In H.-J. Kreowski, editor, Recent Trends in Data Type Specification, Informatik-Fachberichte 116, Berlin, 1985. Springer-Verlag

ALGEBRAIC (?) SPECIFICATION OF CONCEPTUAL DATABASE SCHEMATA

(Extended Abstract)

H.-D. Ehrich
Inst.f.Informatik, PF 3329, D-3300 Braunschweig

1. Introduction

This contribution does not present any technical results. Rather, it is an attempt to broaden the view towards specification problems and algebraic methods.

Algebraic specification of abstract data types is, by now, a well developed discipline. There are two comprehensive textbooks available [KI83, EM85], and others are likely to follow. It is interesting to note that both textbooks call their subject just "algebraic specification", obviously assuming that there can be no doubt about <u>what</u> is to be specified. Indeed, up to now, interest exclusively concentrated on the specification of abstract data types, modelled as classes of many-sorted algebras, specified by signatures, axioms in some predicate calculus, and algebraic concepts like initiality, terminality, etc. to further constrain the class of models.

There are, however, other specification problems in software development. We claim that, for instance, the problem of specifying a conceptional database schema is different from that of specifying an abstract data type. We will also argue that algebraic concepts and techniques might prove usefull here, too. We have not in mind to model a conceptual database schema as an algebraic data type. This is possible in principle [EKW78, DMW82], but it seems to be inappropiate to do so. At least, this approach is incompatible with all data modelling approaches taken in the database field. The deeper reason is that algebraic data type specification is based on purely applicative concepts providing no means of directly modelling storage concepts like variables, etc., whereas concepts of storage are very central for databases.

2. Database Specification in Layers

Data modelling approaches (see [TL82] for a recent textbook) have a notion of "entity" or "object" that is not just a data element. While a data sort like <u>bool</u>, <u>nat</u>, <u>int</u>, <u>text</u>, etc. is naturally interpreted by a fixed set of data elements (taking, of course, the operations defined on them into account), an object sort like PERSON, PROJECT, etc. essentially denotes a time-varying collection of objects of that sort, and it is the objective of a database to store information about the objects "currently in" the database, like salary or age of a person, manager or set of employees in a project, etc.

Modelling this situation by means of abstract data types would require to add "database states" as an extra data sort, with operations like insert, delete, retrieve operating on it. This approach of viewing a database state as an atomic data element in an abstract data type looks strange to most database people. They are used to view a database state as a large, sometimes sophisticated structure. This view is nicely captured in a suggestion made in [GMS83] to consider database states as "possible worlds" in a modal system of algebras. We have taken up this approach [ELG84, LEG85], concentrating on dynamic database constraints expressed in a temporal logic. Updates and views are treated in [KMS85] using this approach. Temporal logic has been used before in database specification [Se80, CCF82, CF82, Ku84], following more or less the abstract data type point of view.

One of the main features of our modal approach is that the specification of data and objects in a conceptual database schema can be separated into two layers:

<u>data layer</u>: here we imagine a set of abstract data types like <u>bool</u>, <u>int</u>, etc., specified algebraically by one of the approaches advocated in the literature. The abstract data types have a fixed interpretation not changing in time. Thus, there is only one "fixed world of data elements" (that is part of every possible world constituting a database state).

object layer: here we have the objects, attributes and relationships that determine the structure of database states. Database states change in time, and it is the objective of the object specification to characterize the permissible states as "possible worlds" in a modal logic, together with the permissible state changes.

There is another layer on top of the object layer, specifying the state-dependent operations or "transactions". These consist of aggregate functions like average or minimal salary of persons in the database state, and state changing operations like insert, delete and update. We will not consider transactions here, but concentrate on data and object layers.

Attributes and Relationships are uniformly modelled in the functional approach to data modelling [BF79, Sh81]. We adopt this approach, since it opens the way for applying algebraic concepts to the object level.

3. Object specifications

In order to make the above ideas precise, let $\Sigma_D=(S_D,\Omega_D)$ be a data signature, where S_D is a set of data sorts, and Ω_D is an $S_D^*\times S_D$-indexed family of data operators. We assume a fixed data algebra A that serves as the standard interpretation of Σ_D in each database state, thus building an invariant part of the database.

An object signature Σ_O is a signature extension of the data signature,

$$\Sigma_O = \Sigma_D + (S_O,\Omega_O).$$

S_O is a set of object sorts, and Ω_O is an $S^*\times S$-indexed set of object functions, where $S=S_D\cup S_O$. A (database) state is an interpretation of Σ_O, i.e. a mapping σ associating a set $\sigma(s)$ with each sort $s\in S$ and an operation $\sigma(f)$ with each operator $f\in\Omega=\Omega_D\cup\Omega_O$, such that $\sigma(f) : \sigma(s_1)\times \ldots \times\sigma(s_n) \rightarrow \sigma(s_0)$ for each $f : s_1\times \ldots s_n \rightarrow s_0 \in\Omega$ and $\sigma|\Sigma_D=A$. In this sense, the object layer is "built upon" the data layer. For $s\in S_O$, we call $\sigma(s)$ the set of "actual" objects, and for $f\in\Omega_O$. we call $\sigma(f)$ the "actual" object function in state σ. In order to constrain the class of possible states, we assume a set $\pi(s)$ of "possible" objects for each $s \in S_O$, and $\sigma(s) \subseteq \pi(s)$ for each $s \in S_O$. We call $\pi(s)$ the <u>universe</u> of object sort s.

Thus, database states are algebras. It is also possible to extend the universe π by setting $\pi(\Sigma_D)=\sigma(\Sigma_D)=A$ and also adding some meaningful object functions to it, so that universes are algebras, too. We will elaborate on this point in a forthcoming paper.

Attributes are object functions of the special form $a:s\to t$, where $s \in S_O$ and $t \in S_D$. An example is age : PERSON -> <u>nat</u>. "Object constants" $v:\to t$ of a data sort $t \in S_D$ can serve as simple variables of sort t.

The <u>specification</u> of an object schema has to characterize the permissible states and the permissible state changes. For the data layer, this amounts to specifying the data algebra A, and this is the classical specification problem for abstract data types. Thus, we assume an algebraic specification $D=(\Sigma_D, E_D)$, where E_D is a set of axioms (e.q. equations, initiality constraints, etc.), such that the semantics of D is A (up to isomorphism).

For the object layer, we have to extend the data specification D by Σ_O-axioms C_O,

$$O = D + (\Sigma_O, C_O) \ .$$

We call the axioms in C_O (object or database) <u>constraints</u>. C_O will contain <u>static</u> and <u>dynamic</u> constraints. Static constraints characterize the permissible states, and that can be coped with by giving a loose specification using some (first order) predicate calculus. In order to give dynamic constraints characterizing permissible state changes, however, we have to leave the grounds of conventional algebraic specification and extend our logic by temporal aspects.

Temporal Logic was originated in [RU71] and has been applied to the specification and verification of programs in [MP79]. Application of Temporal Logic to database specification was originated in [Se80] and developed in [CCF82, CF82, Ku84]. Our simplified version of Temporal Logic uses two temporal quantifiers, always.. until.. and sometimes.. before.., as well as quantification $\forall$, $\exists$ over actual objects (states) and $\underline{\forall}$, $\underline{\exists}$ over possible objects (universes). Models of temporal formulas are sequences

$$\underline{\sigma} = (\sigma_0, \sigma_1, \sigma_2, \ldots\ldots)$$

of structures (states, in our case), thus modelling the development of a database in time.

Now, the objective of an object specification $O=D+(\Sigma_O,C_O)$ is to characterize a class of permissible state sequences $\underline{\sigma}$:

$$[\![O]\!] = \{ \underline{\sigma} \mid \underline{\sigma} \models C_O \}$$

The details of our Temporal Logic and its semantics can be found in [LEG85], together with results how to enforce a special class of temporal database constraints operationally, i.e. how to recognize as early as possible whether a given state sequence $\underline{\sigma}$ is permissible, inspecting its prefixes $[\sigma_0,\ldots\sigma_n]$, $n\geq 0$.

4. Conclusions

Many problems are open in the field of conceptual database specification. Of particular interest here is that it would be nice to have an algebraic semantics for an object specification, associating with it a fixed algebra (up to isomorphism) as a standard universe, and characterizing the class of algebras that can represent database states within this standard universe. The class of permissible state sequences is then well defined by the semantics of the Temporal Logic. Having a theoretically well founded and usefull specification methodology for data and objects, however, is only a first step towards the ultimate goal of establishing software correctness and reliability. The logical next step is to elaborate the transaction layer and see how the database constraints can be taken into consideration and, hopefully, be enforced there. Here again, we have another specification problem that might require different concepts and techniques.

References

BF79 Buneman, P. / Frankel, R.E. : FQL - A Functional Query Language. Proc. ACM SIGMOD Int. Conf. on Management of Data 1979, 52-58

CCF82 Castilho, J.M.V. / Casanova, M.A. / Furtado, A.L. : A Temporal Framework for Database Specifications. Proc. 8th Int. Conf. on Very Large Data Bases, Mexico 1982

CF82 Casanova, M.A. / Furtado, A.L. : A Family of Temporal Languages for the Description of Transition Constraints. Proc. Workshop on Logical Bases for Data Bases, Toulouse 1982

DMW82 Dosch, W. / Mascari, G. / Wirsing, M. : On the Algebraic Specification of Databases. Proc. 8th Int. Conf. on Very Large Data Bases, Mexico City 1982

EKW78 Ehrig, H. / Kreowski, H.J. / Weber, H. : Algebraic Specification Schemes for Database Systems. Proc. 4th Int. Conf. on Very Large Data Bases, Berlin 1978

ELG84 Ehrich, H.-D. / Lipeck, U.W. / Gogolla, M. : Specification, Semantics, and Enforcement of Dynamic Database Constraints. Proc. 10th Int. Conf. on Very Large Data Bases, Singapore 1984

EM85 Ehrig, M. / Mahr, B. : Fundamentals of Algebraic Specification 1. Springer-Verlag, Berlin 1985

GMS83 Golshani, F. / Maibaum, T.S.E. / Sadler, M.R. : A Modal System of Algebras for Database Specification and Query /Update Language Support. Proc. 9th Int. Conf. on Very Large Data Bases, Florence 1983

Kl83 Klaeren, H. : Algebraische Spezifikation. Springer-Verlag, Berlin 1983

KMS85 Khosla,S. / Maibaum, T.S.E. / Sadler, M. : Database Specification. Proc. IFIP Working Conf. on Database Semantics, R. Meersman / T.B. Steel (eds.), North Holland, Amsterdam 1985

Ku84 Kung, C.H. A Temporal Framework for Database Specification and Verification. Proc. 10th Int. Conf. on Very Large Data Bases, Singapore 1984

LEG85 Lipeck, U.W. / Ehrich, H.-D. / Gogolla, M. :Specifying Admissibility of Dynamic Database Behaviours Using Temporal Logic. Proc. IFIP Working Conf. on Theoretical and Formal Aspects of Information Systems, A.Sernadas et al. (eds.), North Holland, Amsterdam 1985

MP79 Manna, Z. / Pnueli, A. : A Modal Logic of Programs. Proc. Conf. Automata, Languages and Programming. LNCS 19, Springer-Verlag, Berlin 1979

RU71 Rescher, N. / Urquhart, A. : Temporal Logic. Springer-Verlag, Berlin 1971

Se80 Sernadas, A. : Temporal aspects of Logical Procedure Definition. Inform. Sys. 5(1980), 167-187

Sh81 Shipman, D,W. : The Functional Data Model and the Data Language DAPLEX. ACM TODS 6(1981), 140-173

TL82 Tsichritzis, D.C. / Lochovsky, F.H. : Data Models. Prentice-Hall, Englewood Cliffs 1982