Data Semantics Revisited

Alexander Borgida[1] **and John Mylopoulos**[2]

Abstract. The problem of data semantics is establishing and maintaining the correspondence between a data source and its intended subject matter. We review the long history of the problem in Databases, and contrast it with recent research on the Semantic Web. We then propose two new directions for research on the problem and sketch some open research questions.

1 Introduction

" . . . It struck me that it would be good to take one thing in life and regard it from many viewpoints, as a focus for my being, and perhaps as a penance for alternatives missed. . . . "

[1]

Two panels, held at SIGMOD'98 (Seattle, June 4) and CAiSE'98 (Pisa, June 11), discussed the topic of data semantics and its place in Databases research in the next millennium. The first, titled "Next Generation Database Systems Won't Work Without Semantics" included as panelists Philip Bernstein, Umesh Dayal, John Mylopoulos (chair), Sham Navathe and Marek Rusinkiewicz. The second one, titled "Data Semantics Can't Fail This Time!" included as panelists Michael Brodie, Stefano Ceri, John Mylopoulos (chair), and Arne Solvberg.

Atypically for panels, participants to both discussions generally agreed that data semantics will be *the* problem for Databases researchers to tackle in the near future. Stefano Ceri summed up well the sentiments of the discussions by declaring that

". . . The three most important research problems in Databases used to be 'Performance', 'Performance', and 'Performance'; in years to come, the three most important and challenging problems will be 'Semantics', 'Semantics', and 'Semantics' . . . "

What is the data semantic problem? In what sense did it "fail" in the past? . . . "And why did the experts agree – unanimously – that the situation was about to change?

We review the data semantics problem and its long history in Databases research, noting the reasons why solutions of the past won't work in the future. We then review recent work on the Semantic Web and the directions it is taking. Finally, we sketch two new directions for research on data semantics.

[1] Author.s address: Dept. of Computer Science, Rutgers University, NJ, USA; email address: borgida@cs.rutgers.edu

[2] Author.s address: Dept. of Computer Science, University of Toronto, Toronto, Canada; email address: jm@cs.toronto.edu

2 The Problem and Its History

A data source is useful because it models some part of the real world, its *subject* (or *application*, or *domain of discourse*). The problem of *data semantics* is establishing and maintaining the correspondence between a data source, hereafter a *model*, and its intended subject. The model may be a database storing data about employees in a company, a database schema describing parts, projects and suppliers, a website presenting information about a university, or a plain text file describing the battle of Waterloo.

2.1 Semantic Data Models

The problem has been with us since the very early days of the Relational Model. Indeed, within four years from the publication of Ted Codd's classic paper [2], there were proposals for semantic data models that were more expressive than the Relational Model and were – therefore – capable of capturing more "world knowledge". Specifically, in 1974 Jean-Raymond Abrial proposed a semantic model that was, in fact, an early object-oriented data model [3]. At the very first Very Large Data Bases conference in 1975, there was a whole session on semantic data models, where Peter Chen presented the Entity-Relationship (hereafter ER) model [4]. Dozens of other proposals followed, and the race was on for ever-more expressive semantic data models that would slay the data semantics dragon.

But how was database practice to be influenced by these proposals? Three basic options were considered:

- Offer semantic data models through DBMS technology; this option implies building DBMSs based on a semantic data model, e.g., an Entity-Relationship DBMS;
- Use semantic data models only during design-time – i.e., while the database is being designed – and factor them out completely during run-time;
- Use semantic models as part of the user interface to make the contents of a database more understandable to the end user.

For performance reasons, option two prevailed. This means that the semantics of the data in a database were factored out from the running database system and were distributed to its operational environment, i.e., its database administrator and its applications programs. If you wanted to know what the data really meant, youd have to talk to the administrator of these data and/or check out carefully the applications that accessed and updated these data.

Data Semantics Solution 1 *Data semantics is managed by the operational environment of a database system, i.e., its database administrator(s) and applications programs.*

This is a practical solution that has worked well as long as the operational environment of a database remains closed and relatively stable. In such a setting, the meaning of the data can indeed be factored out from the database proper, and entrusted to the small group of regular users and/or application programs.

Unfortunately, there is a well-known drawback to this solution: *legacy data*. After years of use, organizations have found themselves time-after-time in a situation where no one knows any more what a particular database and its applications really mean. It has been estimated that legacy data cost organizations around the world billions of euros to maintain and reengineer.

When it comes to building database technology, Solution 1 leaves semantic issues and semantic data models out in the cold. In this respect, research on data semantics has largely been sidelined in database conferences since the early '80s. Instead, semantic data models found a place in database design methodologies. They also influenced in a substantial way software modeling languages proposed more than a decade later, including UML.

As noted, Solution 1 assumes that the environment of a database system remains closed and stable. Throughout the '90s, there was steady progress in making software systems ever-more distributed, open, and dynamically reconfigurable. With the advent of web technologies and standards, e-Business, peer-to-peer systems, Grid computing and more, that trend promised to usher in a new era of computing where computer systems were universally connected, open, dynamic and autonomic. *That* was the change panelists at SIGMOD'98 and CAiSE'98 saw forthcoming. And *that* was the reason for predicting renewed interest in and growing importance for the problem of data semantics.

2.2 The Semantic Web

Unlike database-resident data, Web data have until recently only been intended for human consumption. Rightly so, Tim Berners-Lee realized that Web data cant be made machine-processable unless they come with a formal account of their meaning. Hence his call for the Semantic Web, which has enjoyed world-wide interest since it was made in the Spring of 1999 [5].

This call, supplemented by several other publications (e.g., [6]), envisions technologies and methodologies for attaching semantic annotations to web data, so that they can be interpreted and reasoned about by applications. These annotations can be based on formal ontologies of concepts and relationships that provide a formal – and hopefully widely accepted – vocabulary for a particular domain, be it general (e.g., social actions and interactions), or specific (e.g., manufacturing, genomic biology, or cardiology).

Although there are no generally-accepted detailed proposals of how specifically data semantics should be represented on the semantic web, one approach might be to have an XML document, with annotations to ontologies, as in the following example text, where the word "seminar" is disambiguated by pointing to the concept `Course` in some ontology, which is further qualified to be offered at UniTN.

> "...The **<concept subClassOf** `= x:Course,` **hasValue** `= [x:offeredAt , UniTN], ...> seminar </`**concept**`>` covers a lot of material about the Greek philosophers in a short time ..."

Note that this makes it clear that "seminar" does not refer in this case to a one-time lecture presented by a visitor, which is one of its other possible meanings.

There has been considerable effort and progress on formal languages for describing metadata/ontologies. For example, the specification of the `Course` concept might look as follows in the OWL ontology language [7].

```
<owl:Class rdf:ID="Course"> ...
      <rdfs:subClassOf>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#offeredAt" />
              <owl:allValuesFrom rdf:resource="#School" />
            </owl:Restriction>
      </rdfs:subClassOf>
      <owl:disjointWith rdf:resource=" #Student"/>
...
 </owl:Class>
```

The above then provides a second vision of what constitutes a data semantics solution:

Data Semantics Solution 2 *Annotate data with terms defined in a formal ontology.*

There has been much less effort on the use of ontologies, and serious questions remain concerning the scalability of the approach, i.e., can we build scalable technologies for it? Equally questionable are our technologies/methodologies for aligning and interoperating data sources in the presence of multiple ontologies. As well, it remains an open issue whether "mere mortals" (i.e., your average practitioner) can use expressive formal languages. Indeed, experience with formal, logic-based languages such as `SQL`, `Datalog` and `Z` in other areas of Computer Science suggests otherwise. Making Semantic Web languages widely usable will definitely require tools far beyond the state-of-the-art.

We propose to look next at a more precise variant of Solution 2, taking a careful look at the notions of model and modeling.

3 Models and Mappings

What happens if we take to heart the notion that the data we have is a *model* of some application domain? To do so, let us first take a more general look at the notion of *model*. In an insightful philosophical analysis, Ladkin [8] argues, among others, that

> *" . . . 'model of' is a ternary relationship, relating the subject S, the model M, and the purpose P for which the model is built . . . "*

Consider, for example, the case of a geopolitical globe as a model of the Earth. Such a globe shows countries, borders, cities, major rivers and mountains, but not climatic regions. In turn, a model of the Earth for the purpose of studying the motion of planets in the solar system, would likely not be a globe, but instead would reduce our world to a single point, corresponding to its center of gravity.

In the large majority of cases, *the purpose of a model is to answering certain kinds of questions about the subject*. This means that in some ways the model is able to answer these questions more precisely/easily/quickly than the subject itself. In such cases, in order to describe a modeling situation we need at least the following:

– A set Q_S of *questions* about the subject world that we would like to have answered using the model.
– The *model*, which is an information source, capable of answering certain questions Q_M, after it has been built/evolved.
– *A mapping from questions* in Q_S to one (or more) questions in Q_M, and *an inverse mapping from the answers* in M to answers about the subject S.

In the example above, the (informal) questions to be asked have to do with the existence and (relative) position of features on the Earth's surface, but not its interior composition. The questions about the model are answered by direct observation of the model by a human, aided perhaps by a string/ruler/compass (something which is not possible for the life-size subject). The mapping of questions and answers is based on the scale reduction of the Earth's spherical surface to that of the globe.

We can now apply the above framework to information systems, such as databases. Suppose we have a relational database with a table indicating when courses meet. This can be viewed as a model of the (real-world) university, for purposes of answering (natural language) questions such as "When does a course meet?", "When is a room free?", but not "Is there a projector in the room?". The questions for the model are likely expressed in SQL, and the answers are tables of tuples. The mapping between subject questions and model questions is informal, in the mind of the programmer or database user. On the other hand, if there is a natural language interface to the database, then the mapping is in fact computed (heuristically) by the natural language processing system.

Looking at information systems, one recognizes the additional need for explicit operators that construct and update the model itself, as the subject evolves or more information is discovered about it. For example, in the case of the above database, these are SQL DDL statements for defining the database schema, as well as SQL DML statements for inserting/updating appropriate tuples. Of course, the usefulness of the answers depends on the accuracy of this model-building activity.

To summarize, we have identified the need for the following in the case of information models:

– a subject world, for which *a set of questions* Q_S is of interest, with answers of the form A_S.
– a model, equipped with (i) *Declare* operations for describing generic/schema-like aspects of the model; (ii) *Tell* operations for providing detailed information about the current state of the model; (iii) *Ask* operations which take queries in language(s) Q_M and provide answers in language(s) A_M; (iv) a specification of how query answering depends on the told information, and for practical systems, an implementation thereof[3].
– *a mapping f from a question in* Q_S *to one ore more questions in* Q_M;
– *a mapping* ∂ *from answers in* A_M *to* A_S.

[3] The core of such a functional view of information sources was first offered by Hector Levesque [9].

Data Semantics Solution 3 *The semantics of the data in a model resides in its ability to answer questions about the subject, and is hence captured by Declare/Tell/Ask operations, associated languages, and mappings.*

In order to better understand this approach, let us take a brief look at each of the above components.

3.1 Models

We have found it useful to categorize models according to the way in which query answering is specified. In this paper we consider *intensional models*, consisting of collections of sentences in some formal language L. L is assumed to be equipped with an entailment relationship $\models$, on which question-answering is based: the answer to a query q is True if the collection of told sentences KB entails q. Information models based on logical theories, such as Description Logics [10], and Reiter's reconstruction of the Relational Model in First Order Logic [11] are examples of such models.

Models can also be categorized in regards to their support of partial information (nulls, disjunction, closed/open world assumption), inconsistency, inaccuracy (errors and bounds). Moreover, information models can be distinguished on the basis of efficacy of query answering (formal complexity as well as practical implementation), and how this varies depending on the languages supported by the *Tell* and *Ask* operators. These have been key concerns of Databases and Knowledge Representation research over the past decades.

3.2 Subjects

To begin with, the subject domain is often partitioned into *generic*, usually time-invariant aspects (so-called *definitional* or *terminological* component) dealing with human conceptualizations of the domain, and specific facts, describing individuals and their interrelationships in the current state of the world (*assertional* component). These often give rise to distinct components of information models (schemas vs. tuples/documents/...).

When discussing data semantics, one often hears talk about the information system modeling "*the real world*". Although in some situations data may be obtained from sensors, in most cases the view of the world is mediated by some human being(s), so that the subject is more appropriately viewed as *human beliefs about the world*. Finally, recent developments in information processing have made not uncommon situations such as an XML document representing data from a database, or conversely, a database storing an XML document. In this case, the subject is itself a formal system, which allows us to describe the mapping in a formal fashion.

For example, in a database schema designed from an ER diagram, we can think of the entities as unary predicates, relationships as n-ary predicates, and attributes as binary relations, and then express the mapping between the ER and relational model using predicate logic:

$$\texttt{db:enrolment}(\texttt{sname}, \texttt{crsId}, ...) \leftrightarrow \exists X, Y.\texttt{er:Students}(X) \wedge \texttt{er:Course}(Y) \wedge \\ \texttt{er:hasName}(X, \text{sname}) \wedge \texttt{er:hasId}(X, \texttt{crsId}) \wedge \texttt{er:EnrolledIn}(X, Y) \wedge \ldots$$

3.3 Mappings

The sets of questions Q_S and Q_M are usually infinite, and we require some finite means for specifying the mapping from the former to the latter. This can be achieved by making both query languages *compositional*, and then reducing the problem of mapping from Q_S to Q_M to the problem of (i) relating the primitive (non-logical) terms of the two languages, such as their predicate symbols/schemas and constants; and (ii) providing some kind of a homomorphic extension of this mapping to composite formulas.

There are several ways of expressing the base relationships. One approach is an informal/graphical specification of correspondences between components of schemas, as used in the Clio system [12] or in model management [13].

A more formal approach, based on logic, is illustrated in the above example involving the student database and ER schema, where each predicate in the model had associated with it an expression over the subject predicates. This can be seen as an analogue of the Local-as-View (LAV) approach to information integration [14, 15], where the so-called mediated schema is the subject domain. An advantage of this approach is that it provides a way to obtain simple atomic facts that can be told to the system in order to build up the model. Alternatively, in analogy with the Global-as-View (GAV) approach, each predicate of Q_S can be associated with an expression in Q_M e.g.,

$$\texttt{er:Students}(X) \leftrightarrow \exists \texttt{sname}.\texttt{db:enrolment}(\texttt{sname}, \ldots) \wedge X = f(\texttt{sname})$$
$$\texttt{er:hasName}(Y, N) \leftrightarrow \exists \texttt{sname}.\texttt{db:enrolment}(\texttt{sname}, \ldots) \wedge Y = f(\texttt{sname}) \wedge N = \texttt{sname}$$

where f is some injective function, guaranteed to return a different value for each argument, thereby ensuring a different student individual for each student name in the enrolment database relation. An advantage of this approach is that it facilitates translation of queries from the ER subject world to the database model, in marked contrast to the LAV approach discussed above. The – more general – GLAV approach introduced in [16] and used in [17] among others, provides for collections of arbitrary query pairs $q_S(x)$ and $q_M(y)$, each of which returns sets of (tuples of) substitutions, and these can then be related in quite general ways (e.g., by set theoretic containment, membership, or numeric comparison). In fact, [18] suggest that a mapping is an arbitrary formula in some logical language, involving elements of the two models.

Note that in view of such options, the expression "mapping from model M to subject S", implying a directionality, is somewhat misleading since there are different ways of expressing the mapping, some of which seem to be from S to M. Moreover, the mapping from M to S is used to translate queries from S to M. Note also that in cases other than the GAV approach, there may be no precise query translation, and one may be reduced to approximations, as with query answering using views. The formal (query) languages used in specifying mappings can be based on standard First Order Logic, or subsets thereof (e.g., Datalog, SQL, Description Logics) or more complex variants involving structure (e.g., XQuery) or higher-order aspects (e.g., Hylog, Infinitary or Second Order Logics).

For example, suppose that part of the contents of relational tables

```
db:class(cId,cTitle,term)
db:enrolment(cId,sname),
```

concerning course enrolments, is to be published in an XML document with schema described by the DTD

```
<!ELEMENT catalog (course*) >
<!ELEMENT course (title, students) >
<!ELEMENT students  (student*) >
<!ELEMENT student #PCDATA >
<!ELEMENT title #PCDATA >
```

A standard two-step approach is to first map the relations to simple XML trees, where the first level corresponds to relation names, and the second level to tuples

```
<!ELEMENT class (ctuple*)>
<!ELEMENT ctuple (cId,cTitle)>
<!ELEMENT enrolment (etuple*)>
<!ELEMENT etuple (cId,sname)>
<!ELEMENT cId #PCDATA >
<!ELEMENT sname #PCDATA >
```

and then use XQuery to describe the construction of the final desired document:

```
<catalog>
{for $c in $db/class/ctuple
return <course>
<title> $c/cTitle </title>
<students>
   {for $e in $db/enrolment/etuple where $e/cId = $c/cId
   return  $e/sname }
</students>
   </course>
}
</catalog>
```

This XML document is a model of the relational database for purposes of answering questions unrelated to the semester when the course is offered. Hence *the meaning of the XML model is (should be) defined in terms of the meaning of its predecessor, the relational database, using the mapping and the queries to be supported*. This is in line with our third take on data semantics, based on the notion of model introduced at the beginning of this section.

A final note concerning mappings: while normally these are concerned with the intensional/schema aspects of the models, it is useful to also look at the extensional/individual aspects. In particular, mappings between ontologies, for example, tend to assume that the individuals in the subject and model world are identical. However, we have seen that this is more complex in the case of mappings between object-centered models such as ER, and "flat" data models – such as relational databases – where we have to introduce Skolem functions. In general, things can be even more complex, as in the case of a census database about households that is used as a model for information about

individuals. Here, we need to keep a binary relation between each household and the individuals living in it [19].

Of course, the above framework for data semantics, based on the notions of model and mapping, is just a recasting of voluminous previous work in databases and knowledge representation. In particular, research on *data integration* (e.g., [14], [15], [20]) has developed a rich framework where a mediated schema is inserted between users and the heterogeneous information sources they are trying to access, with the users issuing queries against the mediated schema. These are translated into queries about the original data sources, using mappings.

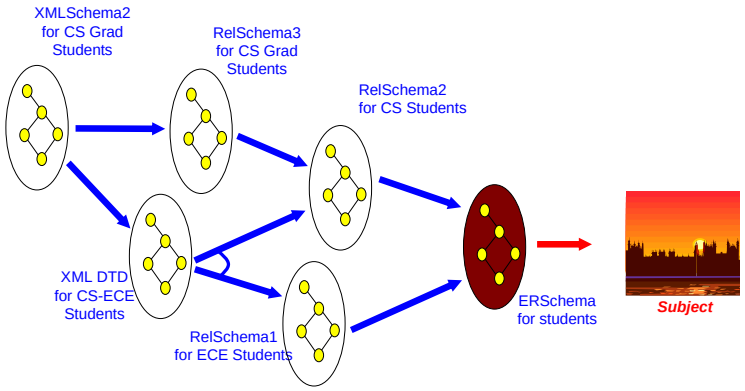

Fig. 1. The mapping continuum.

4 The Mapping Continuum

Consider the following examples of modeling:

- A photo of a landscape is a model of the landscape (its subject matter).
- A photocopy of the photo is a model of a model of the landscape.
- A digitization of the photocopy is a model of the model of the model of the landscape
- etc.

This kind of situation, first considered by Brian Cantwell Smith [21], shows that meaning is rarely a simple mapping from symbol to object; instead, it often involves *a continuum of (semantic) correspondences* from symbol to (symbol to)* object. Succeeding sections discuss how this paradigm can be applied to information systems. This paradigm constitutes our first attempt at a novel approach to data semantics.

4.1 The Mapping Continuum Hypothesis

Consider the chain of information models illustrated in Figure 1. In this case we have a series of models and mappings supporting the modeling relationship:

– The ER schema is a (conceptual) model of the university domain, or at least the part of it dealing with students. (There can be no formal mapping between the ER schema and the (informal) domain of discourse.)
– RelationalSchema1 is a model of the ER conceptual schema, for the purpose of queries concerning computer science students. (We have illustrated this kind of mapping in Section 3.2)
– RelationalSchema2 is a model of the conceptual schema, for the purpose of queries concerning electrical and computer engineering students.
– XML DTD1 supports queries about either CS or ECE students, and is hence a model of both the previous relational schemas. Therefore we show arrows to both, representing the existence of mappings. (We have given an example of this kind of mapping at the end of Section 3.3)
– RelationalSchema3 is a restricted model, dealing with graduate students.
– XMLSchema2 is an XML version of RelationalSchema3, but it can also be used to answer some questions directed to XML DTD1 (those relating to graduate students). This means that there are mappings relating XMLSchema2 to both of them.

Some mappings are *lineal*, connecting a schema to the predecessor subject for which it is intended to be a model. All mappings in Figure 1, except the one from XML Schema2 to XML DTD1, are lineal. Lineal mappings capture the semantics of the schema in the third sense of data semantics, introduced in Section 3 above, and will therefore be called *semantic mappings*. Note that both of the mappings for XML DTD1 are lineal, as would be the case for data warehouses in general, and hence the semantics of such models is more complex.

A situation described by a mapping continuum can be represented by a graph, whose nodes are models/schemas, and whose edges are mappings. In order to avoid circularity, the semantic mappings must form an acyclic subgraph, ending at so-called *ground nodes*. Ground nodes will likely be closer to a conceptual/ontological view of the domain of discourse, and in some sense anchor the semantics of other models. Ideally, there would in fact be a single ground node, corresponding to an ontology of the application domain. However, as with the semantic web, we acknowledge that agreeing on a single, universal ontology is likely to be infeasible. We are now in position to offer a fourth version of a solution to the data semantics problem:

Data Semantics Solution 4 *Every (non-leaf) model in the semantic continuum comes with an explicit semantic mapping to some other model, and its meaning is the composition of the mappings relating it to the ground node(s) reachable from it.*

This solution leads to a research program that includes issues such as:

Mapping composition. Since we are proposing to define the meaning of an information source in terms of the composition of the semantic mappings emanating from it, we must of course clarify the notion of "composition" itself. Fagin et al [18] point to at least two possible interpretations: their own, which is query-independent, and that of [17], which is parameterized by a query language. In our case, this parameter would naturally be determined by the set of queries that the model is intended to answer. Interestingly, these queries may be about the "opposite" end of the composition than in [17]. Once

this issue has been settled, there are several questions, such as the language required to express the composition of two mappings (it may be some variant of infinitary or second order logic), and the computability of the mapping itself, if we are trying to use it to "populate" one model using instance data from another model.

Consistency of semantics. In the case of graphs where nodes have multiple predecessors, we need to consider the problem of how meaning is defined in the case when there are multiple lineal predecessors. For example, do we want all paths to ground models or just the individual mappings? And in general, we need to consider the issue of consistency when there are multiple paths from a model to some other node in the continuum. In such cases, it isnt clear how consistency is to be defined.

Although we started from an analysis of the notion of modeling, and the correspondence continuum, others have proposed similar frameworks. Research on *peer-to-peer data management* [22, 23, 19] has proposed a framework where queries to each peer may be translated, using mappings, into queries of neighboring peers ("acquaintances"), and this process may be repeated. The connection of peers to each other corresponds to the mapping graph of our mapping continuum, although without a requirement for acyclicity.

Likewise, research on *data provenance* [24] considers the situation where one data source is partially populated with information from one or more antecedents, but there are updates that need to be propagated (both backwards and forwards). Moreover, for an answer to a given query, one wants to know the source(s) of each element of the answer. There is an obvious mapping from a data source to its antecedents, although it is not clear that this mapping is semantic – i.e., to what extent the latest information source's semantics are captured *entirely* by that of its antecedents.

The above areas have tended to study the problem at the same level of "semantic abstraction" (e.g., integrating two ontologies or two relational databases). Our research program is distinguished primarily by an emphasis on (directed) semantic mappings, and their path to ground nodes.

4.2 A Detour on Mapping Discovery

The above framework can be interpreted as suggesting that rather than having to reconstruct the semantics of a data source every time it is needed, it might be better to maintain/discover mappings between that data source and others. To make this scenario work, we need some evidence that mapping discovery might be easier than semantic reconstruction (e.g., ontology building and alignment), which we have argued is hard.

As intimated above, we believe that *lineal mappings* ought to be codified and preserved during the development of a new information source. So the ER conceptual model, and the mapping from the relational schema designed from it, ought to be formalized and maintained, so that it can support later data warehousing, for example (e.g., [25]). But what if this did not happen, or if we want to find non-lineal mappings? Experience indicates that most end-users have problems with logical formalisms, even ones as simple as Datalog. To address this problem, we need *tools* that help derive such mappings. A recent successful tool of this sort is Clio [12, 26], which takes as input

correspondences (graphical pairings) between *elements* (usually columns/tags) of relational or XML schemas, and produces a GLAV-style mapping between the two schemas that can be used to transfer data from one information source to the other. To understand better the mapping, examples involving specific data values might also be used as devices for eliciting information from Clio users.

For example, in trying to transfer `cs:Teach` information, about who teaches whom, from source db, the user might indicate that `cs:Teach.student` corresponds to db:Enroll.sname, and that `cs:Teach.prof` corresponds to `db:Course. instructor` (since `db:Enroll` does not explicitly list the course instructor). In Figure 2, this is indicated by two correspondence arrows, `vc1` and `vc2`. Although correspondences indicate connections between different schemas, it is also necessary to find logical/semantic connections between attributes in a single schema. Clio assumes that these are indicated that by co-occurrence of attributes in a single relation, and by foreign keys. So the `db` schema is augmented by foreign key information, as indicated by the dashed arrows in Figure 2. As a result of these foreign keys, and a chase-like process, the right-hand side of the mapping will be

```
db:enroll(pname,ctitle),db:course(ctitle,instructor), db:pupil(pname,...),
          db:educator(instructor,...)
```

while the left-hand side will be

```
cs:teach(prof,studntId)
```

and these will be connected using the equalities

```
instructor=prof, studntId=pname
```

which represent the correspondences, yielding the Horn rule

```
cs:teach(prof,studntId) :-
instructor=prof, studntId=pname
db:enroll(pname,ctitle), db:course(ctitle,instructor),
db:pupil(pname,...), db:educator(instructor,...).
```

To summarize, the hypothesis underlying Clio is that users will find it is easier to specify simple correspondences, and that the actual mappings desired will be among the ones generated by the tool.

We have recently developed a tool, Maponto [27], for uncovering mappings between relational schemas and ontologies. The tool can be used not just in cases when lineal mappings were not preserved, but also when the ontology was independently developed (e.g., a data warehouse or semantic web scenario). Inspired by Clio, this tool also starts from correspondences of table columns to datatype properties in the ontology. It then finds connections between the concepts bearing these attributes in the ontology, viewed as a graph, and orders these connections according to total length, while keeping in mind semantic information available in the relational schema as foreign keys. One aim of this heuristic algorithm is to derive the “natural” mapping induced by the classical relational schema design process from ER diagrams, in the case when the ontology is exactly the ER schema, and the relational schema has not been de-normalized.

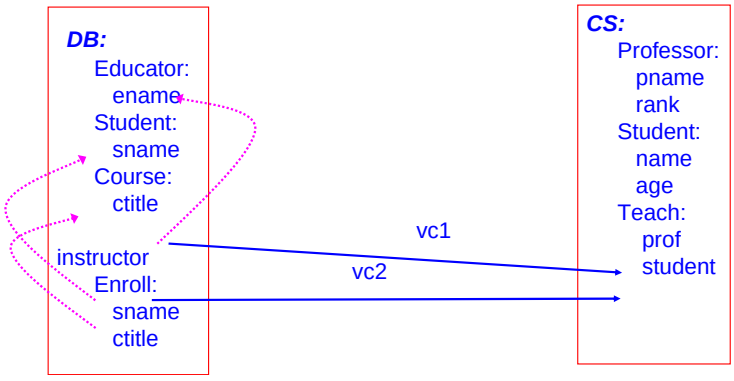

Fig. 2. Schemas and their correspondences.

5 Intentional Semantics

Traditionally, the semantics of data deals with the "*what/when*" aspects of a subject: what are the objects, their inter-relationships, their groupings into concepts, and constraints thereof. To achieve a more nuanced understanding of the semantics, it makes sense to consider "*how*" aspects: distinguish (conceptual) objects that represent activities and processes in the domain. In fact, some conceptual modeling languages proposed several decades ago followed this approach, applying the same paradigm to describe both static and dynamic aspects of a domain. For example, Taxis transactions [28] were classified into taxonomies with inheritance; Taxis scripts [29] extended this to workflows, which were Petri nets that used message passing for communication; and RML [30] used class hierarchies to organize objects, events and assertions while specifying requirements for a software system.

To motivate the need for more, consider a university, where an information system maintains, among others, relations

```
Student(st#,nm,addr,advisor,dept,degree)
Course(crs#,crsname,instr,dept,yr,term,size)
```

Suppose that the enrolment process at this university requires students to sign-up for courses at the end of one term (say, May), and pay for each course at the beginning of the next term (say, September), once the official "add/drop" period is over. (This design in itself is the result of balancing needs ...) Consider now a query such as "Find the total size of courses in a specific semester". The meaning, and *proper use*, of the answer depends on the meaning of `size` in table `Course`. If the size was incremented every time a student signed up for a course, then the sum of the sizes may not be an accurate reflection of the total enrolment in courses for that term, since some students may change their mind, not pay for the course, and therefore fail to be officially enrolled. So the sum of sizes is likely to be an *overestimate* of enrolments. This may be satisfactory from the point of view of the university central administration, which would like high enrolment figures to support its request for more Government funding. However, this

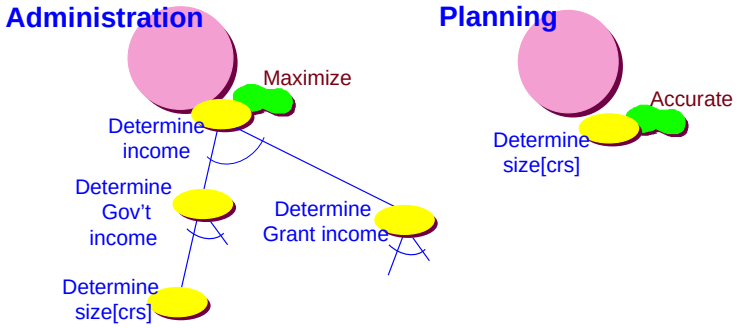

Fig. 3. Actors and their goals.

may be less than satisfactory for the university planning department, which needs the most accurate answer possible.

Such considerations should be part of the data semantics solution because they bring in the dimension of trust in the data we are trying to interpret. This naturally leads to the question of how such information is to be obtained and represented. For this, it is helpful to recall that information systems are software systems, hence subject to the much the same development processes as other kinds of software. Now, the development of software can be viewed as being split into several stages, including:

- *Early requirements*, when analysts are trying to understand an organizational setting; this results in an organizational model;
- *Late requirements*, when analysts formulate (software-based) solutions needed by the organization, resulting in contractual requirements;
- *Design and development* of the software system itself.

The organizational model is concerned mostly with the actors/stakeholders, their goals, and how these are currently met/dependent. Presumably the requirements describe a (software) actor that helps further the goals of (some) organizational stakeholders. We shall use the above university setting to introduce briefly the notions of the *i** notation for capturing early requirements [31], and how this can be applied to capture design decisions during software development, as suggested by the Tropos project [32]. The aim of the presentation – however sketchy and speculative – is to argue for the need to link intentions to data semantics.

To begin with, we model in Figure 3 actors (e.g., Administration) represented by circles. Actor goals (ovals), include determining income for the forthcoming year (for Administration). This goal can be decomposed into a number of subgoals (indicated by edges connected by an arc), such as determining income from the government, and income from grants. In turn, determining government income relies on estimating sizes for courses.

Now we are ready to consider one of the novel features of i*/Tropos: in addition to standard goals, it is possible to also model so-called *softgoals*, which capture general intentions of actors, but which usually dont lead to functional requirements for the new

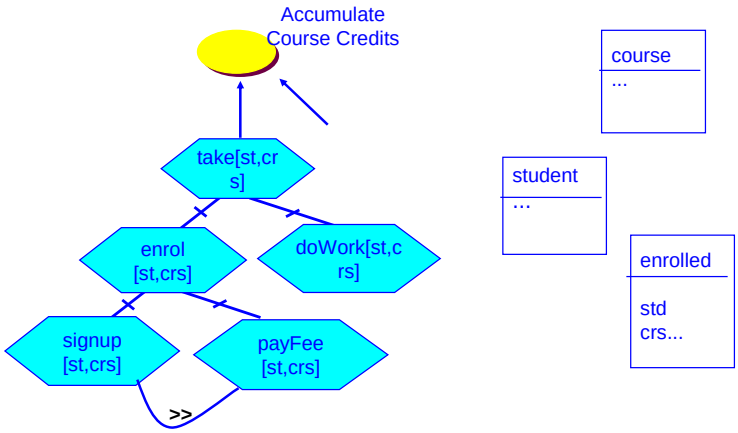

Fig. 4. Goals, tasks and objects.

system. Instead, softgoals are used to make choices between alternatives (architectures, designs, implementations).

In the above case, one of the softgoal of the Administration actor (represented as a cloud-like shape) is to *maximize* income, which in this case results in wanting to maximize various subgoals, including course sizes. As shown in Figure 3, the Planning department is also an actor in this setting, and it happens to also have among its subgoals the determination of course sizes. However, its softgoals associated with its wish to know course sizes are different in that it wants information that is as accurate as possible. (Imagine *maximize* and *accurate* as being among a list of possible qualities, for which a "logic" has been established.)

Before we turn to realizing these goals, we need to introduce one more bit of notation, concerning the activities and data that are used in the implementation. Consider, in Figure 4, the student goal of accumulating course credits. Analyzing it, we realize that it *concerns* several objects – a student and some courses, so that these must become "resources" (data objects) in the implementation. (Further implementation decisions will represent these as relational tables such as those diagramed above in rectangles.) At the same time, the hexagonal box, labeled `take[st,crs]`, with an arrow pointing to the goal, shows an activity that can be used to achieve that goal. (There may be other ways of achieving the goal, such as working or being a teaching assistant.) The diamonds underneath, linked by crossed edges, indicate subactivities, which may be related by double arrows marking temporal precedence. So `take[st,crs]`, involves among others `enrol[st,crs]`, which in turn requires first `signup[st,crs]` and then `payFee[st,crs]`.

Consider now alternative ways of fulfilling the `size[course]` goal. One would be to count how many students have signed up for the course – represented as the activity `count signups`, in the hexagonal box in Figure 5. A different alternative is to count payments. The effect of these alternatives on the two softgoals, `Maximize` and `Accurate`, are represented by arrows labeled with `+` and `-`, indicating positive

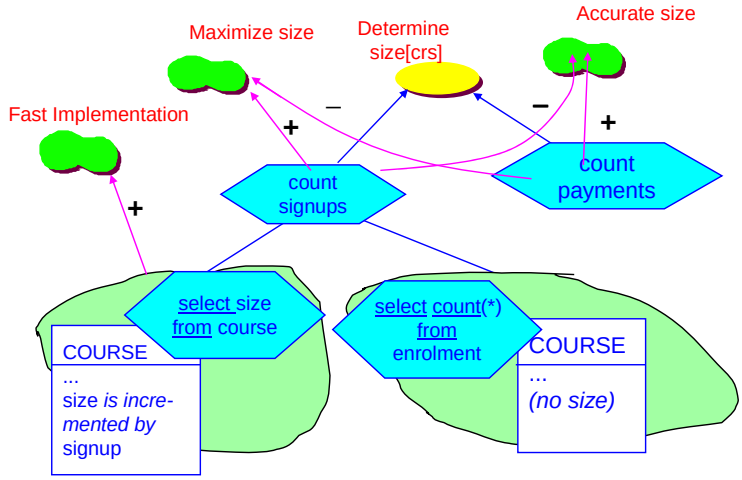

Fig. 5. Different activities for determining course size.

and negative contributions towards them. From this representation it is possible to read off that counting signups will tend to maximize course size but decrease accuracy, in contrast to counting payments.

We also have alternative implementations for the `count signups` process: we can either issue a `select count(*)` type of query over the enrolment relation, or we can essentially "cache" the size of each course, by incrementing it as part of the `signup[st,crs]` activity carried out by each student, as diagrammed in Figure 4. The second approach might be more efficient from the point of view of time, and hence contribute positively to the overall non-functional requirement of fast response time.

Data Semantics Solution 5 *Data semantics a schema, as well as the intentions behind its design.*

Note that Solution 4 dealt with data semantics by answering the question "Where did the schema of the data come from?" Solution 5, on the other hand, focuses on intentions and accounts for data semantics by offering a framework for answering the question "Why is the design of the schema the way it is? What were the alternatives? Why was it chosen among the alternatives?" This is a radical solution, to be sure. But trust is a major issue for data that are created and used in an open, distributed environment such as the Web. And questions of trust cant be accounted for without bringing into the picture "somehow" the intentions of the designers and the managers of an information source. Radical solutions lead to radical research agendas. Here are some issues worth exploring:

– **From goals to schemas.** Traditionally, database design amounts to a series of steps that first construct a conceptual (e.g., ER) schema, and then transform this into a relational one through well-defined transformations. This process needs to be augmented, so that the designer starts with stakeholder goals and softgoals and

through goal analysis generates a set of possible schema designs. The output of the design process is now a set of schemas, along with their evaluation with respect to a set of softgoals. Among these schemas, one is chosen for further refinement using existing techniques. The KAOS project [33] offers a glimpse of what this design process might look like, but focuses on the design of software rather than databases.

– **Trusted query processing.** Suppose that along with a query, we also specify desired qualities (e.g., maximum/accurate course size numbers). Given a database and its schema design, wed like to be able to tell (i) if the data in the database "match" the desired qualities; (ii) (if not) develop techniques for populating exactly/approximately alternative schema designs with the data that exist. For example, if the Government wants accurate course size counts but is getting instead optimistic ones from University administration, perhaps data from previous years can be used to derive approximate, but more accurate course size counts. Such "data correction" mechanisms are bread-and-butter for economists, journalists and political scientists. In the era of the Semantic Web, such mechanisms can form the basis for dealing with issues of intent and trust.

6 Conclusions

We have briefly reviewed the history of the problem of data semantics, as well as recent research trends towards the vision of the Semantic Web, pointing out solutions that worked in the past, or might work in the future. We conclude from this review that the problem of data semantics would have been with us even without the Web and its semantic extensions. The problem arises from general trends towards open, distributed computing, where it is no longer possible to assume that the operational environment of an information source is closed and stable. Accordingly, we should be looking for solutions that are general, i.e., not Web technology-specific.

We expressed concerns about current research towards ever-more expressive modeling languages, both from the point of view of scalability for relevant technologies, and usability for emerging tools. As an alternative to prevailing research directions, we proposed a framework where the meaning of data is determined by its origin(s) through (design-time) lineal mappings. This solution places an emphasis on schema mappings and traceability techniques. As another alternative, we suggested concepts and design techniques adopted from (software) Requirements Engineering for analyzing stakeholder goals and softgoals to generate and select designs. This – admittedly radical – solution focuses on the intentions behind a database design, as a means for understanding the data in the database. We believe that issues of trust (in the data one is trying to understand) will ultimately have to be dealt with in terms of intentions and stakeholders.

Acknowledgements

We are grateful to Yannis Velegrakis for helpful feedback to an earlier draft of this paper.

References

1. Zelazny, R.: 24 views of Mount Fuji. Isaac Asimov's Science Fiction Magazine **7** (1985)
2. Codd, E.: A relational model for large shared data banks. Communications of the ACM **13** (1970) 377–387
3. Abrial, J.R.: Data semantics. In Klimbie, Koffeman, eds.: Data Management Systems, North-Holland (1974)
4. Chen, P.: The entity-relationship model: Towards a unified view of data. In: Proceedings International Conference on Very Large Databases (VLDB75). (1975)
5. Berners-Lee, T., Fischetti, M.: Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor. Harper, San Francisco (1999)
6. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Scientific American (2001)
7. W3C: Web ontology language (owl) version 1.0, http://www.w3.org/tr/2003/wd-owl-ref-20030331 (2003)
8. Ladkin, P.: Abstraction and modeling, research report RVS-Occ-97-04, University of Bielefeld, 1997; http://www.rvs.uni-bielefeld.de/publications/abstracts.html#AbsMod. (Technical report)
9. Levesque, H.: Foundations of a functional approach to knowledge representation. Artificial Intelligence **23** (1984)
10. Borgida, A.: Description logics in data management. IEEE Transactions on Knowledge and Data Engineering **7** (1995) 671–682
11. Reiter, R.: Towards a logical reconstruction of relational database theory. In M. Brodie, J. Mylopoulos, J.S., ed.: On Conceptual Modelling, Springer-Verlag (1984) 191–233
12. Miller, R., Haas, L., Hernadez, M.: Schema mapping as query discovery. In: Proceedings International Conference on Very Large Databases (VLDB00), Cairo. (2000)
13. Pottinger, R., Bernstein, P.: Merging models based on given correspondences. In: Proceedings International Conference on Very Large Databases (VLDB03), Berlin. (2003) 826–873
14. Levy, A., Rajaraman, A., Ordille, J.: Querying heterogeneous information sources using source descriptions. In: Proceedings International Conference on Very Large Databases (VLDB96), Mumbay,. (1996) 251–262
15. Lenzerini, M.: Data integration: A theoretical perspective. In: Proceedings International Conference on Principles of Database Systems (PODS02). (2002) 233–246
16. Friedman, M., Levy, A., Millstein, T.: Navigational plans for data integration. In: Proceedings National Conference on Artificial Intelligence (AAAI99). (1999) 67–73
17. Madhavan, J., Halevy, A.: Composing mappings among data sources. In: Proceedings International Conference on Very Large Databases (VLDB03), Berlin. (2003) 572–583
18. Fagin, R., Kolaitis, P., Popa, L., Tan, W.C.: Composing schema mappings: Second-order dependencies to the rescue. In: Proceedings International Conference on Principles of Database Systems (PODS04). (2004) 83–94
19. Borgida, A., Serafini, L.: Distributed description logics: Assimilating information from peer sources. Journal of Data Semantics (2003) 153–184
20. -: Proceedings of semantic integration workshop, at ISWC03, Sanibel Island, october 2003. http://ceur-ws.org/vol-82 (2003)
21. Smith, B.C.: The correspondence continuum, TR CSLI-87-71, Stanford University. Technical report (1987)
22. Halevy, A., Ives, Z., Suciu, D., Tatarinov, I.: Schema mediation in peer data management systems. In: Proceedings International Conference on Data Engineering (ICDE03). (2003)
23. Bernstein, P., Giunchiglia, F., Kementsietsidis, A., Mylopoulos, J., Serafini, L., Zaihrayeu, I.: Data management for peer-to-peer computing: A vision. In: Proceedings SIGMOD WebDB Workshop. (2002) 89–94

24. Buneman, P., Khanna, S., Tan, W.C.: Why and where: A characterization of data provenance. In: Proceedings International Conference on Database Theory (ICDT01). (2001) 316–330
25. Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Data integration in data warehouses. Journal of Cooperative Information Systems **10** (2001) 237–271
26. Velegrakis, Y., Miller, R., Mylopoulos, J.: Representing and querying data transformations. In: Proceedings International Conference on Data Engineering (ICDE05), to appear. (2005)
27. An, Y., Borgida, A., Mylopoulos, J.: Refining mappings from relational tables to ontologies. In: Proceedings VLDB Workshop on the Semantic Web and Databases (SWDB04), Toronto, August 2004. (2004)
28. Mylopoulos, J., Bernstein, P., Wong, H.: A language facility for designing database-intensive applications. ACM Transactions on Database Systems **5** (1980) 185–207
29. Barron, J.: Dialogue and process design for interactive information systems using Taxis. In: Proceedings ACM SIGOA Conference on Office Information Systems, Philadelphia. (1982) 12–20
30. Greenspan, S., Mylopoulos, J., Borgida, A.: Capturing more world knowledge in the requirements specification. In: Proceedings International Conference on Software Engineering, (ICSE82), Kyoto. (1982) 225–235
31. Yu, E.: Modeling organizations for information systems requirements engineering. In: Proceedings IEEE International Symposium on Requirements Engineering (RE93), San Diego, IEEE Computer Society Press. (1993) 34–41
32. Castro, J., Kolp, M., Mylopoulos, J.: Towards requirements-driven software development methodology: The tropos project. Information Systems **27** (2002) 365–389
33. Dardenne, A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of Computer Programming **20** (1993) 3–50