

# Algebraic Graph-Based Approach to Schema Integration

Ilya Beylin

Chalmers University  
S-41296 Göteborg, Sweden  
ilya@cs.chalmers.se

Boris Cadish\*

Frame Inform Systems  
Elizabetes 23, LV-1234 Riga, Latvia  
diskin@frame.riga.lv

Zinovy Diskin\*

## Abstract

The language of sketches is as concise and powerful as higher order logic and as handy as conventional ER-diagrams. We discuss its benefits in respect to problems of schema integration.

It is evident that a chief property of the next generation information systems is their organization on cooperative principles. Typically, applications involved in a cooperative system have to access data stored in several separate databases. These databases can be differently organized, semantically overlapping and even inconsistent, so that an application programmer encounters severe problems, moreover, in each *multibase* case they have to be solved specifically.

In fact, as it was noted in [13], the situation is similar to the *multifile* situation before invention of DBs, and the analogous solution can be suggested: in order to provide applications with a single integral view of data, local DBs should be integrated into a united *distributed DB*. Under the latter we mean a database system which has a schema like an ordinary DB but its extension is *virtual*, that is not stored immediately but derived from the extensions of local databases.

It is characteristic to multibase systems that local DBs can be autonomous, oriented on applications of different level, and independently developed. This results in the essential heterogeneity of whole systems. Thus, *heterogeneous multibase integration* turns out to be a fundamental issue of cooperative system operation. In particular, in the context of federated database environment, integration is a function regularly performed at different levels and by different services depending on the organization of the federal environment.

The database integration gives rise to several problems:

- *Heterogeneous framework*: If data are stored in different systems, the database schemas can be defined in different languages; we need a way to compare such definitions.
- *Structural conflicts*: Semantically the same data can be presented by different constructs (function or relation; attribute or entity). Note, these are conflicts between presentations, not between data.

---

\*Supported by Grant 93.315 from the Latvian Council of Science

ing virtual integration, that is, building the integrated schema whose extent can be computed. This language should be universal in capturing the diversity of all possible situations of correspondence between local DBs.

These problems are known, and various approaches, techniques, and tools have been proposed (see the surveys [3, 16, 14]). It is worth noting that majority of the existing approaches aim at integration within one or another data model, so the heterogeneous integration remains outside their scope.

Even structural conflicts within a sufficiently rich (in its collection of modeling constructs) data model is still a challenge. Indeed, the taxonomy of conflicts between schemas proposed by Spaccapietra *et al* ([18, 17]) where *semantic*, *descriptive*, *structural* and *heterogeneity* conflicts are distinguished, appears to be an informal description, encompassing the diversity of intuitions behind conflicts, but not a precise formal specification capable to support automated integration.

In contrast, we propose an approach in which

- input schemas can be defined in arbitrary language provided it possesses formal semantics;
- all kinds of conflicts are operated in a unified, concise and fully formalized way;
- all cases of data correspondence are reduced to equalities between graph elements.

We call the approach *AGO* since its general characteristic consists in the Algebraic-Graph-Oriented nature of all its constructs and procedures. This paper outlines the main concepts of our approach relevant to integration of schemas. Reader should consult [8, 6, 7] for more detail and examples. Problems of data integration (that is, computing queries on the virtual schema) are touched upon in [5].

**Sketches.** The core of AGO is a family of graph-based specification languages in which specifications are directed multigraphs endowed with a labeling formalism of a special kind. Certainly, conventional semantic schemas are also graphs endowed with special labeling, however, the question is which graphs should be used and what to be labeled. A distinctive property of AGO-specifications providing essentially all their advantages consists in a special discipline of labeling: it is rigorously formalized, consistent and mathematically justified. We call AGO-specifications *sketches* following the terminology tradition of *categorical logic* (see, *eg*, [20, 2]) which is a graph-based higher-order logic, as opposed to string-based ones, and can be seen as a graph-based counterpart of the ordinary many-sorted relational logic. The essence of the sketch approach to data modeling consists in considering all object classes homogeneous while all type information about objects is concentrated in arrow (in particular, attribute) structures connected with classes.<sup>1</sup> In this respect

---

<sup>1</sup>Removing type information from nodes to arrow diagrams already prevents appearance of some kinds of structural conflicts, *eg*, like those that arise in ER-modeling when a class is represented by an entity node in one view schema and by a relationship node—in another.

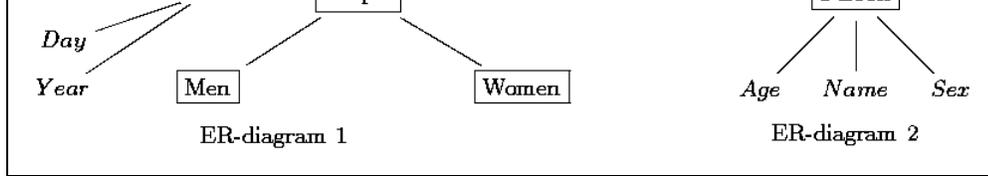


Figure 1: Schemas to be integrated

sketches are very close to the well known family of Functional Data Models ([15, 1, 11, 10]) but are distinguished by a special approach to labeling.

Formally, sketches are graphical constructs consisting of three kinds of items: (i) nodes, to be interpreted as sets, (ii) arrows, to be interpreted as functions between corresponding sets, and (iii) diagram markers, to be interpreted as constraints imposed on diagrams labeled by these markers (we call a diagram any non-empty part of the underlying graph, in contrast to the ER tradition, where diagrams are the complete specifications.) Correspondingly, structural operations on schemas are reduced to operations on diagrams.

An advantage of the sketch specification language is that it is provably universal in the sense that any formal data specification can be replaced by an equivalent sketch of some appropriate type. The proof follows roughly from the following known facts (see, e.g., [12, 2]):

1. as adopted in modern mathematics, any formal specification can be expressed in some version of set theory, higher-order logic, or type theory; all these theories are interpretable as toposes.
2. toposes can be specified by sketches.
3. any topos can be presented as a (kind of nested relational) algebra over the corresponding graph.

The conclusion is that the full power of higher-order logic, including its algebraizability, can be simulated by sketches; or, in other words, everything that can be specified formally can be specified by sketches as well.

**Example.** Figures 1 and 2 show two sample ER-diagrams and their sketch counterparts. The ER-diagrams here are rather informal, some constraints are implicit (*Men* and *Women* are not just subclasses of *People*, but form a partition).

In the sketches the principal structure lies on arrows, whereas constraints are expressed by few markers: double arcs on divergent arrows denote separating family of functions, that is relation, and double arcs on convergent arrows—disjoint sum. Double arrows stand for inclusion functions. The ovals with domain identifiers in it are also markers: the correspondent nodes have predefined semantics, common for all schemas.

Suppose these schemas describe the same fragment of the real world, that is *People* and *Person* refer to the same set, and the *name* arrows—the same function. Then in integrated scheme they should be glued together. However, *bdate* and *age*

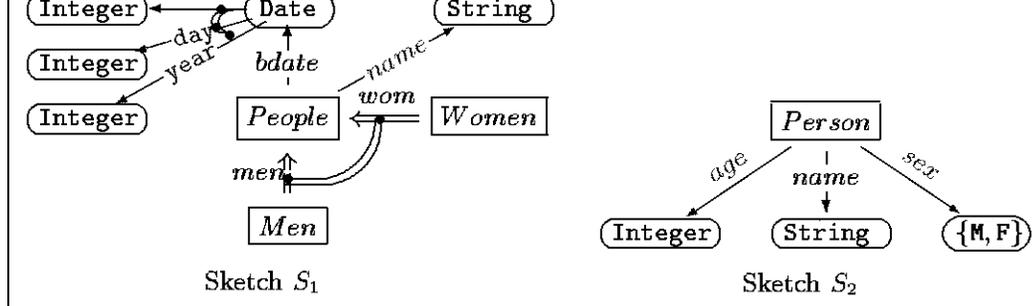


Figure 2: The schemas transformed into sketches

are related indirectly, by an arithmetic expression, and the correspondence between the attribute *sex* and partition *Men/Women* requires whole queries to express elements of one schema in terms of the other.

Still, the sketch language has enough power to express such correspondences. As it was said, any formal specification can be expressed by sketches of appropriate type. Such universal sketch type would be too heavy to use it directly (since it involves entire function domains like  $\text{Real} \rightarrow \text{Real}$ ), but we can approximate it by smaller types that contain just practically used markers (as listed in [6]).

In our example we use the commutativity marker  $\langle \Rightarrow \rangle$  to express

$$x.age = 1995 - x.bdate.year,$$

and the coimage  $\langle \text{CoIm} \rangle$  to get all *Persons* whose *sex* is "M". We also make use of some domain-specific functions (arithmetic for *Integer* and element inclusions for the concrete set  $\{M, F\}$ ).

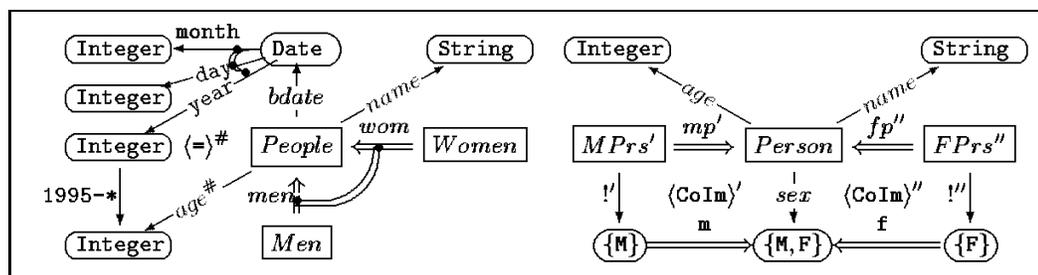


Figure 3: The extended sketches

**Sketch operations vs conflicts.** For AGO, all kinds of conflicts distinguished by Spaccapietra *et al* are special cases of the well known distinction between basic and derived information: data considered as basic in one view (and actually stored in a database) can be safely considered as derived (from another collection of basic data) in another view (and thence actually must be computed if requested). So, the essence of resolving such conflicts consists in search of operations (queries) transforming

constructs denoting derived information *so that the correspondence assertions can be stated as equations.*<sup>2</sup> So, the integration procedure presupposes that a fixed list of sketch-extending operations is predefined, and this is also a constituting component of AGO.

For database integration, each of the sketch-extending operations must be coupled with a procedure computing extent of derived items produced by the operation. Indeed, from a computational view point the arity scheme of an operation is nothing but a graph-based specification of the corresponding query, and the procedure assigned to the operation to compute the answer to this query.

Importantly, mappings from initial view schema into extended ones are always maintained so that input schemas are not lost.

**Correspondence information via equations.** In AGO, view correspondence information, including interschema data, is specified in a fully formal language, moreover, the language is essentially equational.

Let us return to the example. The sketches on figure 3 can be integrated directly, as soon as we list pairs of equal items:

$$\begin{aligned} S_1.People &= S_2.Person, & S_1.age^\# &= S_2.age, & S_1.name &= S_2.name, \\ S_1.Men &= S_2.MPrs', & S_1.men &= S_2.mp', \\ S_1.Women &= S_2.FPrs'', & S_1.wom &= S_2.fp'' \end{aligned}$$

and the integration algorithm gives us the full sketch (Figure 4) together with mappings of local sketches to the integrated one.

In general, specification of the correspondence information involves an additional sketch  $S_{CI}$  with a set of equation  $E_{CI}$ .

Imagine that in the example above the nodes *Person* and *People* are semantically overlapping, but do not coincide. In that case the integrator builds the following *correspondence information specification*:

$$\begin{array}{ccc} M & \xrightarrow{\nu} & P_2 \\ \parallel & \langle \text{PBO} \rangle & \parallel \\ \mu \downarrow & & \downarrow \beta \\ P_1 & \xrightarrow{\alpha} & J \end{array} \quad \boxed{\begin{array}{l} S_1.People = S_{CI}.P_1 \\ S_2.Person = S_{CI}.P_2 \end{array}} \quad E_{CI}$$

$S_{CI}$

Here the symbol  $\langle \text{PBO} \rangle$  is another marker constraining the whole square to be a *meet-join diagram*, that is, to satisfy the condition  $M = P_1 \cap P_2$ ,  $J = P_1 \cup P_2$  with  $\mu, \nu, \alpha, \beta$  being the corresponding inclusion functions. In other words, both nodes

---

<sup>2</sup>We propose a formal definition of operation over sketches so that the process of extending sketches with derived data can be formally specified as a consecutive (free) application of these operations to initial sketches. This is quite similar to building complex algebraic expressions from basic ones by freely applying operation symbols according their arities. However, while the arity of an ordinary operation is a natural number or a list of sorts (domain names), the arity of a sketch operation is a sketch specifying all data involved where a subsketch of input data is designated.

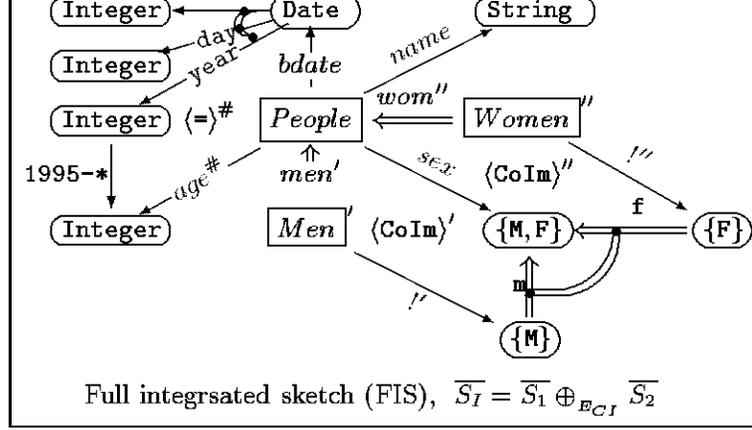


Figure 4: The result of integration

will be kept in the integrated schema, and we make their intersection and union visible as well.

The only two kinds correspondence assertions considered in AGO are 'two nodes are equal' or 'two arrows are equal', and their intended semantics is evident. The original sketches to be integrated are just extended with derived items so that an equational description of the correspondence be possible. If necessary, an additional correspondence sketch is introduced.

It is remarkable that general results of categorical logic guarantee that *any* situation of view integration which can be specified formally, can be also specified by sketches in the above described way.

**Integration and marker conflicts.** From the formal view point, the schema  $S_{CI}$  is quite similar to local schemas so that integration of  $n$  local schemas  $S_1, \dots, S_n$  turns into disjoint merge  $n + 1$  schemas,  $S_1, \dots, S_n, S_{CI}$ , and factorizing the result by the congruence generated by  $E_{CI}$  (ie, gluing together certain items of the merge according to the  $E_{CI}$ -equations).

Quasi-formally, the result of the above-mentioned manipulations can be described as follows:

$$\overline{S_I} = \overline{S_1} \oplus \dots \oplus \overline{S_n} \oplus \overline{S_{CI}} / E_{CI}$$

where  $S_i$  denote sketches corresponding to schemas,  $\overline{S_i}$  denote results of extending them with derived items, the symbol '/' denotes binary operation of factorization the left term by the congruence (generated by the term) on the right.<sup>3</sup>

Actually this procedure is performed in two steps. The first one consists in disjoint merging  $(n + 1)$  graphs underlying local sketches and then factorizing the merge according to  $E_{CI}$ -equations (this step can be mechanized). The second step consists in converting the integrated graph into a sketch integrating diagram markers

<sup>3</sup>In such an algebraic setting, the (meta)operation of integration is immediately commutative and associative because the term above can be formally defined as a composition of categorical operations on diagrams which are commutative and associative; this might be compared with rather involved reasoning on this fact in [4].

In fact the problem of whether markers are consistent is nothing but a graph-based counterpart of the familiar problem (in relational data models) of whether two dependencies are consistent. The latter is known to be undecidable, that implies undecidability of the view consistency problem in general. However it can be decidable in some special but interesting cases. See [9] for some details.

Thus, in respect to the taxonomy of conflicts and correspondence assertions proposed by Spaccapietra *et al* [17, 19, 18], it can be said that owing to introducing additional inter-schema sketches into integration, AGO reduces all that conflicts to conflicts of two kinds—structural conflicts (of being basic/derived) and constraint conflicts (between diagram markers), whereas all kinds of correspondence assertions are reduced to equations between arrows and equations between nodes. Among these conflicts, only marker conflicts are responsible for consistency of views.

**Conclusion** As it was said, an advantage of the sketch specification language is its universality: it follows from general results of categorical logic that any data specification expressible in a conventional formal language of the modern mathematics can be replaced by an equivalent sketch of some appropriate type. So, sketches provide a real possibility to handle heterogeneity in a consistent way.

In addition, rigor semantics of sketches makes it possible to distinguish in the general integration procedure those parts which can be fully automated and those that have to be performed by an integrating person (DBA). In its turn, the latter phases can be also made computer-aided and, on the whole, the sketch framework brings to light limitations and possibilities of automation in schema and data integration.

On the other hand, our experience has shown that sketches turn out to be very handy as a machinery for semantic modeling and integration. In particular, they are convenient for semi-automated phases of integration being performed in an interactive mode. In addition, since the sketch treatment of integration is essentially algebraic, several extensive integration steps can be reduced to formal algebraic manipulations with terms and equations, that provides their efficient computer implementation.

## References

- [1] S. Abiteboul and R. Hull. IFO: a formal semantic database model. *ACM TODS*, 12(4):525–565, 1987.
- [2] M. Barr and C. Wells. *Toposes, Triples, Theories*. Springer, 1985.
- [3] C. Batini, M. Lenzerini, and S. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
- [4] P. Buneman, S. Davidson, and A. Kosky. Theoretical aspects of schema merging. In *Proceedings of EDBT'92*, 1992.

- In *Next Generation of Information Technologies and Systems, NGITS'95*, pages 69–79, Naharia (Israel), 1995. Extended abstract is available on ftp: <ftp://ftp.cs.chalmers.se/pub/users/diskin/ngits95>.\*
- [6] Z. Diskin. Formalizing graphical schemas for conceptual modeling: Sketch-based logic vs. heuristic pictures. Technical Report 9501, FIS/LDBD, Riga, Latvia, 1995. On ftp: <ftp://ftp.cs.chalmers.se/pub/users/diskin/kruse95>.\*
  - [7] B. Cadish and Z. Diskin. Heterogenous view integration via sketches and equations. To appear in *Proc.Int.Symp.on Methodologies for Information Systems, Springer LNAI'?*, 1995.
  - [8] Z. Diskin and B. Cadish. Variable sets and functions framework for conceptual modeling: Integrating ER and OO via sketches with dynamic markers. In *Proc. 14th Int.Conf.on Object-Oriented and Entity-Relationship Modeling, (OO & ER)'95*, Springer LNCS'1021, 1995. On ftp: <ftp://ftp.cs.chalmers.se/pub/users/diskin/er95>.\*
  - [9] Z. Diskin, B. Cadish, and I. Beylin. Algebraic graph-based approach to management of multibase systems, II: Mathematical aspects of schema integration. Technical Report 9502, Frame Inform Systems/LDBD, Riga,Latvia, 1995. On ftp: <ftp://ftp.cs.chalmers.se/pub/users/diskin/tr9502>.\*
  - [10] R. Hull. Four views of complex objects: A sophisticate's introduction. In *Nested relations and complex objects in Databases*, LNCS'361, pages 87–116, 1987.
  - [11] R. Hull and R. King. Semantic database modeling: Survey, applications and research issues. *ACM Computing Surveys*, 19(3):201–260, 1987.
  - [12] J. Lambek and P. Scott. *Introduction to higher order categorical logic*. Cambridge University Press, 1986.
  - [13] A. Motro. Superviews: Virtual integration of multiple databases. *IEEE TOSE*, 13(7):785–798, 1987.
  - [14] E. Pitoura, O. Bukhres, and A. Elmagarmid. Object orientation in multi-database systems. *ACM computing surveys*, 27(2):141–195, 1995.
  - [15] D. Shipman. The functional data model and the data language DAPLEX. *ACM TODS*, 6(1):140–173, 1981.
  - [16] A. Sneth and C. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 1990.
  - [17] S. Spaccapietra and C. Parent. Conflicts and correspondence assertions in interoperable databases. *ACM SIGMOD Record*, 20(4):49–54, 1991.
  - [18] S. Spaccapietra, C. Parent, and Y. Dupont. Model-independent assertions for integration of heterogeneous schemas. *Very Large Databases Journal*, 1(1), 1992.

- [20] C. Wells and M. Barr. The formal description of data types using sketches. In *Mathematical foundations of Programming language semantics*, volume 298 of *Springer LNCS*, 1988.