Mapping a Semantic Database Model to the Relational Model

Peter Lyngbaek
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, California 94304

Victor Vianu(*)
EECS C-014
University of California, San Diego
La Jolla, California 92093

ABSTRACT

The connection between semantic database models and the relational model is formally investigated using the Iris Data Model, which has been implemented using relational database techniques The results focus on properties of relational schemas that are translations of Iris schemas Two new types of constraints, cross-product constraints and multiplicity constraints are introduced to characterize the relational translations of Iris schemas The connection established between Iris and relational schemas also yields new, unexpected information about Iris schemas In particular, a notion of equivalence of Iris schemas is defined using their relational translations, and a result is obtained on simplifying the type structure of Iris schemas

1 Introduction

Relational database technology and semantic data modeling have been two major areas of database research in recent years Relational database technology is based on solid theoretical grounds, and it is understood what constitutes a well-designed relational database schema Semantic data modeling, on the other hand, provides a rich set of data abstraction primitives which can capture additional semantics of the application in the database schema So far, relational database technology and semantic modeling have evolved almost separately In practice, however, some of the better known semantic database systems have been implemented on top of existing relational systems [Ca83,TsZa84] This has lead to a need for establishing and understanding connections between semantic models and the relational model The present paper is a first attempt to formally investigate this connection

The semantic model we use is the Iris Data Model, a typical semantic model developed and partially implemented at Hewlett-Packard Laboratories using relational database techniques Our approach is based on defining formally a translation of Iris schemas into relational schemas and characterizing the relational schemas corresponding to Iris schemas Establishing this formal correspondence has several significant benefits

1 The special properties of relational translations of semantic schemas are identified and can be used to obtain more efficient implementations
2 Semantic database interfaces can be constructed for existing relational databases satisfying certain conditions which are described
3 Questions concerning the semantic model can be reformulated in terms of the relational model, where powerful theoretical tools are already available
4 Relational translations of semantic models can be used to compare different semantic schemas and address issues such as equivalence and simplification of semantic schemas

The results obtained in this paper focus on understanding the properties of relational translations of Iris schemas After establishing the formal mapping between Iris schemas and relational schemas, we characterize the relational schemas which are translations of Iris schemas in terms of the constraints they satisfy The constraints used are unary inclusion dependencies and two new types of constraints called "multiplicity constraints" and "cross-product constraints" While it is shown that there is no finite axiomatization for such constraints, a powerful inferencing mechanism is exhibited The inferencing mechanism is novel in that it involves a set of equations

(*) This author was supported in part by the NSF under grant number IST-8511538
An informal presentation based on an earlier version of this paper was given at the Pre-VLDB Symposium in Beijing, China (1986) No portion of this paper appeared in print

associated with the constraints, and is more powerful than any finite set of traditional inference rules Relational translations of Iris schemas are then investigated with respect to Normal Forms It is shown that such translations are not generally in Boyce-Codd Normal Form due to the presence of unexpected "side-effect" functional dependencies A restriction on Iris schemas is then inferred, which guarantees that their relational translations are in BCNF and have additional desirable properties Moreover, the same restriction is shown to guarantee that a certain finite set of inference rules for the constraints in the relational translations is sound and complete

Finally, relational transactions are used to obtain new insight into properties of Iris schemas A notion of equivalence of Iris schemas is defined based on their relational translations This is then used to obtain a result on simplifying the type structure of Iris schemas Other issues are briefly investigated, such as non-trivial satisfiability of Iris schemas, localeness of Iris schemas, hidden type aliasing, and hidden restrictions on the cardinality of types of Iris schemas

The paper consists of five sections, of which the first is the Introduction In Section 2 the Iris Data Model is introduced, and Iris schemas are formally defined The mapping between Iris schemas and relational schemas, and the characterization of relational translations are exhibited in Section 3, as well as the notion of equivalence of Iris schemas and the result on type simplification Section 4 contains the results on inference rules for the constraints of relational translations of Iris schemas, and on Normal Forms Finally, Section 5 contains concluding remarks

Due to space limitations, some of the definitions and results are presented informally and many details, as well as the proofs, are omitted

2 The Iris Data Model

The Iris Data Model falls into the general category of semantic data models that supports high-level structural abstractions as well as behavioral abstractions We chose this model mainly because it supports modeling constructs that are common to a large class of semantic models and because it has been implemented using relational techniques The roots of the model can be found in previous work on Daplex [Sh81] and its extensions [Ku83], the Taxis language [MyBeWo80], and earlier work at HP Laboratories on the Integrated Data Model [BeFe83]

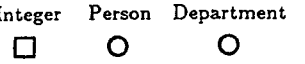
An Iris schema is defined as a directed graph together with a set of constraints (The graphical representation of Iris schemas is inspired by Entity-Relationship diagrams and Nijsen diagrams) The model we describe is close to the subset of the full Iris model[1] supported by the Iris Version 2 0 prototype, an experimental prototype system implemented at Hewlett-Packard Laboratories The presentation emphasizes structural abstractions rather than behavioral abstractions Additional information describing the Iris Data Model can be found in [DeKeLy85, LyKe86, Fi87]

The Iris Data Model is based on *objects, types,* and *functions*[2] Objects represent things and concepts from an application environment, types are sets of objects that share common properties, and functions define properties of objects

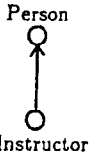
The model distinguishes between literal objects and non-literal objects Literal objects include integers, reals, booleans, and strings They are directly representable Non-literal objects correspond to objects from the application environment that cannot be represented directly in external form For example, non-literal objects can model persons, departments, and bank accounts

Types are uniquely named sets of objects The types Integer, Real, Boolean, and String are called literal types Non-literal types contain objects internally represented by surrogates Types may overlap, for example, an object which represents a given person may be an instance of the types Employee, Taxpayer, and Subscriber

In the graphical representation of an Iris schema, literal types are denoted by squares and non-literal types by circles Both literal and non-literal nodes are labeled Below, the literal type Integer and the two non-literal types Person and Department are shown



Non-literal types are organized in a type structure that supports generalization and specialization [SmSm77] The type structure represents subtype/supertype relationships If a given type is a subtype of another type (represented by a directed edge from the subtype to the supertype), then all the instances of the subtype are also instances of the supertype In the example below, the type Person is the supertype of the type Instructor



A given type may have multiple subtypes The subtypes may be overlapping and they do not necessarily partition the supertype A type may also have multiple supertypes In that case, each object of the subtype must belong to all the supertypes

Properties of objects are expressed in terms of functions, which are defined over types A function may have any number of argument and result types For example, the function

Assignment Instructor -> Course × Semester

is defined on Instructor objects and returns pairs of Course and Semester objects

The Iris functions considered in this paper[3] are implemented by storing the graphs of the functions They may possibly be multi-valued and are therefore not functions

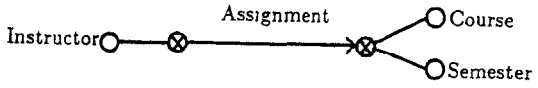
[1] The model described in this paper differs slightly from the actual Iris Data Model as it is currently defined

[2] Types and functions are themselves represented as objects [LyKe86]

[3] In addition to functions whose graphs are stored, the Iris Data Model supports derived functions

in the mathematical sense Rather, they are relations defined over the cross-products of their argument and result types

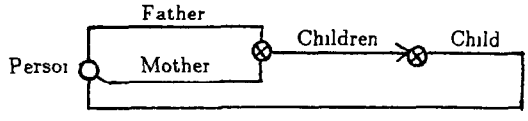
A function is graphically represented by a labeled edge that connects two cross-product vertices The cross-product vertex in which the function edge originates is connected to all the types in the domain of the function and the cross-product vertex in which the function edge terminates is connected to all the types in the range of the function The function Assignment is illustrated below



The objects that are interrelated by a function play certain roles in that function In Assignment, for example, one object plays the role of an instructor, another object plays the role of a course, and a third object plays the role of a semester In some functions, objects of the same type may play different roles The function Children, defined below, is an example of a function in which objects of one type play three different roles

Children Father/Person × Mother/Person -> Child/Person

In order to be able to distinguish the different roles from one another, the roles are uniquely labeled Graphically, this is represented by labels on the edges from the type vertices to the cross-product vertices



If a role name is not explicitly specified it defaults to the name of the associated type

A function is defined not only on the types explicitly mentioned in the function definition, but also on the subtypes of those types This is referred to as inheritance For example, the function Children defined on Person objects are automatically defined on Instructor objects

In order to capture the precise semantics of a given application, a data model should support the distinction between multi-valued and single-valued functions and partial and total functions In the Iris Data Model, the more general concept of *object participation* serves that purpose As an example, consider the function

Major Student -> Department

which is typically partial (i e , a student is not required to have a major) and single-valued (i e , a student has at most one major) These requirements can be defined in the Iris schema by specifying a *lower object participation* (LOP) of zero and an *upper object participation* (UOP) of one for objects playing the Student role in the function Major

Major Student [0,1]

The LOP specifies the minimum number of times each Student object must participate in the relation defined by Major Since the LOP is zero, a Student object is not required to be related to a Department object The UOP specifies the maximum number of times each Student object can participate in the relation defined by Major Since the UOP is one, a Student object can be related to at most one Department object

Lower and upper object participations, which are referred to as *participation constraints*, can be defined for any subset of the argument and result roles of a function Thus, a participation constraint on the two roles Instructor and Semester in the function Assignment can limit instructors to teach at most three courses per semester

Assignment Instructor, Semester [0,3]

Furthermore, the requirement that an instructor can teach a given course only once per semester can be expressed by the following participation constraint

Assignment Instructor, Course, Semester [0,1]

LOP values are restricted to zero and one, UOP values must be positive integers or the special value ∞ If a participation constraint has not been explicitly specified for a given subset of roles in a function it is assumed to be $[0,\infty]$ (i e , participation is not restricted) For literal types with infinite domains, e g , Integer, Real, and String, the LOP must be zero

When several participation constraints are defined on the same function it is important to ensure that they are consistent with each other The participation constraints defined on derived functions[4] are determined by the participation constraints defined on the functions from which they are derived

In addition to supporting semantic integrity control, participation constraints are important for query optimization and organization of data on physical storage devices

We can now define an Iris schema Formally, an Iris schema is a directed, labelled graph together with a set of participation constraints The graph specifies the types, functions, and subtype/supertype relationships defined for a given Iris database The set of constraints specifies the participation constraints imposed on the roles and functions of the graph The definition given below is closely related to the formal definition of the IFO model [AbHu84]

Definition An *Iris schema* is an ordered pair (G, C) where G = (V, E) is a directed graph such that

1 V is the disjoint union of three sets L (labeled literal type vertices), N (labeled non-literal type vertices), and X (cross-product vertices),

2 E is the disjoint union of three sets F (labeled function edges), R (labeled role edges), and S (specialization edges),

3 a function edge connects two cross-product vertices, a role edge connects a literal or non-literal type vertex and a cross-product vertex, and a specialization edge connects two non-literal type vertices,

4 the specialization edges determine an acyclic subgraph of G, and

5 the labels of the type vertices are distinct, the labels of the function edges are distinct, and the labels of the role edges connected to the two cross-product vertices of a given function are distinct,

and C is the collection of sets C_f, f in F, such that

6 for each function edge f, C_f is a set of expressions X[l, u],

[4] We do not describe derived or computed functions in this paper

where $0 \leqslant l \leqslant 1$, $1 \leqslant u \leqslant \infty$, and X is a subset of the labels of the set of role edges connected to the two cross-product vertices of f

An *instance* of an Iris schema is defined in the obvious manner types are mapped to sets of objects and functions are mapped to relations

We now illustrate the Iris model by describing a schema of an example database The database is used by an educational institution to keep track of students, courses, instructors, enrollments, and teaching assignments

Example 2 1 The Iris schema of the example database contains five non-literal types Person, Instructor, Student, Course, and Semester, together with the literal type String Person objects represent people affiliated to the educational institution, such as students and instructors The types Student and Instructor are subtypes of Person Student objects represent past and current students and Instructor objects represent past and present teachers The instances of the type Course correspond to courses offered by the educational institution, and the instances of the type Semesters correspond to semesters in which courses have been offered

Properties of the objects in the example application are captured in the database schema by the following functions

Pname Person -> String
Id Person -> String
Cname Course -> String
Enrollment Student -> Course × Semester × Grade/String
Assignment Instructor -> Course × Semester
Period Semester -> From/String × To/String

The functions represent the facts that persons have names, persons have identification numbers, courses have names, students obtain grades for the courses in which they enroll on a per semester basis, instructors are assigned to courses on a per semester basis, and semesters have beginning and ending dates, respectively

Many kinds of requirements that are appropriate in an educational institution are specified in the database schema as participation constraints For example, the requirement that every person must have at least one name is specified by the constraint

Pname Person [1,∞] (C1)

There are two constraints which specify that each person must have a unique identification number

Id Person [1,1] (C2)
Id String [0,1] (C3)

Constraint C2 specifies that every person must have exactly one identification number Constraint C3 specifies that a given identification number is assigned to at most one person

Course names usage is also restricted by two participation constraints

Cname Course [1,1] (C4)
Cname String [0,1] (C5)

Each course is required by constraint C4 to have exactly one course name, and all course names are required by constraint C5 to be unique

Enrollments of students in courses are restricted by the following participation constraints

Enrollment Course, Semester [1,24] (C6)
Enrollment Student, Semester [0,5] (C7)
Enrollment Student, Course [0,3] (C8)
Enrollment Student, Course, Semester [0,1] (C9)

Each course must be offered every semester and the class sizes cannot exceed a maximum of 24 students (C6), students are limited to five courses per semester (C7), they are allowed to enroll in the same course up to three times (C8), but only once in a given semester (C9)

Instructors are assigned to teach courses under the following restrictions

Assignment Instructor, Semester [0,3] (C10)
Assignment Instructor, Course, Semester [0,1] (C11)

Constraint C10 prevents an instructor from teaching more than three courses per semester, and constraint C11 prevents an instructor from teaching the same class twice in the same semester

Semesters have the following two constraints

Period Semester [1,1] (C12)
Period From, To [0,1] (C13)

Constraint C12 specifies that semesters must have beginning and ending dates, and constraint C13 specifies that no two semesters can have the same pair of beginning and ending dates

Figure 2 1 shows the graphical representation of the Iris schema for the educational institution database Iris schemas that have many types and functions tend to be complicated to represent graphically However, database interfaces that support graphical schema representations should allow end-users to selectively view and manipulate small pieces of a schema

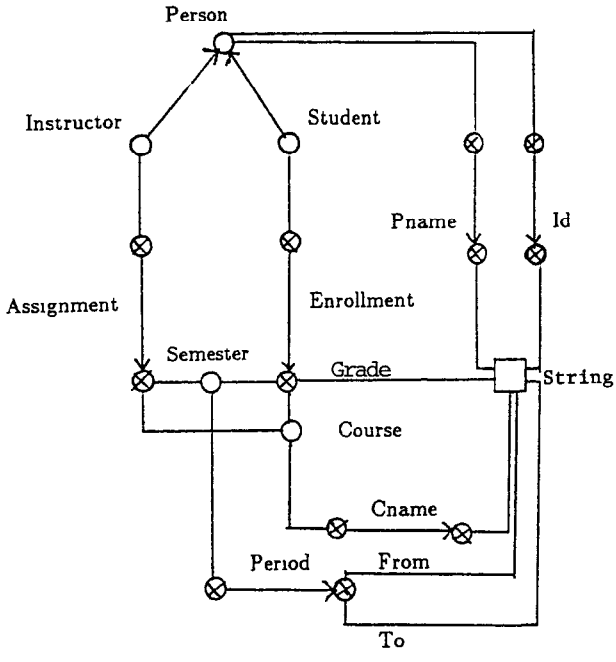


Figure 2 1 Graph of Iris Schema for Educational Institution Example

The example Iris schema supports queries such as "Who taught Biology in the Spring of 1983?", "In what courses did Nancy Slick receive the grade A?", and "What is the grade

point average of the student with identification number 900-23-7239?"

Iris queries are specified as Find operations The Find operation has the following form

FIND (x_1/T_1, ,x_n/T_n) Predicate

The variables x_1, ,x_n, which are called result variables, have types T_1, ,T_n, respectively The result variables are free variables in the predicate The predicate may optionally be existentially-qualified and contain functions, comparisons of variables and values, and logical connectives It is beyond the scope of this paper to formally define the Iris query language

3 Mapping Iris Schemas to Relational Schemas

The Iris model is currently implemented using relational database techniques Thus, every Iris schema is mapped to a relational schema with appropriate constraints, and every Iris instance is implemented as a corresponding relational instance Iris queries are translated into relational select-project-join queries, and Iris updates become relational transactions The usual tools of relational databases can be used to improve schema design, perform query optimization, and maintain database integrity in the course of updates In this section we formally examine the correspondence between Iris schemas and relational schemas We define a mapping[5] of Iris schemas to relational schemas and characterize all relational schemas which are translations of Iris schemas The relational translation of Iris schemas is then used to define a simple notion of equivalence of Iris schemas Based on this notion of equivalence, we exhibit a result concerning simplification of the type structure of Iris schemas

We first present some basic concepts and notation of the relational model, which will be used in the rest of the paper We assume familiarity with the notions of attribute, domain, tuple, relation over a finite set of attributes, projection of a relation or a set of constraints on a set of attributes, functional dependency (fd), inclusion dependency (id) and Boyce-Codd Normal Form (BCNF), as in [Ma83, Ul82] A relation schema is a triple <RelName, R, Σ> where RelName is a symbol called the relation name, R is a set of attributes and Σ is a set of constraints over R A (relational) database schema is a pair <S,Σ> where S is a set of relation schemas with distinct names and Σ a set of inter-relational constraints involving relations in S The logical closure of a set Σ of constraints is denoted by Σ^* If Σ consists of fd's or unary id's, its logical closure can be computed using known sets of inference rules [Ul82, CFP82] We will use two types of constraints in addition to fd's and id's These new constraints arise through the translation of the participation constraints of Iris schemas, and are defined next

Definition Let R be a finite set of attributes A cross-product constraint over R is an expression $\otimes$ X where $X \subseteq R$ A relation r satisfies $\otimes$ X iff $\Pi_X(r) = \times_{A \in X} \Pi_A(r)$ ($\times$ is the cross-product operation[6]) A multiplicity constraint is an expression $X[\leqslant k]$ where $X \subseteq R$ and $1 \leqslant k \leqslant \infty$ A relation r over R satisfies $X[\leqslant k]$ if no more than k tuples in r have identical projections on X

Thus, a cross-product constraint over a subset of the attributes of a relation requires that every combination of values of those attributes appear in the relation, a multiplicity constraint places an upper bound on the number of times each combination may appear in the relation

For example, the relation represented in Figure 3 1 satisfies $\otimes$ BC,BC[$\leqslant$1], AB[$\leqslant$2], and A[$\leqslant$3] It does not satisfy $\otimes$ AB or C[$\leqslant$1]

A	B	C
0	0	1
0	0	0
0	1	0
1	1	1

Figure 3 1

We next describe the mapping of Iris schemas to relational schemas The relational translation of an Iris schema s will be denoted by RelTrans(s) Consider an Iris schema s The mapping is based on associating with each function edge of s a relation whose attributes are the role labels of the edges connecting the two cross-product vertices of the function edge to corresponding type vertices For instance, the function

Enrollment Student -> Course x Semester x Grade/String

of Example 2 1 is represented by a relation with attributes Student, Course, Semester, and Grade Next, subtyping and participation constraints of s induce constraints in RelTrans(s) Specifically, participation constraints induces intra-relational cross-product and multiplicity constraints, and subtyping (together with participation constraints) induces intra and inter-relational unary inclusion dependencies (a detailed example is presented later)

We next define formally the relational schema corresponding to a given Iris schema We will carry out the translation in two stages, denoted RelTrans_1 and RelTrans_2 In the first-stage of the translation we will have, in addition to relations corresponding to functions, one relation corresponding to the cross-product of all non-literal types The "types" relation is not present in the final translation, but is used to more easily infer the constraints which must be satisfied by the "function" relations as a consequence of subtyping and participation constraints The second stage of the translation consists of computing the constraints satisfied by the "function" relations and eliminating the "types" relation We now define the first stage of the translation

Definition Let s = (G,C) (G=(V,E)) be an Iris schema Then $\text{RelTrans}_1(s)$ = <S,Σ>, where S consists of the relation schemas

(i) <f,X,Σ_f>, where f is a function label in G, X is the set of role labels associated with f, and

$$\Sigma_f = \{\otimes Y \mid Y[1,k] \in C_f\} \cup$$
$$\{Y[\leqslant k] \mid Y[n,k] \in C_f, n \in \{0,1\}\}, \text{ and}$$

(ii) <types, N, Σ_{types}> where N is the set of non-literal types in V and

Σ_{types} = {[T_1] $\subseteq$ [T_2] | T_1,T_2 $\in$ N and a specialization edge in E is orginating in T_1 and ending in T_2}

[5] The mapping defined here is a simplification of the mapping used by the Iris prototype implementation

[6] Note that cross-product constraints are special cases of join dependencies

and Σ is the set of inter-relational inclusion dependencies $\{[R] \subseteq [T] \mid R$ is a role label of an edge adjacent to the non-literal type $T\} \bigcup \{[T] \subseteq [R] \mid T$ is a non-literal type and R a role label of an edge adjacent to T, and $R \in Y$ for some Y such that $Y[1,k] \in C_f$ for some function label f and $k \geqslant 1\}$

There is a straightforward one-to-one correspondence between the instances of an Iris schema s and the instances of the relational schema $\text{RelTrans}_1(s)$ We omit here the formal definition of this correspondence

While the relational schemas provided by RelTrans_1 are capable of representing the same information as Iris schemas, they are not entirely satisfactory because they cannot be characterized solely in terms of the kind of integrity constraints used in the schemas This is due to the presence of the "types" relation, which is distinguished from the other relations in that it can not satisfy arbitrary cross-product and multiplicity constraints and plays a special role in the schema The second stage of the translation eliminates this inconvenience by removing the "types" relation[7] and adding to the remaining schema the inclusion dependencies which follow from the presence of the types relation This is defined next

Definition Let RelTrans_2 be the mapping defined on the range of RelTrans_1 as follows For each relational database schema $<S,\Sigma>$ in $\text{range}(\text{RelTrans}_1)$, $\text{RelTrans}_2(<S,\Sigma>) = <S_1,\Sigma_1>$ where

(i) $S_1 = \{<\text{name}, X, \Sigma_{\text{name}}'> \mid <\text{name}, X, \Sigma_{\text{name}}> \in S$, name $\neq$ types, and $\Sigma_{\text{name}}' = \Sigma_{\text{name}} \cup \Pi_X(\Sigma^*)\}$, and

(ii) Σ_1 is the set of inter-relational constraints in Σ^* which do not involve the relation "types"

Notation Let RelTrans be the composition of RelTrans_1 and RelTrans_2

Example 3 1 We will show how to translate a sub-part of the schema defined in Example 2 1 to a corresponding relational schema using mappings RelTrans_1 and RelTrans_2 The Iris sub-schema to be translated contains all the types of the original schema, the functions Id and Enrollment, and all the participation constraints defined on those two functions

Applying RelTrans_1 produces the "types" relation and the function relations Id and Enrollment illustrated in Figure 3 2

types

Person	Instructor	Student	Course	Semester

Id

Person	String

Enrollment

Student	Course	Semester	Grade

Figure 3 2 Relations produced by RelTrans_1

The inclusion dependencies, cross-product constraints, and multiplicity constraints induced by RelTrans_1 are shown in Figure 3 3

$[\text{Instructor}_{\text{types}}] \subseteq [\text{Person}_{\text{types}}]$
$[\text{Student}_{\text{types}}] \subseteq [\text{Person}_{\text{types}}]$
$[\text{Person}_{\text{Id}}] \subseteq [\text{Person}_{\text{types}}]$
$[\text{Person}_{\text{types}}] \subseteq [\text{Person}_{\text{Id}}]$
$[\text{Student}_{\text{Enrollment}}] \subseteq [\text{Student}_{\text{types}}]$
$[\text{Course}_{\text{Enrollment}}] \subseteq [\text{Course}_{\text{types}}]$
$[\text{Semester}_{\text{Enrollment}}] \subseteq [\text{Semester}_{\text{types}}]$
$[\text{Course}_{\text{types}}] \subseteq [\text{Course}_{\text{Enrollment}}]$
$[\text{Semester}_{\text{types}}] \subseteq [\text{Semester}_{\text{Enrollment}}]$

$\otimes \text{Person}_{\text{Id}}$
$\otimes \text{Course}_{\text{Enrollment}}\text{Semester}_{\text{Enrollment}}$

$\text{Course}_{\text{Id}}[\leqslant 1]$
$\text{String}_{\text{Id}}[\leqslant 1]$
$\text{Course}_{\text{Enrollment}}\text{Semester}_{\text{Enrollment}}[\leqslant 24]$
$\text{Student}_{\text{Enrollment}}\text{Semester}_{\text{Enrollment}}[\leqslant 5]$
$\text{Student}_{\text{Enrollment}}\text{Course}_{\text{Enrollment}}[\leqslant 3]$
$\text{Student}_{\text{Enrollment}}\text{Course}_{\text{Enrollment}}\text{Semester}_{\text{Enrollment}}[\leqslant 1]$

Figure 3 3 Constraints produced by RelTrans_1

Applying RelTrans_2 to the relations and constraints of Figure 3 2 and Figure 3 3 produces the relational schema of Figure 3 4 The "types" relation is eliminated together with all its inclusion dependencies Notice how the only inclusion dependency of the schema in Figure 3 4 is deduced from the closure of the inclusion dependencies of Figure 3 3

Remarks (i) While RelTrans_1 is a one-to-one mapping from Iris schemas into relational schemas, RelTrans is not one-to-one Thus, several distinct Iris schemas can be mapped to the same relational schema by the RelTrans mapping

(ii) For each Iris schema s, there is a straightforward correspondence between the instances of s and the instances of the relational schema RelTrans(s) The correspondence is not one-to-one, because objects which do not participate in any Iris function are not represented in the relational translation

(iii) The mapping RelTrans describes the correspondence between Iris schemas and relational schemas at the logical level The actual implementation of Iris schemas differs slightly from this description For instance, a record is maintained of the objects which are members of each non-literal type Also, different relations are sometimes "clustered" to improve performance

[7] The current Iris implementation maintains types in stored relations

Id

Person	String

Enrollment

Student	Course	Semester	Grade

$[\text{Student}_{\text{Enrollment}}] \subseteq [\text{Person}_{\text{Id}}]$

$\otimes\ \text{Person}_{\text{Id}}$
$\otimes\ \text{Course}_{\text{Enrollment}}\text{Semester}_{\text{Enrollment}}$

$\text{Course}_{\text{Id}}[\leqslant 1]$
$\text{String}_{\text{Id}}[\leqslant 1]$
$\text{Course}_{\text{Enrollment}}\text{Semester}_{\text{Enrollment}}[\leqslant 24]$
$\text{Student}_{\text{Enrollment}}\text{Semester}_{\text{Enrollment}}[\leqslant 5]$
$\text{Student}_{\text{Enrollment}}\text{Course}_{\text{Enrollment}}[\leqslant 3]$
$\text{Student}_{\text{Enrollment}}\text{Course}_{\text{Enrollment}}\text{Semester}_{\text{Enrollment}}[\leqslant 1]$

Figure 3 4 Relational schema produced by RelTrans_1 followed by RelTrans_2

The following result characterizes all relational schemas which are images of Iris schemas under the RelTrans mapping As suggested earlier in this section, the characterization involves solely the kinds of constraints present in the schema

3 1 Theorem For each relational database schema r whose only constraints are unary inclusion dependencies, cross-product constraints, and multiplicity constraints, there exists an Iris schema s such that RelTrans(s) = r

The proof of Theorem 3 1 consists of finding a mapping which, for every relational schema r of the type described in the theorem, provides a translation SemTrans(r) to a (semantic) Iris schema such that RelTrans(SemTrans(r)) = r The main component of the construction of SemTrans(r) consists of finding an assignment of attributes (roles) to types In particular, one type is associated with all attributes belonging to a cycle of inclusion dependencies implied by the constraints of r Consequently, the subtyping structure of the resulting Iris schema is guaranteed to be acyclic Furthermore, it can be shown that SemTrans(r) has the property that hidden type "aliasing" cannot occur, that is, two types of the schema cannot be forced to always contain the same objects in every valid instance of the schema (It can be verified that, in arbitrary Iris schemas, "aliasing" can indeed occur, even though the original sub-typing specification is acyclic - see Example 3 4)

The mapping of Iris schemas to relational schemas provides a new, formal mechanism for comparing different Iris schemas In particular, various notions of equivalence of Iris schemas can be defined based on their relational translations We next define the simplest such notion of equivalence

Definition Two Iris schemas s_1 and s_2 are equivalent if and only if[8] $(\text{RelTrans}(s_1))^* = (\text{RelTrans}(s_2))^*$

As a first application of the above definition we next compare different Iris schemas with respect to their type structure In particular, it may be of interest to simplify the type structure of a given Iris schema according to certain criteria, such as the number of different types in the schema This leads to the following

Definition An Iris schema s is type-minimal iff there is no equivalent Iris schema s′ with fewer types

The following result shows that there is exactly one type-minimal Iris schema equivalent to a given Iris schema

3 2 Theorem If s_1 and s_2 are equivalent, type-minimal Iris schemas, then they are isomorphic [9]

It can be shown that each Iris schema which is the translation SemTrans(r) of a relational schema r is type-minimal [If G is the graph whose vertices are the attributes of r and whose edges correspond to inclusion dependencies, the number of types in the Iris translation[10] is | { v| v is a vertex of G, id(v) ⩾ 1, and v is not on a cycle of G } | + | { c| c is a maximal cycle of G } | + sign (| { v| v is an isolated vertex of G } |)] In particular, we can use the relational translation of an Iris schema to find an equivalent Iris schema which is type-minimal Indeed, we have

3 3 Theorem For every Iris schema s, SemTrans (RelTrans(s)) is equivalent to s and type-minimal

In practice, the above results on type minimality can be used as guidelines for designing the type structure of an Iris schema Indeed, it may be the case that the initial breakdown of objects into types is unnecessarily refined, in that the schema does not take advantage of certain type distinctions In this case some of the types can be merged Following is a simple example of a schema where type simplification is possible due to aliasing

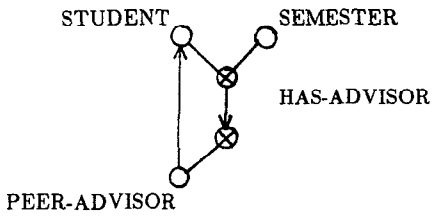
Example 3 4 The Iris schema represented in Figure 3 5 is a fragment of a schema for an academic institution, where each student must be assigned a peer advisor for each semester of the academic year The schema contains the three types STUDENT, PEER-ADVISOR, and SEMESTER Each peer advisor is a student, so PEER-ADVISOR is a subtype of STUDENT The function HAS-ADVISOR assigns a peer-advisor for every student-semester pair

---

[8] For each relational database schema s, s* denotes the schema containing the same relations as s and whose constraints are the logical closure of the constraints of s

[9] Informally, two Iris schemas are isomorphic if they are the same except for the choice of labels We omit the formal definition here

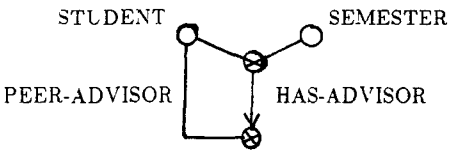
[10] If n is a non-negative integer then sign(n) = 0 if n = 0 and sign(n) = 1 if n > 0



STUDENT SEMESTER [1,∞]
SEMESTER PEER-ADVISOR [0,1]

Figure 3 5

The first participation constraint indicates that each student must be assigned an advisor each semester, the second indicates that a peer advisor can only advise one student in a given semester It can be easily verified that, due to these constraints, each student must also be a peer advisor Thus, the types STUDENT and PEER-ADVISOR are aliases (they contain exactly the same objects) It follows that PEER-ADVISOR can be eliminated as a type and kept simply as a role edge for students The resulting schema (Figure 3 6) is equivalent to the first, and is type-minimal



STUDENT SEMESTER [1,∞]
SEMESTER PEER-ADVISOR [0,1]

Figure 3 6

4 Integrity Constraints in Relational Translations of Iris Schemas

In this section we examine in detail the integrity constraints induced by Iris schemas on their relational translations Since Iris schemas are implemented using relational translations, such constraints are of paramount importance for reasons familiar from relational database theory First, the constraints can be used to maintain the semantic integrity of the database, second, they are necessary for query optimization, and third, the constraints are used to verify that the relational schema satisfies the traditional criteria of good relational schema design (in order to avoid redundancy and update anomalies, and to obtain efficient implementations) As shown in the previous section, relational translations of Iris schemas can be characterized using unary inclusion dependencies, cross-product constraints, and multiplicity constraints Thus, it is important to develop mechanisms for computing logical closure for these constraints, such as a finite set of inference rules Additionally, it is of interest to infer all functional dependencies implied by such sets of constraints, in order to verify that the relational translations are in normal form (BCNF) Suprisingly, our results show that arbitrary Iris schemas are not well-behaved with respect to the above goals, and that other unexpected problems arise In particular, we show that there is no finite axiomatization for unary id's, multiplicity constraints, and cross-product constraints, and that relational translations are not always in Boyce-Codd Normal Form, as was initially assumed However, we do provide a powerful (but incomplete) inferencing mechanism for the constraints The inferencing mechanism is novel in that it involves a set of equations associated with the constraints, and is more powerful than any finite set of traditional inference rules Finally, we exhibit a restriction on Iris schemas which guarantees that they are well-behaved with respect to all the criteria discussed

We next look at properties of relational translations of arbitrary Iris schemas, that is, relational schemas whose constraints are arbitrary sets of unary id's, multiplicity constraints, and cross-product constraints The first problem we address is that of finding inference rules for the constraints The following result shows that, unfortunately, there is no finite axiomatization for such constraints

4 1 Theorem There is no finite set of inference rules for computing the logical closure of a set of (inter and intra-relational) unary inclusion dependencies, (intra-relational) cross-product constraints and (intra-relational) multiplicity constraints

While there is no complete finite axiomatization for our constraints, it is desirable to have as powerful an inference mechanism as possible for such constraints We next present an inference mechanism which uses, in addition to traditional inference rules, a set of equations associated with the constraints It turns out that the inference mechanism is as powerful as an infinite set of independent traditional inference rules Despite this fact, the inferencing mechanism is not complete

Intuitively, the equations used in the inferencing mechanism express relationships between the number of distinct values occurring in different columns of a relation and the number of tuples in the relation These equations, called "cardinality equations," are defined next

Definition Let $s = < S, \Sigma >$ be a relational database schema whose constraints are unary id's, multiplicity constraints, and cross-product constraints Let $k = \max \{ i \mid X[\leqslant i]$ is a multiplicity constraint in some relation schema in S$\}$ For each attribute A, let a be a new symbol The set of symbols associated with s is

$$\text{Symb}(s) = \{ a_{rel} \mid < rel, R, \Sigma_0 > \in S \text{ and } A \in R \} \cup \{ t_{rel} \mid < rel, R, \Sigma_0 > \in S \} \cup \{ j \mid 0 \leqslant j \leqslant k \}$$

Subscripts are omitted when the relation name is understood The symbol a_{rel} represents the number of distinct values in column A of relation rel, the symbol t_{rel} stands for the number of tuples in relation rel

A cardinality equation over s is an expression of the form $u \leqslant v$ or $u = v$, where u and v are finite, non-empty sequences of symbols in Symb(s) of length at most $\max \{ |R| \mid < rel, R, \Sigma > \in S \} + 1$

For instance, consider a relation schema $< rel, AB, \{ A[\leqslant 2] \} >$ Then $t \leqslant 2ab$ and $a \leqslant b$ are examples of cardinality equations over this schema A relation r over the schema satisfies the equation $t \leqslant 2ab$ if

$|r| \leqslant 2 |\Pi_A(r)| \cdot |\Pi_B(r)|$ A relation r satisfies $a \leqslant b$ if $|\Pi_A(r)| \leqslant |\Pi_B(r)|$ We omit here the formal definition of satisfaction of a cardinality equation by an instance of a database schema, since this should be clear from the example Note that there is just a finite number of cardinality equations over a given schema Also, it can be shown that it is decidable whether one cardinality equation implies another cardinality equation Indeed, implication can be expressed as a universally quantified sentence in the first-order language over $\mathbb{N}$ with multiplication, $\leqslant$, and $=$ [For example, the fact that $a \leqslant b$ and $b \leqslant c$ imply $a \leqslant c$ is formally written as

$$\forall\, a\, \forall\, b\, \forall\, c\, ((a \leqslant b \wedge b \leqslant c) \Rightarrow a \leqslant c)\,]$$

The decidability of such sentences follows easily from the fact that the theory of ordered abelian groups is decidable (see [Mo]), so, in particular, the theory of the ordered rational numbers with multiplication is decidable (We omit the details here) The logical closure of a set E of cardinality equations over a schema s is denoted by E^* and consists of all cardinality equations over s implied by E

We next show how to associate a set of cardinality equations with relational database schemas

Definition Let $s = < S, \Sigma >$ be a relational database schema with unary id's, cross-product constraints, and multiplicity constraints Let EQN(s) consist of the following cardinality equations

(i) for each attribute A of relation schema rel, $t_{rel} \geqslant a_{rel}$ is in EQN(s),

(ii) for each relation schema rel over attributes $A_1 \ldots A_n$, $t_{rel} \leqslant a_1 \ldots a_n$ is in EQN(s),

(iii) for each inclusion dependency $[A] \subseteq [B]$ of s, $a_{rel_1} \leqslant b_{rel_2}$ is in EQN(s), where A and B are attributes in the relation schemas rel_1 and rel_2, respectively,

(iv) for each cross-product constraint $\otimes A_1 \ldots A_n$ of a relation schema rel, $t_{rel} \geqslant a_1 \ldots a_n$ is in EQN(s), and

(v) for each multiplicity constraint $A_1 \ldots A_n [\leqslant k]$ of a relation schema rel, $t_{rel} \leqslant k a_1 \ldots a_n$ is in EQN(s)

Note that EQN(s) is associated to the entire database schema s, not just to individual relation schemas

We next exhibit our set of inference rules, denoted by Inf The set Inf consists of two parts The first part, denoted by Inf_1, contains inference rules which involve only constraints The second part, denoted by Inf_2, describes how new constraints can be inferred from constraints in conjunction with equations in $EQN^*(s)$ We now list these rules (X and Y denote subsets of the attributes of a relation schema)

Inf_1

1 For each attribute A, $[A] \subseteq [A]$ and $\otimes A$
2 For each set X of attributes, $X [\leqslant \infty]$
3 $R [\leqslant 1]$, where $< rel, R, \Sigma >$ is in the database schema
4 If $[A] \subseteq [B]$ and $[B] \subseteq [C]$ then $[A] \subseteq [C]$
5 If $X [\leqslant k]$ and $X \subseteq Y$ then $Y [\leqslant k]$
6 If $\otimes X$ and $Y \subseteq X$ then $\otimes Y$
7 If $X [\leqslant k]$ and $n \geqslant k$ then $X [\leqslant n]$

Inf_2

1 If $[A] \subseteq [B]$ and $b \leqslant a$ then $[B] \subseteq [A]$
2 If $A_1 \ldots A_n [\leqslant k]$ and $t \geqslant k a_1 \ldots a_n$ then $\otimes A_1 \ldots A_n$
3 If $\otimes A_1 \ldots A_n$ and $t \leqslant a_1 \ldots a_n$ then $A_1 \ldots A_n [\leqslant 1]$

Let Σ^+ be all the constraints which can be inferred from Σ and $EQN^*(s)$ using the above rules It is easily seen that the rules are sound, i e , $\Sigma^+ \subseteq \Sigma^*$ It is of interest to note that the new constraints generated using the inference rules do not imply new cardinality equations Thus, the closure of the cardinality equations never has to be recomputed Indeed, we have

4 2 Lemma Let s be a relational database schema with unary id's, cross-product constraints, and multiplicity constraints Let s^+ be the database schema with the same relations as s, whose constraints are all the constraints which can be inferred from the constraints of s using Inf Then $EQN^*(s) = EQN^*(s^+)$

Remark As stated earlier, the set of inference rules $Inf_1 \cup Inf_2$ can replace an infinite set of independent traditional inference rules for our constraints We now give an example that shows how our rules involving equations can replace one traditional inference rule Consider the rule

if $\otimes AB$, $[C] \subseteq [A]$, and $BC [\leqslant 1]$

then $AB[\leqslant 1]$, $[A] \subseteq [C]$, and $\otimes BC$

The set EQN of equations associated with $\otimes AB$, $[C] \subseteq [A]$, and $BC[\leqslant 1]$ is $\{ t \geqslant ab, c \leqslant a, t \leqslant bc \}$ It is easily seen that $t \leqslant ab$, $a \leqslant c$, and $t \geqslant bc$ are in EQN^* From $t \leqslant ab$ and $\otimes AB$ we obtain $AB [\leqslant 1]$, using rule 3 of Inf_2 From $a \leqslant c$ and $[C] \subseteq [A]$ we obtain $[A] \subseteq [C]$ using rule 1 of Inf_2 And, from $t \geqslant bc$ and $BC [\leqslant 1]$ we obtain $\otimes BC$ using rule 2 of Inf_2

We have seen that relational translations of arbitrary Iris schemas give rise to certain unexpected problems concerning axiomation of constraints We now briefly discuss some additional problems arising in this context

(i) Side-effect fd's

It is of interest to know whether or not relational translations of Iris schemas are in Boyce-Codd Normal Form Based on intuitive grounds, it has so far been assumed that the only fd's satisfied in these translations are key fd's which arise from explicitly constraining upper-bound participations to 1 In other words, given a relational database s corresponding to an Iris schema, it was assumed that the only fd's holding in a relation schema $< rel, R, \Sigma >$ of s are of the form $X \rightarrow R$, where $X[\leqslant 1]$ is in Σ We informally call all other fd's which might hold in such a schema side-effect fd's The inference rule exhibited in the previous remark shows that side-effect fd's do in fact exist (since the inferred constraint $AB [\leqslant 1]$ implies that AB is a key) However, the side-effect fd produced in the remark does not violate BCNF We now show that there are side-effect fd's which violate BCNF

4 3 Example Consider a relation schema with attributes ABCDEF and constraints $\{ \otimes AC, \otimes AD, \otimes BC, \otimes BD, \otimes CD, \otimes CE, \otimes DF, AC [\leqslant 1], AD [\leqslant 1], BC [\leqslant 1], BD [\leqslant 1], CD [\leqslant 1], E [\leqslant 2], F [\leqslant 2] \}$ It can be shown that the fd $A \rightarrow B$ must then hold in every instance of the schema Thus $A \rightarrow B$ is a side-effect fd violating BCNF

Obviously, it is of interest to infer all fd's satisfied by relational translations of Iris schemas However, no finite set of inference rules exists for computing the fd's implied by the constraints involved in the relational translations

(ii) Non-trivial satisfiability

Clearly, each set of unary id's, cross-product constraints and multiplicity constraints is trivially satisfiable by a database instance consisting of a single tuple in each relation We call a database schema non-trivially satisfiable if it has instances other than the single-tuple instance (Thus, the original Iris schema has instances containing more than one object in each type) It can be seen that there are relational translations of Iris schemas which are not non-trivially satisfiable Some of these schemas can be detected using the associated set of equations EQN (Indeed, if a = 1 is in EQN* for every attribute A, then the only instance of the schema is the single tuple instance) However, it can be shown that there are schemas which are not non-trivially satisfiable, but this cannot be detected using the equations Such a schema can be obtained from the schema in Example 4 3 by adding $\otimes$ AB to its constraints

(iii) Type cardinality restrictions

Some Iris schemas imply restrictions on the maximum number of objects of a given type This can be seen from the fact that, in relational translations of Iris schemas, the associated set of equations may imply constraints such as $a \leqslant k$ for some attribute A For instance, consider the set of constraints $\{ \otimes AB, B[\leqslant 5] \}$ It is easily seen that $a \leqslant 5$ is implied by the equations associated with this set of constraints Thus, the A column can contain at most 5 distinct values, and the type corresponding to the role A in the original Iris schema may contain at most 5 objects

(iv) Non-localness

We call an Iris schema local iff the participation constraints for different functions are independent It can be seen (using relational translations) that not all Iris schemas are local Thus, participation constraints for one function may imply participation constraints for another function This phenomenon can be undesirable and may lead to complications

In the remainder of the section we present a restriction on Iris schemas and their relational translations which guarantees that the schemas are well-behaved with respect to all the criteria discussed so far, in particular finite axiomatization and normal forms Clearly, different such restrictions can be found The one we present here has the advantage of being relatively weak and the disadvantage of being rather complicated However, a variety of simpler (but stronger) restrictions can easily be inferred from the one we present We now formulate our restriction in terms of relational schemas The restriction has two components one involving the intra-relational constraints, and a second involving the inter-relational inclusion dependencies

Definition A relational database schema $< S, \Sigma >$ whose constraints are unary id's, multiplicity constraints and cross-product constraints is restricted if the following conditions hold

(i) For each relation schema $< rel, R, \Sigma_{rel} >$ in S, if $X[\leqslant k]$ and $\otimes Y$ are in Σ_{rel} and $[A] \subseteq [B]$ where $A \in X$ and $B \in Y$, then $Y \subseteq X$, and

(ii) There is no sequence $< rel_i, R_i, \Sigma_{rel_i} >$, $1 \leqslant i \leqslant n$, of distinct relation schemas of S, and attributes $A_1 \ldots A_n$, $B_1 \ldots B_n$, such that $A_i \in R_i$, $B_i \in R_i$ for each i $(1 \leqslant i \leqslant n)$, $[A_i] \subseteq [B_{i+1}]$ $(1 \leqslant i < n)$, $[A_n] \subseteq [B_1]$, and $A_1 \neq B_1$

Informally, condition (ii) states that two distinct attributes of the same relation cannot be "connected" using the inter-relational inclusion dependencies of the schema

For instance, the schema of Example 3 1 (Figure 3 4) is restricted

The formulation of the above restriction in terms of Iris schemas is similar and is omitted (Recall that a constraint of the type $X[\leqslant k]$ corresponds in to an Iris upper-participation k for X, while $\otimes$ Y corresponds to lower participation one for Y)

We next show that the restriction proposed on relational schemas eliminates the problems encountered in unrestricted schemas The following result concerns finite axiomatization for restricted database schemas

4.4 Theorem The set of inference rules Inf_1 is sound and complete for restricted relational database schemas

Note that the rules of Inf_1 do not involve the set of equations associated with the schema Thus Inf_1 is a proper finite axiomatization for the constraints of restricted schemas

The next result concerns the issue of side-effect fd's and BCNF

4 5 Theorem Each relation schema $< rel, R, \Sigma >$ of a restricted relational database schema is in BCNF Furthermore, a non-trivial fd $X \rightarrow A$ holds in the relation schema iff $X[\leqslant 1] \in \Sigma$

In particular, there are no side-effect fd's in restricted relational database schemas

Finally, restricted schemas do not give rise to problems concerning satisfiability, localness, or type cardinality constraints Indeed, the following can be shown

(i) Each restricted relational database schema is non-trivially satisfiable

(ii) Each restricted relational database schema s is local (i e , the constraints in s^* which apply to a given relation schema $< rel, R, \Sigma_{rel} >$ of s are equal to Σ^*_{rel}, and thus can be computed locally)

(iii) Let s be a restricted relational database schema, for each integer n there exists an instance of s such that each column of a relation in the schema has at least n distinct values Thus, no type cardinality constraints are implied by the constraints of the schema

5 Conclusions

A formal framework for understanding the connection between semantic data models and the relational model was developed in this paper The translation of Iris schemas to relational schemas was shown to give rise to constraints different than those usually encountered in relational databases The results obtained so far concern mostly these types of constraints and their connection with traditional Normal Forms involving fd's The formal approach to these problems reveals that they are more complex than was assumed originally based on intuitive grounds For instance, relational translations of Iris schemas are not always in

BCNF, as was expected, due to the presence of "side-effect" fd's, and the constraints of these schemas are not finitely axiomatizable Other subtle problems concerning Iris schemas were also brought to light, such as type aliasing, trivial satisfiability, and non-localness The restriction exhibited on schemas provides guidelines as to how such problems can be avoided when designing Iris schemas

The Iris model only contains simple constructs common to most semantic models However, the techniques developed for Iris provide insight into the problems arising when more complex constructs are allowed in the semantic model For instance, semantic schemas allowing types which are disjoint unions or cross-products of types give rise to relational constraints for which the implication problem is undecidable (Indeed, specialization involving types that are cross-products of types gives rise to non-unary inclusion dependencies which, in conjunction with fd's induced by participation constraints, lead to undecidability of implication (see [ChVa83]), in the case of disjoint unions of subtypes, the equations associated with the schema involve multiplication as well as addition, so implication involves the theory of integers with addition and multiplication, which is undecidable [Mo76]) In particular, it is undecidable whether relational translations of such schemas are in BCNF, whether they are non-trivially satisfiable, whether aliasing occurs, or whether type cardinality restrictions are implied Such results provide new insight into trade-offs between expressiveness and tractability of semantic schemas

Several questions concerning Iris schemas which were not mentioned in this paper can fruitfully be translated in relational terms One such problem is computing the participation constraints for derived functions Another problem is determining whether a given Iris schema allows all objects of non-literal types to be uniquely identified using literal types and Iris queries (In relational terms, this corresponds to finding a select-project-join query whose result is a relation with both representable and non-representable types, satisfying certain fd's) Such problems will be explored in future work

Finally, note that the mapping between models described in this paper concerns exclusively static aspects of the models It is of interest to include in this correspondence dynamic aspects of the models Iris schemas with update operations would then translate to relational schemas involving both constraints and transactions, such as those discussed in [AbVi85] The results already obtained concerning the connection between transactions and constraints in relational databases could be used to validate update semantics in Iris schemas

Acknowledgements

The authors would like to thank Richard Hull and Bill Kent for their helpful comments on earlier versions of this paper

References

[AbHu84] Abiteboul, S and Hull, R IFO A Formal Semantic Database Model (Preliminary Report) *Proceedings ACM SIGACT-SIGMOD Symp on Principles of Database Systems,* 1984

[AbVi85] Abiteboul, S and Vianu, V Transactions and constraints *Proceedings ACM SIGACT-SIGMOD Symp on Principles of Database Systems,* 1985

[BeFe83] Beech, D and Feldman, J S The Integrated Data Model A Database Perspective *Proceedings of the Ninth International Conference on Very Large Data Bases,* Florence, Italy, October 1983

[Ca83] Cattell, R G G Design and Implementation of a Relationship-Entity-Datum Data Model Technical Report CSL 83-4, Xerox Corporation, Palo Alto Research Center, May 1983

[CFP82] Casanova, M A , Fagin, R , Papadimitriou C H Inclusion dependencies and their interaction with functional dependencies *Proceedings ACM SIGACT-SIGMOD Symp on Principles of Database Systems,* 1982, pp 55-62

[ChVa83] Chandra, A and Vardi, M The implication problem for functional and inclusion dependencies is undecidable IBM Research Report, RC 9980 (44299), 1983

[DeKeLy85] Derrett, N , Kent, W , and Lyngbaek, P Some Aspect of Operations in an Object-Oriented Database *IEEE Database Engineering Bulletin* December, 1985

[Fi87] Fishman, D et al Iris An Object-Oriented DBMS *ACM Transactions on Office Information Systems,* 4(2), April 1987 (to appear)

[Ku83] Kulkarni, K G *Evaluation of Functional Data Models for Database Design and Use* Ph D Thesis, University of Edinburgh, 1983

[LyKe86] Lyngbaek, P and Kent, W A Data Modeling Methodology for the Design and Implementation of Information Systems *Proc Int'l Workshop on Object-Oriented Database Systems,* Pacific Grove, California, September 1986

[Ma83] Maier, David *The Theory of Relational Databases* Computer Science Press, 1983

[Mo76] Monk, Donald *Mathematical Logic* Springer Verlag, 1976

[MyBeWo80] Mylopoulos, J , Bernstein, P A , and Wong, H K T A Language Facility for Designing Database-Intensive Applications *ACM Transactions on Database Systems* 5(2) 185-207, June, 1980

[Sh81] Shipman, D The Functional Data Model and the Data Language DAPLEX *ACM Transactions on Database Systems* 2(3) 140-173, March, 1981

[SmSm77] Smith, J M and Smith D C P Database Abstractions Aggregation and Generalization *ACM Transactions on Database Systems* 2(2) 105-133, June, 1977

[TsZa84] Tsur, S and Zaniolo, C An Implementation of GEM - supporting a semantic data model on a relational back-end *Proceedings of the ACM SIGMOD International Conference on Management of Data* Boston, Ma , June, 1984

[Ul82] Ullman, Jeffrey *Principles of Database Systems* Computer Science Press, 1982