The constraint set Σ is $\mathcal{C}$-closed by Theorem 44. Hence there is a population $\underline{r}^s$ which satisfies Σ, but violates $card(\underline{r}, \underline{c}) = \{0, 1\}$ and therefore violates $\underline{r} : Y \to \underline{r}$, too. As an example for $\underline{r}^s$, we may choose the population constructed in Lemma 40.

Otherwise let Y be of size at least two. Σ_K does not contain any of the unary key dependencies $\underline{r} : \{\underline{c}\} \to \underline{r}$ with $\underline{c} \in Y$. Consequently, none of the cardinality constraints $card(\underline{r}, \underline{c}) = \{0, 1\}$ is in Σ_C. Thus, for every $\underline{c} \in Y$, the set $\text{CanDeg}(\underline{r}, \underline{c})$ contains a candidate degree $d^* \geq 2$.

Let $\underline{r}^t$ be a population with codomains $\underline{c}^t$, $\underline{c} \in \text{Co}(\underline{r})$, such that $\underline{r}^t$ satisfies Σ_C. As usual, we denote the size of this population and its codomains by $g(\underline{r})$ and $g(\underline{c})$, respectively. Put $\mu = g(\underline{r})^{n-1}$, where n is the arity of $\underline{r}$. We aim at constructing a population $\underline{r}^s$ of size $h(\underline{r}) = \mu g(\underline{r})$. For every component $\underline{c} \in \text{Co}(\underline{r})$, the codomain $\underline{c}^s$ will just be the disjoint union of μ copies of $\underline{c}^t$. Thus $\underline{c}^s$ will be of size $h(\underline{c}) = \mu g(\underline{c})$.

We are going to construct the desired representation graphs $G^s(\underline{r}, \underline{c})$ as subgraphs of the complete graph $K_{h(\underline{r})}$. Let e be a fixed edge in this complete graph. By Theorem 48, we find subgraphs $H(\underline{r}, \underline{c})$ such that each of them is isomorphic to the clique graph $\mu K_{g(\underline{r})}$ and contains the fixed edge e exactly when $\underline{c}$ lies in Y.

Now consider the representation graphs for the old population $\underline{r}^t$. $G^t(\underline{r}, \underline{c})$ is a clique graph on $g(\underline{r})$ vertices and with clique sizes in $\text{CanDeg}(\underline{r}, \underline{c})$. On replacing every clique in $H(\underline{r}, \underline{c})$ by a copy of the corresponding graph $G^t(\underline{r}, \underline{c})$, we obtain the desired representation graph $G^s(\underline{r}, \underline{c})$ for our new population $\underline{r}^s$.

It remains to check whether the population $\underline{r}^s$ with these representation graphs has the claimed properties. Clearly, $\underline{r}^s$ satisfies exactly the same cardinality constraints as $\underline{r}^t$ does. In particular, $\underline{r}^s$ satisfies Σ_C.

Let $\underline{r} : X \to \underline{r}$ be a key dependency in Σ_K. We treat two cases. If X is of size one, say $X = \{\underline{c}\}$, then Σ_C contains the cardinality constraint $card(\underline{r}, \underline{c}) = \{0, 1\}$. The population $\underline{r}^s$ satisfies Σ_C and therefore the unary key dependency $\underline{r} : X \to \underline{r}$, too. Conversely, let X be of size two or larger. Since Σ_K is $\mathcal{K}$-closed, X is not a subset of Y. By Theorem 48, the intersection of the graphs $H(\underline{r}, \underline{c})$, $\underline{c} \in X$, is edgeless. Due to our construction, the same holds for $G^s(\underline{r}, X)$ which is the intersection of the representation graphs $G^s(\underline{r}, \underline{c})$, $\underline{c} \in X$. Hence the key dependency under discussion is satisfied by $\underline{r}^s$.

Finally, consider the subset Y. The graphs $H(\underline{r}, \underline{c})$, $\underline{c} \in Y$, have exactly the fixed edge e in common. We want to ensure the same for the graphs $G^s(\underline{r}, \underline{c})$. Therefore, we have to replace the clique containing the edge e in such a way by a copy of $G^t(\underline{r}, \underline{c})$, that e will not be deleted. This is easy to guarantee as long as $G^t(\underline{r}, \underline{c})$ is not a collection of isolated vertices. For this reason, we have to guarantee for every $\underline{c} \in Y$ that the old representation graph $G^t(\underline{r}, \underline{c})$ contains a clique of size at least 2.

Fortunately, every set CanDeg$(\underline{r}, \underline{c})$, $\underline{c} \in Y$, contains a candidate degree $d^* \geq 2$ as pointed out above. It suffices to start with a population $\underline{r}^t$ which supplies evidence of every $(\underline{r}, \underline{c}, d^*)$ where $\underline{c}$ is in Y and d^* is the corresponding candidate degree. The existence of such a population is guaranteed by Theorem 22. In this case, the new population $\underline{r}^s$ violates the key dependency $\underline{r} : Y \to \underline{r}$ as desired. □

11.4 Closed sets of cardinality constraints and key dependencies

We are now in the position to establish the main result of this section. It provides a complete characterization of closed sets in the joint class of cardinality constraints and key dependencies.

Theorem 50. *$\Sigma = \Sigma_C \cup \Sigma_K$ is $(\mathcal{C} \cup \mathcal{K})$-closed if and only if Σ_C is $\mathcal{C}$-closed, Σ_K is $\mathcal{K}$-closed and Σ contains every constraint implied by Σ due to (KC) or (CK).*

Proof. (*Necessity.*) Suppose Σ is $\mathcal{C}$-closed as well as $\mathcal{K}$-closed. By Theorem 44, also Σ_C is $\mathcal{C}$-closed and contains all constraints derived by (KC). Since Σ_K consists of all key dependencies in Σ, it is $\mathcal{K}$-closed, too. The correctness of (CK) is due to Lemma 39.

(*Sufficiency.*) Again by Theorem 44, Σ is $\mathcal{C}$-closed under the specified conditions. It remains to verify that Σ is $\mathcal{K}$-closed. Consider an arbitrary key dependency σ not in Σ, that is, not in Σ_K. We have to find a database instance which is legal for Σ, but violates σ. This is easily done if σ is a unary key dependency: According to (CK), Σ does not contain the corresponding cardinality constraint, and this yields the existence of the desired database instance since Σ is $\mathcal{C}$-closed.

Hence we may restrict ourselves to non-unary key dependencies $\sigma \notin \Sigma$. Let σ be the key dependency $\underline{r} : Y \to \underline{r}$ where $\underline{r}$ is a relationship type in $\vec{S}$ and $Y \subseteq \mathrm{Co}(\underline{r})$ holds. Clearly, Σ does not contain any cardinality constraint $card(\underline{r}, \underline{c}) = \{0, 1\}$ with $\underline{c} \in Y$. For each component $\underline{c} \in Y$, we fix a candidate degree $d^* \geq 2$. By Theorem 22, there is a database instance $\vec{S}^t$ which is legal for Σ_C and supplies evidence of every $(\underline{r}, \underline{c}, d^*)$ under discussion. As usual, let $g(\underline{v})$ denote the size of the population $\underline{v}^t$ in $\vec{S}^t$.

Now we apply Theorem 45 to obtain a database instance $\vec{S}^{t'}$ which is legal for Σ. Its populations $\underline{v}^{t'}$ are of size $qg(\underline{v})$ where q is some sufficiently large integer. Unfortunately, $\vec{S}^{t'}$ will usually not violate $\underline{r} : Y \to \underline{r}$ as desired. On the other hand, Lemma 49 gives us a population $\underline{r}^s$ which satisfies Σ, but violates $\underline{r} : Y \to \underline{r}$. This population is of size $h(\underline{r}) = \mu g(\underline{r})$ and has codomains $\underline{c}^s$ of size $h(\underline{c}) = \mu g(\underline{c})$.

The idea is to combine both results. For that, we take the disjoint union of μ copies of $\vec{S}^{t'}$. In the resultant database instance $\vec{S}^{t''}$ we replace the population $\underline{r}^{t''}$ by the disjoint union of q copies of $\underline{r}^s$. This provides a database instance which is still legal for Σ, but violates the key dependency $\underline{r} : Y \to \underline{r}$ under inspection.

Due to discussion above, Σ is $\mathcal{K}$-closed and, consequently, $(\mathcal{C} \cup \mathcal{K})$-closed as claimed. □

The condition on the constraints derived from Σ via (CK) or (KC) may also be expressed as follows.

Corollary 51. *$\Sigma = \Sigma_C \cup \Sigma_K$ is $(\mathcal{C} \cup \mathcal{K})$-closed if and only if Σ_C is $\mathcal{C}$-closed, Σ_K is $\mathcal{K}$-closed and we have*

$$\{(\underline{r}, \underline{c}) \in L : (card(\underline{r}, \underline{c}) = \{0, 1\}) \in \Sigma_C\} = \{(\underline{r}, \underline{c}) \in L : (\underline{r} : \{\underline{c}\} \to \underline{r}) \in \Sigma_K\}. \quad (16)$$

Moreover, Theorem 50 gives us an efficient method to decide implication in $\mathcal{C} \cup \mathcal{K}$.

Algorithm 52. *Given Σ_C and Σ_K specified on $\vec{S}$, the algorithm determines the $(\mathcal{C} \cup \mathcal{K})$-closure $\overline{\Sigma}$ of the constraint set $\Sigma = \Sigma_C \cup \Sigma_K$:*

1. *Determine Σ_K^{Card} from Σ_K by applying (KC) to every suitable link.*
2. *Determine the $\mathcal{C}$-closure $\overline{\Sigma}_C$ of $\Sigma_C \cup \Sigma_K^{Card}$ as proposed in Section 10.*
3. *Determine $\overline{\Sigma}_C^{Key}$ from $\overline{\Sigma}_C$ by applying (CK) to every suitable link.*
4. *Determine the $\mathcal{K}$-closure $\overline{\Sigma}_K$ of $\Sigma_K \cup \overline{\Sigma}_C^{Key}$ as proposed above.*
5. *Join $\overline{\Sigma}_C$ and $\overline{\Sigma}_K$, and denote the union by $\overline{\Sigma}$.*

Proof. Every constraint in $\overline{\Sigma}$ is implied by the given constraint set Σ. It suffices to check whether $\overline{\Sigma}$ is $(\mathcal{C} \cup \mathcal{K})$-closed as claimed. We shall apply Theorem 50 to the derived constraint set $\overline{\Sigma}$. Note, that $\overline{\Sigma}_C$ consists of all cardinality constraints in $\overline{\Sigma}$, while $\overline{\Sigma}_K$ consists of all key dependencies in $\overline{\Sigma}$. Clearly, $\overline{\Sigma}_C$ is $\mathcal{C}$-closed due to step 2, and $\overline{\Sigma}_K$ is $\mathcal{K}$-closed due to step 4. Furthermore, $\overline{\Sigma}_C^{Key}$ is a subset of $\overline{\Sigma}$ due to step 3.

It remains to show that $\overline{\Sigma}_K^{Card}$ is a subset of $\overline{\Sigma}$, too. Every unary key dependency in $\overline{\Sigma}_K$ belongs to Σ_K or is implied by $\Sigma_C \cup \Sigma_K^{Card}$. Consequently, every constraint in $\overline{\Sigma}_K^{Card}$ lies in Σ_K^{Card} or is implied by $\Sigma_C \cup \Sigma_K^{Card}$. Thus $\overline{\Sigma}_K^{Card}$ has to be a subset of the $\mathcal{C}$-closure of $\Sigma_C \cup \Sigma_K^{Card}$, that is, a subset of $\overline{\Sigma}_C$ due to step 2. □

Theorem 53. *There exists a database instance over $\vec{S}$ which is $(\mathcal{C} \cup \mathcal{K})$-Armstrong for $\Sigma = \Sigma_C \cup \Sigma_K$ if and only if Σ implies a finite cardinality constraint for every link in $\vec{S}$.*

Proof. Without loss of generality, we may suppose that Σ is $(\mathcal{C} \cup \mathcal{K})$-closed. Obviously, the condition is necessary as in Theorem 38.

(*Sufficiency.*) We are looking for a database instance which satisfies a constraint σ from $\mathcal{C} \cup \mathcal{K}$ just when σ belongs to Σ. Recall that $\mathcal{K}$ was the set of all possible key dependencies with respect to the given database schema $\vec{S}$. Clearly, this set is finite. For every key dependency $\sigma \in \mathcal{K}$ which is not in Σ, we take a database instance which is legal for Σ but violates σ. We constructed such a database instance in the proof of Theorem 50. Now we take the disjoint union of all these database instances. The resultant database instance $\vec{S}^s$ is $\mathcal{K}$-Armstrong for Σ.

Furthermore, Theorem 38 yields a database instance $\vec{S}^t$ which is $\mathcal{C}$-Armstrong for Σ_C. As supposed, Σ is $(\mathcal{C} \cup \mathcal{K})$-closed. This gives us $\Sigma_K^{Card} \subseteq \Sigma_C$ by Theorem 50. Hence, Theorem 44 provides a database schema $\vec{S}^{t'}$ which is $\mathcal{C}$-Armstrong for Σ. Finally, the disjoint union of $\vec{S}^{t'}$ and $\vec{S}^s$ is $(\mathcal{C} \cup \mathcal{K})$-Armstrong for Σ and establishes the claimed result. □

12 Interactions with functional dependencies

Let Σ_C, Σ_K and Σ_F denote sets of cardinality constraints, key dependencies and functional dependencies, respectively, declared on a database schema $\vec{S}$. In addition, we usually put $\Sigma = \Sigma_C \cup \Sigma_K \cup \Sigma_F$. It is well-known that the *Armstrong rules* [2, 19] provide a solution to the implication problem for functional dependencies. We record these rules for future reference.

Lemma 54 (Armstrong). *Let $\underline{r}$ be a relationship type, and X, Y and Z be non-empty subsets of $Co(\underline{r})$. The following rules are correct:*

(F1) $\Sigma \models (\underline{r} : X \to Y)$ *whenever* $Y \subseteq X$ *holds.*

(F2) *If* Σ *contains* $\underline{r} : X \to Y$ *and* $\underline{r} : Y \to Z$, *then* $\Sigma \models (\underline{r} : X \to Z)$.

(F3) *If* Σ *contains* $\underline{r} : X \to Y$, *then* $\Sigma \models (\underline{r} : X \cup Z \to Y \cup Z)$.

Due to their definition, key dependencies happen to be special functional dependencies. Some observations may be noted immediately.

Lemma 55. *Let $\underline{r}$ be a relationship type, and X be a non-empty subset of $Co(\underline{r})$. The following rules are correct:*

(KF) *If Σ contains $\underline{r} : X \to \underline{r}$, then $\Sigma \models (\underline{r} : X \to Co(\underline{r}))$.*

(FK) *If Σ contains $\underline{r} : X \to Co(\underline{r})$ and $\underline{r} : Co(\underline{r}) \to \underline{r}$, then $\Sigma \models (\underline{r} : X \to \underline{r})$.*

Armstrong [2] proved the rules given in Lemmas 54 to be correct and complete. Hence, Σ_F is $\mathcal{F}$-closed if and only if it contains all constraints obtained by applying (F1) to (F3). Similarly, $\Sigma_K \cup \Sigma_F$ is $(\mathcal{K} \cup \mathcal{F})$-closed if and only if it contains all constraints derived by (K), (F1) to (F3), (KF) and (FK).

The Armstrong rules gave rise to a considerable number of papers concerning applications, consequences and alternative rules. For a discussion, see [47]. In [2, 25], one finds methods for constructing $\mathcal{F}$-Armstrong populations.

Theorem 56. *Let Σ_F contain non-unary functional dependencies only, and let $\vec{S}^t$ be a given database instance. There exists a database instance $\vec{S}^{t'}$, which is legal for Σ_F and satisfies exactly the same cardinality constraints as $\vec{S}^t$ does.*

Proof. We apply the construction presented in the proof of Theorem 41 to $\vec{S}^t$. In the resultant database instance $\vec{S}^{t'}$, any two relationships in a population $\underline{r}^{t'}$ agree in at most one component. Hence, $\vec{S}^{t'}$ trivially satisfies every non-unary functional dependency. □

As for Theorem 41 in the case of key dependencies, we derive a couple of interesting consequences from Theorem 56 for a set $\Sigma = \Sigma_C \cup \Sigma_K \cup \Sigma_F$ containing cardinality constraints, key and functional dependencies. The proofs are analogous to the proofs of Theorem 42, Corollary 43 and Theorem 44. The major tool in these proofs is the construction introduced to verify Theorem 41. The resultant database instance $\vec{S}^{t'}$ satisfies *every* possible non-unary key or functional dependency.

Corollary 57. *Let Σ_F contain non-unary functional dependencies only.*

(i) An object type $\underline{v}$ is redundant for $\Sigma = \Sigma_C \cup \Sigma_K \cup \Sigma_F$ if and only if it is redundant for $\Sigma_C \cup \Sigma_K$.

(ii) Σ is consistent if and only if $\Sigma_C \cup \Sigma_K$ is consistent.

(iii) Σ is $\mathcal{C}$-closed if and only if $\Sigma_C \cup \Sigma_K$ is $\mathcal{C}$-closed.

The question whether $\Sigma_C \cup \Sigma_K$ is closed with respect to the class of cardinality constraints may be decided according to Theorem 44. We continue our study by investigating closed sets with respect to the joint class of cardinality constraints, key and functional dependencies. Again, representation graphs $G^t(\underline{r}, \underline{c})$ and their intersections are profitable to derive the desired characterization.

Lemma 58. *The population $\underline{r}^t$ satisfies the functional dependency $\underline{r} : X \to Y$ if and only if $G^t(\underline{r}, X)$ is a subgraph of $G^t(\underline{r}, Y)$.*

Proof. In $G^t(\underline{r}, X)$ two vertices r_1 and r_2 are connected by an edge whenever the restrictions $r_1[X]$ and $r_2[X]$ of the relationships r_1 and r_2 are equal. The definition of the functional dependency $\underline{r} : X \to Y$ verifies the claimed result. □

Attention to combinatorial properties of functional dependencies has been drawn e.g. in [6, 7, 21, 22]. As already pointed out by Codd [17] and Armstrong [2], functional dependencies can be represented by closure operations. For a relationship type $\underline{r}$ and any non-empty subset $X \subseteq \underline{r}$, we put

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

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

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

is semantically closed in the class $\mathcal{F}$ of functional dependencies. A complete theory of closure operations to handle sets of functional dependencies was developed in [12].

Next, we give a result analogous to Lemma 49, but for key and non-unary functional dependencies. For that, we need an interesting observation that enables us to reuse the populations constructed to verify Lemma 49.

Lemma 59. *Let $\underline{r}$ be a relationship type, and X, Z be non-empty subsets of $Co(\underline{r})$. The population $\underline{r}^s$ constructed in Lemma 49 satisfies a non-unary functional dependency $\underline{r} : X \to Z$ if and only we have either $X \not\subseteq Y$ or $X \cup Z \subseteq Y$.*

Proof. Suppose the subset $Y \subseteq \mathrm{Co}(\underline{r})$ in Lemma 49 is of size 1. The corresponding population $\underline{r}^s$ generated in Lemma 49 satisfies every possible non-unary functional dependency $\underline{r} : X \to Z$. Since Y is of size one and X of size at least two, we evidently have $X \not\subseteq Y$.

Otherwise, suppose Y is of size 2 or larger. The population $\underline{r}^s$ satisfies $\underline{r} : X \to Z$ exactly when $G^s(\underline{r}, X)$ is a subgraph of $G^s(\underline{r}, Z)$. We treat two cases. If $X \not\subseteq Y$ holds, the $G^s(\underline{r}, X)$ is edgeless by our construction and in particular by Theorem 48). In this case, $G^s(\underline{r}, X)$ is a trivial subgraph of $G^s(\underline{r}, Z)$.

Conversely, if $X \subseteq Y$ holds, the graph $G^s(\underline{r}, X)$ contains a single edge, namely the fixed edge e. However, $G^s(\underline{r}, Z)$ contains this edge just when Z is a subset of Y, too. This gives us $X \cup Z \subseteq Y$ and proves the claim. □

Lemma 60. *Let all functional dependencies in Σ_F be non-unary, trivial or implied by Σ_K. Moreover, let $\Sigma_C \cup \Sigma_K$ be $(\mathcal{C} \cup \mathcal{K})$-closed and $\Sigma_K \cup \Sigma_F$ be $(\mathcal{K} \cup \mathcal{F})$-closed. Consider a relationship type $\underline{r}$ and non-empty subsets $Y, W \subseteq Co(\underline{r})$. If Σ_F does not contain the functional dependency $\underline{r} : Y \to W$, then there is a population $\underline{r}^s$ which satisfies $\Sigma = \Sigma_C \cup \Sigma_K \cup \Sigma_F$, but violates $\underline{r} : Y \to W$.*

Proof. A population violates $\underline{r} : Y \to W$ if and only if it violates $\underline{r} : cl(Y) \to W$. Thus we may suppose that Y is functionally closed under Σ_F, that is, $Y = cl(Y)$ holds.

Let us consider the population $\underline{r}^s$ constructed in Lemma 49, which satisfies $\Sigma_C \cup \Sigma_K$ and violates the key dependency $\underline{r} : Y \to \underline{r}$. As we shall see below, this population is also suitable for our purposes here.

We have to check that $\underline{r}^s$ satisfies Σ_F. Of course, it suffices to discuss the non-unary functional dependencies in Σ_F. Let $\underline{r} : X \to Z$ be an arbitrary non-unary functional dependency in Σ_F. Note, that $X \subseteq Y$ gives us $Z \subseteq cl(X) \subseteq cl(Y) = Y$ and thus $X \cup Z \subseteq Y$. Therefore, the population $\underline{r}^s$ satisfies every non-unary functional dependency in Σ_F by Lemma 59.

Further, we shall verify that $\underline{r}^s$ violates $\underline{r} : Y \to W$. By Lemma 59, $\underline{r}^s$ violates $\underline{r} : Y \to W$ if and only if $W \not\subseteq Y$ holds. However, since Σ_F is $\mathcal{F}$-closed and $\underline{r} : Y \to W$ is not in Σ_F, we have $W \not\subseteq cl(Y) = Y$. This concludes the proof. □

From Lemma 60, we derive a nearly complete characterization for closed sets in the class of cardinality constraints, key and functional dependencies.

Theorem 61. *Let all functional dependencies in Σ_F be non-unary, trivial or implied by Σ_K. Then $\Sigma = \Sigma_C \cup \Sigma_K \cup \Sigma_F$ is $(\mathcal{C} \cup \mathcal{K} \cup \mathcal{F})$-closed if and only if $\Sigma_C \cup \Sigma_K$ is $(\mathcal{C} \cup \mathcal{K})$-closed and $\Sigma_K \cup \Sigma_F$ is $(\mathcal{K} \cup \mathcal{F})$-closed.*

Proof. The necessity is obvious. To prove the sufficiency we proceed as in Theorem 50.

(*Sufficiency.*) By Theorem 44, Σ is $\mathcal{C}$-closed under the specified conditions. It remains to check that Σ is also $\mathcal{K}$-closed and $\mathcal{F}$-closed. First, we show that Σ is $\mathcal{F}$-closed. Consider a functional dependency σ not in Σ, that is, not in Σ_F. We have to find a database instance which is legal for Σ, but violates σ. Similar to Theorem 50, we may construct such a database instance involving the population established in Lemma 60. This proves Σ to be $\mathcal{F}$-closed.

Next, we verify that Σ is $\mathcal{K}$-closed. Consider a relationship type $\underline{r}$ in the database schema. Suppose Σ does not contain any key dependency declared on $\underline{r}$, i.e. Σ_K is empty for $\underline{r}$. It suffices to find a database instance which is legal for Σ and violates $\underline{r} : \text{Co}(\underline{r}) \to \underline{r}$. Take the database instance $\vec{S}^{t''}$ generated in Theorem 50 with $Y = \text{Co}(\underline{r})$. It is legal for $\Sigma_C \cup \Sigma_K$ and violates the key dependency $\underline{r} : \text{Co}(\underline{r}) \to \underline{r}$ under discussion. In addition, it satisfies every non-unary functional dependency in Σ_F due to our construction and Lemma 59. Hence, this database instance is legal for the whole constraint set Σ. On the other hand, it violates $\underline{r} : \text{Co}(\underline{r}) \to \underline{r}$ and consequently every possible key dependency on $\underline{r}$.

Conversely, suppose Σ contains at least one key dependency declared on $\underline{r}$. Since Σ_K is $\mathcal{K}$-closed, it contains the key dependency $\underline{r} : \text{Co}(\underline{r}) \to \underline{r}$ due to (K). Under this presumption, the key dependency $\underline{r} : Y \to \underline{r}$ is semantically equivalent to the corresponding functional dependency $\underline{r} : Y \to \mathrm{Co}(\underline{r})$. Thus, for any key dependency $\underline{r} : Y \to \underline{r}$ not in Σ, we simply take the database instance established above which is legal for Σ and violates the functional dependency $\underline{r} : Y \to \underline{r}$ and thus the key dependency under inspection. This proves Σ to be $\mathcal{K}$-closed, too. $\square$

As in Section 11, we also obtain a condition on the existence of Armstrong databases. To verify this result, we proceed as is the proof of Theorem 53, but use the databases constructed in Theorem 53 and in Theorem 61.

Theorem 62. *Let all functional dependencies in Σ_F be non-unary. There exists a database instance over $\vec{S}$ which is $(\mathcal{C} \cup \mathcal{K} \cup \mathcal{F})$-Armstrong for Σ if and only if Σ implies a finite cardinality constraint for every link in $\vec{S}$.*

13 Unary functional dependencies

Until now, we usually excluded unary functional dependencies. In this section, we 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 via (CK) and (KF).

Lemma 63. *Let $\underline{r}$ be a relationship type, and $\underline{v}, \underline{w}$ be components of $\underline{r}$. Σ_C implies a functional dependency $\underline{r} : \{\underline{v}\} \to \{\underline{w}\}$ if and only if Σ_C implies $card(\underline{r}, \underline{v}) = \{0, 1\}$.*

Proof. (*Necessity.*) Suppose Σ_C does not imply $card(\underline{r}, \underline{v}) = \{0, 1\}$. We have to check that Σ_C does not imply $\underline{r} : \{\underline{v}\} \to \{\underline{w}\}$ either. We do this by constructing suitable representation graphs for $\underline{v}$ and $\underline{w}$. Note that the representation graph $G^t(\underline{r}, \underline{v})$ may contain cliques of size 2 or larger. Thus it is not forced to be edgeless. We first choose $G^t(\underline{r}, \underline{w})$ with at least two cliques. Afterwards, we choose $G^t(\underline{r}, \underline{v})$ such that it contains an edge which does not occur in $G^t(\underline{r}, \underline{w})$. Then the functional dependency $\underline{r} : \{\underline{v}\} \to \{\underline{w}\}$ does not hold. Hence Σ_C does not imply this dependency.

(*Sufficiency.*) Let Σ_C imply $card(\underline{r}, \underline{v}) = \{0, 1\}$. Then $\{\underline{v}\}$ is key, and the claimed functional dependency holds due to (CK) and (KF). $\square$

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

Example. Consider the schemas in Figure 7. The specified sets of cardinality constraints are both consistent. Assume, we are given the functional dependency $\underline{r} : \{\underline{v}\} \to \{\underline{w}\}$, too. In every legal database instance, the representation graph $G^t(\underline{r}, \underline{v})$ has to be a subgraph of $G^t(\underline{r}, \underline{w})$.

For the first schema, the cliques of $G^t(\underline{r}, \underline{v})$ are complete graphs on 4 to 6 vertices, while the cliques of $G^t(\underline{r}, \underline{w})$ are complete graphs on 5 to 7 vertices. The only

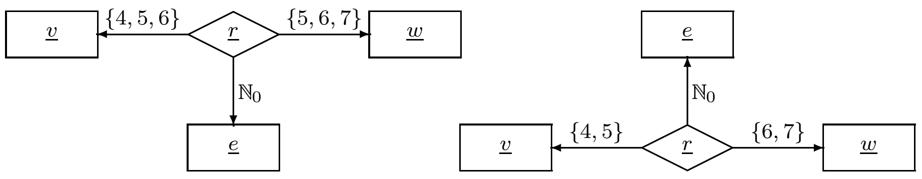

Figure 7: Examples for interactions with a functional dependency $\underline{r} : \{\underline{v}\} \to \{\underline{w}\}$.

possible decomposition of cliques of $G^t(\underline{r}, \underline{w})$ into those of $G^t(\underline{r}, \underline{v})$ is the trivial one. Hence, both representation graphs are isomorphic. This gives us the new constraints $\underline{r} : \{\underline{w}\} \to \{\underline{v}\}$ as well as $card(\underline{r}, \underline{v}) = \{5, 6\}$ and $card(\underline{r}, \underline{w}) = \{5, 6\}$.

For the second schema in Figure 7, the situation is even worse. A similar argumentation shows the specified set of constraints to be inconsistent: The object types $\underline{r}$, $\underline{v}$ and $\underline{w}$ are redundant for this constraint set.

The following result summarizes the observations indicated by the examples. A rigorous exploitation of the argumentation gives further results. But this is out of the scope of the present paper.

Lemma 64. *Let $\underline{r}$ be a relationship type, and $\underline{v}, \underline{w}$ be components of $\underline{r}$. Then we have:*

(CUF1) $\Sigma_C \cup \{\underline{r} : \{\underline{v}\} \to \{\underline{w}\}\}$ *implies* $card(\underline{r}, \underline{v}) = \{z \in \mathbb{N}_0 : z \leq \beta(\underline{r}, \underline{w})\}$.

(CUF2) $\Sigma_C \cup \{\underline{r} : \{\underline{v}\} \to \{\underline{w}\}\}$ *implies* $card(\underline{r}, \underline{w}) = \{z \in \mathbb{N}_0 : z \geq \alpha(\underline{r}, \underline{v})\}$.

(CUF3) $\Sigma_C \cup \{\underline{r} : \{\underline{v}\} \to \{\underline{w}\}\}$ *implies* $\underline{r} : \{\underline{v}\} \to \{\underline{w}\}$ *whenever* $\beta(\underline{r}, \underline{w}) < 2\alpha(\underline{r}, \underline{v})$.

Proof. To verify the claimed statements, we consider the representation graphs $G^t(\underline{r}, \underline{v})$ and $G^t(\underline{r}, \underline{w})$ for a population $\underline{r}^t$ which satisfies $\Sigma_C \cup \{\underline{r} : \{\underline{v}\} \to \{\underline{w}\}\}$. Due to the functional dependency $\underline{r} : \{\underline{v}\} \to \{\underline{w}\}$, the first one will always be a subgraph of the second one. In $G^t(\underline{r}, \underline{v})$ all components are of size at least $\alpha(\underline{r}, \underline{v})$, while in $G^t(\underline{r}, \underline{w})$ all components are of size at most $\beta(\underline{r}, \underline{w})$. This motivates the first two implications. In the third case, it is not difficult to see, that both graphs have to be isomorphic. This gives the unary functional dependency $\underline{r} : \{\underline{v}\} \to \{\underline{w}\}$, too. □

An interesting idea to handle unary functional dependencies was introduced by Biskup et al. [10]. They suggested to use these constraints to decompose relationship types by *pivoting*. Though pivoting was originally introduced in an object-oriented data model, it is not difficult to translate this idea to the entity-relationship model. In our example from Section 3, the unary functional dependency LECTURE:{HALL} → {BUILDING} may be used for this approach. As a result we would obtain the schema in Figure 8. In the transformed database schema, this functional dependency is naturally enforced due to the structure of the schema.

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 if every

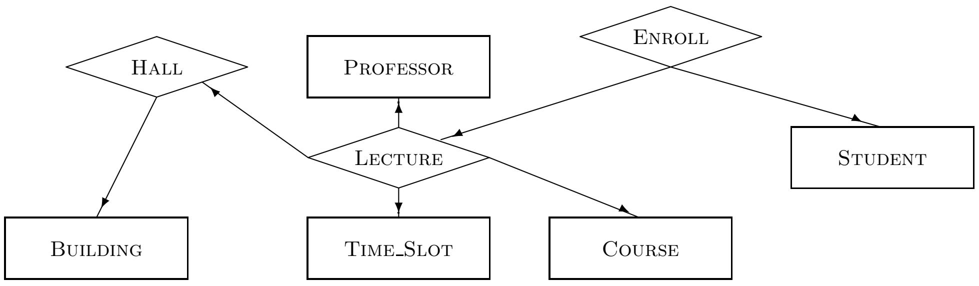

Figure 8: The transformed database schema after decomposition by pivoting.

functional dependency $\underline{r} : X \to Y$ in Σ is trivial or Σ contains the key dependency $\underline{r} : X \to \underline{r}$, too. Applying Theorem 61 and Corollary 59, we conclude as follows.

Theorem 65. *Let every relationship type $\underline{r}$ in the database schema $\vec{S}$ be in BCNF with respect to $\Sigma = \Sigma_C \cup \Sigma_K \cup \Sigma_F$.*

(i) Σ is consistent if and only if $\Sigma_C \cup \Sigma_K$ is consistent.

(ii) Σ is $(\mathcal{C} \cup \mathcal{K} \cup \mathcal{F})$-closed if and only if $\Sigma_C \cup \Sigma_K$ is $(\mathcal{C} \cup \mathcal{K})$-closed and $\Sigma_K \cup \Sigma_F$ is $(\mathcal{K} \cup \mathcal{F})$-closed.

Recall that $(\mathcal{C} \cup \mathcal{K})$- and $(\mathcal{K} \cup \mathcal{F})$-closures can be determined applying Algorithm 52 and the Armstrong rules, respectively. This gives us an efficient procedure to check implication under the assumption of BCNF.

14 Concluding remarks and open problems

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 schemas, or to detect conflicts among the constraints under discussion.

There are at least three open 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 profitable to have a characterization of semantically closed sets in the presence of unary dependencies without additional assumptions such as BCNF.

As mentioned, pivoting may be used to decompose relationship types in order to naturally enforce unary functional dependencies. Here a second problem arises. So far, pivoting has not been studied in the presence of cardinality constraints. We are particularly interested in the question which cardinality constraints declared on the original database schema may be translated to the transformed schema after pivoting.

The third problem concerns Armstrong databases. The Armstrong databases constructed in the present paper are rather huge. In order to use Armstrong databases for database mining, they have to be of reasonable size. Thus we are looking for Armstrong databases of minimum size.

For key and functional dependencies this has been done in [21]. Even though far from being solved this problem gave rise to a considerable number of partial results. We refer the interested reader to [6, 7, 28, 52] for various research papers. In [23], Demetrovics, Katona and Sali give a survey on recent results and show how this question leads to the design-theoretic concept of *orthogonal double covers (ODCs).* The investigation of ODCs turned out to be of interest for its own sake, as well. For details, we refer to [1, 31, 32, 36] and a forthcoming survey paper by Gronau et al. [29].

For cardinality constraints a similar problem was asked in [35]. Unfortunately, it turned out to be NP-complete to decide whether there exists a legal database whose population sizes are bounded by a given integer.

References

[1] B. Alspach, K. Heinrich, and G. Liu. Orthogonal factorizations of graphs. In Dinitz and Stinson [26], chapter 2.

[2] W. W. Armstrong. Dependency structures of database relationship. *Inform. Process.*, 74:580–583, 1974.

[3] P. Atzeni and V. D. Antonelles. *Relational database theory.* Benjamin/Cummings, Redwood, 1993.

[4] C. Batini, S. Ceri, and S. Navathe, editors. *Conceptual database design.* Benjamin/Cummings, Redwood, 1992.

[5] C. Beeri and B. Thalheim. Identification as a primitive of database models. In T. Polle et al., editor, *Fundamentals of information systems*, pages 19–36. Kluwer, Dordrecht, 1999.

[6] F. E. Bennett and L. Wu. On minimum matrix representation of closure operations. *Discrete Appl. Math.*, 26:25–40, 1990.

[7] F. E. Bennett and L. Wu. On minimum matrix representation of Sperner systems. *Discrete Appl. Math.*, 81:9–17, 1998.

[8] T. Beth, D. Jungnickel, and H. Lenz. *Design Theory.* BI, Mannheim, 1985.

[9] A. Binemann-Zdanowicz. Systematization of approaches to equality-generating constraints. In J. Stuller et al., editor, *Current issues in databases and information systems*, volume 1884 of *LNCS*, pages 307–314. Springer, Berlin, 2000.

[10] J. Biskup, R. Menzel, T. Polle, and Y. Sagiv. Decomposition of relationships through pivoting. In B. Thalheim, editor, *Conceptual modeling*, volume 823 of *LNCS*, pages 28–41. Springer, Berlin, 1996.

[11] B. Bollobás. *Extremal graph theory.* Academic Press, London, 1978.

[12] G. Burosch, J. Demetrovics, and G. O. H. Katona. The poset of closures as a model of changing databases. *Order*, 4:127–142, 1987.

[13] D. Calvanese and M. Lenzerini. Making object-oriented schemas more expressive. In *Proc. Thirteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, pages 243–254. ACM Press, Minneapolis, 1994.

[14] P. P. Chen. The entity-relationship model: towards a unified view of data. *ACM Trans. Database Systems*, 1:9–36, 1976.

[15] P. P. Chen, B. Thalheim, and L. Y. Wong, editors. *Future Directions of conceptual modeling*, volume 1565 of *LNCS*. Springer, Berlin, 1996.

[16] S. Chowla, P. Erdős, and E. G. Straus. On the maximal number of pairwise orthogonal Latin squares of a given order. *Canad. J. Math.*, 12:204–208, 1960.

[17] E. F. Codd. A relation model of data for large shared data banks. *Comm. ACM*, 13:377–387, 1970.

[18] C. J. Colbourn and J. H. Dinitz, editors. *The CRC handbook of combinatorial designs.* CRC press, Boca Raton, 1996.

[19] C. Delobel and R. Casey. Decompositions of a database and the theory of Boolean switching functions. *IBM Research Dev.*, 17:374–386, 1973.

[20] J. Demetrovics. On the equivalence of candidate keys and Sperner systems. *Acta Cybernet.*, 4:247–252, 1979.

[21] J. Demetrovics, Z. Füredi, and G. O. H. Katona. Minimum matrix representations of closure operations. *Discrete Appl. Math.*, 11:115–128, 1985.

[22] J. Demetrovics and G. O. H. Katona. A survey on some combinatorial problems concerning functional dependencies in relational databases. *Ann. Math. Artificial Intelligence*, 7:63–82, 1993.

[23] J. Demetrovics, G. O. H. Katona, and A. Sali. Design type problems motivated by database theory. *J. Statist. Plann. Inference*, 72:149–164, 1998.

[24] J. Demetrovics and V. Thi. Relations and minimal keys. *Acta Cybernet.*, 8:279–285, 1988.

[25] J. Demetrovics and V. Thi. Some results about functional dependencies. *Acta Cybernet.*, 8:273–278, 1988.

[26] J. H. Dinitz and D. R. Stinson, editors. *Contempory design theory.* Wiley, New York, 1992.

[27] P. Erdős. Extremal problems in graph theory. In F. Harary, editor, *A seminar in graph theory*, pages 54–64. Holt, Rinehart and Winston, 1967.

[28] B. Ganter and H.-D. O. F. Gronau. On two conjectures of Demetrovics, Füredi and Katona on partitions. *Discrete Math.*, 88:149–155, 1991.

[29] H.-D. O. F. Gronau, M. Grüttmüller, S. Hartmann, U. Leck, and V. Leck. On orthogonal double covers of graphs. Manuscript, 2001, submitted.

[30] H.-D. O. F. Gronau and R. C. Mullin. On a conjecture of Demetrovics, Füredi and Katona. Unpublished manuscript, 1990.

[31] H.-D. O. F. Gronau, R. C. Mullin, and A. Rosa. Orthogonal double covers of complete graphs by trees. *Graphs Combin.*, 13:251–262, 1997.

[32] H.-D. O. F. Gronau, R. C. Mullin, and P. J. Schellenberg. On orthogonal double covers of K_n and a conjecture of Chung and West. *J. Combin. Des.*, 3:213–231, 1995.

[33] A. Hajnal and E. Szemerédi. Proof of a conjecture of Erdős. In P. Erdős, A. Renyi, and V. T. Sós, editors, *Combinatorial theory and its applications*, volume 4 of *Colloq. J. Bolyai Math. Soc.*, pages 601–623. North-Holland, Amsterdam, 1970.

[34] S. Hartmann. Graphtheoretic methods to construct entity-relationship databases. In M. Nagl, editor, *Graphtheoretic concepts in computer science*, volume 1017 of *LNCS*, pages 131–145. Springer, Berlin, 1995.

[35] S. Hartmann. On the consistency of int-cardinality constraints. In T. Ling, S. Ram, and M. Li, editors, *Conceptual Modeling*, volume 1507 of *LNCS*, pages 150–163. Springer, Berlin, 1998.

[36] S. Hartmann. Orthogonal decompositions of complete digraphs. *Graphs Combin.*, 1999, to appear.

[37] D. Jungnickel. *Graphen, Netzwerke und Algorithmen.* BI, Mannheim, 1994.

[38] M. Lenzerini and P. Nobili. On the satisfiability of dependency constraints in entity-relationship schemata. *Inform. Systems*, 15:453–461, 1990.

[39] S. W. Liddle, D. W. Embley, and S. N. Woodfield. Cardinality constraints in semantic data models. *Data Knowledge Eng.*, 11:235–270, 1993.

[40] H. F. MacNeish. Euler's squares. *Ann. of Math.*, 23:221–227, 1922.

[41] H. Mannila and K. Räihä. Design by example: an application of Armstrong relations. *J. Comput. Syst. Sci.*, 33:126–141, 1986.

[42] H. Mannila and K. Räihä. *The design of relational databases.* Addison-Wesley, Reading, 1992.

[43] A. McAllister. Complete rules for n-ary relationship cardinality constraints. *Data Knowledge Eng.*, 27:255–288, 1998.

[44] H. Noltemeier. *Graphentheorie.* de Gruyter, Berlin, 1976.

[45] A. Schrijver. *Theory of linear and integer programming.* Wiley, Chichecter, 1986.

[46] T. Takaoka. Subcubic cost algorithms for the all pairs shortest path problem. *Algorithmica*, 20:309–318, 1998.

[47] B. Thalheim. *Dependencies in relational databases.* Teubner, Stuttgart, 1991.

[48] B. Thalheim. Foundations of entity-relationship modeling. *Ann. Math. Artificial Intelligence*, 6:197–256, 1992.

[49] B. Thalheim. Fundamentals of cardinality constraints. In G. Pernul and A. M. Tjoa, editors, *Entity-relationship approach*, volume 645 of *LNCS*, pages 7–23. Springer, Berlin, 1992.

[50] B. Thalheim. *Entity-relationship modeling.* Springer, Berlin, 2000.

[51] D. Theodorates. Deductive object oriented schemas. In B. Thalheim, editor, *Conceptual modeling*, volume 1157 of *LNCS*, pages 58–72. Springer, Berlin, 1996.

[52] K. Tichler. Minimum matrix representation of some key system. In K.-D. Schewe and B. Thalheim, editors, *Foundations of information and knowledge systems*, volume 1762 of *LNCS*, pages 275–287. Springer, Berlin, 2000.