Detection of Redundant Arcs in Entity Relationship Conceptual Models

David S. Bowers

Computing Department, The Open University
Walton Hall, Milton Keynes, MK7 6AA, UK
d.s.bowers@open.ac.uk

Abstract. One measure of the quality of a conceptual model is the quality of design that can be derived from it. Redundant relationships in an Entity Relationship model cause a generated relational schema to be un-normalised. Since a relationship is redundant only if some other path in the model implies both its set theoretic signature and its semantics, determination of redundancy is not mechanical, and always requires interaction with the client or user. A path composition and search algorithm is presented to detect potentially redundant relationships, and strategies are discussed for the incorporation of this type of algorithm in a CASE environment.

1. Introduction

The principal justification for quality in conceptual models is that such models form the basis for designs and implementation. Well-understood, deterministic rules to transform a conceptual model into a design can be automated, in which case, a conceptual model of poor quality would result in an inadequate implementation.

Within the scope of a short paper, a tractable example is the generation of a relational schema from an Entity Relationship (ER) Model. Various authors, including Teorey [1] and Bowers [2], have proposed algorithmic approaches for the synthesis of a relational schema from an Entity Relational Model, and such techniques are now widely adopted. Despite the essential rigour of such techniques, however, normalization is invariably applied to the generated schemas, since they reproduce faithfully any redundancy present in the original conceptual models.

This paper addresses the detection of redundant relationships within ER models. A relationship is redundant when both its mapping and its semantics are implied by some alternative path. Relational schemas synthesized from conceptual models that include redundant relationships may not be in Third Normal Form, and will not be in Fifth Normal Form. Conversely, posted-key synthesis techniques, such as that described by Bowers [2], should generate schemas always in fourth normal form, and often in fifth normal form, from ER models that are both free of redundant relationships and contain neither multi-valued attributes nor merged entity types.

Detection of redundant relationships within an ER model supports interaction with the client at the conceptual level to resolve redundancy. The traditional alternative of deferring the resolution to the normalization of the generated schema seems both to be

A. Olivé (Eds.): ER 2003 Ws, LNCS 2784, pp. 275-287, 2003.

too late and, moreover, inappropriate: too late because much of the semantic content of the model has been discarded prior to normalization, and inappropriate because normalization based solely on functional dependency analysis can itself introduce ambiguity into the schema.

The algorithms presented in this paper can be incorporated into CASE tools to support the systematic detection of redundancy, and its resolution through a dialogue with the tool user. Such support allows modellers to err on the side of including rather than excluding relationships, but still to address the issue of redundancy.

The context for this discussion is presented in Section 2, and the interaction between redundancy in ER models and normalization criteria are reviewed in Section 3. The basic algorithm for determining potential path equivalence is introduced in Section 4, and strategies for its application are discussed in Section 5. Finally, issues still to be resolved are outlined in Section 6.

2. Context

The issue of redundancy within Entity Relationship models results from what is both the greatest strength and the greatest weakness of the approach: its perceived simplicity. This has led to the almost universal adoption of ER techniques, or one of its derivatives, for modelling data structures, with widespread belief that such models are readily comprehensible by both analyst and client. Unfortunately, there is ample evidence that the comprehension of ER models is difficult, and that their construction can be error-prone [7,8,9,10]. The quality of an ER model, and of any database schema derived from it, remain fundamental issues. Whilst the former has attracted some attention [11,12,13], the latter seems still to be subsumed into a pervasive assumption that any “initial” relational schema, whatever the “quality” of the ER model from which it has been derived, will always need to be “improved” by normalization[14] – an unprovable act of faith that, in many cases, is not justified by the resulting normalised model. Indeed, normalization is normally a technical activity effected entirely on the basis of perceived dependencies between fields of a schema; the semantic information expressed in an ER model is invariably discarded during its transformation to a relational schema. Furthermore, normalization is often regarded as a purely implementation issue, and therefore performed without reference to the client, quite possibly without full understanding of the client’s domain.

It is irrelevant to argue that expert analysts would not make mistakes in ER diagrams. The principal purpose of such diagrams is to act as a communication medium between an analyst and a client , who is almost certainly a novice, at least as far as ER diagrams are concerned. During consultation with the analyst, it is quite possible that the client might suggest - or insist on - a relationship being drawn between two entities, even if it is already implied by another path. Thus, redundant relationships could be introduced merely through interaction with a client..

Systematic approaches to redundancy in ER Models seem to have attracted relatively little attention in the literature. Simple redundancy between a single relationship and paths of two relationships are explored exhaustively by means of occurrence diagrams in [3]. The authors suggest a set of heuristics that an analyst could use to determine redundancy in an ER model.

Our work seeks resolution of redundancy at a semantic level, rather than by mechanical consideration of rules against possible, or actual, data occurrences. Further, our work considers paths of arbitrary length, although the equivalences we deduce for paths of length one and two are the same as in [3].

The focus of the superficially similar approach in [4] seems to be on the rearrangement of the generated relational schema to reduce redundancy. There is, however, an interesting analysis of implicit relationships, represented within an ER diagram by "posted" keys not derived from any explicit relationship. Whilst such "errors" might be contrary to the ethos of ER modelling, it would be interesting to explore how the techniques of [4] could be integrated with those presented here.

An apparently related issue is explored in [5]. Given multiple paths between entities, cardinality constraints, both explicit and implicit, might prevent instantiation of the model. However, a redundant relationship and its alternate path imply precisely the same mapping; hence, they cannot affect the satisfiability of any model.

Similarly, the exploration in [6] of extensional "Int-cardinality" constraints is concerned primarily with satisfiability rather than redundancy, and is therefore also not relevant to the approach presented here.

The problem of identifying redundant relationships in an ER model is similar to that of identifying redundant connections within logic circuits [15,16]. Similar problems arise also with redundancy in production rules [17]. The path composition algorithms are only slightly simpler than those presented in Section 4.

Finally, path composition relates closely to transitive closure algorithms, as developed extensively for knowledge-based systems and deductive databases; which are known to be expensive [18, 19]. Such algorithms have been used for the analysis of functional dependencies [20, 21], but not yet, it seems, related directly to conceptual-level models, such as ER. Also, although efficiency of the transitive closure algorithms is crucial in the examples cited, it is perhaps less so for design activities, where the problems do have to be solved, but, in principle, only once.

3. Redundancy, Synthesis, and Normalization Criteria

Normalization, at least to Third Normal Form (3NF), has long been accepted as a quality criterion for relational schemata. This section summarizes the "errors" in an ER model that, if synthesized into a relational schema using a "posted key" algorithm would cause violations of each of the common Normal Forms.

The absence of redundant relationships in a conceptual model is one of three correctness criteria for an ER model enumerated in [12], although, curiously, it is distinguished from "third normal form violations". This separation may be somewhat misleading, since synthesised redundant relationships would, specifically, violate the criteria for third normal form, by introducing transitive dependencies.

Repeating groups in a schema, which would violate first normal form (1NF), could arise either from multi-valued attributes or from the incorrect "posting" of a relationship. In relational synthesis, a relationship is represented by "posting" the identifier of the entity at the "one" end of a relationship into the relation representing the entity at the "many" end; thus, a posted key can only be single valued. Hence, a

schema synthesised from an ER model will always satisfy 1NF, provided that no entity type has multi-valued attributes.

This is sufficient also to ensure that there are no multi-valued dependencies in the schema. Thus, if the synthesised schema is in 3NF, it will also be in 4NF.

Synthesis algorithms never merge relations for separate entities: at most, keys are posted to represent relationships. Hence, attributes in a synthesised relation are always fully dependent on the identifier of the corresponding entity, provided that no entity in the model has attributes drawn from more than one "real-world" entity.

underlying entity, or they are posted keys of entities in a many-to-one (or one-to-one) relationship Third Normal Form (3NF) requires that there must be no transitive dependencies. Attributes functionally dependent on the primary key are either attributes of the with that entity. Since the non-key attribute on which a transitively-dependent attribute depends must itself be a (primary) key of some entity, it follows that the determining non-key attribute must be a posted (foreign) key.

Hence, the issue is how a relationship between two entities - represented by the determination of the transitively-dependent attribute by a foreign key - might be included in the relation for a third entity. In the simplest case, it will be because there is a redundant relationship in the ER model.

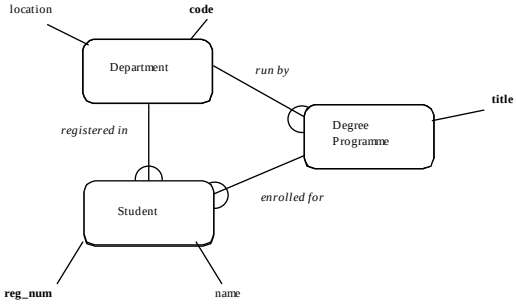

Fig. 1. A simple example of potential redundancy

This is best illustrated by an example, as in Figure 1. Synthesis of this model results in the following schema, in which in the relation *Student* contains an apparent transitive dependency of *department_code* on *reg_num* via *degree_programme_title*.:

Department (code, location)
Degree_programme (title, department_code)
Student (reg_num, name, degree_programme_title, department_code)

The problem lies in the ER model of Figure1, since the relationship *registered in* might be implied completely by the pair of relationships *enrolled for* and *run by*. The potential implication depends on whether or not the fact that a *student* is *enrolled for* a *degree programme run by* a particular *department* means that the *student* is also *registered in* that *department*. If the relationship is redundant, then the resulting schema violates Third Normal Form.

In the more general case, in which the relationships involved are not constrained to have degree one-to-many, the synthesised schema will contain a join dependency, rather than a transitive dependency. However, the elimination of redundant relationships alone is not sufficient to ensure that synthesised schemas will satisfy 5NF, since, rather than being "redundant", the relationship may be part of a representation of a higher-order relationship between three or more entity types, such as the cyclic many:many-many:many-many:many structure.

4. Path Composition and Equivalence

An algorithm to detect (potentially) redundant relationships within an ER diagram needs three components: a means of composing transitive relationships, a strategy for comparing each relationship with every alternative (composed) path between the entities it connects, and a way of comparing the semantics of alternative paths that have the same signature. A path composition algorithm is presented in this section, the available strategies are discussed in section 5.

Each relationship in an ER diagram has a set theoretic signature which is a {partial or total} {function or multifunction} from one entity (say, *A*) to a second (*C*), and an inverse signature, drawn from the same set, from *C* to *A*. Two paths between *A* and *C* are equivalent if they have both the same signatures **and** the same semantics; at this stage, we ignore any explicit numerical cardinality constraints, but take account of optionality in the relationships. Thus, in Figure 2, the relationship *C-A*, which is optional for entity *A*, represents a partial multifunction from *A* to *C*, and a total function from *C* to *A*. These are the same as the composed signatures of relationships *B-A* and *C-B*; however, although the two paths from *A* to *C* have the same signatures, they are equivalent **only** if they also have the same meaning.

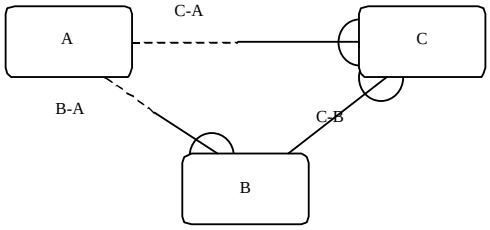

Fig. 2. A generic Entity Relationship loop

More generally, consider an arbitrary path between two entities, E_1 and E_n. Denoting a path from entity i to entity j as $R_{i,j}$, a composite path from E_1 to E_n will comprise:

$$R_{1,n} = [E_1], R_{1,2}, [E_2], R_{2,3},\ldots..R_{n-1,n}, [E_n] \quad (1)$$

An informal proof of the composition $R_{1,n} = R_{1,2}.R_{2,3}\ldots.R_{n-1,n}$ follows.

Consider first the signature of the path $R_{1,n}$, from E_1 to E_n.

Degree: If any $R_{i,j}$ (j > i) of the relationships $R_{1,2}.R_{2,3}….R_{n-1,n}$ is a multifunction, then the whole path is a multifunction.

Proof: If R_{ij} is a multifunction, then some $e_i \in E_i$ maps to several $e_j \in E_j$. Even if the degree of all the remaining $R_{j,j+1}..R_{n-1,n}$ is one, e_i therefore maps to several $e_n \in E_n$. Further, even if the degree of all $R_{1,2}$ .. $R_{i-1,i}$ is one, so that some $e_1 \in E_1$ maps only to e_i, it follows that e_1 will map to several e_n.

Optionality: If any $R_{i,j}$, from E_i to E_j, of the relationships $R_{1,2}.R_{2,3}….R_{n-1,n}$ is partial (j>i), then the whole path from E_1 to E_n is partial.

Proof: If $R_{i,j}$ is partial, then there can be some $e_i \in E_i$ that does not map to any $e_j \in E_j$. Since the path from E_i to E_n is via E_j, then e_i may therefore not map to any $e_n \in E_n$. Further, any $e_{i-1} \in E_{i-1}$ that maps only to e_i may also not map to any e_n, and so, recursively, any $e_1 \in E_1$ which maps only to e_i can not map to any e_n. Hence, there may be an e_1 which does not map to an e_n, and $R_{1,n}$ is partial.

The converses of each of these proofs – that path $R_{1,n}$ is a function if all $R_{i,j}$ are functions, and that path $R_{1,n}$ is total if no $R_{i,j}$ is partial – follow trivially.

Hence, the overall degree of $R_{1,n}$ is one IFF the degree of ALL $R_{i,j}$ in $R_{1,n}$ is one, and the path is mandatory IFF all $R_{i,j}$ are mandatory. Analogous arguments apply to $R_{n,1}$ for the inverse signature.

Thus, if the optionality, $O_{i,j}$, of relationship $R_{i,j}$ (i.e., the required participation of E_i in the relationship $R_{i,j}$) is represented by boolean values '1' if it is mandatory and '0' otherwise, then the optionality of the composition $R_{1,n}$, using the same convention, is

$$O_{1,n} = O_{1,2} \wedge O_{2,3} \wedge \ldots \wedge O_{n-1,n} \tag{2}$$

Similarly, representing the degree, $D_{i,j}$, of $R_{i,j}$ by '1' if the degree is one and '0' otherwise, the degree of the composition $R_{1,n}$ is

$$D_{1,n} = D_{1,2} \wedge D_{2,3} \wedge \ldots \wedge D_{n-1,n} \tag{3}$$

The four components of the complete signature, $S_{i,j}$, of a binary relationship, $R_{i,j}$, may be coded as four binary values; the order corresponds to the graphical representation of the relationship:

$$S_{i,j} = (D_{j,i}, O_{i,j}, O_{j,i}, D_{i,j}) \tag{4}$$

For example, the relationship $R_{a,c}$ ("C-A") in Figure 2 has signature $S_{a,c}$ = (1010). The overall signature of a composite path, $R_{1,n}$, is found as:

$$S_{1,n} = S_{1,2} \wedge S_{2,3} \wedge \ldots \wedge S_{n-1,n.} \tag{5}$$

For the alternate path in Figure 2, $S_{a,b}$ = (1010) and $S_{b,c}$ = (1110). Hence, the signature of the alternate path, $S'_{a,c}$ = (1010) $\wedge$ (1110) = (1010).

This coding provides a general method for composing the relationships in a path within an Entity Relationship model, and for comparing alternate paths. For paths of length 2, the results are the same as those obtained by exhaustive search in [3].

5. Detection Strategies

Given an algorithm to compose the signature of a path of arbitrary length, it is necessary then to have a strategy to apply that algorithm.

There are three obvious strategies, all of which imply an exhaustive search of the ER model. The first is to consider each relationship in turn, and all alternative paths, if any, between the entities connected by that relationship. The second is to focus on each entity in turn, and the set of relationships starting from that entity, and to compose all possible paths from that entity, until all possible alternative routes have been considered for each relationship. The third strategy is to consider the ER model as a whole, building, first, all paths of length 2, comparing those paths with the initial relationships, and, after eliminating any redundant relationships implied by paths of length 2, continuing the search with paths of length 3, and so on.

The first two strategies are fundamentally similar, differing only slightly in their termination criteria. Termination depends, in both cases, on inconsistency between the signatures of a path and a relationship. If the signature of the relationship *R* is S_R, and that of the path *P* is S_P, then, applying Equation (5), *P* is inconsistent with *R* if:

$$S_R \wedge S_P \neq S_R \tag{6}$$

In both cases, particular sub-paths may be composed several times, and both also suffer from the lack of clear termination conditions, discussed in Section 6, in the presence of loops, cycles and multiple relationships between pairs of entities.

Nevertheless, the first strategy has been used as the basis for a prototype CASE environment constructed using Peerlogic's Toolbuilder meta-CASE tool [22]. That prototype has demonstrated that it is feasible to generate an appropriate dialogue to guide the interaction with the analyst to resolve the issue of semantic equivalence, and, furthermore, to modify the ER diagram as a result of that interaction.

The third strategy seems attractive, since it reduces a graph search problem to a one analogous to matrix multiplication. The ER model is represented by an adjacency matrix, in which each cell contains the signature S of the relationship from its "row" entity to its "column" entity. The signatures for paths of length 2 are then found by pre-multiplying the adjacency matrix by itself, to give a matrix of order 2. In this matrix, each cell contains a list of signatures, derived using equation (5), each associated with an intermediate entity, so that the path can be reconstructed as ("from", "intermediate", "to"). Paths of greater length are found by pre-multiplying the higher-order matrix, repeatedly, by the original matrix, appending new "intermediate" entities to the cell entries to represent the appropriate partial paths.

At each stage, the path signatures are compared with those of the relationships in the initial matrix. If any path signature is equal to that of the corresponding simple relationship, then the relationship is potentially redundant, and clarification can be sought from the client. Relationships found to be redundant are removed from the base matrix, and paths containing them from the higher order matrices.

We introduce here an assumption that renders the algorithm tractable, sound but incomplete; the extent of the incompleteness is discussed in Section 6. We assume that no path may visit any entity more than once - that is, loops (and cycles) are not considered. This implies both that there can be no diagonal elements in the adjacency matrix, and that any paths which include an entity more than once can be discarded.

The algorithm is illustrated by an example. Figure 3 represents a scenario for a small software house that uses contract employees to work on projects sponsored by clients. The contract employees' time is billed directly to the client who sponsors their project, and the projects are managed by managers within the software house. The managers may also act as the contact for specific clients, and may supervise a number of contract employees.

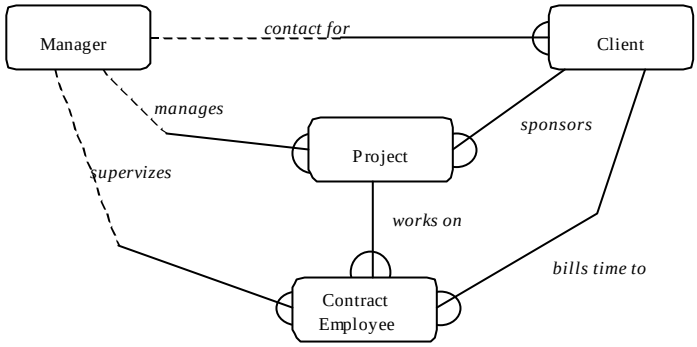

Fig. 3. An ER diagram with potential redundancy

Table 1 is the initial adjacency matrix for this simple scenario, although the signatures are represented by relationship lines rather than by the corresponding 4-tuple of (4). It is clear that the matrix is anti-symmetric, as are all matrices of higher order. In higher-order matrices, not only are the directions of the relationships reversed in diagonally symmetric cells, but so also is the order of entities in any partial path. These entities are represented in the partial paths by the letters M (*Manager*), C (*Client*), P (*Project*) and CE (*Contract Employee*).

Table 1. The Initial Adjacency Matrix

From \ To	Manager	Client	Project	Contract Emp
Manager		- - - ——∈	- - - ——∈	- - - ——∈
Client	∋—— - - -		————∈	————∈
Project	∋—— - - -	∋————		————∈
Contract Emp	∋—— - - -	∋————	∋————	

In Table 2, there are four paths (each appearing twice) that have the same signature as the corresponding simple relationship. These paths are highlighted in the table, and suggest that the corresponding relationships may be redundant. So, for example, it may be that a *Manager* manages a *Project* because (s)he is the contact for the *Client* who sponsors that *Project*; thus, the path *Project* - *Client* - *Manager* may imply the relationship "manages". Similarly, the path *Contract Employee* - *Project* - *Client* may imply the relationship "bills time to", but it may be that, in this particular company, neither of the paths *Contract Employee* - *Project* - *Manager* or *Contract Employee* - *Client* - *Manager* imply the relationship "supervizes".

Table 2. Adjacemcy Matrix of Order 2

From \ To	Manager	Client	Project	Contract Emp
Manager		P ∋- - - ——∈ CE ∋- - - —∈	C - - - ——∈ CE ∋- - - —∈	C - - - ——∈ P - - - ——∈
Client	P ∋—— - - -∈ CE ∋— - - -∈		M ∋- - - - - -∈ CE ∋———∈	M ∋- - - - - -∈ P ————∈
Project	C ∋—— - - - CE ∋— - - -∈	M ∋- - - - - -∈ CE ∋———∈		M ∋- - - - - - -∈ C ∋———∈
Contract Emp	C ∋—— - - - P ∋—— - - -	M ∋- - - - - -∈ P ∋————	M ∋- - - - - -∈ C ∋———∈	

Removing the two redundant relationships (*manages* and *bills time to*) from the initial matrix, the 2nd order matrix reduces to Table 3. This is then premultiplied by the (reduced) initial matrix, to give the 3rd order matrix in Table 4. The matrix is, again, anti-symmetric, but, for clarity, the lower triangle includes the four paths formed during the multiplication that are discarded because they are loops.

There is just one path in the 3rd order matrix with the same signature as the corresponding simple relationship: *Manager - Client - Project - Contract Employee.* As before, let us assume that this path does not imply the relationship "supervizes".

Table 3. Adjacancy Matrix, Order 2, after removal of redundant relationships

From \ To	Manager	Client	Project	Contract Emp
Manager			C - - - ——∈ CE ∋- - - —∈	
Client				M ∋- - - - - -∈ P ————∈
Project	C ∋—— - - - CE ∋— - - -∈			
Contract Emp		M ∋- - - - - -∈ P ∋————		

If loops are not permitted, the longest path possible in a model with four entities will involve three relationships. Hence, the order 3 matrix is the last that needs to be calculated, and the final model is that shown in Figure 4. In general, the algorithm terminates sound at the matrix of order N-1, when the ER model contains N entities.

Clearly, however, the contents of the matrix grow rapidly as paths of increasing length are derived. Fortunately, this can be mitigated by several heuristics. First, the removal of redundant relationships at the earliest opportunity can reduce significantly the number of paths in higher order matrices.

Second, the complexity of paths retained in the matrix can be limited. For example, if all the relationships in the ER diagram had degree no more than one to many, a

criterion could be set to exclude all paths "less constrained" than one to many. Defining the signature S_0 to be that for a fully optional one to many signature, (1,0,0,0), and S_0^T as its inverse, (0,0,0,1), the criterion for inclusion of a path P, with signature S_P, in a higher-order matrix is then:

$$(S_0 \wedge S_P = S_0) \vee (S_0^T \wedge S_P = S_0^T) \tag{7}$$

Finally, the initial adjacency matrix will normally be sparse, unlike the relatively dense example discussed here. Furthermore, the large number of paths that populate the higher order matrices are precisely the same as those that would be derived in either of the first two strategies, but that, using the matrix approach, the partial paths are derived only once rather than for each entity or relationship.

Table 4. Adjacancy matrix of Order 3

From \ To	Manager	Client	Project	Contract Emp
Manager		CE.P ∋- - - —∈		C.P - - - —∈
Client	P.C ∋— - - -∈ P.CE ∋— - - -∈		M.CE ∋-- - - -∈	
Project		CE.M ∋ - - - -∈ CE.P ∋——∈		C.M ∋- - - - -∈
Contract Emp	P.C ∋— - - - P.CE ∋— - - -∈		M.C ∋- - - - -∈ M.CE ∋-- - - -∈	

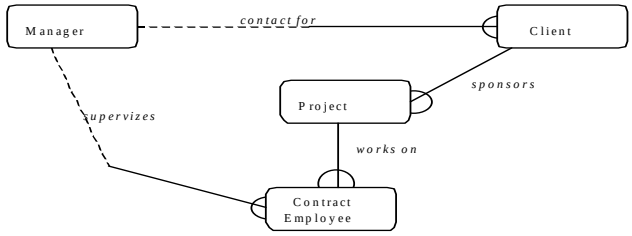

Fig. 4. ER Diagram after removal of redundant relationships

6. Outstanding Issues

The algorithm presented herein is an exhaustive search for closed paths within an ER model, and a search for possible redundancies within each closed path. As such, it does not scale well: the complexity of the matrix algorithm can be shown to be potentially $O(N^{N+1})$, where the model contains N entities. Restricting the search to

cycles, in which no entity is visited more than once, reduces the complexity to $O(N^3.(N-2)!)$, which still seems somewhat impractical.

Fortunately, the initial adjacency matrix is likely to be sparse. If the average number of relationships in which an entity participates is 2ϕ, then the likely complexity of the search could fall to $O(N^3.\phi^N)$. Experience suggests that ϕ is likely to be of the order of 2 or less, indicating that the sheer computational complexity of the approach may be high but, conceivably, attainable.

This analysis ignores the interaction with the client. Whilst such interaction may result in the removal of one or more redundant relationships, the more significant issue is that the number of interactions with the client could, itself, be huge. Even a model containing no redundant relationships may contain several which *could* be, if the semantics were ignored. Each potential redundancy requires an interaction with the client for the distinct semantics to be confirmed. Empirical studies of the scale of this problem will be explored as the next phase of this research.

A related issue is the order in which potential redundancies might be tested. The algorithm detects cycles in order of increasing length, and removal of any redundancies thus exposed can eliminate longer cycles before they are even detected.

Further, adoption of a filter such as (7) could restrict the search to potentially redundant 1:many relationships. Not only would this reduce the effective complexity of the search, but clarification would no longer be sought for a large number of "implied" unconstrained (many to many) relationships. The matrix algorithm is particularly suited to successive relaxation of such a filter, and, again, empirical studies will be undertaken with the implemented algorithm.

Empirical studies will be required also to establish the effectiveness of asking the client - albeit working with an analyst - to decide whether or not a path carries the same semantics as a single relationship. The comprehension difficulties that trigger the inclusion of a redundant relationship, rather than relying on its implication by a path, may make it difficult to decide on the equivalence of the two sets of semantics.

Although there is a tacit assumption in Section 5 that there will be only a single relationship in the ER model between any pair of entities, it is clear that a simple extension of the matrix representation could accommodate reflexive or multiple relationships. Unfortunately, this would also complicate the termination condition for the algorithm. Specifically, it may not be possible to justify restricting the search to simple cycles; indeed, it is possible to envisage examples, such as that in Figure 8, in which the reflexive "supervizes" relationship might be implied by the fact that one employee manages a project on which other employees work.

Hence, the algorithm presented in this paper, whilst sound, is not complete, in that it is possible to conceive ER models containing specific types of redundancy, involving loops and parallel relationships, that might not be detected. These issues are being explored currently, and extensions to the algorithm are being considered.

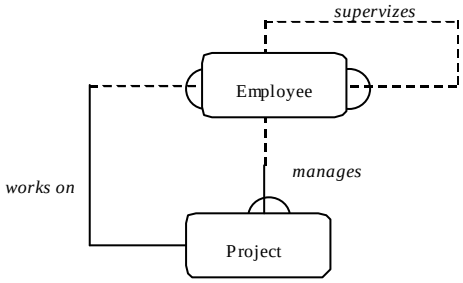

Fig. 5. An ER model in which a relationship may be implied by a cycle

A further issue is the identification of structures that misrepresent higher-order relationships. The best known of these is the cyclic many:many:many structure. Typically, such structures are "resolved" either by the introduction of an entity to represent a ternary relationship between the three entities, or by recognising the join dependencies between relations representing the explicit relationships. In the latter case, the schema violates 5^{th} normal form. The algorithm presented here could draw attention to the cyclic structure, but can suggest no appropriate resolution until the detection of such cliches has been incorporated.

This investigation has been pursued in the context of Entity Relationship models, since the underlying semantics of the notation are well understood. Once the problems identified in this section have been resolved, it should be trivial to extend the approach to other graphical notations - such as Class Diagrams in UML.

7. Conclusions

The issue of redundant relationships within Entity Relationship models has been identified as one that is significant for model quality. A technique has been presented for locating relationships whose set theoretic signature is implied by some alternative path within the model. This alone is insufficient for the relationships in question to be redundant; it is necessary for the semantics also to be equivalent, and determining this always requires interaction with the client for whom the model has been constructed.

Although the algorithm is sound, it is both computationally expensive and incomplete for certain special cases. This suggests a number of issues, including the most important question of usability, which are currently receiving attention.

Nevertheless, the technique as presented in this paper seems already to be a useful tool for improving the quality of Entity Relationship models, and which should reduce the requirement for normalization of generated relational schemas. Furthermore, the approach should generalise to more complex notations.

References

1. Teorey, T.J., Yang, D., Fry, J.P.: A logical design methodology for relational databases using the extended entity-relationship model. ACM Computing Surveys **18**(2) (1986) 197 - 222
2. Bowers, D.S.: From Data to Database. Van Nostrand Reinhold, London (1987), 164 - 179
3. Dullea, J., Song, I-Y.: An Analysis of Cardinality Constraints in Redundant Relationships, Proc. CIKM 97 (1997) 270-277
4. Rosenthal, A., Reiner, D.:Tools and Transformations - Rigorous and Otherwise - for Practical Database Design, ACM Transactions on Database Systems **19** (2) (1994) 167 - 211
5. Lenzerini, M., Nobili, P.: On the Satisfiability of Dependency Constraints in Entity-Relationship Schemata, Proc. 13th International Conference on Very Large Databases, Brighton (1987), Morgan Kaufmann, 147-154
6. Hartmann, S.: On the Consistency of Int-cardinality Constraints, Proc. ER-98, 17th International Conference on Conceptual Modelling, Singapore (1998), 150-163
7. Shanks, G., Darke, P.: Understanding corporate data models, Information & Management **35**, (1999) 19-30
8. Batra, D., Antony, S. R.: Novice errors in conceptual database design, European Journal of Information Systems, **3**(1) (1994) 57-69
9. Atkins, C., Patrick, J.: NaLER: A natural language method for interpreting E-R models, Proc. International Conference Software Engineering: Education and Practice, (1998) 2-9
10. Hay, D.C.: Making data models readable, Information Systems Management, **15**(1), (1998) 21-33
11. Kesh, S.: Evaluating the quality of entity relationship models, Information & Software Technology, **37**(12) (1995) 681-689
12. Moody, D.L..: Metrics for evaluating the quality of entity relationship models, Proc. ER-98, 17th International Conference on Conceptual Modelling, Singapore (1998), 211-225
13. Moody, D.L., Shanks, G.G., and Darke, P, Improving the quality of entity-relationship models – experience in research and practice, Proc. ER-98, 17th International Conference on Conceptual Modelling, Singapore (1998), 255-276
14. Morris, A., The Key, the Whole Key, and Nothing But…, Application Development Adviser, **2**(4) (1999) 64-67
15. Agrawal, W.D., Bushnell, M.L, Lim, P.: Redundancy Identification Using Transitive Closure, Proc. ATS-96, 5th Asia Test Symposium (1996) 4-9
16. Chang, S.-C., Cheng, D.I., Yeh, C.-W.: Removing multiple redundancies in combinational circuits, IEE Proc. Comput. Digit. Tech. **149**(1) (2002) 1-8
17. Schmolze, J.G., Snyder, W.: Detecting redundancy among production rules using term rewrite semantics, Knowledge Based Systems **12** (1999) 3-11
18. Dar, S., Ramakrishnan, R.: A Performance Study of Transitive Closure Algorithms, Proc. ACM - SIGMOD - 94, (1994) 454-465
19. Agrawal, R., Borgida, A., Jogadish, H.V.: Efficient Management of Transitive Relationships in Large Data and Knowledge Bases, ACM - SIGMOD - 89 (1989) 253 - 262
20. Sridhar, R., Iyengar, S.S.: Efficient parallel Algorithms for Functional Dependency Manipulations, Proc. 2nd International Symposium on Databases in Parallel and Distributed Systems, Dulbin (1990) 126 - 137
21. Ausiello, G., D'Atri, A, Sacca, D.: Graph Algorithms for Functional Dependency Manipulation, Journal of the ACM **30** (4) (1983), 752 - 766
22. Bowers, D, Database Schema Improvement Techniques for CASE Tools, proc. UKAIS 2000 (2000) 266-275