ON THE REALIZABILITY OF CARDINALITY CONSTRAINTS IN CONCEPTUAL DATA MODELS

by

Andrew J. McAllister
Joseph D. Horton

TR99-123, May 1999

Faculty of Computer Science
University of New Brunswick
Fredericton, N.B. E3B 5A3
Canada

Phone: (506) 453-4566
Fax: (506) 453-3566
E-mail: fcs@unb.ca
www: http://www.cs.unb.ca

On the Realizability of Cardinality Constraints in Conceptual Data Models

Andrew J. McAllister and Joseph D. Horton

Faculty of Computer Science
P.O. Box 4400, University Of New Brunswick
Fredericton, NB, Canada, E3B 5A3
andrewm@unb.ca

Abstract

A new property of conceptual data models is introduced. *Realizability* extends strong satisfiability as defined by Lenzerini and Nobili, which is held by a data model when it must be possible to create at least one non-empty database in which no cardinality constraints are violated. A constraint is *realizable* when at least one database can be created in which the number of associations involving a specific entity instance is equal to the limit imposed by the constraint. When all constraints in a data model are realizable, then the entire model is said to be realizable. It is possible for a data model to be strongly satisfiable but not realizable, which means the latter is a more stringent test for model correctness. We define bounds imposed by cardinality constraints on the relative sizes of entity sets. A data model is shown to be realizable if and only if the bounds for any cycle of relationships are either both equal to one, or the lower bound is strictly less than one while the upper bound is strictly greater than one.

1. Introduction

The entity relationship approach [1] has influenced a wide variety of conceptual data modeling techniques (e.g. [4, 10]) including most recently several object-oriented approaches such as OMT [9] and the Unified Modeling Language (UML) [12, 13]. A common characteristic of such modeling techniques is the use of entities (or data objects) as well as relationships that represent associations between the entities. Virtually all of the most widely used data modeling techniques use cardinality constraints to define the nature of relationships. Figure 1 provides an example using the UML notation. This example defines two relationships between Employee and Project. The first relationship indicates that each employee is assigned to exactly one project and there may be two to three employees assigned to each project. The second relationship indicates that certain employees serve as occasional resources (advisors, sources of information, etc.) for specific projects. Figure 1 indicates that each employee must serve as a resource for one or more projects, while each project may optionally have up to two resources.

The set of cardinality constraints in a data model restricts the database states that are permitted by the model. For instance, any database state in which a given employee is assigned to two projects is prohibited by Figure 1. A database state that violates no constraints is said to satisfy the model.

It is possible to specify cardinality constraints so they are difficult to satisfy. For instance, assume Figure 1 is modified so that each project can have only zero or one resource employees. In this case the Assigned To relationship specifies that the database must include at least twice as many employees as projects, whereas the Resource For relationship restricts the number of employees in the database to be less than or equal to the number of projects. The only database state that satisfies this model involves zero employees and zero projects. The modified model is satisfiable but only in a very restricted sense.

Lenzerini and Nobili [6] argue that this concept of satisfiability is of limited usefulness for data models. They suggest that a data model should exhibit *strong satisfiability*, which means it is possible to create at least one non-empty database state that satisfies the model. Figure 2 provides a sample database state for the model in Figure 1 involving four employees and two projects. None of the constraints in Figure 1 are violated in Figure 2. Therefore the data model in Figure 1 is strongly satisfiable.

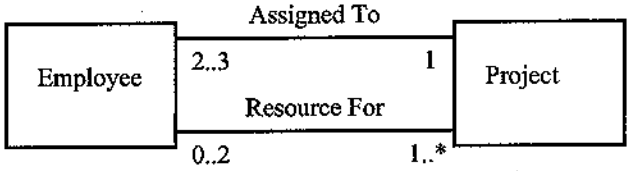

Figure 1: A sample conceptual data model

Employees	Projects
e1	p1
e2	p2
e3	
e4	

Assigned To		Resource For	
Empl.	Proj.	Empl.	Proj.
e1	p1	e1	p1
e2	p2	e2	p1
e3	p1	e3	p2
e4	p2	e4	p2

Figure 2: A database state consistent with Figure 1.

Despite the fact that Figure 1 is strongly satisfiable, the model includes constraints that make little sense. The Assigned To relationship specifies that the database must include *at least twice* as many employees as projects, whereas the Resource For relationship restricts the number of employees in the database to be *at most twice* the number of projects. Therefore the only database states that satisfy this model include exactly twice as many employees as projects. Each project must have exactly two employees assigned and each employee must be a resource for exactly one project. The maximum cardinality constraint of 3 in Figure 1 is meaningless in the sense that it is impossible for any project to be assigned three employees without violating some constraint. Similarly no project can have more than one resource employee nor can any employee to be a resource on less than two projects. A new property for data models is required to ensure that such meaningless constraints can be avoided.

Intuitively, a cardinality constraint can be *realized* if at least one database state can be created in which the number of associations involving at least one specific entity instance is equal to the limit imposed by the constraint. (A more complete definition is provided in Section 4.) For example the constraint limiting the number of resources per project to two is realized in Figure 2 since projects p1 and p2 each have two resources. A data model is said to be *realizable* if each constraint in the model can be realized in at least one database state. Based on the preceding discussion the model in Figure 1 is strongly satisfiable but not realizable.

Realizability is a more stringent criterion for "correctness" of cardinality constraints than is strong satisfiability. Indeed realizability is the more appropriate condition to check since it makes little sense to specify constraints that can never be realized.

Lenzerini and Nobili [6] define a means to determine if a model is strongly satisfiable. Thalheim [11] extends this work by providing an alternative means for testing the same property. Hartmann [3] provides approaches for generating database states with various properties (e.g. minimum size) for strongly satisfiable data models. McAllister [8] introduces the concept of realizability of a single relationship and defines a complete set of rules for checking whether the constraints specified for a single n-ary relationship are realizable.

This paper defines the means to test whether an entire entity-relationship model is realizable. The remainder of the paper is organized as follows. Section 2 defines the terms and concepts required in this paper. Bounds imposed by cardinality constraints on the relative sizes of entity sets are defined in Section 3, setting the stage for a discussion of realizability in Section 4. Conclusions and opportunities for further work are presented in Section 5.

2. Entity-relationship terms and concepts

Employee in Figure 1 is an example of an *entity*. One or more *attributes* are defined for each entity, which specify the types of pertinent values for each entity. For example, Employee might have attributes such as Employee#, Employee-Name, Employee-Address, etc. Each attribute has a domain from which values for that attribute are selected. For example, values for Employee# may be selected from the domain of four-digit integers. An *entity instance* is a tuple consisting of a value for each attribute of a specific entity. An instance of the Employee entity might have values such as (1062, "John Smith", "100 Main St.",.... etc.).

In a conceptual model for a database, an *entity* represents the need for the database to be able to store a set of instances for that entity. For an entity a, the notation $[a]$ is used to represent an instance of a, and (a) represents a set of zero or more instances of a. The term *entity set* is also used to refer to a set of entity instances. The notation $|a|$ refers to the number of entity instances in (a).

A *relationship* represents the need to store associations between entity instances. Each relationship has two or more *roles*, each of which links a specific entity to the relationship. The Assigned To relationship in Figure 1 has two roles and is referred to as binary. The term *n-ary* is used to describe any relationship with n roles where $n>2$. A *role name* can be assigned to any role, however role names are typically used only when a given entity participates in multiple roles for a given relationship. For example, the Supervises relationship in Figure 3 has two roles, both of which are named.

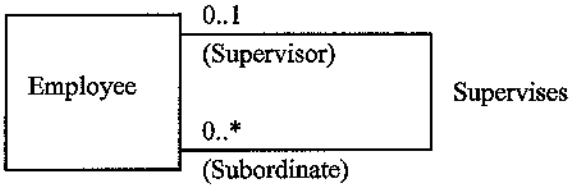

Figure 3: A relationship with two named roles

Attributes can be defined for relationships in the same manner as for entities. In this paper, however, relationship attributes are not considered since their presence or absence has no impact on realizability. In the absence of relationship attributes, a *relationship instance* is a tuple of the form $(e_1, e_2, ..., e_n)$ where n is the number of roles and each e_i identifies an instance of the entity participating in the i'th role. As an illustrative example, Figure 2 shows a possible set of instances for each of the two entities and two relationships defined in Figure 1.

For a relationship r, the notation $[r]$ is used to represent an instance of r, and (r) represents a set of zero or more instances of r. The term *relationship set* is also used to refer to a set of relationship instances. The notation $|r|$ refers to the number of entity instances in (r).

A *cardinality constraint* is used to restrict, for a given relationship, the number of different entity instances with which a given entity instance can be associated. For the UML notation used in Figures 1 and 3, a cardinality constraint for each relationship role can be specified in the form of a range of allowable values, in the form "minimum..maximum". For example, the number 2 specified next to Employee for the Assigned To relationship in Figure 1 specifies that any given instance of Project must be associated with a minimum of 2 instances of Employee by the Assigned To relationship. In addition, the number 3 in Figure 1 specifies that any given instance of Project can be associated with a maximum of 3 instances of Employee by the Assigned To relationship.

The notation $\text{Cmin}(r,a,b)$ represents a minimum cardinality constraint, which specifies the minimum number of different $[b]$'s with which any given $[a]$ can be associated in any (r). Similarly, $\text{Cmax}(r,a,b)$ specifies the maximum number of different $[b]$'s with which any given $[a]$ can be associated in any (r). Abbreviating Employee, Project and the Assigned To relationship as E, P and AT respectively, example constraints in Figure 1 include: Cmin(AT,P,E) = 2, Cmax(AT,P,E) = 3, Cmin(AT,E,P) = 1 and Cmax(AT,E,P) = 1.

A Cmin constraint can be any integer zero or larger. A Cmax constraint can be any integer one or larger, or "*", which represents infinity or "many". $\text{Cmax}(r,a,b) = *$ indicates there is no restriction on the maximum number of $[b]$'s associated with any given $[a]$ in (r).

$\text{Cmax}(r,a,b)$ is *violated* in (r) if for any given [a] the number of associated [b]'s in (r) is more than $\text{Cmax}(r,a,b)$. It follows that no $\text{Cmax}(r,a,b) = *$ can be violated in any (r). Similarly, $\text{Cmin}(r,a,b)$ is *violated* in (r) if for any given [a] the number of associated [b]'s in (r) is less than $\text{Cmin}(r,a,b)$, which means that no $\text{Cmin}(r,a,b) = 0$ can be violated in any (r).

A *database state* for a data model is made up of a set of instances for each of the entities and relationships in the model. A database state that violates no constraints is said to *satisfy* the data model.

A *path* p involving m relationships ($m \geq 1$) is defined as the ordered list $p = (a_1, r_1, a_2, r_2, a_3, ... a_m, r_m, a_{m+1})$ where each relationship r_i associates the entities a_i and a_{i+1}. The notation p^{-1} denotes a path involving the same entities and relationships as p, but in reverse order. A *cycle* is a path where the first and last entities are the same. In cases where the definition of a path makes the direction of relationship traversal ambiguous, specification of entities can be augmented with relationship role names to remove the ambiguity. For example, the following path specification for the model in Figure 3 is ambiguous: (Employee, Supervises, Employee). Two unambiguous paths are: p = (Employee (Supervisor), Supervises, Employee(Subordinate)) and p^{-1} = (Employee(Subordinate), Supervises, Employee (Supervisor)).

For a cycle c and a relationship r in c, c-r denotes the path obtained by removing r from c.

3. Bounds on the relative sizes of entity sets

The basic issue involved with both strong satisfiability and realizability is that cardinality constraints can do more than restrict participation of entities in relationships. The constraints can also place bounds on the relative sizes of entity sets. For example, Section 1 describes how Figure 1 restricts the number of employees to be exactly twice the number of projects. The first step in exploring the concept of realizability is to understand in general how a relationship can restrict the relative sizes of two entity sets.

Consider the Assigned To relationship in Figure 1, ignoring for the moment the Resource For relationship. Assume we wish to create a database state with the minimum number of employees relative to the number of projects in the database. This is achieved by assigning the minimum number of employees to each project and by assigning each employee to the maximum number of projects. In other words, the ratio of employees to projects has a lower bound determined by Cmin(Assigned To,P,E) and Cmax(Assigned To,E,P). This lower bound is defined in general as follows:

$$\text{lb}(r,a,b) = \frac{\text{Cmin}(r,a,b)}{\text{Cmax}(r,b,a)} \leq \frac{|b|}{|a|}$$

For the Assigned To relationship (abbreviated as AT) this works out to:

$$\text{lb}(\text{AT,P,E}) = \frac{\text{Cmin}(\text{AT,P,E})}{\text{Cmax}(\text{AT,E,P})} = \frac{2}{1} \leq \frac{|\text{E}|}{|\text{P}|}$$

There is also an upper bound on the ratio of employees to projects, which is defined in general as follows:

$$\frac{|b|}{|a|} \leq \text{ub}(r,a,b) = \frac{\text{Cmax}(r,a,b)}{\text{Cmin}(r,b,a)}$$

and for the Assigned To relationship as:

$$\frac{|\text{E}|}{|\text{P}|} \leq \text{ub}(\text{AT,P,E}) = \frac{\text{Cmax}(\text{AT,P,E})}{\text{Cmin}(\text{AT,E,P})} = \frac{3}{1}$$

In other words, any database state consistent with the constraints defined for the Assigned To relationship must include between two and three times as many employees as projects. The converse of this statement also holds: any database state must include between one-third and one-half as many projects as employees. The following is easily derived from the definitions of lb and ub provided above:

$$\text{lb}(r,a,b) = \frac{1}{\text{ub}(r,b,a)}$$

This means that lb(AT,E,P) and ub(AT,E,P) are $\frac{1}{3}$ and $\frac{1}{2}$, respectively.

Since Cmin constraints can be zero and Cmax constraints can be *, it is possible for a given relationship to place no bounds on the relative size ratio of the participating entity sets. Consider for example the Resource For relationship (abbreviated RF) in Figure 1 (ignoring for now the Assigned to relationship). The formulae above result in the following:

$$\text{lb}(\text{RF,E,P}) = \frac{1}{2} \leq \frac{|\text{P}|}{|\text{E}|} \leq \text{ub}(\text{RF,E,P}) = \frac{*}{0}$$

In other words, the Resource For relationship imposes no upper bound on the ratio of projects per employee. The following are defined in the context of calculating ub and lb, where x represents any valid Cmin or Cmax value:

$$\frac{0}{x} = 0 \qquad \frac{x}{0} = * \qquad \frac{*}{x} = * \qquad \frac{x}{*} = 0$$

Theorem 1 shows that lb and ub define precisely when the relative sizes of the instance sets for two participating entities allow a given relationship to be satisfied.

Theorem 1: A binary relationship r involving two distinct entities a and b can be satisfied in a database state d with non-empty (a) and (b) iff $\text{lb}(r,a,b) \leq \frac{|b|}{|a|} \leq \text{ub}(r,a,b)$.

Proof: *If-part* — Assume a set of instances for r is generated using Algorithm Genr.

Algorithm Genr.
Input: A definition for r. A database state d with non-empty (a) and (b). Assume $|a| > \text{Cmin}(r,b,a)$ and $|b| > \text{Cmin}(r,a,b)$.
Output: A set (r).
Method:

1. Generate (r) so that $|r| = |a| \times |b|$ in the following manner
   - 1.1 Create $|a|\times|b|$ empty $[r]$'s.
   - 1.2 Create a pattern that includes each $[b]$ once. E.g. $[b]_1, [b]_2, [b]_3, \ldots, [b]_{|b|}$.
   - 1.3 Repeat this entire pattern $|a|$ times in (r), i.e. until each $[r]$ includes a $[b]$ and each $[b]$ appears $|a|$ times in (r).
   - 1.4 Create a pattern that includes each $[a]$ once. E.g. $[a]_1, [a]_2, [a]_3, \ldots, [a]_{|a|}$.
   - 1.5 Repeat the current pattern of $[a]$'s $\frac{|b|}{\text{gcd}(|a|,|b|)}$ times in (r). (The gcd function returns the greatest common divisor of two integers.)
   - 1.6 Shift the current pattern of $[a]$'s by 1 position in a cyclic fashion (e.g. $[a]_1, [a]_2, [a]_3, \ldots, [a]_{|a|}$ becomes $[a]_2, [a]_3, \ldots, [a]_{|a|}, [a]_1$.)
   - 1.7 Repeat steps 1.5 and 1.6 until all $[r]$'s have an $[a]$ participant.
2. Retain the minimum number of $[r]$'s required to satisfy $\text{Cmin}(r,a,b)$ and $\text{Cmin}(r,b,a)$ as follows:
   - 2.1 $n = \max(|a|\times\text{Cmin}(r,a,b), |b|\times\text{Cmin}(r,b,a))$. (The max function returns the largest of two numbers.)
   - 2.2 Retain only the first n $[r]$'s.

As an example of applying Genr, assume r is the Assigned To relationship as defined in Figure 1, a is Employee, b is Project, |Employee| = 5 and |Project| = 2. The result is the set of Assigned To instances in Figure 4.

The relationship r is satisfied by the resultant (r) if no constraints defined for r are violated. The calculation of n ensures that each $[a]$ participates in at least $\text{Cmin}(r,a,b)$ $[r]$'s and that each $[b]$ participates in at least $\text{Cmin}(r,b,a)$ $[r]$'s. Therefore the two Cmin constraints defined for r are not violated.

Assigned To	
Empl.	Proj.
e1	p1
e2	p2
e3	p1
e4	p2
e5	p1

Figure 4: Result of Algorithm Genr for Assigned To

$\frac{n}{|a|}$ gives the average number of [b]'s associated with each [a] in (r). Genr ensures that after a given [a] is associated with a [b], all other [a]'s are associated with one more [b] before the given [a] is associated with another [b]. Therefore the minimum and maximum number of [b]'s associated with each [a] in (r) differ at most by 1. The maximum is defined by $\frac{n}{|a|}$ rounded up to the nearest integer. Cmax(r,a,b) is not violated in (r) iff $\frac{n}{|a|} \le$ Cmax(r,a,b). If $n = |a|\times$Cmin(r,a,b) then $\frac{|a|\times\text{Cmin}(r,a,b)}{|a|} \le$ Cmax(r,a,b) is required, which simplifies to Cmin(r,a,b) $\le$ Cmax(r,a,b) and is true by definition. Otherwise (i.e. if $n = |b|\times$Cmin(r,b,a)) then $\frac{|b|\times\text{Cmin}(r,b,a)}{|a|} \le$ Cmax(r,a,b) is required. This transforms to $\frac{|b|}{|a|} \le$ ub(r,a,b), which is given. It follows that Cmax(r,a,b) is not violated in (r). Similarly, Cmax(r,b,a) is shown not to be violated by showing that $\frac{n}{|b|} \le$ Cmax(r,b,a). It follows that the (r) produced by Genr satisfies r.

Only-if part — Assume lb(r,a,b) > $\frac{|b|}{|a|}$. This transforms to $\frac{\text{Cmin}(r,a,b)\times|a|}{|b|}$ > Cmax(r,b,a). The left hand side gives a lower bound for the average number of [r]'s in which each [b] must participate. This average can only be achieved by violating Cmax(r,b,a). It follows by contradiction that r can be satisfied only if lb(r,a,b) $\le \frac{|b|}{|a|}$. A similar argument shows that r can be satisfied only if $\frac{|b|}{|a|} \le$ ub(r,a,b). □

Given that a single relationship r_1 can impose a restriction on the relative sizes of (a_1) and (a_2), and that a second relationship r_2 can impose a restriction on the relative sizes of (a_2) and (a_3), it makes sense that a path p = (a_1, r_1, a_2, r_2, a_3) can restrict the relative sizes of (a_1) and (a_3). In general, for a path p involving m relationships, we define:

$$\text{lb}(p,a_1,a_{m+1}) = \prod_{i=1}^{m} \text{lb}(r_i,a_i,a_{i+1})$$

$$\text{ub}(p,a_1,a_{m+1}) = \prod_{i=1}^{m} \text{ub}(r_i,a_i,a_{i+1})$$

The proof for Lemma 1 shows how these definitions are derived.

Lemma 1: A path p involving m relationships can be satisfied in a database state d with non-empty (a_1) and (a_{m+1}) iff $\text{lb}(p,a_1,a_{m+1}) \le \frac{|a_{m+1}|}{|a_1|} \le \text{ub}(p,a_1,a_{m+1})$.

Proof: *Base case*—Assume m=1. By Theorem 1 p can be satisfied in d iff $\text{lb}(p,a_1,a_2) \le \frac{|a_2|}{|a_1|} \le \text{ub}(p,a_1,a_2)$.

Induction step—Assume that a subset of p defined as q = (a_1, r_1, ... a_j, r_j, a_{j+1}) can be satisfied in d iff $\text{lb}(q,a_1,a_{j+1}) \le \frac{|a_{j+1}|}{|a_1|} \le \text{ub}(q,a_1,a_{j+1})$. By Theorem 1 r_{j+1} can be satisfied in d iff $\text{lb}(r_{j+1},a_{j+1},a_{j+2})\times|a_{j+1}| \le |a_{j+2}| \le \text{ub}(r_{j+1},a_{j+1},a_{j+2})\times|a_{j+1}|$. The leftmost and rightmost terms define a minimum and a maximum for $|a_{j+2}|$ relative to $|a_{j+1}|$, respectively. Replacing $|a_{j+1}|$ in these two terms with the minimum and maximum (respectively) for $|a_{j+1}|$ relative to $|a_1|$ (such that q can be satisfied) results in: $\text{lb}(r_{j+1},a_{j+1},a_{j+2}) \times \text{lb}(q,a_1,a_{j+1}) \times |a_1| \le |a_{j+2}| \le \text{ub}(r_{j+1},a_{j+1},a_{j+2}) \times \text{ub}(q,a_1,a_{j+1}) \times |a_1|$. Therefore q' = (a_1, r_1, ... a_j, r_j, a_{j+1}, r_{j+1}, a_{j+2}) can be satisfied in d iff $\text{lb}(q',a_1,a_{j+2}) \le \frac{|a_{j+2}|}{|a_1|} \le \text{ub}(q',a_1,a_{j+2})$. Lemma 1 follows by induction. □

Theorem 1 and Lemma 1 show that binary relationships can be satisfied if the relative sizes of the instance sets for the participating entities fall within specific bounds. At least one cycle is necessary in a model to prevent strong satisfiability (and also to prevent realizability). This can be thought of in at least two different ways. First, consider any two entities a and b in a cycle. There are two paths from a to b in the cycle. Each of these paths imposes bounds on $\frac{|b|}{|a|}$. If the two sets of bounds conflict (for example if they define nonoverlapping allowable ranges for $\frac{|b|}{|a|}$) then it can be impossible to strongly satisfy (or to realize) the cycle.

By the definition of lb and ub for paths (as provided above), the following equations are true for any two entities a and b in a cycle c:

$$\text{lb}(c,a,a) = \text{lb}(c,b,b)$$

$$\text{ub}(c,a,a) = \text{ub}(c,b,b)$$

Therefore for a cycle c the bounds can be referred to more simply as lb(c) and ub(c). We define lb(c) to be the same as lb(c,a,a) for any entity a in c, and ub(c) to be the same as ub(c,a,a) for any entity a in c. It follows easily that:

$$\text{lb}(c) = \frac{1}{\text{ub}(c^{-1})} \quad \text{and} \quad \text{ub}(c) = \frac{1}{\text{lb}(c^{-1})}$$

As a slightly different way to think of strong satisfiability of cycles, consider any entity a in a cycle. The cycle defines a (directed) path from a to a, which imposes bounds on the range of $\frac{|a|}{|a|}$ allowed by the path. This leads to the following lemma:

Lemma 2: A cycle c is strongly satisfiable iff $\text{lb}(c) \le 1 \le \text{ub}(c)$.

Proof: By Lemma 1 a cycle c involving entity a is strongly satisfiable iff $\text{lb}(c,a,a) \le \frac{|a|}{|a|} \le \text{ub}(c,a,a)$. Since $\frac{|a|}{|a|} = 1$, Lemma 2 follows easily. □

If all cycles in a given data model are strongly satisfiable, then the model is strongly satisfiable. This restates the result of Lenzerini and Nobili [6] in terms of our definitions of lb and ub.

4. Testing for realizability

We define the terms "realized" and "realizable" as follows. An integer cardinality constraint Cmin(r,a,b) or Cmax(r,a,b) is *realized* in (r) iff for at least one [a] the number of different [b]'s associated with this [a] in (r) is equal to the constraint. An integer cardinality constraint is *realizable* iff it is possible to create at least one (r) in which the constraint is realized without violating any constraints in the data model. Cmax(r,a,b) = * is *realizable* iff for at least one specific [a], regardless of the number of [b]'s associated with this [a] in (r), the number of [b]'s associated with this [a] in (r) can be increased without violating any constraints in the data model. A relationship is realizable iff all constraints specified for the relationship are realizable. A data model is realizable iff all relationships in the model are realizable.

Realizability is similar to strong satisfiability in that both depend on the nature of the cycles in a data model. Realizability involves a more stringent requirement than strong satisfiability, however, as defined by Theorem 2.

Theorem 2: A data model is realizable iff for any cycle c in the model: $\text{lb}(c) = 1 = \text{ub}(c)$ or $\text{lb}(c) < 1 < \text{ub}(c)$.

Proof: *If-part* — The if-part addresses the following cases:
(1) A relationship not in any cycle;
(2) A cycle c for which $\text{lb}(c) = 1 = \text{ub}(c)$;
(3) A cycle c for which $\text{lb}(c) < 1 < \text{ub}(c)$.
The remainder of the if-part is divided into portions numbered to be consistent with the above list.

(1) Assume relationship r involving entities a and b is not part of any cycle. r is the only relationship that places a restriction on $\frac{|b|}{|a|}$. Cmax(r,a,b) and Cmin(r,b,a) are realized simultaneously by setting $\frac{|b|}{|a|}$ to ub(r,a,b) and using Genr to generate (r). The proof for Theorem 1 shows that no constraints are violated in this (r). The other two constraints for r are realized in the same manner by reversing a and b. Thus any r not in a cycle is realizable.

(2) Assume a cycle c for which $\text{lb}(c) = 1 = \text{ub}(c)$. From the definition of lb(c) and ub(c) we know:

$$2.1: \prod_{r_i \in c} \text{lb}(r_i,a_i,a_{i+1}) = \prod_{r_i \in c} \text{ub}(r_i,a_i,a_{i+1})$$

Now assume for some $r_i \in c$ that $\text{lb}(r_i,a_i,a_{i+1}) < \text{ub}(r_i,a_i,a_{i+1})$. Then for 2.1 above to be true, it must also be true that for some $r_j \in c$, $\text{lb}(r_j,a_j,a_{j+1}) > \text{ub}(r_j,a_j,a_{j+1})$, which is impossible by the definition of lb and ub. Therefore for all $r_i \in c$, $\text{lb}(r_i,a_i,a_{i+1}) = \text{ub}(r_i,a_i,a_{i+1})$, which means that any database state constructed for c that satisifies the data model must realize all cardinality constraints defined for the relationships in c. By Lemma 2 c is strongly satisfiable, so it is possible to construct such a database instance. It follows that any cycle c for which $\text{lb}(c) = 1 = \text{ub}(c)$ is realizable.

(3) Assume a cycle c for which $\text{lb}(c) < 1 < \text{ub}(c)$. For every r in c, either $\text{lb}(r,a,b) = \text{ub}(r,a,b)$ or $\text{lb}(r,a,b) < \text{ub}(r,a,b)$. If $\text{lb}(r,a,b) = \text{ub}(r,a,b)$ then any non-empty (r) must realize all four constraints defined for r. By Lemma 2 we know c is strongly satisfiable, so a non-empty (r) can be created. Therefore r is realizable.

Assume $\text{lb}(r,a,b) < \text{ub}(r,a,b)$. Create non-empty ($a$) and ($b$) so that both of the following are true:

$$\text{lb}(r,a,b) < \frac{|b|}{|a|} < \text{ub}(r,a,b)$$

$$\text{lb}(c\text{-}r,a,b) \le \frac{|b|}{|a|} \le \text{ub}(c\text{-}r,a,b)$$

The latter condition enables c-r to be strongly satisfied. It is possible for these conditions to be simultaneously true exactly when the ranges defined by the conditions overlap. The two ranges overlap iff $\text{lb}(r,a,b) < \text{ub}(c\text{-}r,a,b)$ and $\text{lb}(c\text{-}r,a,b) < \text{ub}(r,a,b)$. Since $\text{lb}(c) < 1$:

$$\text{lb}(r,a,b) \times \text{lb}(c\text{-}r,b,a) < 1$$

$$\frac{\text{lb}(r,a,b)}{\text{ub}(c-r,a,b)} < 1$$

$$\text{lb}(r,a,b) < \text{ub}(c\text{-}r,a,b)$$

Similarly, $\text{lb}(c\text{-}r,a,b) < \text{ub}(r,a,b)$ follows from $1 < \text{ub}(c)$. Therefore it is possible to create (a) and (b) so that the conditions defined above are true.

Use Algorithm Genr2 to create (r) in which $\text{Cmin}(r,a,b)$ is realized using (a) and (b) as created above. Then use Algorithm Genr2 again to create a different (r) in which $\text{Cmax}(r,a,b)$ is realized using the same (a) and (b). $\text{Cmin}(r,b,a)$ and $\text{Cmin}(r,b,a)$ can be realized in the same way by replacing c with c^{-1} and swapping a and b.

Algorithm Genr2.
Input: A definition for r. A database d with non-empty (a) and (b). A constraint to be realized—either $\text{Cmin}(r,a,b)$ or $\text{Cmax}(r,a,b)$.
Output: A set (r) in which the given constraint is realized.
Method:

1. If the given constraint is $\text{Cmin}(r,a,b)$ then:
   $m = \text{Cmin}(r,a,b)$.
   Else:
   If $\text{Cmax}(r,a,b) = *$ then:
   $m =$ any integer $> \text{Cmin}(r,a,b)$.
   Else:
   $m = \text{Cmax}(r,a,b)$.
2. Generate (r) so that $|r| = |a| \times |b|$ in the following manner:
   2.1 Create $|a| \times |b|$ empty $[r]$'s.
   2.2 Create a pattern that includes each $[b]$ once. E.g. $[b]_1, [b]_2, [b]_3, \ldots, [b]_{|b|}$.
   2.3 Repeat this entire pattern $|a|$ times in (r), i.e. until each $[r]$ includes a $[b]$ and each $[b]$ appears $|a|$ times in (r).
   2.4 Insert $[a]_1$ in each of the first m $[r]$'s.
   2.5 Create a pattern that includes each $[a]$ once, with the exception that $[a]_1$ is not included. E.g. $[a]_2, [a]_3, \ldots, [a]_{|a|}$.
   2.6 Repeat the current pattern of $[a]$'s $\frac{|b|}{\gcd(|b|,|a|-1)}$ times in (r).
   2.7 Shift the current pattern of $[a]$'s by 1 position in a cyclic fashion (e.g. $[a]_2, [a]_3, \ldots, [a]_{|a|}$ becomes $[a]_3, [a]_4, \ldots, [a]_{|a|}, [a]_2$.)
   2.8 Repeat steps 2.6 and 2.7 until all $[r]$'s have an $[a]$ participant.
3. Retain the minimum number of $[r]$'s required to satisfy $\text{Cmin}(r,a,b)$ and $\text{Cmin}(r,b,a)$ as follows:
   3.1 $n = \max(m + (|a|\text{-}1) \times \text{Cmin}(r,a,b), |b| \times \text{Cmin}(r,b,a))$.
   3.2 Retain only the first n $[r]$'s.
4. If any constraints defined for r are violated in (r) then:
   Double $|a|$ and $|b|$.
   Repeat steps 2 to 4.

By the definition of realizability, the relationship r is shown to be realizable iff $\text{Cmin}(r,a,b)$ and $\text{Cmax}(r,a,b)$ are each realized in their respective (r) and no constraints are violated in either (r). The proof of this is the same regardless of whether $\text{Cmin}(r,a,b)$ or $\text{Cmax}(r,a,b)$ is the constraint to be realized. In Genr2, steps 1 and 2.4 combine to ensure that the specified constraint is realized in the resultant (r). It remains to be shown that no constraints defined for r are violated in (r).

The calculation of n in step 3.1 ensures that each $[a]_i$ $(i>1)$ participates in at least $\text{Cmin}(r,a,b)$ $[r]$'s and that each $[b]$ participates in at least $\text{Cmin}(r,b,a)$ $[r]$'s. The calculation of m in step 1 ensures that $[a]_1$ participates in at least $\text{Cmin}(r,a,b)$ $[r]$'s. Therefore the two Cmin constraints defined for r are not violated in (r).

The calculation of m ensures that $\text{Cmax}(r,a,b)$ is not violated for $[a]_1$.

If $n = m+(|a|\text{-}1)\times\text{Cmin}(r,a,b)$ then the number of $[b]$'s associated with each of $[a]_2$ to $[a]_{|a|}$ in (r) is $\text{Cmin}(r,a,b)$, so $\text{Cmax}(r,a,b)$ is not violated. To avoid violating $\text{Cmax}(r,b,a)$ the following must be true:

$$m+(|a|\text{-}1)\times\text{Cmin}(r,a,b) \le |b|\times\text{Cmax}(r,b,a)$$

which can be transformed as follows:

$$\frac{m}{\text{Cmax}(r,b,a)} + (|a|\text{-}1)\times\text{lb}(r,a,b) \le |b|$$

$$\text{lb}(r,a,b) \le \frac{|b| - \frac{m}{\text{Cmax}(r,b,a)}}{|a|-1}$$

Step 4 in Genr2 increases $|a|$ and $|b|$ so that $\frac{|b|}{|a|}$ remains constant. This can be represented in the above formula by multiplying each of $|a|$ and $|b|$ by a variable k (which can increase in value) as follows:

$$\text{lb}(r,a,b) \le \frac{k\times|b| - \frac{m}{\text{Cmax}(r,b,a)}}{k\times|a|-1}$$

The right hand term approaches $\frac{|b|}{|a|}$ as $k \to \infty$. Since $\text{lb}(r,a,b) < \frac{|b|}{|a|}$, then for some value of k the above condition becomes true. Therefore $\text{Cmax}(r,b,a)$ is not violated in (r).

If $n = |b| \times \text{Cmin}(r,b,a)$ then the number of $[a]$'s associated with each $[b]$ in (r) is $\text{Cmin}(r,b,a)$, so $\text{Cmax}(r,b,a)$ is not violated. To avoid violating $\text{Cmax}(r,a,b)$ the following must be true:

$$|b| \times \text{Cmin}(r,b,a) \leq m + (|a|-1) \times \text{Cmax}(r,a,b)$$

which can be transformed to:

$$\frac{|b| - \frac{m}{\text{Cmin}(r,b,a)}}{|a|-1} \leq \text{ub}(r,a,b)$$

Introducing a variable k (as above) results in:

$$\frac{k \times |b| - \frac{m}{\text{Cmin}(r,b,a)}}{k \times |a| - 1} \leq \text{ub}(r,a,b)$$

The left hand term approaches $\frac{|b|}{|a|}$ as $k \to \infty$. Since $\frac{|b|}{|a|} < \text{ub}(r,a,b)$, then for some value of k the above condition becomes true. Therefore $\text{Cmax}(r,a,b)$ is not violated in (r). It follows that any cycle c for which $\text{lb}(c) < 1 < \text{ub}(c)$ is realizable.

Only-if part — By Lemma 2, any cycle c for which $\text{lb}(c) > 1$ or $\text{ub}(c) < 1$ is not strongly satisfiable and is therefore not realizable.

The only remaining case is a cycle c where $\text{lb}(c) = 1 < \text{ub}(c)$. Note that $\text{lb}(c) < 1 = \text{ub}(c)$ is the same case and is equivalent to $\text{lb}(c^{-1}) = 1 < \text{ub}(c^{-1})$.

Since $\text{lb}(c) < \text{ub}(c)$ there must be at least one r in c for which $\text{lb}(r,a,b) < \text{ub}(r,a,b)$. Assume r is such a relationship. Since $\text{lb}(c) = 1$, by the definition of $\text{lb}(c)$ we know the following:

$$\text{lb}(r,a,b) \times \text{lb}(c\text{-}r,b,a) = 1$$

$$\text{lb}(r,a,b) = \frac{1}{\text{lb}(c-r,b,a)} = \text{ub}(c\text{-}r,a,b)$$

This means the only value for $\frac{|b|}{|a|}$ that permits both r and c-r to be strongly satisfied is $\text{lb}(r,a,b)$. Assume an (r) is created and $\frac{|b|}{|a|} = \text{lb}(r,a,b)$. Every $[a]$ must be in (r) exactly $\text{Cmin}(r,a,b)$ times and every $[b]$ must be in (r) exactly $\text{Cmax}(r,b,a)$ times. Since $\text{lb}(r,a,b) < \text{ub}(r,a,b)$ then one or both of the following must be true:

$$\text{Cmin}(r,a,b) < \text{Cmax}(r,a,b)$$

$$\text{Cmin}(r,b,a) < \text{Cmax}(r,b,a)$$

Therefore it is impossible to realize at least one of $\text{Cmin}(r,b,a)$ and $\text{Cmax}(r,a,b)$. It follows that any cycle c for which $\text{lb}(c) = 1 < \text{ub}(c)$ is not realizable. □

Theorem 2 makes it possible to check a data model for unrealizable cardinality constraints by searching for cycles that do not meet the criteria defined by the Theorem. This type of searching can be performed in polynomial time. For example, the Floyd-Warshall algorithm or Dijkstra's algorithm determine the shortest path between two nodes in a graph and can be modified to identify unrealizable cycles. These algorithms can be found in any standard algorithms text, such as [2].

The concepts in this paper are defined in terms of binary relationships. These concepts can easily be extended to handle generalization (is-a) relationships [12, 13] and n-ary relationships. An is-a relationship can be treated as a binary relationship with implied cardinality constraints. The standard semantics of "*a* is-a *b*" implies that $\text{Cmin}(r,a,b) = \text{Cmax}(r,a,b) = 1$ and $\text{Cmax}(r,b,a) = *$. $\text{Cmin}(r,b,a)$ can be either 0 or 1 depending on whether $[b]$ can exist independently of an associated $[a]$.

When two of the roles defined for an n-ary relationship form part of a cycle, then it is necessary to understand the cardinality constraints that affect only those two roles. Several authors describe how such binary constraints form part of the definition of n-ary relationships [5, 7, 8]. McAllister proves that if the constraints defined for a given n-ary relationship conform to a specific set of rules, then all of the constraints for that relationship are realizable [8].

Further details concerning how to check for realizability of data models that include generalization and n-ary relationships are omitted due to space constraints.

5. Conclusions

Data models with unrealizable cycles should be avoided since they define system requirements that are impossible to achieve. The approach defined in this paper can be applied manually to ensure that a given model is realizable. The primary application, however, is likely to be the inclusion of this approach in the automated consistency checking capability of computer-aided software engineering (CASE) tools for systems analysis and design, as well as database design tools.

References

1. Chen, P.P. The Entity-Relationship Model - Toward a Unified View of Data. *ACM Transactions on Database Systems,* 1, 1 (1976) 9-36.
2. Cormen, T.H., Leiserson, C. E. and Rivest, R.L. *Introduction to Algorithms*, McGraw-Hill, 1990.

3. Hartmann, S. Graph-Theoretical Methods to Constuct Entity-Relationship Databases. In *Proc. 21st Annual Workshop in Graph-Theoretic Concepts* (1995), 131-145.

4. Hull, R. and King, R. Semantic Database Modeling: Survey, Applications, and Research Issues. *ACM Computing Surveys*, 19, 3 (September 1987), 201-260.

5. Jones T. and Song, I.-Y. Analysis of Binary/Ternary Cardinality Combinations in Entity-Relationship Modeling, *Data & Knowledge Engineering*, 19, 1 (1996), 39-64.

6. Lenzerini, M. and Nobili, P. On the Satisfiability of Dependency Constraints in Entity-Relationship Schemata. *Information Systems*, 15, 4 (1990), 453-461.

7. Lenzerini, M. and Santucci, G. Cardinality Constraints in the Entity-Relationship Model. In *Proc. 3rd Internat. Conf. on ER Approach* (1983) 529-549.

8. McAllister, A. Complete Rules for N-ary Relationship Cardinality Constraints. *Data & Knowledge Engineering*, 27, 3 (August 1998), 255-288.

9. Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F. and Lorenson, W. *Object-Oriented Modeling and Design*, Prentice Hall, 1991.

10. Teorey, T., Yang, D. and Fry, J. A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. *ACM Computing Surveys*, 18, 2 (June 1986), 197-222.

11. Thalheim, B. Fundamentals of Cardinality Constraints. In *Proc. 11th Int. Conf. on ER Approach* (1992), 7-23.

12. *Unified Modeling Language Notation Guide: Version 1.1*, Online: http://www.rational.com/uml/index.shtml, 1997.

13. *Unified Modeling Language Semantics: Version 1.1*, Online: http://www.rational.com/uml/index. shtml, 1997.