Proc. 9th Int. Symposium on Computer and Information Science (ISCIS'94), S. Kuru, M.U. Caglayan, E. Gelenbe, H.L. Akin, C. Ersoy (Eds.), Bogazici University Printhouse, Istanbul, pp. 33-40 (1994)

On Formal Semantics of Some Semantic Data Models

Martin Gogolla*
TU Braunschweig, Informatik, Abt. Datenbanken
Postfach 3329, D-38023 Braunschweig, Germany
e-mail: gogolla@idb.cs.tu-bs.de

Abstract

We explain how different semantic data models like the ER model, the ERC approach and the OMT structural modeling method could be transformed into a semantically well-defined extended Entity-Relationship model and calculus.

Key words: Databases, Software Engineering, Semantic Data Models, Semantics of Data Models, Database Design, Entity-Relationship Model, Object Modeling Technique.

1 Introduction

Within the database and information system community it is widely accepted that a data model consists of at least three components, namely (1) a structural, (2) an integrity specification, and (3) a data manipulation component [Cod81]. For instance, for the relational data model [Mai83] these components are constituted by (1) the attribute and relation names, (2) the declaration of key attributes and referential integrity requirements, and (3) queries and operations for inserting, deleting, and updating tuples in relations. But things get more involved if one considers data models whose features go beyond the relational model, especially the numerous so-called semantic data models [HK87, PM88] which have been proposed in the last twenty years. The notion of semantics in this context does not refer to formal mathematical semantics but to the fact that more knowledge of the application domain can be expressed already in the structural component of the data model. This usually means that a sharp distinction between the structural and the integrity specification component is canceled, and some integrity constraints are integrated into the database schema often denoted in a graphical language. But, frequently the corresponding constraints are only explained in natural language and not in an exact formalism.

We precisely define the semantics of different kinds of integrity constraints usually denoted in a graphical way in schemas for semantic data models. The kinds of considered constraints touch complex attributes, cardinality requirements for relationships and

*Work reported here has been partially supported by the CEC under Grant No. 6112 (COMPASS).

multi-valued attributes, weak entity types, optional attributes, and disjoint, hierarchical components. By this we also shortly discuss various graphical formalisms for structural modeling. As typical examples we consider three data models developed in the seventies, eighties, and nineties, respectively: The original Entity-Relationship model (ER) [Che76], the Entity-Relationship-Complex model (ERC) [PRYS89], and the structural description method of the Object Modeling Technique (OMT) [RBP+91]. The way we define the semantics of the models is to translate them into a structurally clear and plain extended Entity-Relationship model but which comes along with a precisely defined and expressive calculus [HG88, GH91, Hoh93, Gog94].

2 The Original Entity-Relationship Model

One of the first semantic data models developed was the Entity-Relationship model [Che76]. The following diagram is part of a diagram presented in the original article by Chen. It graphically declares a (1:N) relationship between departments and employees (an employee works in exactly one department) and a weak entity type (dependents can only exist if they are related to an employee).

The graphical representation of this diagram in the extended Entity-Relationship model is a very plain one, because it only depicts plain entity and relationship types.

In [Gog89, Gog94] we have shown how to make the notion of an Entity-Relationship schema precise. One can think of it as a signature describing a class of algebras corresponding to the possible database states. In our example this would look as follows.

SORTS
  DEPARTMENT, EMPLOYEE, DEPT-EMP, ...
OPERATIONS
  DEPARTMENT : DEPT-EMP $\rightarrow$ DEPARTMENT,
  EMPLOYEE : DEPT-EMP $\rightarrow$ EMPLOYEE,
  $\mathrm{MAKE}_{\mathrm{DEPT\text{-}EMP}}$: DEPARTMENT $\times$ EMPLOYEE $\rightarrow$ DEPT-EMP, ...

The interpretation of the entity sorts is restricted to finite sets because only a finite number of entities can be stored in a database, and the interpretation of relationship sorts is restricted to a subset of the cartesian product of the interpretation of the participating entity sorts.

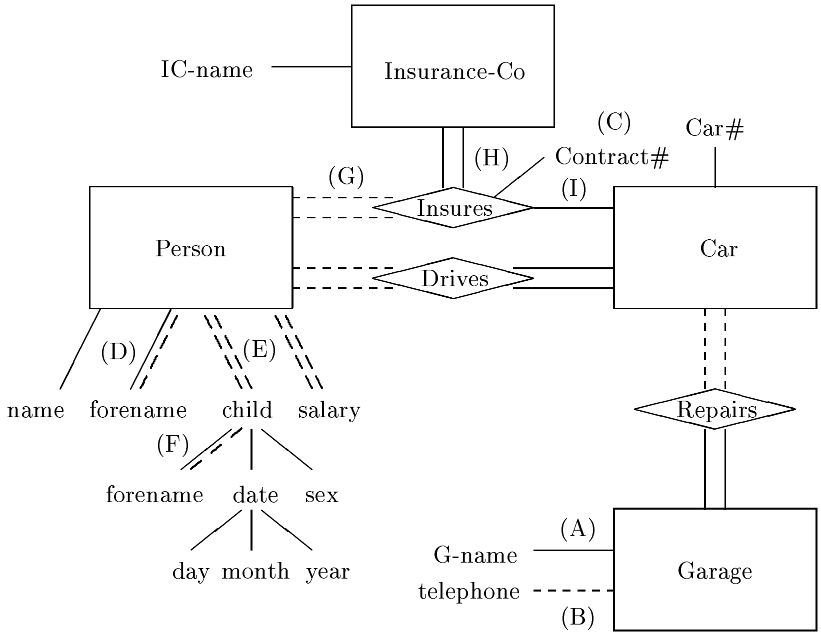
Now in order to achieve the same intended instances as for the original Entity-Relationship diagram we have to restrict the possible interpretations by appropriate intergrity constraints formulated in the extended Entity-Relationship calculus (The letters (A) - (C) refer to the respective parts of the original diagram).

(A) $\forall$ e:EMPLOYEE (CNT -[d | d:DEPARTMENT $\wedge$ DEPT-EMP(d,e)]-) = 1
(B) $\forall$ d:DEPARTMENT (CNT -[e | e:EMPLOYEE $\wedge$ DEPT-EMP(d,e)]-) $\geq$ 0
(C) $\forall$ d:DEPENDENT $\exists$ e:EMPLOYEE (EMP-DEP(e,d))

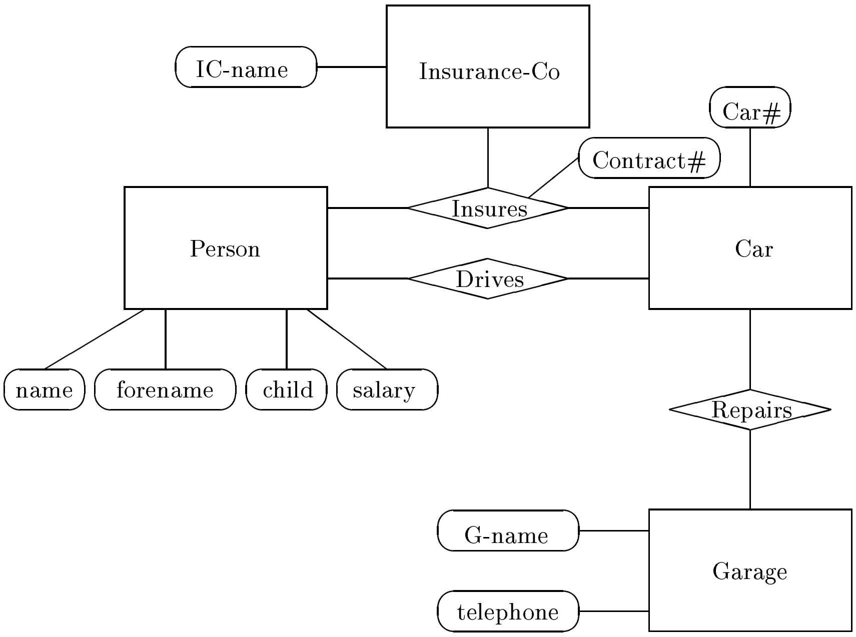
The query term -[d | d:DEPARTMENT $\wedge$ DEPT-EMP(d,e)]- is of sort BAG(DEPARTMENT). The first 'd' is the result term, 'd:DEPARTMENT' is a declaration for the variable 'd', and 'DEPT-EMP(d,e)' is a restricting formula. The valuation of the term yields all DEPARTMENT entities which are related to EMPLOYEE entity e. Furthermore, the extended Entity-Relationship calculus provides operations working on such multi-valued domains. For instance, 'CNT' counts the number of elements in a set, bag, or list. Thus the complete formula states that there is exactly one DEPARTMENT entity related to each EMPLOYEE entity by relationship DEPT-EMP. The constraint (A) refers to the '1'-part in the (1:N) relationship, whereas the constraint (B) is responsible for the 'N'-part. We have given the second formula for symmetry reasons only; it does not restrict the possible interpretations, because the formula is fulfilled by all instances of the above diagram. The constraint (C) realizes the requirement for the weak entity type DEPENDENT.

3 The Entity-Relationship-Complex Approach

The Entity-Relationship-Complex (ERC) model [PRYS89] allows the graphical description of complex attributes and of cardinality constraints for relationships and attributes. We here present part of an original Entity-Relationship diagram taken from [PRYS89] where persons, insurance companies, cars and garages are the entity types of interest.



Above, cardinality constraints are visualized as follows [PRYS89]: One single continuous line denotes a (1:1) cardinality (mandatory mono-valued), one single dashed line a (0:1) cardinality (optional mono-valued), a double continuous line a (N:M) cardinality (mandatory multi-valued), a double dashed line a (0:N) cardinality (optional multi-valued), and one continuous together with one dashed line a (1:N) cardinality for attributes. The corresponding extended Entity-Relationship diagram is given as follows.



In order to model the same schema as in the ERC case we have to introduce the new data types which appear in this application domain. We give specifications for the types dateDescriptor and childDescriptor which in turn employs the type dateDescriptor for its definition. These specifications use the pre-defined type constructors RECORD and LIST; additionally the constructor SET and BAG are available in the extended Entity-Relationship approach. In [Gog94] it is pointed out how equivalent equational specifications [EM85, EGL89, EM90, Wir90] can be given for these type constructors and their corresponding operations used in the extended Entity-Relationship calculus.

DATA TYPES
  dateDescriptor =
    RECORD day:1..31; month:1..12; year:0..3000 END
  childDescriptor =
    RECORD forename:LIST(string); date:dateDescriptor; sex:{female,male} END

On this basis we can now determine the exact type of the attributes which are left unspecified in the ERC diagram. Again the pre-defined type constructors can be used in the declaration of the attributes.

ATTRIBUTES
  IC-name:string, name:string, forename:LIST(string),
  child:LIST(childDescriptor), salary:LIST(integer),
  telephone:string, G-name:string, Contract#:string, Car#:string

An obvious advantage of separating the declaration of data types and attributes is that the data types can be used more than once in our schema whereas in the original ERC approach the structure defining the data types has to be repeated (as this is done in the original article [PRYS89]). After having declared the attributes we can now proceed to restrict the possible values of the attributes to the intended ones. Therefore we give intergrity constraints formulated in the extended Entity-Relationship calculus.

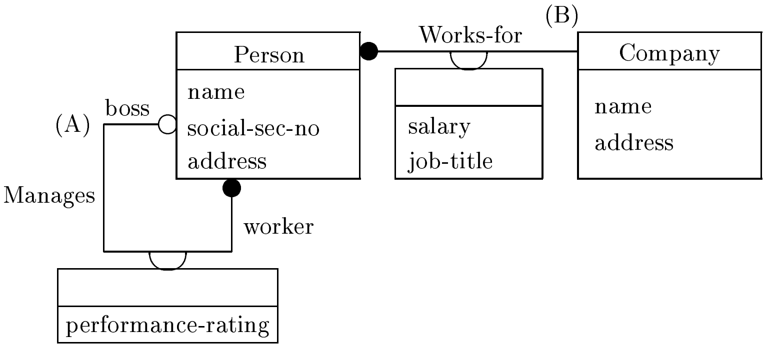
(A) $\forall$ g:Garage DEF(G-name(g))
(B) By default, attributes may be undefined.
(C) $\forall$ ins:Insures DEF(Contract#(ins))
(D) $\forall$ p:Person CNT(forename(p)) ≥ 1
(E) By default, lists may have length zero.
(F) $\forall$ p:Person $\forall$ i:IND(child(p)) CNT(forename(SEL(child(p),i))) ≥ 1
(G) Optional, multi-valued relationship participation is the default.
(H) $\forall$ ic:Insurance-Co (CNT -[ins | ins:Insures $\land$ ins.Insurance-Co=ic]-) ≥ 1
(I) $\forall$ c:Car (CNT -[ins | ins:Insures $\land$ ins.Car=c]-) $= 1$

Some remarks on these constraints are necessary. Consider formula (A). In the extended Entity-Relationship model all attributes are optional and therefore may yield the special value undefined ($\perp$). If one wants to exclude this one simply requires that the attribute has to yield a defined value, for instance by stating DEF(G-name(g)). The attribute forename in constraint (D) is regarded as a function from Person to LIST(string). As mentioned above multi-valued terms can be argument to certain operations like CNT for counting the elements in a list. In formula (F) the declaration i:IND(child(p)) of variable i is a declaration with parameter p. It employs the overloaded function IND which in this case has the signature IND : LIST(childDescriptor) $\rightarrow$ SET(integer). It yields the set of indices of a list, e.g., IND(child(p)) = $\{1,2,3\}$ if the list child(p) has length 3. The function SEL selects the i-th element of the list child(p).

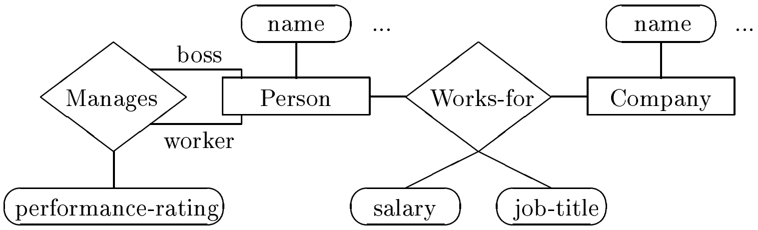
Analogous constraints like (A) have to be specified for all mandatory attributes like IC-name, etc. Analogous constraints like (H) and (I) have to be given for the relationships Drives and Repairs.

4 The OMT Structural Modeling Method

A rather recent and quite popular approach to software development is the object modelling technique (OMT) [RBP$^+$91]. As part of its method it includes an object model notation which is in its basic concepts very close to the Entity-Relationship approach. We cannot give a complete overview of the graphical features offered by OMT for structural modeling but point out some peculiarities by means of two examples. The following OMT diagram introduces classes (entity types) for persons and companies and associations (relationship types) between them. The multiplicity of associations is denoted in OMT by (1) a plain line (exactly one; right side of Works-for), (2) a line with a dot at the end (many; left side of Works-for) or (3) a line with a circle at the end (optional; boss part of Manages).



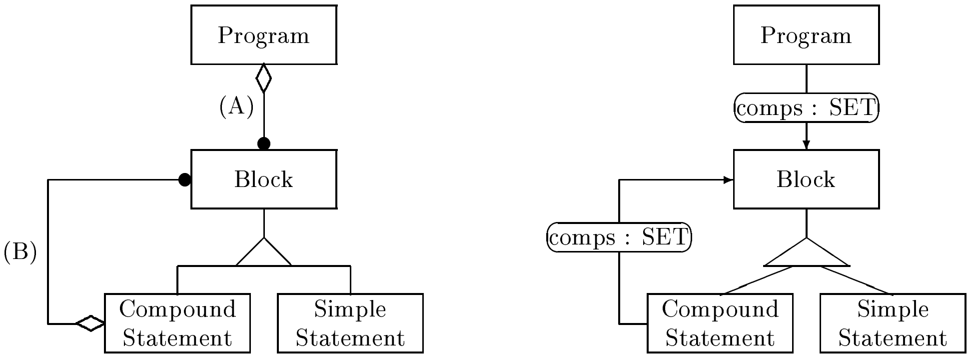
Again, if we translate the above diagram into the extended Entity-Relationship approach we achieve a rather plain diagram; in this case we only have entity and relationship types and attributes. The diagram does not give all attributes of the OMT schema but mentions only the most important ones.



As for the Entity-Relationship and Entity-Relationship-Complex schemas the graphical notation is made precise by specifying corresponding constraints in the extended Entity-Relationship calculus.

(A) $\forall$ p:Person $0 \leq$ (CNT -[m | m:Manages $\wedge$ m.worker=p]-) ≤ 1
(B) $\forall$ p:Person (CNT -[wf | wf:Works-for $\wedge$ wf.Person=p]-) = 1

Another language feature in OMT allows to specify aggregation. The OMT diagram below states the following: Programs may consist of many blocks; blocks are either compound or simple statements; compound statements may consist of many blocks. The corresponding extended Entity-Relationship diagram is pictured next to the OMT schema. It employs a feature called type construction to express generalization and specialization abstractions. This feature is denoted by a triangle where the entity types on the base line of the triangle are the input types (in this case CompoundStatement and SimpleStatement) and the entity types on the other side of the triangle are the output types (in this case Block).



In order to model the intended situation in the extended Entity-Relationship approach again constraints can be specified.

(A) $\forall$ b:Block $\forall$ p_1,p_2:Program
(b IN comps(p_1) $\wedge$ b IN comps(p_2)) $\Rightarrow$ p_1=p_2

(B) is-part-of(b_1:Block,b_2:Block) :$\Leftrightarrow$ b_1 IN comps(CompoundStatement(b_2))
$\forall$ b:Block $\neg$ is-part-of$^+$(b,b)
$\forall$ b:Block $\forall$ b_1,b_2:Block (is-part-of(b,b_1) $\wedge$ is-part-of(b,b_2)) $\Rightarrow$ b_1=b_2

Let us comment on the requirements (B): Type constructions also have their impact on the extended Entity-Relationship calculus. Here we can regard CompoundStatement as a function from Block to the entity type CompoundStatement. For a given block it either returns the compound statement the block corresponds to or undefined if the block in question is a simple statement. Furthermore, our calculus allows to express the transitive closure of a binary relation between entity types. The details of the formulation can be found in [Gog94]. We here use the abbreviation that r^+ denotes the transitive closure if the relation r is given. In our example the (derived) relationship is-part-of has been declared before.

These formulas require on the one hand that aggregation is an exclusive relationship (for instance, a block is a component of at most one program; this is a disjointness property) and on the other hand that aggregation is cycle free (no block is a component of itself; thus the blocks form a hierarchy). These requirements constitute disjoint, hierarchical components and are expressed in the original OMT approach only in natural language but not in a formal setting.

5 Conclusion

We have pointed out how different popular graphical formalisms for database and information system design can be made precise by translating them into extended Entity-Relationship schemas as proposed in [HG88, GH91, Hoh93, Gog94]. Additionally, restricting constraints for database states were specified in order to achieve the intended semantics described informally only in the considered approaches [Che76, PRYS89, RBP+91]. These three models have been chosen as rather typical candidates for structural modeling, and the proposed way to define semantics carries over to other structural modeling approaches. The constraints were formulated in an extended Entity-Relationship calculus provided with a mathematical semantics. We hope our considerations have made a contribution towards the integration of frequently used semi-formal methods and mathematically precisely defined approaches.

Acknowledgements

Discussions with TU Braunschweig colleagues in the KORSO project, namely Stefan Conrad, Grit Denker, Rudolf Herzig, and Nikolaos Vlachantonis, led to improvements. Especially Rudolf Herzig provided constructive insights into peculiarities of some semantic data models.

References

[Che76] P.P. Chen. The Entity-Relationship Model – Towards a Unified View of Data. *ACM Trans. on Database Systems*, 1(1):9–36, 1976.

[Cod81] E.F. Codd. Data Models in Database Management. *ACM SIGMOD Record*, 11(2):112–114, 1981.

[EGL89] H.-D. Ehrich, M. Gogolla, and U.W. Lipeck. *Algebraische Spezifikation abstrakter Datentypen – Eine Einführung in die Theorie*. Teubner, Stuttgart, 1989.

[EM85] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 1: Equations and Initial Semantics*. Springer, Berlin, 1985.

[EM90] H. Ehrig and B. Mahr. *Fundamentals of Algebraic Specification 2: Modules and Constraints*. Springer, Berlin, 1990.

[GH91] M. Gogolla and U. Hohenstein. Towards a Semantic View of an Extended Entity-Relationship Model. *ACM Trans. on Database Systems*, 16(3):369–416, 1991.

[Gog89] M. Gogolla. Algebraization and Integrity Constraints for an Extended Entity-Relationship Approach. In J. Diaz and F. Orejas, editors, *Proc. Theory and Practice of Software Development (TAPSOFT'89)*, pages 259–274. Springer, Berlin, LNCS 351, 1989.

[Gog94] M. Gogolla. *An Extended Entity-Relationship Model – Fundamentals and Pragmatics*. Springer, Berlin, LNCS 767, 1994.

[HG88] U. Hohenstein and M. Gogolla. A Calculus for an Extended Entity-Relationship Model Incorporating Arbitrary Data Operations and Aggregate Functions. In C. Batini, editor, *Proc. 7th Int. Conf. on the Entity-Relationship Approach*, pages 129–148. North-Holland, Amsterdam, 1988.

[HK87] R. Hull and R. King. Semantic Database Modelling: Survey, Applications, and Research Issues. *ACM Computing Surveys*, 19(3):201–260, 1987.

[Hoh93] U. Hohenstein. *Formale Semantik eines erweiterten Entity-Relationship-Modells*. Teubner, Stuttgart, 1993.

[Mai83] D. Maier. *The Theory of Databases*. Computer Science Press, Rockville (MD), 1983.

[PM88] J. Peckham and F. Maryanski. Semantic Data Models. *ACM Computing Surveys*, 20(3):153–189, 1988.

[PRYS89] C. Parent, H. Rolin, K. Yetongnon, and S. Spaccapietra. An ER Calculus for the Entity-Relationship Complex Model. In F.H. Lochovski, editor, *Proc. 8th Int. Conf. on Entity-Relationship Approach*, pages 75–98. ER Institute, Pittsburgh (CA), 1989.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modeling and Design*. Prentice-Hall, Englewood Cliffs (NJ), 1991.

[Wir90] M. Wirsing. Algebraic Specification. In J. Van Leeuwen, editor, *Handbook of Theoretical Computer Science, Vol. B*, pages 677–788. North-Holland, Amsterdam, 1990.