Multirelational Semantics for Extended Entity-Relationship Schemata With Applications*

Sebastiano Vigna

Dipartimento di Scienze dell'Informazione

Università degli Studi di Milano

Via Comelico 39/41, I-20135 Milano MI, Italy

`vigna@acm.org`

Abstract

This paper describes a multirelation semantics for a fragment of the Extended Entity-Relationship schemata formalism based on the bicategorical definition of multirelations. The approach we follow is elementary—we try to introduce as few notions as possible. We claim that bicategorical algebra handles gracefully multirelations and their operations, and that multirelations are essential in a number of applications; moreover, the bicategorical composition of multirelations turns out to correspond to natural joins. From the formal semantics we derive an algorithm that can establish statically the possibility of building parallel ownership paths of weak entities. The ideas described in this paper have been implemented in a free tool, ERW, which lets users edit sets and multirelations instantiating an EER schema *via* a sophisticated web interface.

1   Introduction

Entity-Relationship (ER) schemata are a popular conceptual model. They were originally introduced by Chen [Che76], and later extended in several ways; the more common extensions are usually termed *Extended ER* (EER) [Tha00]. The basic idea is that using sets and relations we can model objects of the real world and their inter-relationships. An EER schema is used to design conceptually a database: then, a *reification* process produces a logical database schema (e.g., in SQL) [EN94].

The starting point of this paper was the creation of a set of specifications and tools implementing the following idea: that conceptual design is sufficient to completely define an application (at least in a default, standardized form). If our goal is to maintain a set of entities and relationships whose types are defined by an EER schema, then the schema should be sufficient to produce automatically a database and a decent user interface.

Since our first requirement was platform independence, we decided that the user interface should be web-based. Only a few years ago, poor standardization of browser document object models, and lack of flexibility, would have made this goal impractical, but this is no longer true.

---

*Note that this version is slightly amended w.r.t. the one presented at ER 2002. In particular, Definition 5 has been corrected.

The result is ERW, a system that, essentially, lets you edit instances of EER schemata. We stress the fact that the user edits such instances, rather than databases, because in our view the underlying relational database is just a support for a clearly defined multirelational semantics for schema instances.

More in detail, one defines an EER schema using ERL, an XML-based language (currently only a fragment of the EER formalism is supported, the most notable omissions being *n*-ary relations and multiple-value attributes). Then, a Java[TM] tool named ERtool reifies the schema into a set of SQL tables using a standard algorithm (ERtool is also able to generate SGML documentation about the reification process). Moreover, it produces a set of definition files that are used by a run-time environment written in PHP, a powerful and very popular server-side scripting language. The run-time environment creates forms that let the user interact with the schema instance, with natural operations such as "associate this entity to this entity", and so on. ERW is free software downloadable from `http://erw.dsi.unimi.it/`.[1]

During the development of ERW, we decided to found the manipulation of entities and relationships on a clear, completely formal semantics. This was made necessary by the complexity of the database management code, which has to handle cardinality constraints, subtypes and ownership. In applications, moreover, we found that *multirelations* are very useful. In a multirelation, two entities can be related "more than once". For instance, if a library has an entity type for persons and one for books, the "loan" relationship type (with, say, start and end date as attributes) will have to be instantiated by a multirelation with attributes, as a customer can borrow a book several times.

Since defining multirelations in terms of tuples leads to the same problems that plague improperly defined multisets [Mon87, Bli89], we decided to try a different approach, by using a natural generalization of the categorical definition of relation. The result is a very natural semantics, also because the object and arrows in categories parallel exactly the notions of entity type and relationship type in EER schemata.

Another obstacle met during the development of ERW was the "weak status of weak entity types" [BS99]. Detecting double ownership is a nontrivial problem if subtyping gets in the way, as ownership can be inherited (if subtyping is not allowed, detecting double ownership is trivial).

This paper presents the solutions we have found for the problems above. First of all, it extends the semantics of EER schemata by providing a clear, mathematically sound description of multirelations and their operations. Note that this does not require a change in the EER syntax: multirelations are simply an extended *cardinality constraint*, that complements the well-known (0:1), (1:N) and so on. The semantics is given in the same spirit of the original paper by Chen [Che76]: entities and relationships are *not* tuples; rather, they have a well-defined semantics on their own; then, attributes are defined as functions mapping entities and relationships into a suitable domain.[2]

Finally, we present a static double-ownership detection algorithm that works for multiple inheritance and multiple owners, and prove it sound and complete using the formal semantics above. The algorithm is currently implemented in ERtool.

---

[1]Note that WebML [CFB00] has a certain intersection with ERW. There are two main differences between ERW and WebML. The first is in focus: ERW does not generate web sites, but just sophisticated user interfaces accessible with a browser. On the other hand, the database support of ERW is much wider than that of WebML, and includes relations with attributes, SQL-expressions based default attribute values, static and dynamic enumerative types, weak entities, subtyping and so on. The suite of *Extended Entity-Relationship Database Tools* [MS94] has a much wider EER support, but no web user interface.

[2]Of course, several other authors have given formal semantics to EER schemata (see, for instance, [GH91] and the references therein). However, to our knowledge none has allowed multirelations.

A final note for the reader: Sometimes our notation may seem unorthodox; we have made an effort to adopt a terminology as close to current practice in the ER community as possible, and we hope the reader will forgive us if sometimes we use an unfamiliar mathematical notation. Sometimes, moreover, we give hints on possible applications of categorical ideas to EER semantics; in this case, we do not give full definitions, and we refer the reader to a standard textbook on category theory (e.g., the classic [Mac71]).

2   The Categorical View of Binary Relations

A binary relation R from set X to set Y is a subset of the cartesian product $X \times Y$, that is, a set of pairs of the form $\langle x, y \rangle$. This is the well-known set-theoretical notion of relation. *Composition* of relations is easily obtained with the "common middle" operation: if R goes from X to Y and S goes from Y to Z then RS is defined as the set of pairs $\langle x, z \rangle \in X \times Z$ for which there is a $y \in Y$ such that $x \ R \ y$ and $y \ S \ z$.

Category theorists see relations in a different way [SCK84]: a relation is a set R (*not* a set of pairs), endowed with two functions, the *left (first) projection* and the *right (second) projection*, often called *left leg* and *right leg* because of the typical way they are drawn:

$$
\begin{array}{ccc}
 & R & \\
\pi_0^R \swarrow & & \searrow \pi_1^R \\
X & & Y
\end{array}
$$
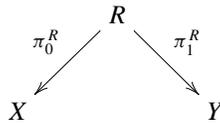
Moreover, we require that that two legs are *jointly monic*: by this we simply mean that there are no elements $r, s \in R$ such that $\pi_0^R(r) = \pi_0^R(s)$ and $\pi_1^R(r) = \pi_1^R(s)$.

A reader acquainted with EER schema reification will certainly notice "this looks like an SQL table reifying a relationship type", and indeed the interesting point about the categorical definition of relation is that it is very close to the relational algebra used to formalize relational databases.

Of course, the interest of category theorists lies elsewhere: *this* definition of relation is valid for many other mathematical structures: groups, topological spaces and so on. Once the objects you manipulate and the *morphisms* (functions, on the case of sets) that connects them are defined, the diagram above tells you what is a relation without even referring to elements (there is a way to specify the joint monicity condition without referring to elements, but we need not to be concerned with that).

Clearly, a categorical relation can be mapped to a relation, just by taking the set

$$
\left\{ \langle \pi_0^R(r), \pi_1^R(r) \rangle \ \middle| \ r \in R \right\}.
$$

Conversely, any set-theoretic relation can be made into a category-theoretic relation taking the two natural projections as legs.

2.1   Straight To Multirelations

We have not yet seen the real contribution of the categorical viewpoint. As we stated in the introduction, our goal is to give a firm mathematical foundation to a multirelational semantics of schemata.

How should we define multirelations? If we start from the set-theoretical definition, we encounter a number of difficulties. Should we assign a natural number of each pair, denoting its multiplicity, or use some other trick?

In the categorical view, instead, multirelations are *more natural* than relations. The definition of a multirelation is simply obtained by *dropping the joint monicity condition*, that is, via a true generalization process. We state this formally:

Definition 1 A (binary) multirelation from set X to set Y is a set M endowed with two functions, the *left leg M_0* and the *right leg M_1*:

$$
\begin{array}{ccc}
 & M & \\
M_0 \swarrow & & \searrow M_1 \\
X & & Y
\end{array}
$$
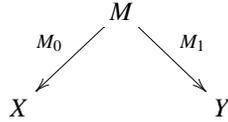
The reader should notice that now it can happen that two elements $r, s \in M$ satisfy $M_0(r) = M_0(s)$ and $M_1(r) = M_1(s)$. In this case, the elements $M_0(r)$ and $M_1(r)$ are related more than once.

A category theorist calls a set with two such functions a *span*; indeed, Definition 1 is simply the categorical definition of a span. The interesting point is that spans form themselves a *bicategory* (a structure slightly weaker than a category), and have a well-defined composition [SCK84]. Since we have now a clear definition of a multirelations, but no hint on their composition, we will look into the standard composition of spans.

2.2 Composition

Let us introduce the last categorical concept we need, the *pullback*. A pullback of two functions with the same codomain, say $f : X \to Z$ and $g : Y \to Z$, is defined as the universal commutative square of functions

$$
\begin{array}{ccc}
P & \xrightarrow{\alpha_0} & X \\
{\scriptstyle \alpha_1}\downarrow & & \downarrow{\scriptstyle f} \\
Y & \xrightarrow{g} & Z
\end{array}
$$
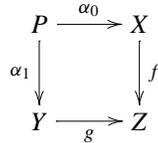
By *commutative*, we mean that paths starting and ending at the same points of the diagram give the same functional composition. In the case above, it means that doing α_0 followed by f is the same as α_1 followed by g, that is, that the equation

$$
f \circ \alpha_0 = g \circ \alpha_1 \tag{1}
$$

is satisfied. By *universal*, we mean that this diagram is the best possible, in the following sense: given any other commutative diagram with the same shape, say

$$
\begin{array}{ccc}
Q & \xrightarrow{\beta_0} & X \\
{\scriptstyle \beta_1}\downarrow & & \downarrow{\scriptstyle f} \\
Y & \xrightarrow{g} & Z
\end{array}
\tag{2}
$$

4

there is *exactly one* function $Q \to P$ such that the following diagram commutes:

$$
\begin{array}{ccc}
Q & & \\
 & \searrow^{\beta_0} & \\
\beta_1 \downarrow & P \xrightarrow{\alpha_0} X & \\
 & \alpha_1 \downarrow \quad \downarrow f & \\
 & Y \xrightarrow{g} Z &
\end{array}
\qquad (3)
$$

In this case, P, together with the *projections* α_0 and α_1, is called the pullback of f and g. Note that commutativity of (3) is a fairly complicated condition to write as a set of equations, since there are many paths of arrows between two nodes.

All this may seem very abstract, but it's closer to databases than it could seem. Let us try to build a set P satisfying the requirements above: first of all we notice that to each element p of P we can assign an element of X ($\alpha_0(p)$) and an element of Y ($\alpha_1(p)$). Thus, we can assign to each element of P a pair $\langle x, y \rangle$. Of course, not all pairs are possible: indeed, since (1) must be satisfied, it must always happen that $f(x) = g(y)$. Moreover, it is not possible that two elements p and p' of P get the same pair $\langle x, y \rangle$: indeed, consider as set Q a singleton $\{q\}$ and define $\beta_0(q) = x$, $\beta_1(q) = y$. This would define a commutative diagram as in (2), and thus it should happen that there is a *unique* function from Q to P making (2) commutative: but this is not true! There are at least *two* such functions, the one mapping q to p and the one mapping q to p'. Thus no two elements of P can be mapped to the same pair of elements from $X \times Y$.

All in all, we can build a pullback P using the subset of $X \times Y$ that satisfy $f(x) = g(y)$, and using as projections the standard projections. Note that from a categorical viewpoint, this subset of $X \times Y$ is indeed *one* of the many possible ways of building a pullback of f and g. We insist on this point because it will be fundamental in what follows.

It is now time to show why ever we need pullbacks. How do category theorists define the composition of multirelations (spans)? It's easy: consider two multirelations

$$
\begin{array}{ccccc}
 & M & & N & \\
M_0 \swarrow & & \searrow M_1 \quad N_0 \swarrow & & \searrow N_1 \\
X & & Y & & Z
\end{array}
$$

Now, note that the "v"-shaped pair of maps looks really like a candidate for building a pullback. This brings us to

$$
\begin{array}{ccccc}
 & & P & & \\
 & \alpha_0 \swarrow & & \searrow \alpha_1 & \\
 & M & & N & \\
M_0 \swarrow & & \searrow M_1 \quad N_0 \swarrow & & \searrow N_1 \\
X & & Y & & Z
\end{array}
$$

5

By composing the α_i's with the external legs, we finally get our composition:

$$
\begin{array}{ccc}
 & P & \\
M_0 \circ \alpha_0 \swarrow & & \searrow N_1 \circ \alpha_1 \\
X & & Z
\end{array}
$$
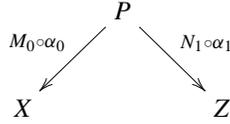
Which are the elements of P? Using the construction above, we would say that they are the pairs $\langle r, s \rangle \in M \times N$ such that $M_1(r) = N_0(s)$; the legs of the span map these elements to $M_0(r)$ and $N_1(s)$, respectively. But remember, this is just one of the possible constructions for the composition of M and N.

This does not tell us much. We can get more intuition about this operation if we take M and N to be relations, represented in the standard set-theoretical way, and their projections. Using again the construction above, we obtain that the elements of P are $\langle \langle x, y \rangle, \langle y', z \rangle \rangle$ such that $\alpha_1(\langle x, y \rangle) = \alpha_0(\langle y', z \rangle)$, that is, $y = y'$; all in all, P would contain the tuples of the form $\langle \langle x, y \rangle, \langle y, z \rangle \rangle$ (with $\langle x, y \rangle \in M$ and $\langle y, z \rangle \in N$).

This really looks like relational algebra: indeed, if M and N were relations we would have just defined the *equijoin* operator of relational algebra. Note that usually the *natural join*, in which duplicated columns are omitted, is taken as a distinct operation, but in our case this is not necessary: the set of triples $\langle x, y, z \rangle$ such that $\langle x, y \rangle \in M$ and $\langle y, z \rangle \in N$ *can as well be taken as the pullback of M_1 and N_0*, as the pullback is defined by a property of its projections, not as a specific set; in categorical-theoretic terms, the two definitions are *naturally*[3] *equivalent*. The projections simply take out of the triples the corresponding elements of X and Z, so that we can compose again.

The example of equijoin and natural join points out the important fact that *composition of multirelations is not unique, but defined up to isomorphism*. We shall not pursue the point here, but from the definition of pullback one can show that all possible pullbacks are naturally isomorphic. Equijoins and natural joins are just two possible pullbacks for relations. A consequence is that multirelations form a *bicategory*, as opposed to a *category*: composition is not unique, but defined up to a unique isomorphism.

All in all, spans are a nicely structured and clearly defined model for multirelations. Moreover, their natural composition reduces to the natural join in the case of relations, and thus the relational algebra is easily extended to an algebra of multirelations. We will not pursue here the details of this extension, but we believe that the connection between the bicategorical algebra of relations and relational algebra is very strong. By means of pullbacks, the usual properties of joins can be proved also for multirelations, and an EER calculus that defines precisely the semantics of queries can be derived.

3   Schemata and Instances

For the rest of the paper, we shall not be concerned with attributes. Our interest is in giving a clear semantics to EER schema instances using sets and multirelations, and attributes would uselessly clutter our presentation. It is not difficult to extend the bicategorical semantics given in the previous section adding an attribute map for every set involved (including the set defining a multirelation, which gives us also attributes for relationship types). At that point, one can easily discuss keys,

---

[3]The original purpose of category theory was exactly to give a precise definition of the term "natural" in mathematics.

attribute constraints and so on. Our interest now is mainly in a formal semantics that is sufficient to express cardinality constraints on multirelations and to validate the ownership structure of a schema.[4]

In our simplified view, an EER schema (of binary relations) $\mathscr{S}$ is given by a set $\mathscr{E}$ of entity types, a set $\mathscr{R}$ of relationship types, a source function $s : \mathscr{R} \to \mathscr{E}$ and a target function $t : \mathscr{R} \to \mathscr{E}$ (note that in order to give a formal semantics to an EER schema without roles you need to take directed relationship types). Moreover, each relationship type has a source and a target *cardinality constraint*, which is a symbol out of `(0:1)`, `(1:1)`, `(0:N)`, `(1:N)`, `(0:M)`, `(1:M)`. The ordered pair of cardinality constraint of a relationship type is usually written as `(-:-)→(-:-)`.

Finally, a relationship type may be marked optionally as either `ISA` or `WEAK`. In the first case, its constraint must be `(1:1)→(0:1)`; in the second case, it must be `(1:1)→(-:N)`.

Whenever there is an `ISA` relationship type from E to F, E is said to be a *direct subtype* of F, and F a *direct supertype* of E. A *subtype* of E is either E or a direct subtype of a subtype of E (analogously for supertypes).

With respect to typical EER schema definitions, we introduce two new cardinality types: `(0:M)` and `(1:M)`. The informal meaning of these types is that a relationship type with such specifiers can be instantiated by a multirelation; when `(0:N)` and `(1:N)` are used, instead, the relationship type must be instantiated by a relation. In this way we do not disturb the standard syntax of EER schemata to introduce our new semantics.

There is an important syntactic rule that must be respected: if one constraint of a relationship type is given by a `(-:M)` specifier, than the same must happen for the other one. Indeed, while all other constraints combine freely, if you allow a multirelation on the source you must do the same for the target, and viceversa.

3.1 Instances

We are finally in a position to give the definition of the multirelation-based schema semantics.

Definition 2 An instance σ for a schema $\mathscr{S}$ is given by a map σ assigning to each entity type E in $\mathscr{E}$ a set $\sigma(E)$ and to each relationship type R in $\mathscr{R}$ a span $\sigma(R)$ satisfying the following properties:

1. The left leg of $\sigma(R)$ must end in $\sigma(s(E))$, and the right leg in $\sigma(t(E))$; in other words, if R is a relationship type from entity type E to entity type F, then $\sigma(R)$ must be a span

$$\sigma(R)$$
$$\sigma(E) \qquad\qquad \sigma(F)$$
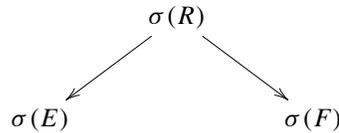
   that is, a span with a left leg ending in $\sigma(E)$ and a right leg ending in $\sigma(F)$[5].

2. A cardinality constraint of the form `(1:-)` requires that the corresponding leg be a surjective function.

---

[4]The same approach is used in the original paper by Chen [Che76], where entity sets and relationship sets are first used to give semantics to the relational structure of a schema, and attributes are defined afterwards by means of suitable mappings from those sets.

[5]The categorically-minded reader will have noticed that this condition represent simply the definition of a *functor* (from a graph to the bicategory of spans of sets). Indeed, from the definition above one could immediately derive a mathematical definition of *isomorphism* and of *transformation* of instances, using natural transformations [Mac71].

3. A cardinality constraint of the form `(-:1)` requires that the corresponding leg be an injective function.

4. A cardinality constraint of the form `(-:N)` on either leg requires that the span be jointly monic (i.e., it must represent a relation).

5. Whenever a relationship type is marked `ISA`, its bijective leg is the identity, and its injective leg is an inclusion.

6. Whenever entity types E and F have a common supertype and $x \in \sigma(E) \cap \sigma(F)$, there is a common subtype G of E and F such that $x \in \sigma(G)$[6].

Note how cardinality constraints map very naturally on function properties if we look at multirelations as spans. The reader should have no difficulty in showing that the above definitions correspond exactly to the usual constraints when we restrict to relations. For instance, if a relationship type R from entity type E to entity type F has a constraint `(0:1)` on E, then elements of $\sigma(E)$ should have at most one relationship with elements of $\sigma(F)$ (that is, we assume the *participation interpretation* of cardinality constraints). But this is exactly what condition (3) says: if the leg ending on $\sigma(E)$ is injective, it cannot happen that an $x \in \sigma(E)$ is related with two elements of $\sigma(F)$: this would require the existence of two elements $r, s \in \sigma(R)$ mapped to x, which is impossible if the leg ending in $\sigma(E)$ is injective.

The definition above treats subtyping as *set inclusion*: all elements assigned to a certain type *are also* assigned to supertypes. This is essentially the original semantics of subtyping given in [Che76], and also ERW treats subtypes this way. Indeed, the multirelation instantiating an `ISA` relationship type must have the following form:

$$
\begin{array}{ccc}
 & X & \\
{}^{\mathbf{1}_X}\swarrow & & \searrow^{\iota} \\
X & & Y
\end{array}
$$
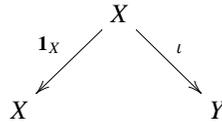
where $\iota : X \hookrightarrow Y$ is the inclusion of X as a subset of Y. This actually forces to interpret subtypes as subsets (of course, attributes of an element of $\sigma(X)$ viewed as an element $\sigma(X)$ or as an element of $\sigma(Y)$ will be different).

Condition (6) is essential to give a precise definition of what we mean by an *entity*. It forces, for each entity type E and each $x \in \sigma(E)$, the existence of a *unique* subtype F of E such that $x \in \sigma(F)$ but $x \notin \sigma(G)$ for any proper subtype G of F; such an F is called the *type of $x \in \sigma(E)$*. An *entity* is now a pair $E, x \in \sigma(E)$ such that E is the type of x. Note that we did not restrain entity sets to be disjoint, so an x whose type is E and an x whose type is F are actually *distinct entities*.

The definition we gave of cardinality constraints is also valid for multirelations. Since it is very natural, and it fits easily the classical case, we believe it is a step in the right direction.[7]

---

[6]The last two conditions are an elementary phrasing of the following condition: consider the binary-meet completion of the type order, and extend σ by union (i.e., for a new element x set $\sigma(x) = \bigcup_{y<x} \sigma(y)$); then, we require that σ is *stable*, that is, that it preserves the meet of elements with a common upper bound [Ber78].

[7]The reader should note that there is another possibility for condition (3): we could allow more than one relationship, but always with the same element. However, this would give rise to a rather clumsy definition, and we see no motivation for such a change.

3.2 Down To SQL

As we mentioned in the introduction, the semantics we have defined has the purpose of specifying what exactly ERW stores in a schema instance. Thus, we must guarantee that when the EER schema is reified by ERtool, the reification algorithm supports the multirelation-based semantics we have just discussed.

ERW's approach is very simple, and inspired by common practice in databases. To superimpose our abstract semantics on the tuple semantics of relational databases, we first need to be able to speak about elements and multirelations without referring to attributes. Thus, all tables generated by the ERW reification algorithm contain an `id` column, which is a natural number and acts as primary key for all entities and relationships[8]. Then, ERW follows standard practice, using a single SQL column to represent relationship types that instantiate to partial functions, and a support table for the remaining ones. Note that reification by means of a support table gives raise exactly to a span: each row is a separate element (because of the `id` column), but there are two additional columns pointing to the tables reifying the source and destination entity types; these columns represent the values of the left and right leg. We see again that the apparently abstract categorical view of relations is really tied to databases.

This concrete structure can be exactly mimicked by our bicategorical semantics: we just have to restrict our sets to sets of natural numbers, and we will have a faithful mathematical reproduction of our database. Composition of multirelations (by means of pullbacks) now models exactly the natural join on foreign keys.

4 Checking Subtyping and Ownership

ERW supports multiple inheritance and multiple owners. This means than every entity type can be defined as a subtype of any set of entity types (using `ISA` relationship types), and can have any number of owner entity types (defined using `WEAK` relationship types; in this case the relationship type is instantiated to an *identifying function*).

There are of course constraints on the directed[9] graph defined by these relationship types. Certainly, if we single out the `ISA` arcs we have to require that the resulting graph contains no cycles.

Checking the ownership relation is more complicated. Indeed, it has been proposed that weak entity types should be resolved [BS99] (i.e., eliminated and substituted by other constructs); the authors note in [BS99] that "[...] there can be multiple definition identification paths between a weak entity type to one of its owners (possibly indirect), or there can be cycles of ownership relations, ect. The understanding of such structures becomes hard, the unclear semantics turn the schemas incoherent, [...]".

Designing ERW we decided to solve the problem in the opposite direction: since we have a completely clear formal semantics, we shall give a polynomial algorithm that guarantees that it never happens for an entity to own another entity by means of two ownership paths[10].

Note that in some applications one is interested in checking that double ownership is not created *dynamically*, but does not forbid it in principle. For this cases our check will be too strict: nonetheless,

---

[8]Of course, this is not very far from the concept of *object identifier* in object-oriented databases. However, we can have the same identifier for two entities whose type is different.

[9]Recall that in definition a schema we gave source and target.

[10]Since the empty path is a legal ownership path, by which an entity owns itself, this definition also forbids cycles.

we shall prove soundness and completeness theorems showing that the check we perform is the best possible among the static ones.

The tricky part lies in the interaction with subtyping. If e is of type F which is an E, and an F owns W, then e could own entities of type W. If, in turn, W owns E, then we have realized a cycle, even if no cycle of WEAK types exists.

A tempting idea would be to analyze parallel paths between entity types, taking however also into consideration the arcs representing supertyping. This would solve the above problem.

However, it is easy to see that this test is not sufficient: consider a situation in which the entity type E owns W, F is an E and W owns F. In this case there is no cycle, but an entity of type F could own itself through W.

4.1 The Algorithm

We start with two definitions from graph theory [Ber85]:

Definition 3 A path is *simple* if it does not contain the same node twice, unless it is the first and the last node of the path. Two paths of a graph are said to be *parallel* if they start on the same node and end on the same node.

The problem in analyzing statically ownership is that WEAK relationship types link *entity types*, but ownership paths link *entities*. This is similar to static type-checking in object-oriented languages: one must always take into consideration the fact that the dynamic type of an object can be any subtype of its static type; in our case, it can happen that $x \in \sigma(E)$ but E is not the type of x.

To get a correct algorithm, we refer again to our bicategorical semantic for schema instances (in this case, the multirelations we handle are constrained to be functions, so the same proof applies also to the standard semantics).

Definition 4 A *path* in an instance of a schema $\mathscr{S}$ as a sequence of entities and relationships x_0, r_0, x_1, r_1, ..., x_{n-1}, r_{n-1}, x_n and types E_0, R_0, E_1, R_1, ..., E_{n-1}, R_{n-1}, E_n of $\mathscr{S}$ such that

1. E_i is the type of $x_i \in \sigma(E_i)$ for $0 \leq i \leq n$ and $r_i \in \sigma(R_i)$ for $0 \leq i < n$;

2. E_i is a subtype of the source of R_i for $0 \leq i < n$;

3. E_{i+1} is a subtype of the target of R_i for $0 \leq i < n$;

4. the left leg of $\sigma(R_i)$ maps r_i to x_i, and the right leg maps r_i to x_{i+1} for $0 \leq i < n$, that is, each relationship in the sequence relates the entities immediately before and immediately after.

If all relationship types R_i are marked WEAK for $0 \leq i < n$, we say that the path is an *ownership path*. A *cycle* is a path satisfying $x_0 = x_n$ and $E_0 = E_n$.

We believe this formal definition captures exactly the notion of ownership path; we now define a static condition on a schema that is equivalent to the possibility of building two distinct ownership paths between entities. By *distinct* we mean that the list of entities or the list of relationship types of the paths are different (note, however, that in our case we can equivalently check the list of relationship types only).

Given an EER schema $\mathscr{S}$, consider the graph having as nodes entity types. Now, for each relationship type R marked WEAK going from E to F, for each subtype E' of E and each subtype F' of F add an arc from E' to F' coloured by R. Denote this graph with $W(\mathscr{S})$.

Definition 5 We say that $\mathscr{S}$ has *double ownership* if $\mathscr{S}$ has a nonempty cycle or two parallel paths whose first colour is different.

This definition formalizes the idea that searching statically for double ownership we cannot move up and down the type hierarchy as much as we like: we *cannot go down after having gone up*. Indeed, if we expand a path of $W(\mathscr{S})$ into a walk[11] of ISA arcs and WEAK arcs, we will see that an ISA arc is never followed by a reversed ISA arc.

Theorem 1 If an EER schema $\mathscr{S}$ has not double ownership, no instance of $\mathscr{S}$ contains two parallel ownership paths.

Proof. See the appendix. ∎

Of course, the previous theorem is just half of what we need: it just guarantees that double ownership is strong enough to avoid parallel ownership paths, but it could be too strong. However, it turns out this is not the case:

Theorem 2 If an EER schema $\mathscr{S}$ has double ownership, there is a subschema $\mathscr{S}'$ of $\mathscr{S}$ and an instance of $\mathscr{S}'$ containing two distinct parallel ownership paths.

Proof. See the appendix. ∎

Note that the above theorem does not state the there is an instance *of* $\mathscr{S}$ that contains parallel ownership paths. The only reason for this limitation is that, in principle, complex interaction between cardinality constraints could make it impossible putting in an instance the entities which are necessary to build the paths. To all practical purposes, however, this distinction is irrelevant.

4.2 Running Time

Definition 5 can be turned into a reasonably fast implementation:

Theorem 3 Double ownership for a schema with n entity types, m WEAK relationship types and largest type hierarchy of size t can be checked in time $O\left(n^3 + tmn + t^2m^2\right)$.

Proof. See the appendix. ∎

Of course, t is bounded by n and m is bounded by n^2. However, in real schemata they are much smaller, as the underlying graph is usually sparse.

4.3 An example

In Fig. 1 you can see a fragment of the ownership/subtyping graph of a schema $\mathscr{S}$. Arrows ($\rightarrow$) denote subtyping, whereas double arrows ($\Rightarrow$) denote ownership (the owner is at the end of the arrow). The graph $W(\mathscr{S})$ is shown in Figure 2. The graph contains the the following pair of parallel paths:

$$J \xrightarrow{R} M \qquad\qquad J \xrightarrow{T} C \xrightarrow{U} E \xrightarrow{S} M$$

---

[11]Recall that a *walk* in a graph is a sequence of nodes and arcs $x_0, a_0, x_1, \ldots, x_{n-1}, a_{n-1}, x_n$ such that for each $0 \leq i < n$ either a_i goes from x_i to x_{i+1}, or a_i goes from x_{i+1} to x_i.
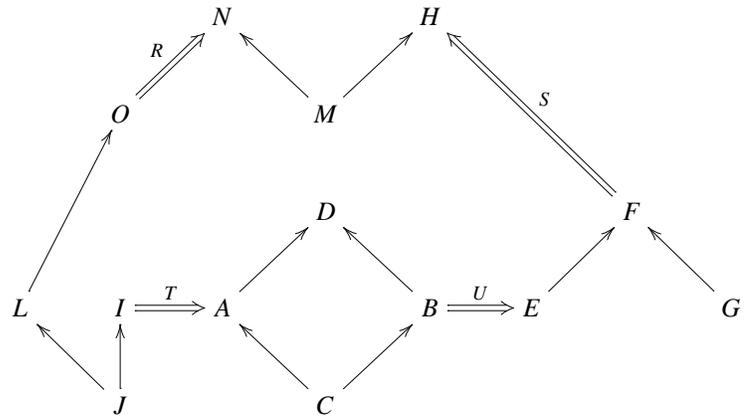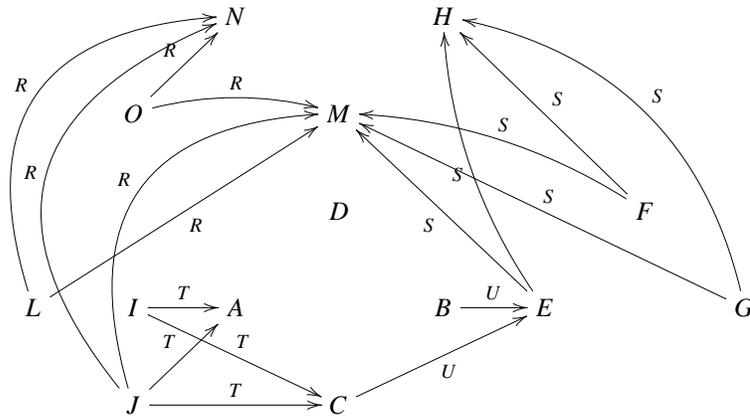
Figure 1: An example of ownership/subtyping graph.

Figure 2: The graph $W(\mathscr{S})$ for the example of Figure 1.

Now we can easily build the ownership paths: we could have an entity m of type M owning an entity j of type J (as M is a subtype of N, N owns O, and O is a supertype of J); on the other end, m could own an entity e of type E (as M is a subtype of H, H owns F, and E is a subtype of F), e could own an entity c of type C (as E owns B, and B is a supertype of C); finally, c could own j (as C is a subtype of A, which owns I, which is a supertype of J).

5 Acknowledgements

6 Conclusions

The semantics that ERW uses in modifying EER schema instances is firmly based on the definition of multirelations as spans, and their properties. As programmers, we found this description very practical. Moreover, the fact that pullbacks (a concept introduced more than fifty years ago!) models exactly natural joins of relation, and thus provides naturally a definition of join for multirelations, is a witness of the connection between the bicategorical algebra of spans and databases. Of course, nothing prevents to extend the multirelational semantics to n-ary relations, by using a set with n legs, or using more general cardinality constraints, by suitably revising Definition 2.

Spans have received a certain attention on their own recently in the theoretical computer science community; [BG01] contains many citations towards papers exploiting spans for several purposes. The authors note that their results have counterparts in the theory of relational algebras, thus supporting the viewpoint of this paper.

References

[Ber78]   G. Berry. Stable models of typed λ-calculi. In G. Ausiello and C. Böhm, editors, *Proceedings of the 5th Colloquium on Automata, Languages and Programming*, volume 62 of *Lecture Notes in Computer Science*, pages 72–89, Udine, Italy, 1978. Springer–Verlag.

[Ber85]   Claude Berge. *Graphs*. North–Holland, Amsterdam, 1985.

[BG01]   R. Bruni and F. Gadducci. Some algebraic laws for spans (and their connections with multirelations). In W. Kahl, D.L. Parnas, and G. Schmidt, editors, *Relational Methods in Software*, volume 44.3 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2001.

[Bli89]   Wayne D. Blizard. Multiset theory. *Notre Dame J. Formal Logic*, 30(1):36–66, 1989.

[BS99]   Mira Balaban and Peretz Shoval. Resolving the "weak status" of weak entity types in entity-relationship schemas. In *Proc. of 18th Int'l Conference on Conceptual Modeling ER'99*, number 1728 in Lecture Notes in Computer Science, pages 369–383, 1999.

[CFB00]   Stefano Ceri, Piero Fraternali, and Aldo Bongio. Web Modeling Language (WebML): a modeling language for designing web sites. In *Proc. Ninth World Wide Web Conference*, 2000.

[Che76]   Peter Pin-Shan Chen. The entity-relationship model: Toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, 1976.

[EN94]   Ramez Elmasri and Shami Navathe. *Fundamentals of Database Systems*. Benjamins/Cummings, 1994.

[GH91]   Martin Gogolla and Uwe Hohenstein. Towards a semantic view of an extended entity-relationship model. *ACM Transactions on Database Systems (TODS)*, 16(3):369–416, 1991.

[Mac71] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer–Verlag, 1971.

[Mon87] G. P. Monro. The concept of multiset. *Z. Math. Logik Grundlag. Math.*, 33(2):171–178, 1987.

[MS94] V. M. Markowits and A. Shoshani. An Overview of the Lawrence Berkeley Laboratory Extended Entity-Relationship Database Tools. In P. Loucopoulos, editor, *Proceedings of the 13th International Conference on the Entity-Relationship Approach*, volume 881 of *Lecture Notes in Computer Science*, pages 333–350, Manchester, United Kingdom, 1994. Springer-Verlag.

[SCK84] Ross Howard Street, Aurelio Carboni, and Stefano Kasangian. Bicategories of spans and relations. *J. Pure Appl. Algebra*, 33:259–267, 1984.

[Tha00] B. Thalheim. *Entity-Relationship Modeling*. Springer–Verlag, 2000.

Appendix: Proofs of Theorems

Proof of Theorem 1.   Suppose by contradiction that we have an instance of $\mathscr{S}$ containing two distinct parallel ownership paths.

If both ownership paths are not cycles, we can assume without loss of generality that the first relationship type on each path is different. Indeed, since WEAK types are instantiated to identification functions, if the first two relationship types on the paths are the same, then the second entities along the path must be the same (because of Condition (6) in the definition of instance), so we could strip the first relationship type and still have a pair of distinct parallel ownership paths. If both ownership paths are cycles, at least one must be nonempty.

We show how to turn these paths into parallel paths of $W(\mathscr{S})$. Let $x_0, r_0, x_1, r_1, \ldots, x_{n-1}, r_{n-1}, x_n$ be an ownership path with types $E_0, R_0, E_1, R_1, \ldots, E_{n-1}, R_{n-1}, E_n$.

Notice that each relationship type R_i has a corresponding arc in $W(\mathscr{S})$ starting from E_i and ending in E_{i+1}, because by definition E_i is a subtype of $s(R_i)$ and E_{i+1} is a subtype of $t(R_i)$. Thus, the resulting sequence of arcs in $W(\mathscr{S})$ forms a path from E_0 to E_n with colours $R_0, R_1, \ldots, R_{n-1}$.

Thus, either we get a nonempty cycle, or we get two parallel paths whose first colour is different, which is a contradiction. ▮

Proof of Theorem 2.   Let $E_0, a_0, E_1, \ldots, E_{n-1}, a_{n-1}, E_n$ be a path in $W(\mathscr{S})$ satisfying Definition 5 (here the E_i's are nodes, the a_i's are arcs, and a_i goes from E_i to E_{i+1}). Let $R_0, R_1, \ldots, R_{n-1}$ the list of colours (relationship types) on the path.

By definition of $W(\mathscr{S})$, E_i is a subtype of $s(R_i)$ and E_{i+1} is a subtype of $t(R_i)$. Thus, if we take entities e_i of type E_i we can choose relationships r_i of type R_i so to obtain an ownership path from E_0 to E_n in an instance of the subschema of $\mathscr{S}$ containing exactly the E_i's and the R_i's.

Since double ownership implies the existence of a nonempty cycle or of two paths in $W(\mathscr{S})$ whose first colour is different, we conclude that a nonempty ownership cycle or two distinct parallel ownership paths exist in an instance of a subschema of $\mathscr{S}$. ▮

Proof of Theorem 3. The problem in getting the time bound we have described is that the construction of $W(\mathscr{S})$ requires adding a large number of arcs. Indeed, for each WEAK relationship type R we could add t^2 arcs (where $t \leq n$ is the size of the largest weakly connected component in the graph having entity types as nodes and ISA relationship types as arcs). Since we will need to examine pairs of arcs, this would lead to a $O(t^4 m^2) = O(n^8)$ time bound.

We can obtain a better bound by noting that there is a redundance in the definition of $W(\mathscr{S})$: indeed, if there are several ways to get from entity type E to entity type F moving down and up the type hierarchy (for instance, when there are two types G and H that are both E and F), we will add

14

correspondingly many arcs. However, if we want to check double ownership the only relevant data is that one can get from E to F.

Consider again the graph having entity types as nodes and `ISA` relationship types as arcs. We call a *simple type walk* in this graph a simple walk formed by a sequence by reversed arcs followed by a sequence of directed arcs.

Consider now a graph $G(\mathscr{S})$ having as nodes entity types. We shall add some arcs coloured on the set $\mathscr{R} + \bar{\mathscr{R}}$ (i.e., a colour is a relationship type R or a "barred" relationship type $\bar{R}$).

For each `WEAK` relationship type R and each entity type E reachable *via* a type walk from $t(R)$ we add an arc from $s(R)$ to E coloured by R. Analogously, for each `WEAK` relationship type R and each entity type E reachable *via* a type walk from $s(R)$ we add an arc from E to $t(R)$ coloured by $\bar{R}$.

We claim that there is a nonempty cycle in $G(\mathscr{S})$ entirely made of non-barred arcs iff $W(\mathscr{S})$ has a nonempty cycle. Analogously, there are two nodes E, F of $G(\mathscr{S})$, and two paths α, β from E to F such that α is entirely made of non-barred colours, and β is entirely made of barred colours, and the first colours on the two paths are not the same (discarding bars), iff $W(\mathscr{S})$ has two parallel paths whose first colour is different.

To see why this is true, it is sufficient to note that if we expand α and β in $\mathscr{S}$, and concatenate α with the reverse of β we obtain a closed walk (the same happens with a cycle).

We now divide in blocks the walk: each block is a maximal subpath formed by a `WEAK` arc preceded by directed `ISA` arcs and followed by reversed `ISA` arcs. Each block is clearly an arc of $W(\mathscr{S})$, and this correspondence makes trivial to build two parallel paths in $W(\mathscr{S})$ (or a cycle) satisfying Definition 5.

The reverse implication can be obtained analogously, dividing in blocks the walk obtained by expanding a cycle in $W(\mathscr{S})$ so that each block is made by a `WEAK` arc and a type walk (note that in this process several cycles of $W(\mathscr{S})$ with the same sequence of colours will give rise to the same cycle of $G(\mathscr{S})$, as we reduced redundancy). Analogously, if we have two parallel paths we divide the resulting closed walk in blocks so that one of the paths turns into a sequence of `WEAK` arcs followed by a type walk, and the other into a sequence of type walks followed by a `WEAK` arc, thus providing a barred and a non-barred path that are parallel.

How much time does it take now to check for the existence of such paths in $G(\mathscr{S})$? Using standard depth-first visiting techniques, we can proceed as follows:

1. first of all, we compute reachability by means of `ISA` arcs in $\mathscr{S}$; this takes $O(t^2 n)$;

2. then, we compute reachability by means of type walks in $\mathscr{S}$; this takes $O(t^2 n)$;

3. then, we build $G(\mathscr{S})$; since each of the m `WEAK` arcs cannot be replicated more than $2t$ times, this takes $O(n + tm)$;

4. we now compute reachability in $G(\mathscr{S})$ by means either of entirely barred, or of entirely non-barred, paths; since each visit explores $O(tm)$ arcs, this takes $O(n(n + tm))$;

5. at this point, we can already check for cycles;

6. then, we compute for each pair of nodes x, y whether there is a node reachable from x *via* a non-barred path, and from y *via* a barred path; this takes $O(n^3)$;

7. finally, we consider all pairs of arcs with a common source such that the first one is not barred, whereas the second one is, and endowed with different labels (discarding the bars): if their

targets have a common reachable node as in the previous step, we conclude that there is double ownership; since each pair of arcs is examined once, this takes $O\big(tm(n + tm)\big)$.

Note that the procedure above does not list *all* possible double ownership paths (or it would be exponential), but in a schema with double ownership it makes it possible to rebuild at least one pair of such paths. ∎

6.1 An example (continued)

Figure 3 shows the graph $G(\mathscr{S})$ for the example of Figure 1. There are two paths satisfying the
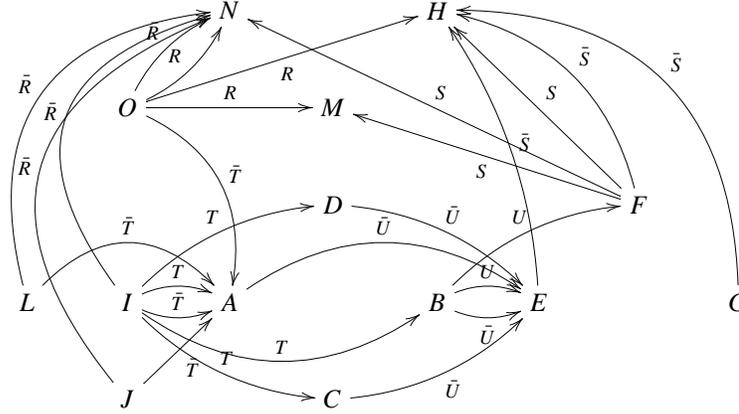

Figure 3: The graph $G(\mathscr{S})$ for the example of Figure 1.

condition stated in the proof of Theorem 3:

$$I \xrightarrow{T} B \xrightarrow{U} F \xrightarrow{S} N \qquad I \xrightarrow{\bar{R}} N.$$

When expanded into the graph having as nodes entity types and as arcs relationship types we get the closed walk

$$I \overset{T}{\Rightarrow} A \leftarrow C \rightarrow B \overset{U}{\Rightarrow} E \rightarrow F \overset{S}{\Rightarrow} H \leftarrow M \rightarrow N \overset{R}{\Leftarrow} O \leftarrow L \leftarrow J \rightarrow I.$$

Following the proof of Theorem 3, we get the two walks

$$J \rightarrow I \overset{T}{\Rightarrow} A \leftarrow C \rightarrow B \overset{U}{\Rightarrow} E \rightarrow F \overset{S}{\Rightarrow} H \leftarrow M$$

and

$$J \rightarrow L \rightarrow O \overset{R}{\Rightarrow} N \leftarrow M$$

which give two paths of $W(\mathscr{S})$ satisfying Definition 5.

16