Specifying a Semantically Adequate Structure for Information Systems and Databases

by

C.N.G. Dampney
Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
N2L 3G1

Research Report CS-87-28
April 1987

Specifying a Semantically Adequate Structure for Information Systems and Databases

Abstract

Information system specification using the entity relationship approach is widespread. However there are many variations in the methodology, and difficulties in continuing the specification process through to the dynamics of the system. We introduce the semantically adequate structure condition as a way of testing a specification so that it can be successfully expanded and refined. The condition requires we test the specification by its information and data semantics, respectively against 1) the (user's perception of) the real world, and 2) satisfiability on the computer.

Our objective is to demonstrate specification through refinement stages starting with a base structure and then systematically adding constraints. The methodology uses familiar diagrammatic E-R notation so the specification can be easily validated as it proceeds. What each refinement stage must satisfy is that its constraints are satisfiable on the model so far specified. In the process we succeed in characterising entity subtypes in a way that satisfies both intuition and rigour. This enables us to formulate the E-R model within a model theoretic framework and take advantage of the mathematical tools that theory provides.. This ensuresthe correctness of the specification process without becoming diverted by too much formality.

Finally an examination of the relational data model succeeds in gaining insight into what multi-valued and cyclic join dependencies really mean. These insights are a result of being able to a) disambiguate the semantics of a table, and b) properly deal with cycles in E-R models.

C.N.G. Dampney

(Computing Discipline, Macquarie University, New South Wales 2109, AUSTRALIA)

1. Introduction

What constitutes a good specification for a databased information system and how does one begin? This question is faced by the professional information system analyst, the database engineer, the experienced database researcher, and neophyte alike. The question is important because in creating a new database one must be assured that its foundations are sound. We, who use the database, need to be sure that the data stored in it correctly reflects what is happenning in the real world.

In general terms the answer to the question is that the specification for a databased information system should begin by defining *structure*. The answer coincides with positive experience gained and demonstrated by many practitioners using variants of the *Entity-Relationship approach* [1,2,3] for information (systems) analysis over the last decade. By structure we mean *mathematical structure* on which *theories (formal expressions)* can be *satisfied* [e.g.4] . The structure specified must be able to satisfy the kinds of constraints that information sytems analysts use in practice.

The paper emphasizes structure and semantic notions. We show that the essential criterion that must be satisfied by the structure is that it must be *semantically adequate* in a sense more fully

described below. This criterion causes us to distinguish clearly between modelling a) real world information, and b) data on the computer - a distinction long recognised and by many authors [5,6,7].

Section 2 has self contained examples of specifying structure using a refinement of the E-R approach. We show how a specification can be built while checking that it satisfies the requirements of real world information needs. Thus specification and validation proceed hand in hand. The *connection* constructor is introduced as a method for unambiguous specification of the scope of the structure over which the constraints apply. We then provide a method for specifying *subtypes* and show how subtypes define a base for *transition constraints* on which the dynamics of an information system can be further specified. These specification methods form the *structure specification level* on which the next level of refinement - *behavioural specification* can be applied.

In section 3 we provide a model theoretic framework for specifying this *refined* E-R structure. The constraints that have long been used on the E-R model are functional (FDs) and inclusion (IDs) constraints. We prove that constraint specification must also include *the constarints' scope of the application*, which is that part of the structure to which the constraint applies. We also derive overall *restrictions* that an E-R model must have for satisfiability. In particular the issue of cycles in the structure is examined.

In section 4 we consider the Relational DataBase (RDB) data model. The *refined E-R model* sheads insight into the nature of relational data model *constraints*. We show that multi-valued dependencies and cyclic join dependencies are characterised by IDs. The implication problem for functional and inclusion constraints [7,8,9] specified on a refined E-R structure (that meets the restrictions referred to in the above paragraph), turns out to be decidable.

There is an underlying assumption in this paper which should be made explicit. It is assumed that we are dealing with information systems for which specification is possible. Such information systems are those, for example, used for business and administration where the real world system being simulated on the computer is sufficiently well understood that we know enough to specify it. In contrast, Artificial Intelligence tackles problems with the objective of discovering their specifications (knowledge) assisted by automated means.

1.1. Specification and Semantic Adequacy

We return to the question in the first paragraph of the paper and develop the definition for a *semantically adequate structure*. For a database to be *"well-specified"* it would have to satisfy the following condition -

> The semantically adequate structure condition (version 1). *That the data in a "well specified" database must relate correctly to the real world information it is meant to represent.*

We note that the real world is as it is perceived as *information* in the mind of the user. Let us accept that we perceive and are able to describe information about the real world in terms of things (entities) and the relationships between them (figures 4 and 5).

We define correctness on the basis of the laws of (some) logic. Such laws are expressed as rules of inference. This allows us to *model* 1) how the user deduces and checks the correctness of information, and 2) how the database can check correctness by determining whether data being stored satisfies the model.

There are thus two separable parts of the condition that respectively concern the database and the perceived real world. The condition now becomes -

The semantically adequate structure condition (version 2). *That (i) (information condition) the information about the real world is correct, and (ii) (data condition) the data stored and the deduction of derived data on the computer is correct.*

These conditions are made decidable if a *model* can be developed which 1) supports the description of real world information, and 2) can be implemented on the computer.

The distinction between the information and data conditions is important because there are semantic issues which respectively belong to 1) *information analysis* which aims to discover satisfactory models for real world phenomena, and 2) *database design* which aims to *approximate* and translate the information model to a computer based data model. In practice the approximation may instead be accomplished during information analysis. Approximation is often necessary because a computer based model must be *countable*. The processes of information analysis and database design are illustrated in figure 1.

In the past we have confused information analysis and database design. Indeed papers on Relational DataBase design used to assert that RDB structures (relations, operators and constraints) are appropriate mathematical structures which can describe the real world. As is well documented [10], and evident in recent success increasing the semantic power of data models, RDB structures are semantically inadequate to the task. This issue is considered more carefully in section 4.

1.2 What is meant by semantics?

The word "semantics" takes on many shades of meaning according to the context in which it is used. As semantics itself is concerned with meaning the notion is hard to capture generally. We take as our informal definition for semantics (Montague[11]):-

Definition of Semantics. *Semantics is the relation between expressions and objects.*

The philosophical starting point for this paper is illustrated in figure 1. We have two kinds of semantic relations, viz:-

1) *information semantics* between the model and real world objects;
2) *data semantics* between the model and computer based objects.

Computer objects being man made are well understood - their general properties can be formulated in terms of decidability. We thus have a final version for the semantically adequate structure condition -

The semantically adequate structure condition (final version). *That (i) (**information**) the model structure can support the information needs, constraints, and behaviour of the real world system, and (ii) (**data**) the model structure elements are countable and the finite satisfiability of constraints on the model's structure is (semi-)decidable.*

The *data condition* is suggested by Bry and Mathey's recent result [12] that *finite satisfiability is semi-decidable* on the computer. By definition of the discrete componentry in a computer the number of model elements must be countable. The real world can not be so well formalised. We

as perceivers of the real world are ourselves the means to test the *information condition* by judging whether the information semantics of a model matches our perception. Providing we have suitable methods to express constraints the data conditions can be decided formally and objectively whereas the information condition must be subjectively judged by us.

The notable omission from the above condition is that it does not include the requirement that behaviour be computable. We separate the issue of behaviour specification and look at it briefly at the end of section 2.3.

1.3 The Thesis

The thesis in this paper is that a base entity-relationship structure refined by systematically adding constraints through the stages outlined in section 2 is a good methodology for specifying an information system.

2. Satisfaction of the Information Condition While Specifying Structure.

In this section we show how an information system specifcation can be constructed starting with a base structure created using the E-R approach. From the base structure the specification is refined by adding constraints until the possible states of the model match the possible states of the real world counterpart. During the process the information component of the semantically adequate structure condition can be checked. Thus specification and validation against the real world occur together.

As Furtado and Neuhold [13] do, we identify structure as the first level of specification. We call it the *structure level* of specification, rather than their term "information level", to prevent confusion with the use of the terms "information model" and "information analysis".

The structure level of specification can be divided into three separate stages of refinement.

2.1 Stage 1 - Specification of a well formed base structure.

The base structure must be able to support information needs and the constraints which further refine it. We now consider the information needs. The diagrammatic notation used to represent an E-R model is given in figure 2.

Figure 3 presents an E-R model of a course information system. Experience has shown that such a diagram, or its variants with other notation, are fairly easily understood by someone who also understands the real world information system being described. The information semantics of the model is thus tested and checked against that person's perception of the real world. The advantage of the diagrammatic notation over the corresponding entity set, attribute and relationship definitions in text form, is that the whole can be grasped at once.

There are considerable semantics even in the base structure. Figures 4 and 5 highlight that the E-R model is dealing with real world things of interest (entities). Note especially that an entity set is a set of the real world things themselves and is NOT the computer representation of those entities. Because an identifying attribute is one to one and the ID's are unique each entity instance, at least of a given type, should also be distinct. In contrast values sets have no *semantic significance* in their own right. For example we see in figure 4 that Location (STD_ADDRESS) has been specified as a value set. The classification of sets into entity sets or value sets is an information semantic issue and is therefore decided by the information analyst.

All many to many to ... n-ary relationships are decomposed into n unary functions from a new entity corrsponding to that relationship. Thus every relationship instance (*connection*) between entites is uniquely identifiable and their semantics can thus be fully explored. This is important when determining whether or not a relationship can be formed by composing others, because if it can, it is semantically redundant.

This base structure of entities, relationships and attributes defines a *well formed structure* [14]. It must be necessary (no redundant relationships or attributes) and sufficient to *support* the information needs.

2.1.1 Definition of the base structure functions.

It will be shown that it is the functions which really govern the structure, and hence it is important to carefully describe them.

Attribute functions map entity sets to values sets. Each entity set has a special *identifying attribute* which is an injective (one-to-one) total function onto its identifying attribute value set. The inverse of the identifying attribute function is therefore also a function. In database terms this attribute acts as a surrogate for the entity. The descriptive attributes are used to record facts about an entity. Attribute functions are not necessarily total functions, but if partial they are specifically called *optional attributes* .

Relationship functions map entity sets to entity sets. Relationship functions are either partial or total, injective (one-to-one) or non-injective, and surjective (onto) or non-surjective. The E-R approach uses the terms:- a) *mandatory participation of an entity* in a relationship if an entity is the domain (range) of a total (onto) function, and b) *optional participation of an entity* in a relationship if an entity is the domain (range) of a partial (into) function.

Thus E-R structures consist of:-

1) two classes of sets - *entity sets* and *value sets*, and
2) two classes of functions - *relationship functions* and *attribute functions*.

2.2 Stage 2 - Constraint Specification.

Constraints must be discovered by the information analyst and then specified. Figures 6 and 7 respectively exemplify binary functional and binary inclusion constraints. The examples demonstrate the following:-

> *When a constraint is specified, its scope of application over the structure must also be specified.*

While Maier [15] in particular makes this point, it is often not made clear in database design, and yet it is crucial.

Consider the functional dependency (STUDENT,TIME) --> CLASS in figures 3 and 6. In this case the constraint is over the derived structure defined by STUDENTs who have ENROLMENTs in COURSEs with CLASSes on at given TIMEs. While in this case this is the obvious structure, there are other possible, but semantically unlikely, structures that might include CANGIVE and STAFF. In a cyclic structure in particular, and most E-R models contain cycles, an implicit definition of constraint scope is ambiguous because there are alternative parts of the structure over which the constraints between attribute values could apply.

Inclusion constraints between (possibly composed) relationships are also important in information analysis. Most cycles in the structure will contain at least one binary inclusion dependency - see also section 3.4. Inclusion constraints also occur because, as we will see, an entity instance may be a member of more than one entity set. For example, someone could be both a STUDENT and a member of STAFF (figure 9). A constraint such as "No PERSON may be a STUDENT in a CLASS that he *Takes* as a STAFF member" can be specified as an exclusion constraint.

The information analyst must test and determine if the model structure can satisfy the *functional dependency, inclusion dependency and other constraints over their specified scope.* An unambiguous method for specifying the scope and constructing it using *the connection operator* was exemplified in figures 6 and 7. We now describe the method

2.2.1 Specifying Constraint Scope Unambiguously

Constraint scope is specified by deriving the structure to which it applies from the base structure. The heart of the specification method is the *connection* operator ("*") which defines ***connection*** $(A)f\,{}^*g$ analogous to (forward) ***composition*** $(A)f;g$. [Note that $(A)f;g$ is equivalent to $g \bullet f\ (A)$].

Suppose f : A -->> B and g : B --> C, then

$$(A)f\,{}^*g = \{<a,b_1,c_1>, <a,b_2,c_2>,...\}$$

whereas $(A)f;g = \{<a,c_1>, <a,c_2>,... \}$.

Thus ***connection*** retains all intermediate elements. Tables of tuples are thus constructed representing the derived structure so that the constraints can be tested

Connection produces, with one important exception, results similar to the ***join*** operator applied between relations with only one matching attribute. The exception is that as, and if, cycles of connections are completed attribute value sets are re-named to distinguish them from their previous occurrence - see example in section 4.4. The table constructed by connection contains additional columns which show the presence or absence of relationship instances between the entity sets (e.g. figure 6). This is done rather than display entity instances which of course cannot exist in the computer.

> *The advantage of the* ***connection*** *operator is that it produces semantically unambiguous tables* compared to the ***join*** when the entity sets involved are cyclically connected.

2.3 Stage 3 - Subtype Specification.

Now consider the way an entity participates in its relationships. Figure 8 shows that some participations may require other participations. Each subtype is defined by requiring mandatory participation in all its attributes and relationships. We can then define the subtype membership hierarchy and overlap membership structure from the real world information system's semantics as further constraints on the system. Specifically subtypes are constrained by inclusion and exclusion constraints. Figures 9 and 10 provide further examples. Figure 11 defines an entity's substructure in set theoretic form.

So far the functions defined in the base structure have been allowed to be partial functions. We now see that subtypes require all functions to be total and relationship functions to be surjective

(onto). The subtypes specify in increasing detail a refined substructure consisting of subsets E^n_m within each entity set E^n.

Thus **it is the functions that actually define entity (sub)types.** It is also important to notice that while within each (sub)type the entity instances must be distinct, an entity instance may be (simultaniously) a member of more than one entity (sub)type.

Transitions between states simulate the behaviour of the real world system. Therefore the model structure satisfies this aspect of the information condition.

Subtype specification provides a base for dynamic specification because an entity instance changes subtypes as its relationships change. In a database such transitions are caused by transactions. Independently the transitions between entity (sub)types may be constrained in another specification refinement stage. This stage belongs to the next level of specification [13], that is behavioural specification. We do not consider this further in this paper.

3 The Refined E-R Model.

We now provide a formal definition of the refined E-R model in model theoretic terms. This definition will be used to test that as the structure level of information systems specification is built, the specification satisfies the data part of the semantically adequate structure condition.

A model theoretic framework [e.g. 16,17] can be used to define the refined E-R structure. In particular we use a many-sorted model theoretic language. A (sub)type is a *sort*. Such a model theoretic framework ensures that the specification is well defined providing the constraints can be satisfied.

3.1 The model

We define:-

(1) A *many sorted vocabulary* :- $\mathbb{V} = (\tau, x, ...)$ where $\tau = \{s, ..., \boldsymbol{R}, ..., f, ..., c, ...\}$

where:-

s,...	are sort symbols: one for each entity (sub)type and value type;
R,...	are binary relation symbols (predicates) for each sort;
f,...	are unary function symbols such that $f : s_i \text{ --> } s_j$ for every pair of sorts (not necessarily distinct),
c,...	are constants of each sort, and
x,...	are variables over the constants of each sort;

and where:-

i) Each function symbol and constant is equipped with a sort symbol,
ii) The argument place in each unary function is equipped with a sort symbol,
iii) $\boldsymbol{R},... = (\{=,...,>,<,....\}, f^{-1})$ where there is a) a "=" for every sort; and b) ">","<", and other such predicates for the value type sorts. The sort symbols in both the argument places of each predicate are the same. The f^{-1} are relations defined by function inverses.

(2) A *(refined E-R) many-sorted structure* :- $\mathbb{S} = (S,...,\underline{R},...,\underline{f},...,\underline{c},... \}$

where:-

$S,...$ = (E,..., V,...) are the entity (sub)type sets and value sets corresponding to each sort,

$\underline{R},...$ are (interpreted) binary relations - *interpretations* of the predicates R,

$\underline{f}$ = (f_{attr} ,..., f_{rel},...) are the attribute and relationship (interpreted) functions -*interpretations* of the functions f,

$\underline{c},...$ = (e,...,v,...) are entity and value individuals - *interpretations* of the constants c.

The class of τ structures will be denoted by *Structure* [τ] where *Structure* : $\{\tau \text{ --> } \mathbb{S}\}$ is also called in first order logic the (set of) interpretations. We let $\tau_{\mathbb{S}}$ be the vocabulary of $\mathbb{S}$.

(3) An *E-R theory* :- $\mathbb{T} = \{constraint,...\}$

where each *constraint* = (γ,σ)... , and where :-

$\gamma \in$ {FD, ID,..} are well formed formulas expressing the complex functional , inclusion and other constraints.

$\sigma \in (\{s,...\}, \{f^{-1},...\})$ is a subset of the sorts and relations in the vocabulary τ that defines the scope of γ. σ characterises σ, a subset of the vocabulary $\tau \supseteq \sigma$, because only the following are retained:-

(i) those symbols of τ that have all their sort symbols in σ, and
(ii) the inverse function relations f^{-1} specified.

(γ,σ) means that γ must be satisfied on $\mathbb{S}\lceil\sigma$, the σ-*reduct* of $\mathbb{S}$, which using [16] is defined as the σ structure obtained from $\mathbb{S}$ by "forgetting" S for $s \notin \sigma$, and the $\underline{R}$ for $(\{=,...,>,<,....\},f^{-1}) \notin \sigma$

3.2 Logic

This model is studied within the framework of a multi-sorted logic ($\mathbb{L}$, $\models_{\mathbb{L}}$) which with its quantifers (e.g. $\forall$, $\exists$) and connectives (e.g. ~,$\wedge$,$\vee$,$\rightarrow$) defines the class $\mathbb{L}(\tau)$ of well formed formulas permitted. The satisfaction relation $\models_{\mathbb{L}}$ is a relation between structures and well formed formulas. We drop the subscript for $\models$, as it can be presumed by context.

Given a particular logic we consider $\gamma \in \mathbb{L}(\tau)$ and its models *Model* (γ) where

Model (γ) = $\{\mathbb{S} \in$ *Structure* [τ] such that $\mathbb{S} \models \gamma \}$

By the *Reduct Property* of logic [e.g.16] we have that if $\gamma \in \mathbb{L}(\sigma)$ and $\tau_{\mathbb{S}} \supseteq \sigma$, then

$\mathbb{S} \models \gamma$ iff $\mathbb{S}\lceil\sigma \models \gamma$

The if and only if property is very important because it proves:-

Theorem: *A constraint must always have its scope of application specified*

3.3 Satisfaction of an E-R theory

We now consider whether the connection constructor is a satisfactory method for defining a *Reduct* on the base structure. Connection constructors are defined by their functions and their inverse functions. Sorts are defined also by their functions. Thus a connection constructor does characterise a set of sorts, and inverse functions as required. Therefore we have proved:-

Theorem *All constraints that are satisfied on a derived structures are also satisfied on the base structure.*

We prove the satisfaction of FDs and IDs in section 4.

A *proper cyclic structure* on the refined E-R model is defined as a structure with a cycle of (*total*) functions between sorts. We call it *proper* to distinguish it from the looser definition often used for cyclic structures on "unrefined" E-R models. Note that on an "unrefined" E-R model the relationship functions are not necessarily, and indeed are generally not, *total* functions.

3.4 Restrictions on the E-R model satisfaction relation

Having provided the framework we now consider restrictions on the satisfaction relation $\models$ that is appropriate for E-R modelling.

3.4.1 Closed Connections

We would like to ensure that all the derived structures required for the constraints $\{(\gamma,\sigma),...\}$ of the E-R theory $\mathbb{T}$ can be represented by tables constructed by the connection constructor. In some cases we are interested in derived structures that represent all distinct connections to an element in a cyclic structure including multiple times around the cycle of sorts.

The problem is that these connections may be infinitely long and therefore cannot be captured on a finite data structure. This problem can be overcome by ensuring that where all such connections are required there are no infinite unclosed sequence of connections. This property can be stated as:-

The Closed Connection Cycle Property requires that in a cycle of sorts in the structure $\mathbb{S}$:-

$$\exists \text{ sorts } s_a,s_b \text{ such that } \forall\, x_a,x_b\ \boldsymbol{R_1}\ (x_a,x_b) \supset \boldsymbol{R_2}\ (x_a,x_b)$$

where $\boldsymbol{R_1}$ and $\boldsymbol{R_2}$ respectively have their scopes defined so they cover the cycle and only intersect on the sorts s_a,s_b.

We see from the examples of section 2 and the analysis in section 4 that this property is met in practice where we may have some interest in considering all connections.

3.4.2 Individual Distinctness

A particular interpretation $(\mathbf{I} : \tau \text{ --> } \mathbb{S}) \in$ *Structure* must according to the methodology of section 2 satisfy the following:-

Restriction on Individual Distinctness: *Individuals from two different entity (sub)type sets or two different value sets are not necessarily distinct, but all individuals in a given sort are distinct.*

3.5 An E-R Model is Theory and Structure

An E-R model is thus seen as being part specified by theory and part by structure. What we have now done is confirm our philosophical starting point of figure 1. For those caught in the nemesis of first order logic, recognise that models restricted by both countable sets of individuals and axioms are very uninteresting. A major problem in specifying a system by its axioms is that some structures are either not axiomatizable or the axioms are too hard to discover. An example of such a structure is the natural numbers with arithmetic. Additional elements beyond the natural numbers have to be added to satisfy the Peano axioms.

4. Satisfaction of the Data Condition

We now use Relational DataBase (RDB) theory to analyse data semantics of the refined E-R model. The word "table" is used rather than "relation" to emphasize that we are dealing with implemented data structures.

4.1 Expressing an E-R structure as RDB tables.

We first note that an E-R base structure and its derived structures can be readily translated to RDB tables:-

1) A relationship function can be represented by a table consisting of binary tuples that satisfy functional dependencies.
2) An entity with n attributes can be represented by a table consisting of n-tuples over the attribute value sets. The tuples satisfy the (n-1) functional dependencies from the identifying attribute's set to the decriptive attribute sets - see figure 4.
3) A derived structure formed by using the connection constructor can be represented as a table as shown in figure 6.

This last table shows a set of tuples over columns of attribute values and relationship existences with boolean values. The relationship existence columns show the presence or absence of a connection (relationship instance) between the identifying attributes of each pair of entities in the structure. These columns are needed in a cyclic structure because without them it is ambiguous which connections are being represented. This ambiguity has confused RDB theory because it is not clear from a table of attribute values what connections are being represented.

It is also evident that the constraints specified on the E-R model are in RDB terms either functional dependencies (FDs) or inclusion dependencies (IDs). For example:-

1) All relationship functions R(A,B) from entity $E_1(A,...)$ to entity $E_2(B,...)$ must satisfy

$$E_1.A \supseteq R.A \quad \text{and} \quad E_2.B \supseteq R.B$$

2) Mandatory participation of an entity E with identifying attribute A in a relationship R(A,B) can be specified as $R.A \supseteq E.A$. Therefore, by 1) $R.A = E.A$. Such a

constraint we call a *paired unary inclusion constraint.*

3) The E-R model inclusion and functional dependency constraints are specified as IDs and FDs on base or derived structures represented by tables.

4) Subtypes are specified by unary inclusion constraints between identifying attributes of subtype entities and their supertype entities.

There are well documented definitions of the FD and ID constraints in terms of what they must satisfy on a table to be certain the satisfiability of such constraints can be determied. We have thus shown that:-

Result *The refined E-R model satisfies the data condition part of the semantically adequate structure condition.*

We now further examine the E-R model to see what insight it provides into the RDB data model.

4.2 Well-formed E-R models translate to well normalised RDBs.

We now show that a well formed E-R model happens to be in at least fourth normal form. We also show that the "cyclic join dependencies" that are recognised when refining RDB relations to fifth normal form are rather strange so far as information semantics are concerned

An E-R model structure is in fourth normal form by construction, because it assumes that all attribute functions and relationship functions needed to describe the real world have been recognised. Relationship function recognition requires that all n-aryrelationships have been recognised, and then decomposed to relationship functions.

4.3 Enforcement of complex functional dependencies

More complex functional dependency constraints were shown in section 2 to be specified over derived structures. These derived structures may contain multi-valued dependencies (MVDs), as in figure 12. We have the following theorem:-

Theorem. *Consider a bnary functional dependency FD specified on a derived structure S constructed over a base of unary functions $\{f_i\}$ and sets. If $\{f_i\}$ does not imply FD, then either :*

a) the derived structure S necessarily will contain a MVD, or
b) if in addition the base functions are total, then FD and $\{f_i\}$ mplies alternative, possibly composite, keys on one or more of the sets.

Proof.

Consider a binary functional dependency *FD : (X,Y) --> Z* over a structure defined by the sets X, Y, Z and two base functions between those sets. Two base functions are necessary and sufficient to connect the three sets together and to unambiguously define the scope of the constraint over the entire structure.

The proof proceeds by considering the various pairs of base functions possible and eliminating those that imply FD. The remaining cases satisfy the theorem.

There are six possible base functions are X --> Y, Y-->X, X-->Z, Z-->X, Y-->Z, Z-->Y. Each base function can only form pairs with 4 others to ensure the three sets are connected. This produces 12 pairs. As the theorem is symmetric in X and Y we need only consider bases in the following 7 cases:-

1) X-->Y, X-->Z 2) X-->Y, Z-->X 3) X-->Y, Y-->Z 4) X-->Y, Z-->Y
5) X-->Z, Y-->Z 6) X-->Z, Z-->Y 7) Z-->X, Z-->Y

In cases 1, 3, 5, and 6 the functional dependency (X,Y) -->Z is implied. Now consider the remaining cases:-

Case 2. We have (f_1 : Z-->X , f_2 : X-->Y, and (X,Y)-->Z). From (f_2 : X-->Y, and (X,Y)-->Z) we get Y-->Z, and from (f_1 : Z-->X , f_2 : X-->Y) we get Z-->Y.

Both Y-->Z and Z-->Y are defined on the table with columns (Z, f_1?, X, f_2?,Y) representing the derived structure constructed from [Z]*f_1*[X]*f_2*[Y], and therefore only contain full tuples which satisfy all these functional dependencies. From Y-->Z, Z-->X, and X-->Y we can infer that Y-->X and X-->Z. Therefore (X<-->Y, Y<-->Z, and Z<-->X) on these tuples. If f_1 and f_2 are total functions then X,Y and Z are alternative keys on the derived structure and, if null alternative keys are allowed, then for X,Y and Z as well.

* Result - b) The structure can be reduced to one entity type with 3 alternative keys if null keys are permitted.

Case 4 We have (X-->Y and Z-->Y) on the structure, which implies that (Y-->>X and Y-->>Z) which describes an MVD on the derived structure.

* Result - a) If the base functions are partial, both the base and the derived structure and the base structure are required for enforcing all dependencies

Case 7 We have (Z-->X, and Z-->Y) which implies Z --> (X,Y), and therefore Z<-->(X,Y) so that if f_1 and f_2 are total then Z and (X,Y) are alternative keys on the derived structure and for Z.

* Result - b). The derived structure is not required if a mechanism exists for identifying an entity by a composite key from neighbour entitties. This result is exemplified with the ENROLMENT entity in figure 6.

Therefore the table representing a derived structure may only be useful, so far as constraint checking is concerned, for checking the FDs specified over all of it. The derived structure will prove difficult to update when it contains an MVD.

The theorem can be generalised to include a larger base for specifying a binary FD. Three sets only are appropriate but now it is possible that the binary relationships would be many to many so that the derived structure would contain an embedded MVD. If the binary relationships are not many to many, they would be functions and hence the same proof would be applicable. It is conjectured that derived structures for more complex FDs would have similar restrictions on their usefulness for constraint checking.

What this means for the relational data model is that while it is possible to have tables on which all FDs can be enforced, either all complex FDs required are implied, or alternative keys are implied, or the table necessarily contains MVDs.

4.4 MVDs and CJDs are Characterised by Inclusion Dependencies

MVDs

Consider the relation R4(A,B,C) as representing a ternary relationship between the entities A,B, and C. Suppose that:-

$$R4 = \pi_{AB}(R4) \textit{ join } \pi_{BC}(R4),$$

where π_X (R) represents the projection of R over the set of attributes X (See AppendixA.5). If it is not true that 1) AB satisfies B-->A and BC satisfies B-->C, then 2) by definition R4 contains a MVD that is not a FD because B-->>A or B-->>C.

This equation is interpreted as the E-R model in figure 12. R4 with connection columns added is constructed by the connection construction:-

$$(A)\ *id_A\ *\mathbf{R_{A\_B}}\ *id_B\ *\mathbf{R_{B\_C}}\ *id_C \text{ --> } T4 : (A\ ?_{R_{A\_B}}\ B\ ?_{R_{B\_C}}\ C)$$

We also have:- $$R4 = \pi_{ABC}(T4)$$

In terms of E-R constraints we note that B mandatorily participate in the relationships $\mathbf{R_{A\_B}}$ and $\mathbf{R_{B\_C}}$, which are expressed as inclusion dependencies. This gives the result.

Theorem *A MVD over three entities connected by two binary relationships is characterised by two pairedunary inclusion constraints.*

CJDs

Now consider the relation R5(A,B,C) as representing a ternary relationship between the entities A, B, and C, which have identifying attributes A, B, and C respectively. Suppose that:-

$$R5 = \pi_{AB}(R5) \textit{ join } \pi_{BC}(R5) \textit{ join } \pi_{CA}(R5).$$

This equation by definition satisfies a join dependency that is not a MVD. Because it is around a cycle we call it a *cyclic join dependency* (CJD) constraint. The equation is interpreted as the E-R model in figure 12. Now consider the connection construction which produces the table T5 :-

$$(A)\ *id_A\ *\mathbf{R_{A\_B}}\ *id_B\ *\mathbf{R_{B\_C}}\ *id_C\ *\mathbf{R_{C\_A}}\ *id_A \text{ -->}$$
$$T5 : (A\ ?_{R_{A\_B}}\ B\ ?_{R_{B\_C}}\ C\ ?_{R_{C\_A}}\ A')$$

In this case A ≡ A' because the relationships satisfy a cyclic join dependency constraint and therefore:-

$$R5 = \pi_{ABC}(T5)$$

This clearly illustrates the distinction between the ***join*** and ***connection*** constructors. By symmetry on the problem R can also be constructed by any connection construction formed by a permutation of A, B and C.

As well:- $?_{R_{A\_B}} \wedge ?_{R_{B\_C}}$ --> $?_{R_{C\_A}}$, and therefore $\mathbf{R_{C\_A}} \supset \mathbf{R_{A\_B}}\ ;\ \mathbf{R_{B\_C}}$

In terms of E-R constraints we note that A, B and C mandatorily participate in the relationships $\mathbf{R_{A\_B}}$, $\mathbf{R_{B\_C}}$, and $\mathbf{R_{C\_A}}$ which we can express as a set of paired unary inclusion dependencies between each of the three relationships and their two entities. The three different ways of

constructing R5(A,B,C) with their conditions on the repeated column being identical, implies that the following three binary inclusion constranits are satisfied:-

$$R_{A\_B} \supset R_{B\_C} \; ; \; R_{C\_A}$$
$$R_{B\_C} \supset R_{C\_A} \; ; \; R_{A\_B}$$
$$R_{C\_A} \supset R_{A\_B} \; ; \; R_{B\_C}$$

These can also be expressed as binary inclusion dependencies. This gives the result:-

Theorem *A CJD over three entities connected in a cycle with three binary relationships is characterised by six paired unary inclusion constraints and three binary inclusion constraints.*

Such cycles of paired unclusion constraints are rare in practice. In the case of CJDs it seems best to maintain the entities and relationships involved in one table only because updates to the components need to happen "all at once".

We now note that these binary inclusion constraints were given in section 3 as a required property to prevent the possibility of an infinte unclosed cycle of connections through the entities A,B and C. It gives us a way of understanding what 5NF is trying to say:-

CJDs occur because a cycle of relationships can be represented in a (flat) table. Fifth normal form is thus an artifact of the relational data structure (table) representation.

4.5 The Implication problem for FDs and IDs

It has been shown that the implication problem for FDs and typed, acyclic IDs is decidable [8,9]. All inclusion dependencies used in the E-R model are typed. The condition of acyclicity can now be weakened slightly to allow closed cycles of implication dependencies:-

Theorem (FD and ID implication) Providing there is no unconstrained cycle of inclusion dependencies, then the implication problem for the constraints specified for the E-R model are decidable.

This is implied by Bry and Mathey's *finite satisfiability is semi-decidable* theorem [12].

What must be avoided in practice is a unary cycle of unconstrained inclusion dependencies, which is possible if all entities in a cycle of relationships mandatorily participate in those relationships. Such a situation is easily determined by inspection. The cost in this rare circumstance is that the satisfiability of the implication problem might be checked before trying to discover implied constraints.

6. Conclusions

The issue that has been recognised in the E-R approach to specification is that some problems presume a structure on which they are specifiable and on which the specification's satisfiability can be decided.. Unlike other areas, information analysts and database designers proceed from a base defined in terms of structure rather than axiom.

The refined E-R Model has been shown to be specified using both theory and structure.

The term "model" is used in model theory to mean a structure that satisfies a particular theory

mapped to it by a satisfaction relation of its logic. Thus the result of an adequate specification is a model - that is a (countable) structure which satisfies the theory and is therefore *consistent.* However, note that **there is no requirement that the structure is completely defined by the theory it satisfies, nor that the theory is a complete axiomatisation of the structure.**

What we have also shown is that semantics as used in first order logic is not sufficiently powerful. As obvious as it sounds, it is quite clear that we have two ways external to an E-R model of understanding what it means. These two ways are to test its information semantics and its data semantics according to the semantically adequate structure condition.

What now must be done is to pursue the question of specifying state transitions, that is the behaviour of an E-R model. While there have been many proposals, it now seems that using the complete structure of the model including its subtypes, we have a base on which further refinement stages of the specification can proceed.

Acknowledgements

This paper was completed while I was a visiting professor in Computer Science at the University of Waterloo, and on leave from Macquarie University. I thank both universities and the Australian Computer Research Board for their financial support related to this work.

References

[1] Chen,P.P.S., 1976. "The entity-relationship model - towards a unified view of data". **ACM Transactions on DataBase Sysems**, Vol. 1, No.1, pp9-36.

[2] Howe,D.R., 1983, **Data Analysis for Database Design.** Edward Arnold, London, 306pp.

[3] Parkin, A., 1980. **Systems Analysis**. Edward Arnold, London, 220pp.

[4] Bartree, J., 1985. "Model-theoretic logics: background and aims" *in* Barwise,J. and S.Feferman (Eds.), 1985. **Model-Theoretc Logics**. Springer-Verlag, New York

[5] Langfors,B., 1974. "Theoretical aspects of information systems." **IFIP Congress 1974**, pp937-945

[6] Bubenko,J.A., 1980. "Information modelling in the context of system development." **IFIP Congress 1980, Information Processing**, pp 395-411.

[7] Arisawa,H., and T.Miura, 1986. "On the properties of extended inclusion dependencies." **IEEE Transactions on Software Engineering**, vol. SE-12, No. 11, pp1098-1101

[8] Cosmadakis,S.S., and P.C.Kanellakis, 1984. "Functional and inclusion dependencies: a graph theoretic approach." **Proc. 4th. ACM Symposium on Principles of Database Systems,** 1984, pp29-37.

[9] Mitchell,J.C., 1983. "Inference rules for functional and inclusion dependencies." **Proc. 43rd. ACM Symposium on Principles of Database Systems,** 1983, pp.58-69

[10] Codd,E.F.,1979. "Extending the database relational model to capture more meaning." **ACM Trans on Database Systems** vol.4, No.4, pp397-434.

[11] Montague,R.,1974. "On the nature of certain philosophical entities." *in* **Formal Philosophy**, R.H. Thomason (ed.), Yale University Press, 148-187 [reprinted from **The Monist,** vol.53 (1960), pp159-194]

[12] Bry,F., and R. Manthey, 1986. "Checking consistency of database constraints: a logical basis." **Proc of the Twelth International Conference on Very Large Data Bases**, Kyoto, August, 1986, pp13-20.

[13] Furtado, A.L., and E.J.Neuhold, 1986. **Formal Techniques for Data Base Design**. Springer-Verlag, New York.

[14] Brady, L.I., 1985. "A universal relation assumption based on entities and relationships." **IEEE Proc. 4th. International Conference on the Entity-Relationship Approach**, Chicago, U.S.A., Oct 28-30, 1985, pp208-215.

[15] Maier,D., 1983. **The Theory of Relational Databases**. Computer Science Press, 637pp.

[16] Ebbinghaus,H.-D., 1986. "Extended logics:the general framework." *in* Barwise,J. and S.Feferman (Eds.), 1985. **Model-Theoretc Logics**. Springer-Verlag, New York

[17] Chang,C.C., and H.J.Keisler ,1973. **Model Theory**, North Holannd, Amsterdam.

Figures

Figure 1. The semantics of the Entity-Relationship Model

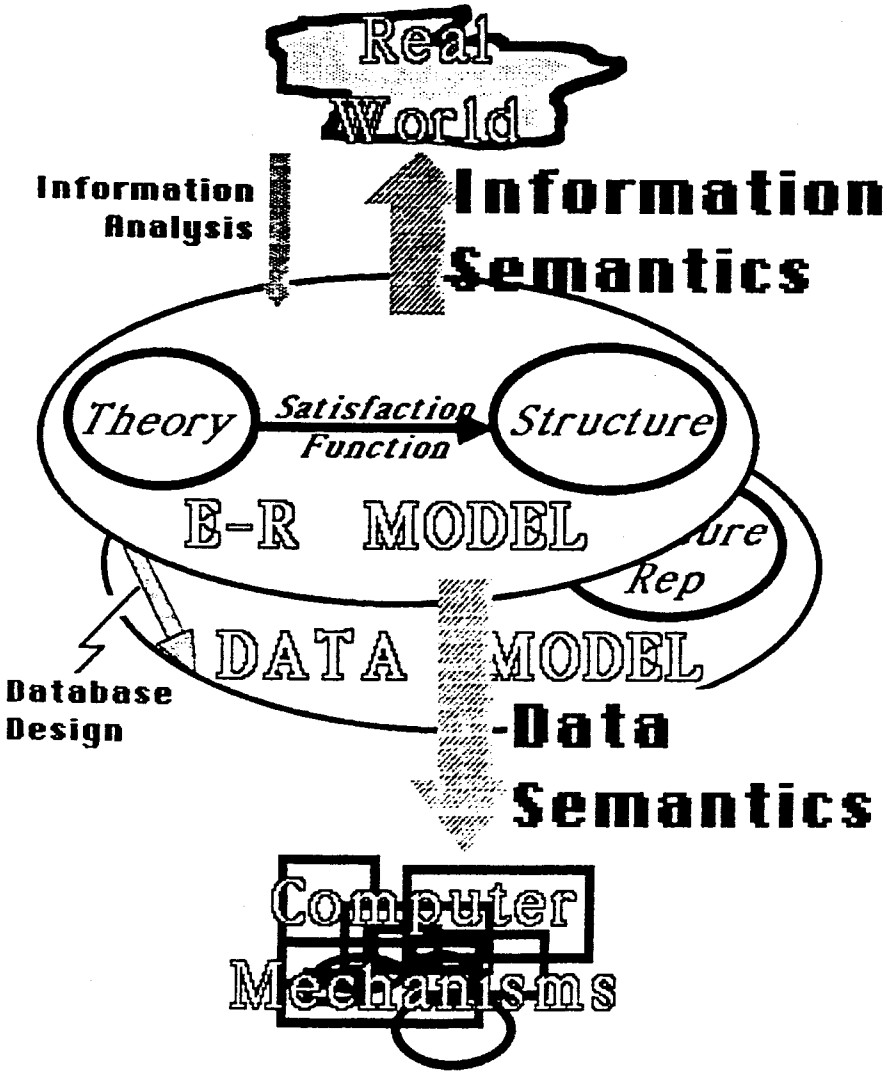
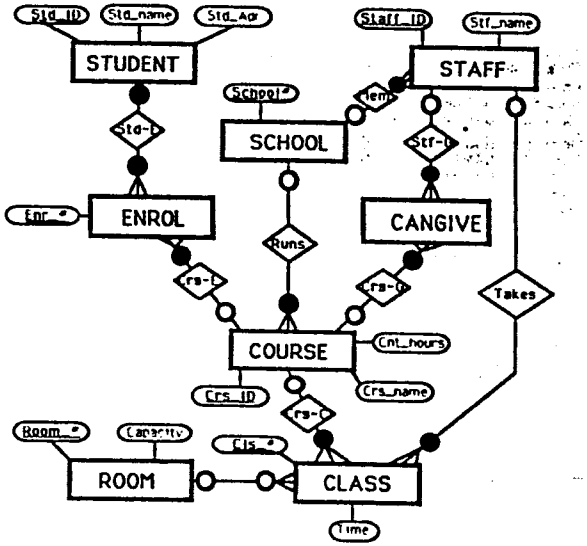
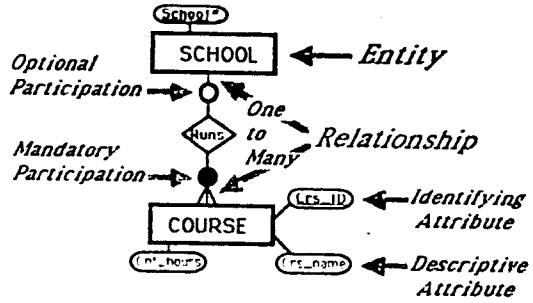


Figure 3 The E-R model of a Course Information System





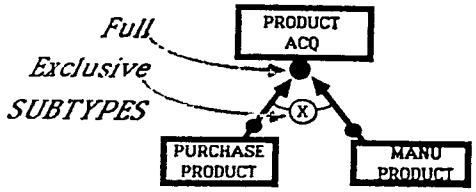


Figure 2. Summary of Entity-Relationshp Diagrammatic Notation

Figure 4 The E-R model of an entity type showing its attribute functions and their value sets

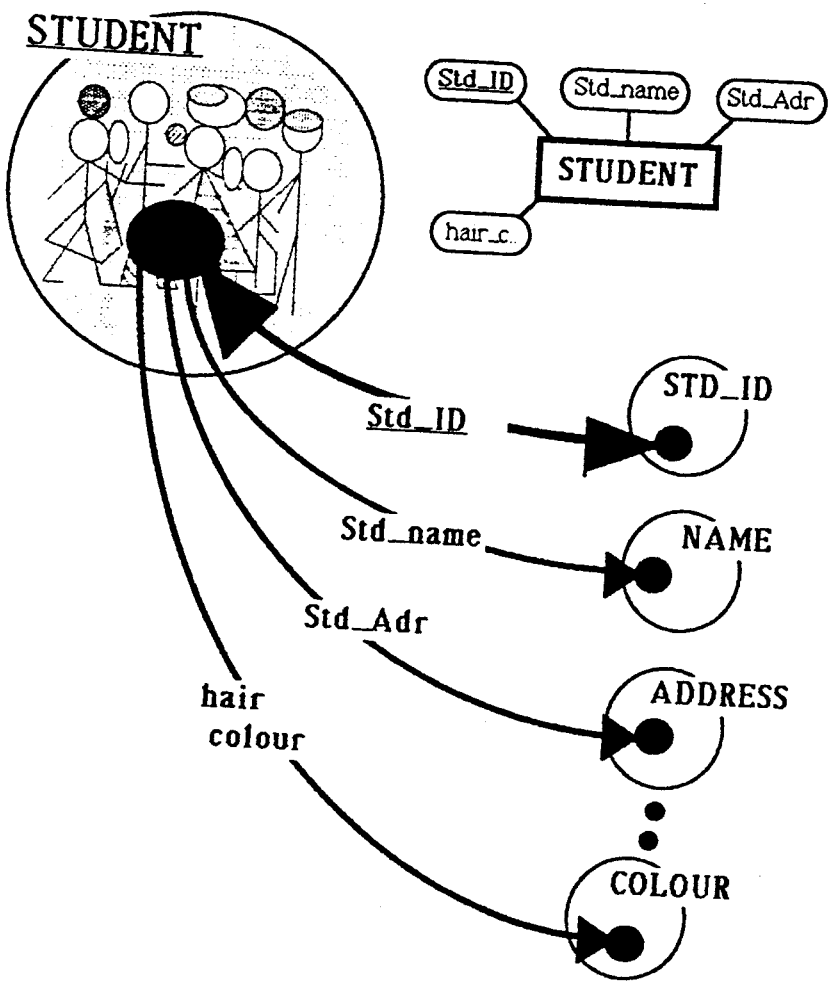


Figure 5 The E-R model of a relationship function *Takes* : CLASS --> STAFF.

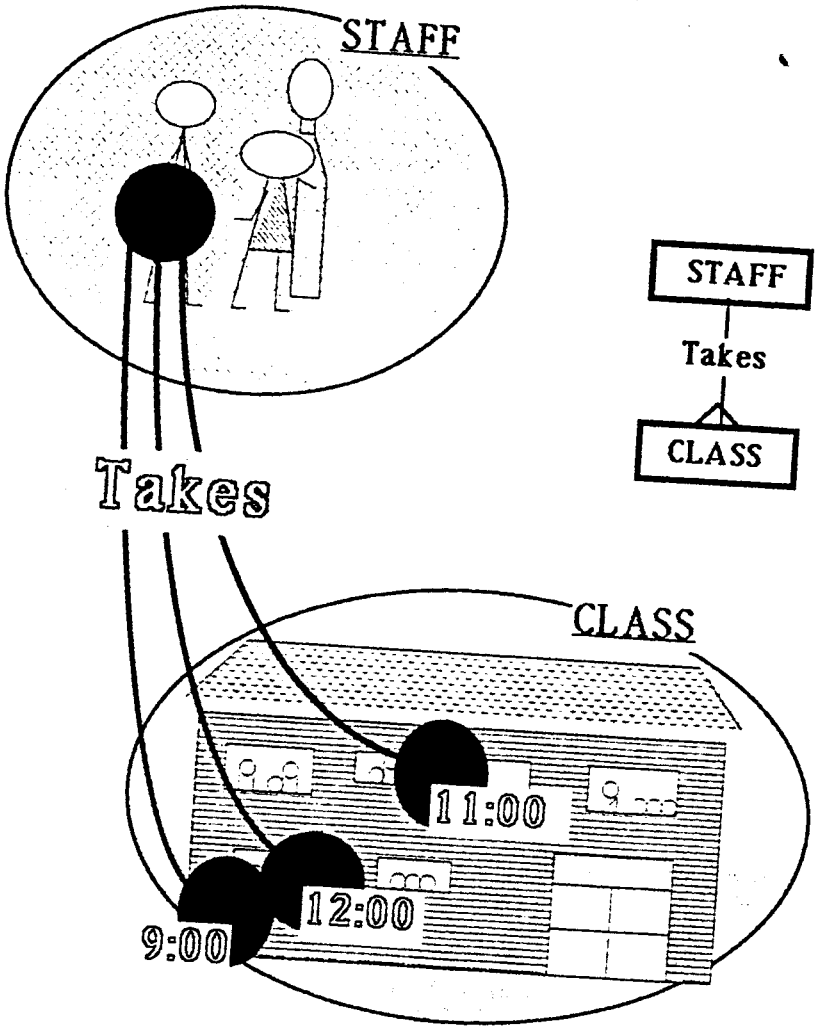
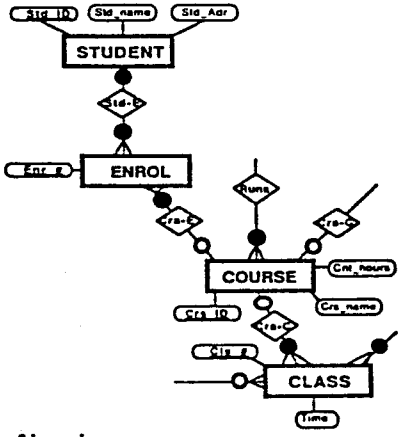


Figure 6 Examples of binary function constraints.

Example 1. Student enrolled in a class



Description of invariant

A student may only be enrolled in one course that is on at a given time.

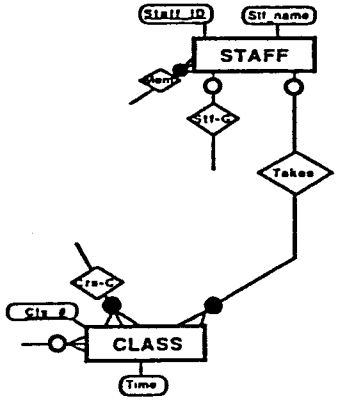
Definition of the table

(STUDENT) **Std_ID* * Std-E^{-1} * Crs-E * Crs-C **Cls_#* **Time* -->
(STD_ID ?$_{(Std\text{-}E^{-1},Crs\text{-}E,Crs\text{-}C)}$ CLS_# TIME)

Definition of Invariant over the table

(STD_ID, TIME) --> CLS_#

Example 2. Staff taking a class



Description of invariant

A member of staff can only take one course at a given time

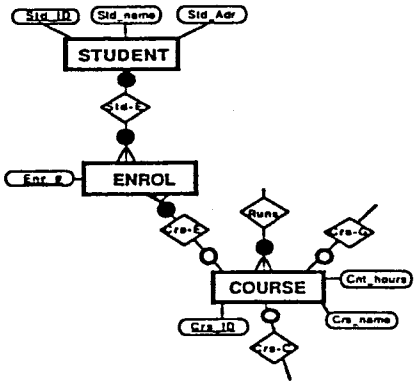
Definition of the table

(STAFF) **Staff_ID* * Takes^{-1} **Cls_#* **Time*
--> STAFF_ID ?$_{(Takes^{-1})}$ CLS_# TIME

Definition of Invariant over the table

(STAFF_ID, TIME) --> CLS_#

Example 3. Student enrolments in courses



Description of invariant

A Student may only enrol once in a given course.

Definition of the table

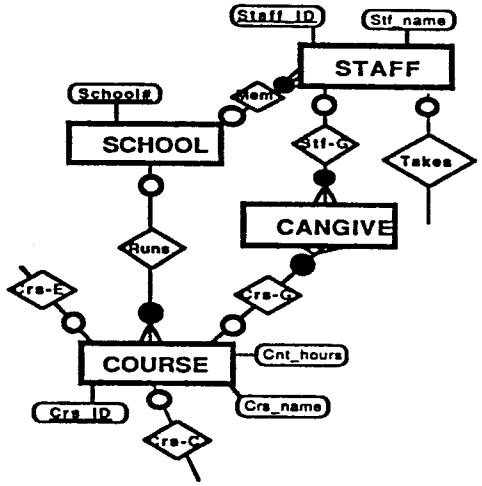
(STUDENT) **Std_ID* * Std-E^{-1} **Enr_#* * Crs-E **Crs_ID*
--> (STD_ID ?$_{(Std\text{-}E^{-1})}$ ENR_# ?$_{(Crs\text{-}E)}$ CRS_ID)

Definition of Invariant over the table

(STD_ID, CRS_ID) --> ENR_#

Figure 7 Examples of binary inclusion constraints.

Example 1. Courses given by a member of staff



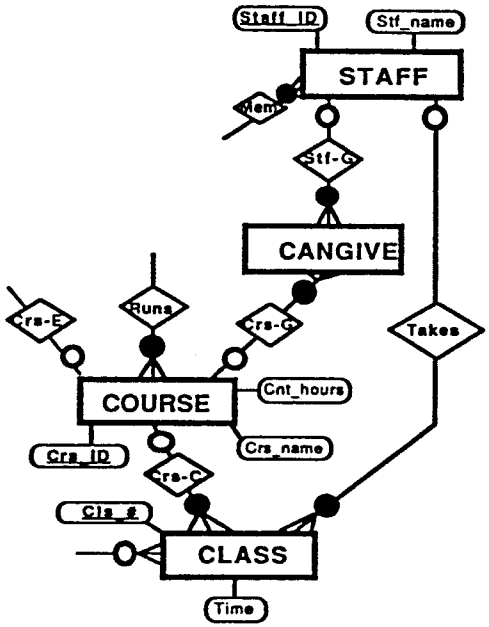
Description of invariant

A Course can only be given by staff who are members of the school that runs the course.

Definition of Invariant over the relationships

$$\textbf{Runs} ; \textbf{Mem}^{-1} \supset \textbf{Crs-G}^{-1} ; \textbf{Stf-G}$$

Example 2. Classes taken by a member of staff



Description of invariant

A Class can only be taken by a member of staff who can give the course that the class is part of.

Definition of Invariant over the relationships

Crs-C ; Crs-G^{-1} ; Stf-G ⊃ Takes

Figure 8 The Subtypes of the COURSE entity

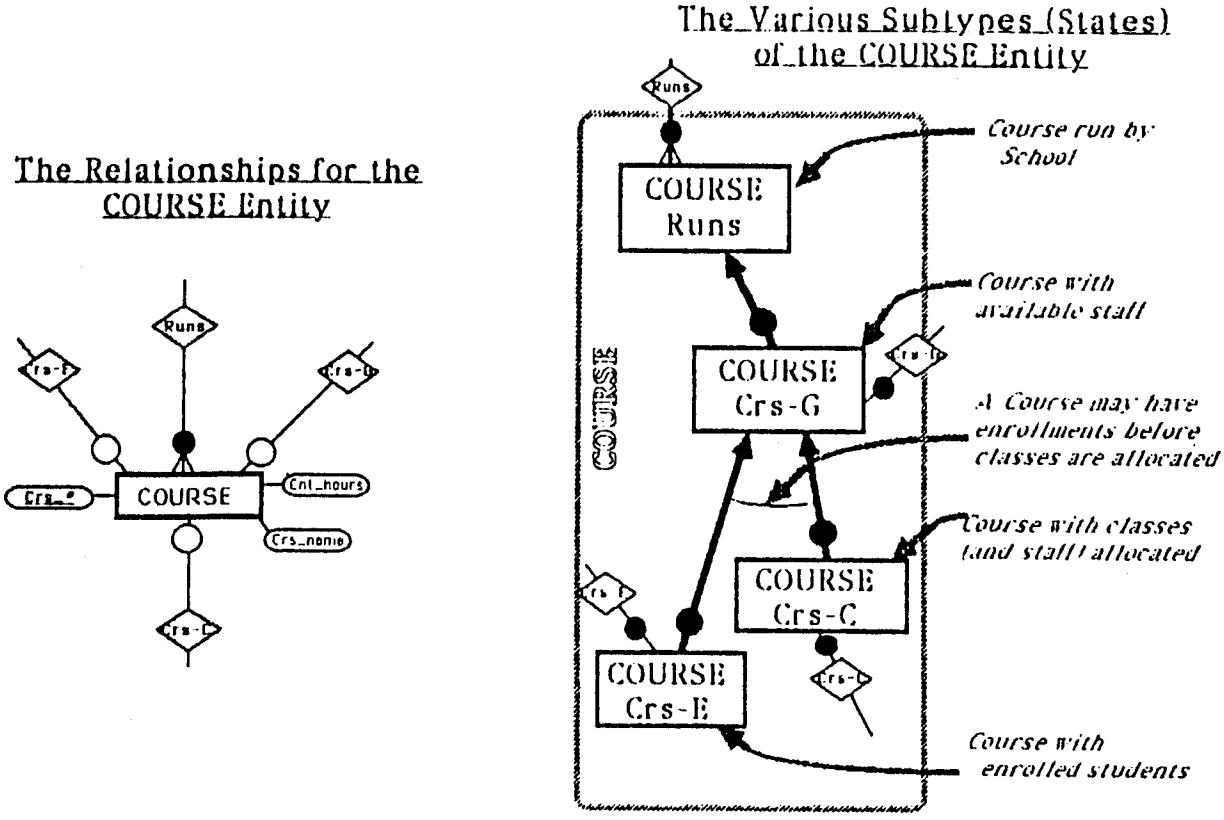


Figure 9 All the sub-types of the Course Information System inckuding STAFF and STUDENT as subtypes of PERSON.

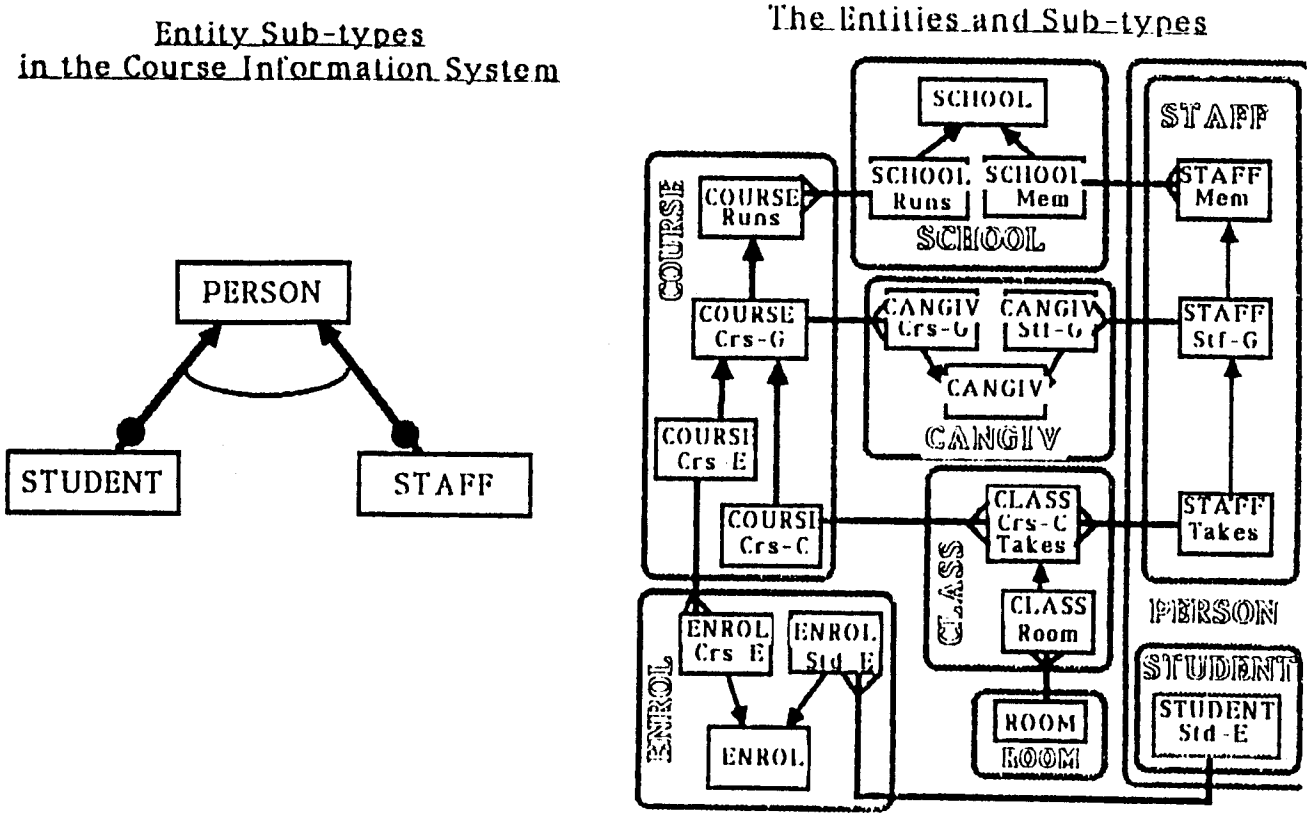


Figure 10 A Manufacturing Information System illustrating a more comples set of subtypes

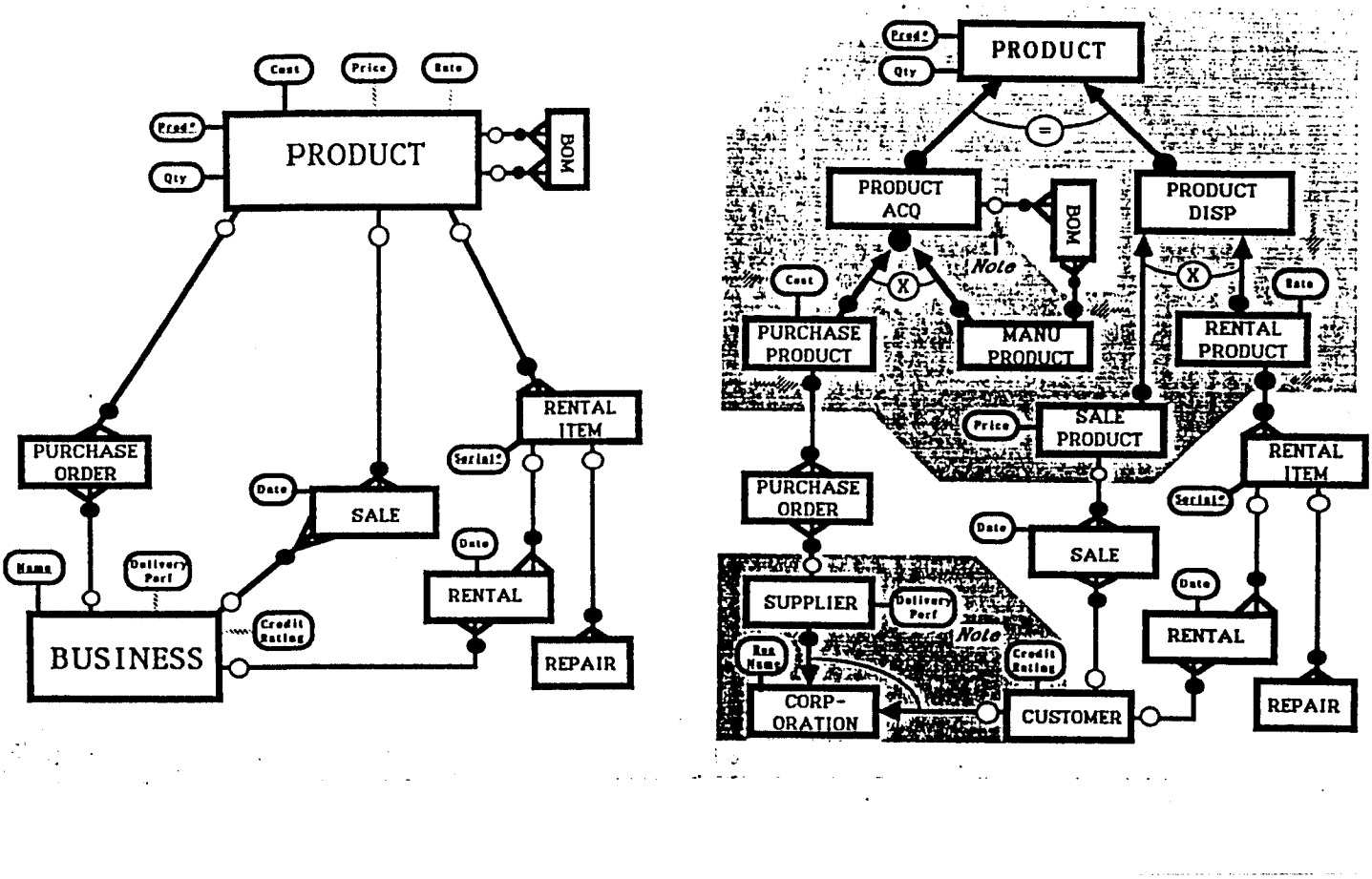


Figure 11 Set theoretic specification of subtypes

Example 1. Course Information System Entities

COURSE subtypes - COURSE_runs ⊃ COURSE_Crs-G
COURSE_Crs-G ⊃ COURSE_Crs-E
COURSE_Crs-G ⊃ COURSE_Crs-C

ENROL subtypes - ENROL_Crs-E = ENROL_Std-E

SCHOOL subtypes - SCHOOL_Runs = SCHOOL_Mem

CANGIV subtypes - CANGIV_Crs-G = CANGIV_Std-G

CLASS subtypes - CLASS_Crs-C&Takes ⊃ CLASS_Room

STAFF subtypes - STAFF_Mem ⊃ STAFF_Stf-G
STAFF_Stf-G ⊃ STAFF_Takes

PERSON subtypes - PERSON ⊃ STAFF
PERSON ⊃ STUDENT

Example 2. Product Sub-type Entities

PRODUCT ⊃ PRODUCT_ACQ

PRODUCT_ACQ = PRODUCT_DISP

PRODUCT_ACQ = (PURCHASE_PRODUCT ∪ MANU_PRODUCT)

(PURCHASE_PRODUCT ∩ MANU_PRODUCT) = ϕ

PRODUCT_DISP ⊃ (SALE_PRODUCT ∪ RENTAL_PRODUCT)

(SALE_PRODUCT ∩ RENTAL_PRODUCT) = ϕ

Fugure 12 E-R diagrams illustrating Relational Database Normal Forms