Reasoning about participation constraints and Chen's constraints

Sven Hartmann

Information Science Research Centre
Massey University
Palmerston North, New Zealand
s.hartmann@massey.ac.nz

Abstract

Cardinality constraints are often considered as one of the basic constituents of the entity-relationship approach to database design. In his original proposal of this model, Chen [6] defined cardinality constraints as look-across constraints. Alternatively, however, cardinality constraints may also be defined on the basis of the participation or look-here interpretation.

While both definitions correspond to each other for binary relationships, they differ for n-ary relationships (with $n \geq 3$). Participation constraints restrict the number of relationships a fixed object may participate in. Chen-style constraints limit the number of objects that co-occur with a given tuple comprising instances of the remaining $n-1$ components of the relationship type under discussion.

In our paper we present a sound and complete system of inference rules for a class of generalized cardinality constraints containing both, participation constraints and Chen-style constraints. It turns out that both constraint classes are almost independent, which justifies their juxtaposition in conceptual database design. Similar results will be presented in the presence of additional functional dependencies. The paper concludes with an axiomatization for the joint class of generalized cardinality constraints and functional dependencies.

Keywords: cardinality constraint, ER modelling, functional dependency, axiomatization

1 Introduction

The entity-relationship model is still today the most popular approach towards conceptual modelling. Its concepts are easy to understand and sufficiently powerful to model real-world problems. Due to its popularity, the ER model is nowadays taught in almost all university classes on data modelling. The ongoing work on the ER model has led to a huge amount of literature, both research articles and textbooks.

Cardinality constraints are commonly considered as one of the basic constituents of the entity-relationship model and its extensions. A cardinality constraint is a restriction that bounds the number of elements in a set. For example, if A is a set of persons and B a set of teams, we may require that every team in B consists of at most 11 players and that every person plays for at most one team. The entity-relationship model is often acclaimed for its simple and comprehensible graphical representation. In the ER diagram, see Figure 1, there are two possible ways to represent the mentioned requirements: either we label the link between PlaysFor and Person by 11, and the link between PlaysFor and Team by 1, or we do it just the other way round.

Figure 1: A binary relationship type.

When it comes to drawing the ER diagram there is an extraordinary amount of disagreement among data modelers. Ferg [10] who was the first to point out this problem called these two approaches *Chen notation* and *Merise notation*, respectively. Song, Evans and Park [21] called them *look-across* and *look-here* approach, respectively. Depending on which approach we prefer, the diagram in Figure 2 tells us that every Person owns up to 4 houses and every house has only one owner (this is look-across approach) or that every Person owns at most one house and every house can have up to 4 owners (this is look-here approach).

Figure 2: A binary relationship type with labels for cardinality constraints.

Both approaches are frequently used in literature and taught in university courses. The look-across approach goes back to Chen [6] and is used e.g. in [3, 17, 20, 23, 27]. The look-here approach goes back to the French data modelling technique Merise [19] and is used e.g. in [2, 4, 8, 9, 15, 25, 28]. Surprisingly almost nobody is very disturbed by this problem - even though it is a permanent source of confusion. And in fact, for binary relationship types the difference between these two approaches is simply a matter of style. Both approaches can be used to express pretty much the same semantic information, and both representations can easily be translated to each other by exchanging the labels at the links.

For ternary relationship types, however, the problem becomes more crucial. Suppose we want to illustrate in Figure 3 that every professor teaches at most 2 courses per weekday. If we follow the look-across approach we shall label the link between Teach and Course by 2. But what should we do if we decide to

use the look-here approach? On the other hand, suppose we want to illustrate that every course is taught at most 5 times. If we use the look-here approach we shall label the link between TEACH and COURSE by 5. But what should we do if we are going to follow the look-across approach? And even worse, which approach shall we use if we are to illustrate both constraints simultaneously?

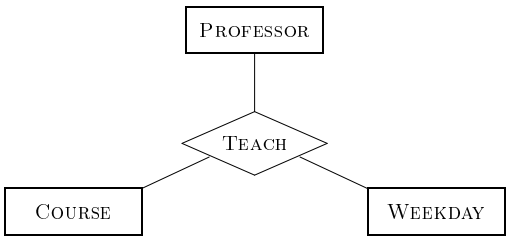

Figure 3: A ternary relationship type.

As pointed out by some authors [10, 18, 24] none of the two approaches may be replaced by the other one since the semantic constraints behind them may not be expressed by each other. Nevertheless, most of the standard textbooks on ER modelling still seem to neglect this observation and - in order not to confront readers with that uncomfortable situation - present only one of the possible approaches. This solution, however, falls far short with respect to practical challenges in data modelling.

The objective of this paper is to further discuss the interaction between the two kinds of cardinalities behind the two different labelling approaches. We are going to present a sound and complete system of inference rules for the two kinds of cardinality constraints under discussion. Note that we are not concerned about the graphical representation of these constraints but about their expressive power.

We present our results for a class of generalized cardinality constraints introduced in [10, 16, 24] which comprise the constraints behind both approaches, look-here and look-across. In addition, we also study interactions between generalized cardinality constraints and functional dependencies which are without any doubt the most popular constraint class in data modelling. The main result of this paper is a sound and complete system of inference rules for the joint class of functional dependencies and generalized cardinality constraints.

2 Popular concepts for cardinality constraints

The entity-relationship approach to conceptual design, first introduced by Chen [6], considers the target of a database as consisting of entities and relationships. To begin with, we 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 an excellent survey on entity-relationship modelling, we refer to [25].

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 modelled by a type E form an *object set* E^t. Its members are said to be *objects* or *instances* of type E. Entities are instances of entity types, while relationships are instances of relationship types.

Designing a database usually starts with declaring entity types to model the basic real-world objects in the target of the database. Afterwards, relationship types may be specified hierarchically via aggregation. A *relationship type* R is characterized by its component set $Co(R)$, its attribute set $Attr(R)$, and its primary key $id(R)$. Herein, $Co(R)$ consists of all the object types $C_1, \ldots, C_n$ involved in R. Roughly speaking, R reflects real-world connections between objects of the types $C_1, \ldots, C_n$. A relationship type with n components is said to be *n-ary*. Relationship types of arity 2 or 3 are, in particular, called *binary* and *ternary*. Moreover, a relationship type R may possess additional attributes. $Attr(R)$ denotes the set of attributes $A_1, \ldots, A_m$ used to further describe the relationship type R. Each of the attributes A_j has its domain $dom(A_j)$. The *primary key* $id(R)$ of R is a non-empty subset of $Co(R) \cup Attr(R)$.

Sometimes it is convenient to allow an object type to occur several times as a component of the same relationship type. In this case, *roles* are associated with the different occurrences. Though not always explicitly set out, all our considerations apply to this case, too.

Given an object set C_i^t for every component $C_i \in Co(R)$, a *relationship* of type R is an element of the cartesian product $C_1^t \times \cdots \times C_n^t \times dom(A_1) \times \cdots \times dom(A_m)$. Hence, every relationship r of type R assigns an object $r(C_i) \in C_i^t$ to each component $C_i \in Co(R)$, and a value $r(A_j) \in dom(A_j)$ to each attribute $A_j \in Attr(R)$, respectively.

A *population* R^t is a set of relationships of type R such that for any two relationships r and r' in R^t their projections $r|_{id(R)}$ and $r'|_{id(R)}$ from $Co(R) \cup Attr(R)$ to the primary key $id(R)$ are distinct. The number of relationships in a population is the *size* of the population. Informally, a population R^t may be regarded as a table whose columns correspond to the components and attributes of R and whose rows correspond to the relationships in R^t.

All object types declared for some application, collectively, form a *database schema*. A major advantage of the entity-relationship approach is its ability to provide a simple graphical representation of a database schema. The *ER diagram* of a database schema is directed graph whose vertices represent the object types in the database schema. As usual, vertices for the entity types are drawn as rectangles and vertices for the relationship types as diamonds. Two vertices R and C are connected by an arc from R to C whenever C is a component of R. The arcs in the ER diagram are also called *links*.

In database design great attention is devoted to the modelling of semantics. Central to this idea is the notion of integrity constraints. Defining and enforcing integrity constraints helps to guarantee that the database correctly reflects the underlying domain of interest. Cardinality constraints are among the most popular integrity constraints used in the entity-relationship model. They were introduced in Chen's original proposal of the ER model [6], and have been frequently used in conceptual design since then. In

fact, Chen himself did not use the term 'cardinality', but used the expressions '1:1 mapping', '1:N mapping' and 'M:N mapping' for binary relationship types with given cardinality constraints and explained each of them.

Later on, many authors formalized and used the Chen style of cardinalities: Let R be a relationship type with component set $Co(R) = \{C_1, \ldots, C_n\}$, and let C_k be one of its components. A *Chen-style constraint* is an expression $card^{Chen}(R, C_k) = b$ with $b \in \mathbb{N}^\infty$, that is, b is a non-negative integer or ∞. This constraint holds in a population R^t if for every choice of objects $c_i \in C_i^t$ with $i \neq k$ there are at most b relationships in R^t such that $r(C_i) = c_i$ for all $i \neq k$. In the ER diagram, the Chen-style constraint $card^{Chen}(R, C_k) = b$ is graphically reflected by labelling the link between R and C_k with the value b, that is, uses the look-across approach.

As an example consider the Chen-style constraint $card^{Chen}(\textsc{Teach}, \textsc{Course}) = 2$ declared on the relationship type in Figure 3. It claims that at most two courses are taught by a certain professor on a certain weekday. Using the look-across approach, this constraint can be illustrated by labelling the link between Teach and Course with a 2.

In his paper [6], Chen also gave an example of an M:N:P ternary relationship with components Supplier, Project, Part, but did not explain how these cardinalities are to be understood. The authors of the ER-like modelling technique Merise [19] proposed an interpretation different from the one mentioned above: A *participation constraint* is an expression $card^{part}(R, C_k) = b$ with $b \in \mathbb{N}^\infty$. This constraint holds in a population R^t if for every object $c_k \in C_k^t$ there are at most b relationships in R^t satisfying $r(C_k) = c_k$. In the ER diagram, the participation constraint $card^{part}(R, C_k) = b$ is graphically reflected by labelling the link between R and C_k with the value b, that is, uses the look-here approach.

For example, the participation constraint $card^{part}(\textsc{Teach}, \textsc{Course}) = 2$ tells us that every course is taught at most twice. Using the look-here approach, this constraint can be illustrated again by labelling the link between Teach and Course with a 2.

For binary relationship types, Chen-style constraints may easily be translated to participation constraints and vice versa: If R has two components C_1 and C_2, then $card^{Chen}(R, C_1) = b$ corresponds to $card^{part}(R, C_2) = b$, and $card^{Chen}(R, C_2) = b'$ corresponds to $card^{part}(R, C_1) = b'$. The only difference is the placement of the labels in the ER diagram. So there is no need to use both kind of constraints simultaneously in a database schema.

In general, however, the difference between the two definitions is more substantial: A Chen-style constraint fixes instances of $n - 1$ components and bounds the number of relationships in the population R^t involving this fixture, while a participation constraint fixes only an instance of a single component and bounds the number of relationships in R^t this fixed object participates in.

This indicates that both kinds of cardinality constraints may be used in conceptual modelling for their own right - as soon as we have an n-ary relationship type with $n \geq 3$. Unfortunately, this matter of fact is widely ignored in textbooks on conceptual modelling. The reason for this is simple: If we want to use Chen-style constraints and participation constraints simultaneously, it becomes more difficult to illustrate them in the diagram. Neither the look-here nor the look-across approach are suitable to illustrate both kinds of constraints. Suppose we specified the constraints $card^{Chen}(\textsc{Teach}, \textsc{Course}) = 2$ and $card^{part}(\textsc{Teach}, \textsc{Course}) = 5$ and want to illustrate them in the diagram. The traditional way is to label the link between the relationship type Teach and its component Course once with a 2 and once with a 5 - which will obviously cause some irritation. The adequate solution for this problem would be a slightly refined method to illustrate constraints in the diagram. Instead, however, most textbooks either suggest not to use n-ary relationship types at all or restrict themselves to one of the two kinds of cardinality constraints.

3 On n-ary relationship types

One way to overcome the uncomfortable situation under discussion is to restrict entity-relationship modelling to binary relationship types. For example, the authors of the Unified Modeling Language (UML) proposed to avoid non-binary associations since the definition of cardinalities is somehow complicated for n-ary relationship types [20].

In our opinion, however, this proposal is not applicable in general. First of all, n-ary relationship types appear quite naturally in conceptual modelling. Potential users of information systems usually express their system requirements in natural language. The abstraction of the requirements specification results in a first version of a database schema which reflects the most important concepts. This first schema is the basis for further communication with the users and for step-by-step refinement.

Requirements engineering provides best results if users are allowed to express their application knowledge and demands in a language they are familiar with. Chen [7] pointed out that English sentences can be straightforward transferred to relationship types. Empirical studies provide similar results for other languages, see [26]. Consider the observation 'John teaches a Java course on Tuesday', or more generally 'A professor teaches a course on a certain weekday'. This can easily be modelled by a ternary relationship type Teach with components Professor, Course and Weekday as shown in Figure 3.

During system analysis the designer must extract conceptual knowledge from requirements given in natural language, that is, descriptions in natural language are transformed into formal descriptions of data and conditions on them. Due to the complexity of natural language sentences, the transformation often results in n-ary relationship types with $n \geq 3$. This observation holds in particular if the designer is assisted by automated design tools, which accept natural language input. For a discussion of this issue, see [26]. Examples for n-ary relationship types occurring in conceptual design are given e.g. in [8, 23, 25]. It should be mentioned, that n-ary relationship types also appear during reverse engineering [22].

Of course, relationship types derived from natural language sentences do not always respect database requirements. In practice, they are sometimes too large or carry along complicated dependencies. These problems should be fixed in later design steps by de-

composing relationship types. This approach usually increases the readability of the schema, reduces potential redundancy and supports constraint enforcement. However, splitting relationship types must not be carried too far. The new structures should be more acceptable than, but at least equivalent to the original ones. Basic requirements are information preservation and constraint preservation.

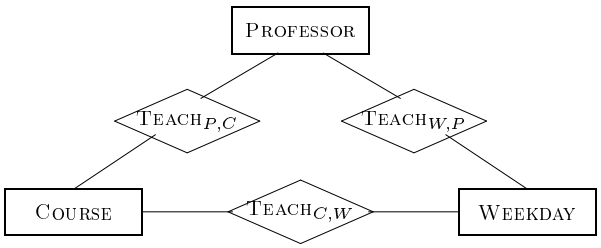

Figure 4: A possible decomposition of the ternary relationship type Teach into three binary relationship types.

As an example consider the ternary relationship type Teach in Figure 3. Figure 4 shows a standard decomposition of this relationship type into three binary ones. A decomposition of an n-ary relationship type R is *lossless* if every original population R^t over R may be rebuild from the corresponding populations over the new binary relationship types, without missing or spurious relationships. Hence, it should be possible to rebuild the population Teach^t in Figure 5 from its projections to the three two-element subsets {Professor,Course}, {Professor, Weekday} and {Course, Weekday} of the component set of Teach, see Figure 6.

Professor	Course	Weekday
John	Java	Tu
John	C++	Tu
Mary	Java	Tu
Mary	Java	Fr
Mary	C++	Fr

Figure 5: A population of the ternary relationship type Teach.

Unfortunately, the decomposition under inspection is not lossless as the tuple (Mary, C++, Tu) is in the join of the populations over the three binary relationship types, but does not occur in the original population Teach^t.

Professor	Course
John	Java
John	C++
Mary	Java
Mary	C++

Course	Weekday
Java	Tu
Java	Fr
C++	Tu
C++	Fr

Weekday	Professor
Tu	John
Tu	Mary
Fr	Mary

Figure 6: The three projections of the population in Figure 5 to the two-element subsets of the component set of Teach.

The question whether a lossless decomposition is possible or not depends on the integrity constraints specified for the relationship type to be decomposed. In the relational data model, this question has been widely studied e.g. for functional or multivalued dependencies. In the entity-relationship model, the question has been discussed in [14, 13, 18] for cardinality constraints. Unfortunately, these investigations concentrate only on ternary relationship types and none of them covers all possible cases, as pointed out in [5, 25].

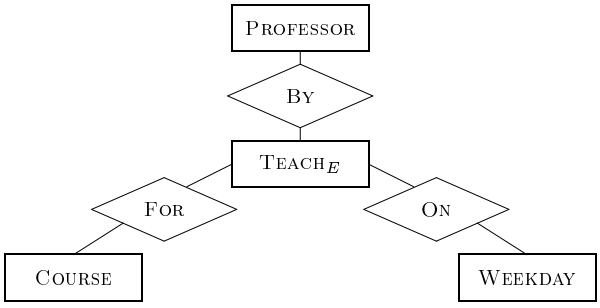

Figure 7: A possible replacement of the ternary relationship type by an entity type.

Another possible way to replace an n-ary relationship type by binary ones is illustrated in Figure 7. This time, an n-ary relationship type R is transformed into an entity type R_E and linked to its former components via n new binary relationship types.

Teach$_E$	Professor
Teach$_1$	John
Teach$_2$	John
Teach$_3$	Mary
Teach$_4$	Mary
Teach$_5$	Mary

Teach$_E$	Course
Teach$_1$	Java
Teach$_2$	C++
Teach$_3$	Java
Teach$_4$	Java
Teach$_5$	C++

Teach$_E$	Weekday
Teach$_1$	Tu
Teach$_2$	Tu
Teach$_3$	Tu
Teach$_4$	Fr
Teach$_5$	Fr

Figure 8: The three populations over the binary relationship types in Figure 7 corresponding to the population in Figure 5.

In practice it is rather easy to transform the original population Teach^t to populations of the new binary relationship types: For every relationship in Teach^t we create a new entity of type Teach$_E$ and insert a new relationship into each of the three new populations. The retransformation, however, is only possible if each entity of type Teach$_E$ participates exactly once in each of three new populations. Note that this condition may be expressed by Chen-style constraints or by participation constraints declared on the new binary relationship types together with additional existence constraints. Moreover, the retransformed relationships have to be mutually distinct on the primary key of the n-ary relationship type. It is not difficult to see that the decomposition under inspection is lossless only if these two conditions is satisfied.

Unfortunately, replacing the ternary relationship type has some disadvantages, too. First of all, it dramatically increases the size of the schema: before the decomposition we had three entity types and one relationship type, afterwards we have four entity types and three relationship types. Moreover, we have to translate cardinality constraints declared on the original relationship type to cardinality constraints de-

clared on the new types. For a participation constraint like $card^{part}(\text{TEACH}, \text{COURSE}) = 2$ this is fairly easy as it corresponds to the participation constraint $card^{part}(\text{FOR}, \text{COURSE}) = 2$ declared on one of the new binary types.

For a Chen-style constraint such as $card^{Chen}(\text{TEACH}, \text{COURSE}) = 2$ the situation is more complicated. Of course, the constraint 'Every professor teaches at most two courses per weekday' should be specified for the transformed schema, too. But this is no longer possible with the help of one of the traditional definitions of cardinality constraints. Rather, we have to define a new kind of 'inter-relationship' cardinality constraints which is more difficult to understand and does not allow a simple graphical representation in the ER diagram.

This example points out the major consequence of forbidding n-ary relationship types: ER diagrams become larger and less understandable for potential users of the database system to be designed. Moreover, decomposing n-ary relationship types into binary ones does not solve any of the problems arising from the simultaneous usage of Chen-style constraints and participation constraints as discussed above.

Observation 1. *n-ary relationship types with $n \geq 3$ are useful in conceptual modelling.*

4 Participation constraints or Chen's constraints

The standard way to avoid the discussion about the graphical representation of cardinality constraints is to introduce only one of the two kinds of cardinality constraints under inspection. Often this is done in the unspoken concensus that both kinds of constraints may be used to express pretty much the same semantic information. Unfortunately, this is not true. As an example, consider the populations in Figures 9–11. They show that both kinds of constraints may hold independently from each other.

PROFESSOR	COURSE	WEEKDAY
John	Java	Tu
John	C++	We
Mary	Java	Tu
Susan	Java	Tu

PROFESSOR	COURSE	WEEKDAY
John	Java	Tu
John	C++	Tu
John	Java	Mo
Mary	Java	Tu

Figure 9: The first population satisfies $card^{part}(\text{TEACH}, \text{PROFESSOR}) = 2$, but violates $card^{Chen}(\text{TEACH}, \text{PROFESSOR}) = 2$. For the second population, the situation is vice versa.

Of course, there is some interaction between both kinds of constraints. For example, if $card^{part}(\text{TEACH}, \text{PROFESSOR}) = 2$ holds in a population TEACH^t then $card^{Chen}(\text{TEACH}, \text{COURSE}) = 2$ holds, too. This is not difficult to see: If a fixed professor teaches at most 2 courses in total, then she/he will teach no more than 2 courses on a fixed weekday.

Observation 2. *Let R be a relationship type with components $C_1, \ldots, C_n$, and let $i, k \in \{1, \ldots, n\}$. If a population R^t satisfies the participation constraint $card^{part}(R, C_k) = b$ then it also satisfies the Chen-style constraint $card^{Chen}(R, C_i) = b$ for every $i \neq k$.*

For binary relationship types, the converse is also true, but not for n-ary relationship types with $n \geq 3$. As an example, consider the populations in Figure 10.

PROFESSOR	COURSE	WEEKDAY
John	Java	Tu
John	C++	Tu
John	Java	We
Mary	C++	Fr

PROFESSOR	COURSE	WEEKDAY
John	Java	Tu
John	C++	Tu
Mary	Java	We
Mary	C++	Fr

Figure 10: The Chen-style constraint $card^{Chen}(\text{TEACH}, \text{COURSE}) = 2$ holds in both populations. Moreover, the first population violates $card^{part}(\text{TEACH}, \text{PROFESSOR}) = 2$, while the second population satisfies this participation constraint.

Further, we should not expect a participation constraint like $card^{part}(\text{TEACH}, \text{PROFESSOR}) = 2$ to imply any Chen-style constraint which is 'sharper' than $card^{Chen}(\text{TEACH}, \text{COURSE}) = 2$. If a fixed professor may teach up to 2 courses in total, then there is no reason why she/he should not teach both courses on the same weekday, see Figure 11.

PROFESSOR	COURSE	WEEKDAY
John	Java	Tu
John	C++	Tu
Mary	Java	We
Mary	C++	Fr

PROFESSOR	COURSE	WEEKDAY
John	Java	Tu
Susan	C++	Tu
Mary	Java	We
Mary	C++	Fr

Figure 11: The participation constraint $card^{part}(\text{TEACH}, \text{PROFESSOR}) = 2$ holds in both populations. In addition, the first population violates $card^{Chen}(\text{TEACH}, \text{COURSE}) = 1$, while the second one satisfies this Chen-style constraint.

Observation 3. *In general, participation constraints cannot be expressed by Chen-style constraints, and Chen-style constraints cannot be expressed by participation constraints.*

5 A unifying concept: Generalized Cardinality Constraints

In order to resolve the confusion around participation constraints and Chen-style constraints, some authors introduced new approaches towards cardinality constraints. Generalized cardinality constraints have been defined e.g. by Ferg [10], by Thalheim [24] within his Higher-order entity-relationship model (HERM) and by Embley et.al. [9] within their Object-oriented System Analysis (OSA). For surveys on classes of cardinality constraints used in entity-relationship modelling the interested reader is referred to [16, 25].

Let R be a relationship type and W a non-empty subset of the component set $Co(R) = \{C_1, \ldots, C_n\}$. A *generalized cardinality constraint* is an expression

$card(R, W) = b$ with $b \in \mathbb{N}^\infty$. This constraint holds in a population R^t if for every choice of entities $c_i \in C_i^t$ with $C_i \in W$ there are at most b relationships in R^t such that $r(C_i) = c_i$ for all $C_i \in W$.

A participation constraint $card^{part}(R, C_k) = b$ corresponds to a generalized cardinality constraint $card(R, W) = b$ with $W = \{C_k\}$. A Chen-style constraint $card^{Chen}(R, C_k) = b$ is just a generalized cardinality constraint $card(R, W) = b$ with $W = Co(R) \setminus \{C_k\}$. It should be emphasized that there are generalized cardinality constraints which are neither Chen-style nor participation constraints. If the primary key of a relationship type contains no attributes, that is, if $id(R) \subseteq Co(R)$, we immediately have the generalized cardinality constraint $card(R, id(R)) = 1$.

6 Inference Rules for Generalized Cardinality Constraints

The constraints satisfied by a population are usually not independent. A single constraint σ *follows* from a constraint set Σ if σ holds in every population R^t which satisfies Σ. We also say that Σ *implies* σ. Two constraint sets Σ and Σ' are *equivalent* if every constraint in Σ' follows from Σ and vice versa.

In practice, of course, we will not inspect all possible populations in order to decide whether a constraint σ follows from a given constraint set Σ or not. Rather, we are interested in inference rules which help us to decide this question. An *inference rule* is an expression $\frac{\Sigma'}{\sigma}\gamma$ where Σ' is a subset of Σ, and γ states some condition on Σ' which has to be satisfied if we want to apply this rule. If Σ contains a subset Σ' satisfying the condition γ, then σ may be *derived* from Σ due to that inference rule. An inference rule is *sound* if Σ implies every constraint σ which may be derived from Σ due to that rule.

We are interested in inference rules which completely describe all the implications of a given constraint set Σ. A *rule system* $\mathcal{R}$ is a set of inference rules. The most prominent example of such a rule system is the Armstrong system for functional dependencies [1]. A set Σ is *syntactically closed* with respect to $\mathcal{R}$ if it contains every constraint σ which may be derived from Σ due to some rule in $\mathcal{R}$. Given a class $\mathcal{Z}$ of integrity constraints, a constraint set Σ is *semantically closed* with respect to $\mathcal{Z}$ if it contains every constraint $\sigma \in \mathcal{Z}$ which is implied by Σ. The general problem is to find a rule system $\mathcal{R}$ such that a given set $\Sigma \subseteq \mathcal{Z}$ is semantically closed w.r.t. $\mathcal{Z}$ if and only if it is syntactically closed w.r.t. $\mathcal{R}$. Such a rule system is said to be *sound and complete* for the implication of $\mathcal{Z}$.

In this section, we will present a suitable rule system for generalized cardinality constraints and verify that this system is sound and complete. As already mentioned above, there is some kind of interaction between generalized cardinality constraints: According to Observation 2, participation constraints always imply certain Chen-style constraints. This observation gives rise to rule (C2) below. The other two rules are obvious: (C1) gives us the trivial constraint providing ∞ as an upper bound, while (C3) allows us to conclude 'weaker' constraints from 'sharper' ones.

Observation 4. *Let R be a relationship type, X, Y be non-empty subsets of $Co(R)$, and $b, b' \in \mathbb{N}^\infty$. The following rules are sound:*

$$\text{(C1)}\ \frac{}{card(R, X) = \infty}$$

$$\text{(C2)}\ \frac{card(R, X) = b}{card(R, Y) = b}\ X \subset Y$$

$$\text{(C3)}\ \frac{card(R, X) = b}{card(R, X) = b'}\ b < b'$$

Let Σ_C be a set of generalized cardinality constraints specified on a relationship type R. Given a non-empty subset $Y \subseteq Co(R)$ let b_Y denote the smallest integer b such that Σ_C contains some constraint $card(R, X) = b$ with $X \subseteq Y$. If no such integer exists, we put $b_Y = \infty$. Informally, we may say that $card(R, Y) = b_Y$ is the 'sharpest' constraint derivable from Σ_C by applying the rules (C1) and (C2).

For example, consider the constraint set containing $card(\textsc{Teach}, \{\textsc{Professor}\}) = 3$, $card(\textsc{Teach}, \{\textsc{Course}\}) = 1$ as well as $card(\textsc{Teach}, \{\textsc{Professor}, \textsc{Weekday}\}) = 4$. We obtain $b_{\{\textsc{Professor}\}} = 3$, $b_{\{\textsc{Course}\}} = 1$, $b_{\{\textsc{Weekday}\}} = \infty$, $b_{\{\textsc{Professor},\textsc{Course}\}} = 1$, $b_{\{\textsc{Professor},\textsc{Weekday}\}} = 3$, $b_{\{\textsc{Course},\textsc{Weekday}\}} = 1$ and $b_{\{\textsc{Professor},\textsc{Course},\textsc{Weekday}\}} = 1$.

Later on, we make use of the following observation.

Observation 5. *Let Σ_C be a given set of generalized cardinality constraints. For any two non-empty subsets X and Y of $Co(R)$ with $X \subseteq Y$ we have $b_X \geq b_Y$.*

Now consider the constraint set Σ_C^B which consists of the constraints $card(R, Y) = b_Y$ for all the non-empty subsets Y of $Co(R)$. We call Σ_C^B the *basic* constraint set corresponding to Σ_C. Note that Σ_C^B provides essentially the same semantic information as Σ_C: The constraints in Σ_C^B may be derived from Σ_C by applying the rules (C1) and (C2). As these rules are sound, Σ_C^B follows from Σ_C. Conversely, every constraint in Σ_C may be derived from Σ_C^B due to rule (C3). Hence, Σ_C^B implies Σ_C.

Observation 6. *Every set Σ_C of generalized cardinality constraints is equivalent to the basic constraint set Σ_C^B corresponding to Σ_C.*

Our objective is to show that the rules (C1)–(C3) presented above form a complete rules system for generalized cardinality constraints. The following result is a first step into this direction. It verifies that sets of generalized cardinality constraints do not imply any constraint which may not be derived from the corresponding basic set due to (C3).

Observation 7. *Let Σ_C be a set of generalized cardinality constraints, let Y be a non-empty subset of $Co(R)$, and let $b < b_Y$ be some positive integer. Then Σ_C does not imply the constraint $card(R, Y) = b$.*

Let σ denote the constraint $card(R, Y) = b$ under inspection. For $b < b_Y$ the value b will be finite. To verify the previous observation we construct a population $R^{t,\sigma}$ which satisfies Σ_C^B but violates σ. To begin with we fix object sets $C_i^{t,\sigma} = \{c_{i,1,\sigma}, \ldots, c_{i,b+1,\sigma}\}$ for every component $C_i \in Co(R)$, and choose values $a_{i,1,\sigma}, \ldots, a_{i,b+1,\sigma} \in dom(A_i)$ for every attribute $A_i \in Attr(R)$. Then, let $R^{t,\sigma}$ consist of $b+1$ relationships $r_{1,\sigma}, \ldots, r_{b+1,\sigma}$ where

$$r_{j,\sigma}(C) = \begin{cases} c_{i,j,\sigma} & \text{if } C_i \notin Y, \\ c_{i,1,\sigma} & \text{if } C_i \in Y, \end{cases}$$

Professor	Course	Weekday
John	Java	Mo
John	C++	Mo
John	Delphi	Mo

Figure 12: Let σ denote the constraint $card(\text{Teach},\{\text{Professor},\text{Weekday}\}) = 2$. The population $R^{t,\sigma}$ above violates σ.

and $r_{j,\sigma}(A_i) = a_{i,j,\sigma}$ for every attribute A_i of R.

Clearly, the resultant population $R^{t,\sigma}$ satisfies Σ_C^B. Moreover, any two relationships coincide in their projections to the fixed subset $Y \subseteq Co(r)$. As $R^{t,\sigma}$ is of size $b+1$ it obviously violates the constraint $card(R,Y) = b$.

Theorem 8. *The rules (C1)–(C3) form a sound and complete system for the implication of generalized cardinality constraints.*

As we have already pointed out the soundness of the three rules it remains to discuss the completeness. Let R be a relationship type with a set Σ_C of generalized cardinality constraints declared on it. Fix some constraint σ, say $card(R,Y) = b$, which may not be be derived from Σ_C due to our rules. Note, that for any non-empty subset $X \subseteq Y$ the constraint set Σ_C may only contain constraints $card(R,X) = b'$ with $b' > b$. By definition, this gives us $b_Y > b$. Now consider the population $R^{t,\sigma}$ constructed above. It satisfies Σ_C, but violates σ. Hence, the constraint under inspection does not follow from Σ_C as claimed.

Thus, the rule system (C1)–(C3) is in fact sound and complete as desired. Hence, Observation 2 above - which is a simple consequence of rule (C2) - describes the only non-trivial implication between participation constraints and Chen-style constraints. This explains why both kinds of constraints are necessary in conceptual data modelling and none of these two approaches should be neglected for the sake of an easier graphical representation of cardinality constraints in ER diagrams.

7 Generalized Cardinality Constraints in the presence of Functional Dependencies

Functional dependencies are considered to be the most important class of integrity constraints used in database design. Of course, functional dependencies are also of interest in entity-relationship modelling. In practice, most applications require functional dependencies as well as cardinality constraints. The objective of this section is to present a sound and complete rule system for the joint class of functional dependencies and generalized cardinality constraints.

A *functional dependency* on R is a statement $R : X \to Z$ where both, X and Z are non-empty subsets of $Co(R) \cup Attr(R)$. This functional dependency holds in the population R^t if we have $r[Z] = r'[Z]$ whenever $r[X] = r'[X]$ holds for any two relationships r and r' in R^t.

As an example consider the populations $R^{t,\sigma}$ introduced above. Any two relationships r and r' in such a population coincide in their projections to Y, that is, $r[Y] = r'[Y]$ but differ in each component $C \notin Y$. Hence, $R^{t,\sigma}$ cannot satisfy some functional dependency $R : X \to Z$ where X is a subset of Y and Z contains some component C not in Y. In fact, it is easy to see that $R^{t,\sigma}$ satisfies a functional dependency $R : X \to Z$ if and only if we have either $X \not\subseteq Y$ or $X \cup Z \subseteq Y$. In the first case, any two relationships in the population $R^{t,\sigma}$ differ in their projections to X, while in the second case any two relationships coincide in their projections to Y, and thus, in their projections to $X \cup Z$.

Observation 9. *Let σ be a generalized cardinality constraint, say $card(R,Y) = b$. The population $R^{t,\sigma}$ constructed above satisfies a functional dependency $R : X \to Z$ if and only if we have $X \not\subseteq Y$ or $X \cup Z \subseteq Y$.*

Usually, one expects to find at least one functional dependency for every relationship type R: As R is supposed to have a primary key $id(R)$, we immediately have the functional dependency $R : id(R) \to Co(R) \cup Attr(R)$.

It is well-known [1] that the Armstrong rules (F1)–(F3) below are sound and complete for the implication of functional dependencies.

$$\textit{(F1)}\ \frac{}{R : X \to Y}\ Y \subseteq X$$

$$\textit{(F2)}\ \frac{R : X \to Y}{R : X \to X \cup Y}$$

$$\textit{(F3)}\ \frac{R : X \to Y,\ R : Y \to Z}{R : X \to Z}$$

Let Σ_F be a set of functional dependencies, and let $X \subseteq Co(R) \cup Attr(R)$ be a non-empty subset of components and/or attributes. As usual, we denote by $X^+ = \{C \in Co(R) \cup Attr(R) : \Sigma_F \models R : X \to \{C\}\}$ the *functional closure* of X under Σ_F, that is, the set of all components and attributes which are determines by X according to the functional dependencies implied by Σ_F.

Given a set Σ_F of functional dependencies, we call $\Sigma_F^B = \{R : X \to X^+ : \emptyset \neq X \subseteq Co(R) \cup Attr(R)\}$ the *basic* constraint set corresponding to Σ_F. In textbooks on database theory, the dependencies in Σ_F^B are sometimes called *right-extended*. It is well-known that Σ_F and Σ_F^B are equivalent, cf. [17].

Let Σ_F be a set of functional dependencies and let σ denote a functional dependency $R : X \to Z$ which is not implied by Σ_F, that is, Z is not a subset of X^+. It is rather easy to find a population $R^{t,\sigma}$ which satisfies Σ_F but violates σ. First, we fix a two-element object set $C_i^{t,\sigma} = \{c_{i,1,\sigma}, c_{i,2,\sigma}\}$ for every $C_i \in Co(R)$ as well as two values $a_{i,1,\sigma}, a_{2,\sigma} \in dom(A_i)$ for every attribute $A_i \in Attr(R)$. Then, let $R^{t,\sigma}$ consist of the two relationships $r_{1,\sigma}$ and $r_{2,\sigma}$ where $r_{1,\sigma}$ is given by $r_{1,\sigma}(C_i) = c_{i,1,\sigma}$ for every $C_i \in Co(R)$ and $r_{1,\sigma}(A_i) = a_{i,1,\sigma}$ for every $A_i \in Attr(R)$, while $r_{2,\sigma}$ is defined by

$$r_{2,\sigma}(C_i) = \begin{cases} c_{i,2,\sigma} & \text{if } C_i \notin X^+, \\ c_{i,1,\sigma} & \text{if } C_i \in X^+, \end{cases}$$

and

$$r_{2,\sigma}(A_i) = \begin{cases} a_{i,2,\sigma} & \text{if } A_i \notin X^+, \\ a_{i,1,\sigma} & \text{if } A_i \in X^+. \end{cases}$$

This time, we may ask which generalized cardinality constraints hold in $R^{t,\sigma}$. The answer is fairly easy: As $R^{t,\sigma}$ is of size 2, it clearly satisfies every generalized cardinality constraint $card(R,Y) = b$ with $b \geq 2$. For $b = 1$, however, the constraint under discussion only holds if Y is not a subset of X^+.

Observation 10. *Let Σ_F be a set of functional dependencies, and let σ be a functional dependency not*

Professor	Course	Weekday
John	Java	Mo
John	Java	Tu

Figure 13: Let σ denote the functional dependency Teach : {Course} $\rightarrow$ {Weekday} and suppose Σ_F^B contains the functional dependency Teach : {Course} $\rightarrow$ {Professor,Course}. The population population $R^{t,\sigma}$ above satisfies Σ_F^B, but violates σ.

implied by Σ_F, say $R : X \rightarrow Z$. The population $R^{t,\sigma}$ constructed above satisfies a generalized cardinality constraint $card(R, Y) = b$ if and only if we have $b \geq 2$ or $Y \not\subseteq X^+$.

The previous observation shows that there is some sort of interaction between functional dependencies and generalized cardinality constraints. The following two rules are simple examples of inference rules involving both kinds of integrity constraints.

Observation 11. *Let R be a relationship type, X, Y be non-empty subsets of $Co(R)$ and $b \in \mathbb{N}^\infty$. The following rules are sound:*

$$\textit{(CF1)}\ \frac{R : X \rightarrow Y,\ card(R, Y) = b}{card(R, X) = b}$$

$$\textit{(CF2)}\ \frac{card(R, X) = 1}{R : X \rightarrow Co(R) \cup Attr(R)}$$

It is easy to check the correctness of these rules. Suppose a population contains $b + 1$ relationships which coincide in their projections to X. If X determines Y, then all these relationships coincide in their projections to Y, too. Then, however, the cardinality constraint $card(R, Y) = b$ would be violated. This verifies (CF1). Rule (CF2) is a simple consequence of the definition of generalized cardinality constraints.

Theorem 12. *The rules (C1)–(C3), (F1)–(F3), (CF1) and (CF2) form a sound and complete system for the implication of generalized cardinality constraints and functional dependencies.*

As the soundness has already been discussed above it remains to verify the completeness of the rule system $\mathcal{R}$ comprising all the rules mentioned in the theorem. Let Σ_C be a set of generalized cardinality constraints and Σ_F be a set of functional dependencies, and let $\Sigma = \Sigma_C \cup \Sigma_F$ denote their union. Clearly, Σ is equivalent to $\Sigma_C^B \cup \Sigma_F^B$. If Σ is syntactically closed w.r.t. the rule system $\mathcal{R}$, then Σ_C contains Σ_C^B by our completeness result for generalized cardinality constraints, and Σ_F contains Σ_F^B due to the completeness of the Armstrong system. The following observation collects some obvious results on the interaction of generalized cardinality constraints and functional dependencies in syntactically closed constraint sets. The first result is a consequence of (CF1), while the second one is caused by (CF2).

Observation 13. *Let $\Sigma = \Sigma_C \cup \Sigma_F$ be syntactically closed with respect to the rule system $\mathcal{R}$ mentioned above, and let X be a non-empty subset of $Co(R)$. We have $b_X = b_{X^+}$, and if $b_X = 1$ then $X^+ = Co(R) \cup Attr(R)$.*

We are now ready to verify Theorem 12. Suppose $\Sigma = \Sigma_C \cup \Sigma_F$ is syntactically closed w.r.t. $\mathcal{R}$. We have to check that Σ does not imply any functional dependency or generalized cardinality constraint which is not already contained in Σ.

Firstly, let $card(R, Y) = b$ be a fixed generalized cardinality constraint which is not in Σ_C. It suffices to present some population which satisfies $\Sigma_C^B \cup \Sigma_F^B$ but violates the fixed cardinality constraint. Of course, we have $b < b_Y = b_{Y^+}$. By σ we denote the generalized cardinality constraint $card(R, Y^+) = b$ and consider the population $R^{t,\sigma}$ constructed above. By construction, it violates σ and thus the fixed constraint $card(R, Y) = b$. On the other hand, it satisfies Σ_C^B and thus every generalized cardinality constraint in Σ_C. By Observation 9 this population satisfies all functional dependencies $R : X \rightarrow Z$ with $X \not\subseteq Y^+$ or $X \cup Z \subseteq Y^+$. Fortunately, this condition holds for each functional dependency in Σ_F^B as $X \subseteq Y^+$ immediately gives $X \cup Z \subseteq X^+ \subseteq Y^+$. Hence, our population $R^{t,\sigma}$ satisfies Σ_F^B and thus Σ_F.

Secondly, let $R : X \rightarrow Z$ be a fixed functional dependency which is not in Σ_F. Note that the existence of such a dependency yields $X^+ \neq Co(R) \cup Attr(R)$ and, by the previous observation, $b_X \geq 2$. Again it suffices to present some population which satisfies $\Sigma_C^B \cup \Sigma_F^B$ but violates the fixed functional dependency. This time, let σ denote $R : X \rightarrow Z$. Consider the relationship $R^{t,\sigma}$ constructed above. It violates σ, but satisfies Σ_F^B and thus Σ_F. Furthermore, by Observation 10, our population satisfies every generalized cardinality constraint $card(R, Y) = b$ with $Y \not\subseteq X^+$ or $b \geq 2$. Fortunately, this condition holds for each generalized cardinality constraint in Σ_C^B as $Y \subseteq X^+$ immediately gives $b_Y \geq b_{X^+} \geq 2$. Therefore, our population $R^{t,\sigma}$ satisfies Σ_C^B, and thus Σ_C.

This concludes the proof of Theorem 12. Consequently, the rule system comprising the inference rules (C1)–(C3) for generalized cardinality constraints, the Armstrong rules and the rules (CF1) and (CF2) happens to be sound and complete. In particular, it shows that the rules (CF1) and (CF2) describe all possible interactions between functional dependencies and generalized cardinality constraints.

8 Conclusion

In this paper we discussed the implication problem for a class of cardinality constraints which generalize the two popular kinds of cardinality constraints used in entity-relationship modelling. We presented a system of inference rules and proved this system to be sound and complete. This result verifies again that Chen-style constraints and participation constraints are almost independent such that both kinds of constraints may be used in conceptual modelling for their own right - as soon as n-ary relationship types come into play. Together, these constraint classes allow us to specify semantic information which may not be expressed when considering only one of the two approaches. This justifies the simultaneous usage of both constraint classes in data modelling. Moreover we studied interactions between generalized cardinality constraints and functional dependencies and again presented a sound and complete system of inference rules.

It should be noted that there are still more general definitions of cardinality constraints proposed in literature. For a survey, see [16, 25]. In the relational data model, Grant and Minker [11] studied numerical

dependencies which generalize functional dependencies and are closely related to generalized cardinality constraints. Unfortunately, the class of numerical dependencies does not allow a finite axiomatization as shown in [11]. McAllister [18] discussed a class of cardinality constraints which are equivalent to numerical dependencies, but are based on some kind of non-classical logic. But again, there is no complete rule system known for this constraint class. In [12] we studied implications for participation constraints which impose not only upper bounds on the number of occurrences of certain objects, but also lower bounds. In future it should be possible to present a sound and complete system of inference rules for generalized cardinality constraints and minimum participation constraints. However, this problem is not in the scope of the present paper and requires more involved methods from combinatorial design theory.

References

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

[2] C. Batini, S. Ceri, and S. B. Navathe. *Database design: An entity-relationship approach.* Benjamin/Cummings, Menlo Park, 1992.

[3] T. Bruce. *Designing quality databases with IDEF1X information models.* Dorset House, 1992.

[4] 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.

[5] R. Camps. From ternary relationships to relational tables. *SIGMOD Record*, 31, 2002.

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

[7] P. P. Chen. English sentence structure and entity-relationship diagrams. *Information Science*, 29:127–149, 1983.

[8] R. Elmasri and S. B. Navathe. *Fundamentals of database systems.* Addison-Wesley, 1999.

[9] D. W. Embley, B. D. Kurtz, and S. N. Woodfield. *Object oriented systems analysis: a model-driven approach.* Yourdon Press Series, Prentice Hall, 1992.

[10] S. Ferg. Cardinality constraints in entity-relationship modeling. In T. J. Teorey, editor, *Proceedings of the 10th International Conference on Entity-Relationship Approach (ER'91), 23-25 October, 1991, San Mateo, California, USA*, pages 1–30. ER Institute, 1991.

[11] J. Grant and J. Minker. Inferences for numerical dependencies. *Theoretical Comput. Sci.*, 1985:271–287, 41.

[12] S. Hartmann. On the implication problem for cardinality constraints and functional dependencies. *Ann. Math. Artificial Intelligence*, 33:253–307, 2001.

[13] T. H. Jones and I.-Y. Song. Analysis of binary/ternary cardinality combinations in entity-relationship modeling. *Data Knowledge Eng.*, 19:39–64, 1996.

[14] T. H. Jones and I.-Y. Song. Binary equivalents of ternary relationships in entity-relationship modeling: A logical decomposition approach. *J. Database Manag.*, 11:12–19, 2000.

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

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

[17] G. Loizou and M. Levene. *A guided tour of relational databases and beyond.* Springer, Berlin, 1999.

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

[19] A. Rochfeld and H. Tardieu. Merise: An information system design and development methodology. *Information Manag.*, 6:143–159, 1983.

[20] J. Rumbaugh, I. Jacobson, and G. Booch. *The unifying modeling language reference manual.* Addison-Wesley, Reading, 1999.

[21] I.-Y. Song, M. Evans, and E. Park. A comprehensive analysis of entity-relationship diagrams. *J. Computer Software Engrg.*, 3:427–459, 1995.

[22] C. Soutou. Extracting n-ary relationships through database reverse engineering. *LNCS*, 1157:392–405, 1993.

[23] T. J. Teorey. *Database modeling and design.* Morgan Kaufmann, San Francisco, CA, 1998.

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

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

[26] A. M. Tjoa and L. Berger. Transformation of requirement specifications expressed in natural language into an EER model. *LNCS*, 823:206–217, 1993.

[27] J. Ullman and J. Widom. *A first course in database systems.* Prentice Hall, 1997.

[28] E. Yourdon. *Modern Structured Analysis.* Prentice Hall, 1989.