Integrating Information Sources Using Context Logic

Adam Farquhar
Angela Dappert
Richard Fikes
Wanda Pratt

Knowledge System Laboratory
Stanford University
{adam_farquhar,dappert,fikes,pratt}@ksl.stanford.edu

Abstract

It is essential to reduce the cost of integrating information sources and to provide a path that allows for incremental integration that can be responsive to users' demands. This paper presents an approach to integrating disparate heterogeneous information sources that uses context logic. Our use of context logic reduces the up-front cost of integration, provides an incremental integration path, and allows semantic conflicts within a single information sources or between information sources to be expressed and resolved.

Introduction

The number of online network-accessible information sources grows daily. The information promises to provide tremendous value for individuals and corporations. The promise will remain unfulfilled, however, until it is possible to integrate and assimilate information from multiple heterogeneous sources. Because it is impossible to predict the users and patterns of usage in our changing information environment, information providers are not willing to pay a high up-front cost to support integration. Thus, it is essential to reduce the cost of integrating information sources and to provide a path that allows for incremental integration that can be responsive to users' demands. This paper presents an approach to integrating disparate heterogeneous information sources. Our approach uses context logic to ease integration and provide for incremental integration.

Existing approaches do not provide either adequate integration (loosely coupled databases [12]), or adequate flexibility (federated databases [16]), or acceptable costs (global schemas [8,9,17]). Recent work has begun to draw on the insights of the artificial intelligence and knowledge representation communities. The Carnot project [3,7] has taken a global schema approach, but with the hope that the extensive CYC knowledge base [10] will provide a comprehensive common representation. The SIMS project [1] has focused more on the issues of query optimization and planning than on the resolution of semantic conflicts. We share a number of goals with the metadata approach [15], but our technical approach differs substantially. McCarthy and Buvac [14] discuss the application of context logic to integrating databases.

Although the techniques we describe are applicable to a wide variety of information sources, this paper focuses on structured information sources such as relational databases. The difficulty with integrating relational databases is that their views of the world differ. Their ontologies vary, as does the intended meaning of their data (even when it appears quite similar). Their schema may differ in naming conventions, structure (an attribute in one system may be represented as a value in another) and most importantly, their semantics.

Integrating heterogeneous systems requires making implicit assumptions explicit enough to avoid semantic errors. Consider the problems that might arise when trying to determine the number of items in inventory by querying several databases. One database may store the number of individual items, a second may store the number of pallets, and a third may record the difference between outstanding orders and items on hand. The result of summing these numbers will be completely meaningless. It might even result in a negative number.

Our goal is to enable meaningful integration of the information across multiple information sources by resolving semantic inconsistencies in an unobtrusive and cost effective fashion. We want to provide users with access to the complete power of the individual information sources, rather than access to a least-common-denominator global schema. We also want users to be able to take advantage of their familiarity with an individual information source by allowing them to pose queries using that source's vocabulary, but which collect data from others.

This paper focuses on the semantics of the data represented *in* the information sources. It is worth noting, however, that our framework also makes it possible to represent information *about* the

Figure 1a: Schema definitions for a product database.

```
create schema     product
key               name:        char
                  size:        int
                  cost:        int;
create schema     product_type
key               name:        char
                  type:        char;
```

Figure 1b: Partial contents of the product database.

Table: product

name	size	cost
television_1	14	256
simm_1	256	14

Table: product_type

name	type
television_1	television
simm_1	memory_chip

information sources such as access protocols, error conditions, cost of access, reliability of information, completeness of information, and so on. Such information is not essential for semantic integration, but is important for query execution.

A Framework for Information Source Integration

We use a context logic [2,6,13] to represent the schemata and views of the individual information sources, shared ontologies, implicit semantics, and integrated information. Context logic can represent global schema, loosely coupled, or federated systems. Importantly, it also allows us to incrementally strengthen the representation of an information source. An initial representation can be generated automatically from an information source's schema, and then strengthened over time by adding axioms that make explicit the assumptions behind the schema and translate from the information source's vocabulary into the terms that are shared with other information sources. Clients may query the information source as in a loosely coupled system that provides a uniform query interface to a number of heterogeneous systems. One can strengthen the representation over time by adding axioms that make the assumptions behind the schema explicit and which translate from the information source's vocabulary into the vocabulary shared by other information sources.

Consider the simple relational database specified in Figure 1. It is straightforward to translate the tables of a relational database into assertions in first order logic, as shown in Figure 2: each base table in the database corresponds to a relation in the logic and each database view definition corresponds to an axiom.

$$(\forall\ x,y,z\ \text{product}(x, y, z) \Rightarrow \text{string}(x) \ \&\ \text{integer}(y) \ \&\ \text{integer}(z))$$

$$\text{relation}(\text{product}) \ \&\ \text{arity}(\text{product}, 3) \ \&\ \text{primary-key}(\text{product}, 1)$$

$$(\forall\ x,y,z\ \text{product_type}(x, y) \Rightarrow \text{string}(x) \ \&\ \text{string}(y))$$

$$\text{relation}(\text{product_type}) \ \&\ \text{arity}(\text{product}, 2) \ \&\ \text{primary-key}(\text{product_type}, 1)$$

Figure 2: Representation of the product and product_type schema of Figure 1a in logic.

There are two basic problems with representing a database schema and its contents in logic:

- Attributes may be used polymorphically within a single table. For example, the size attribute in the product database stores the size in whatever unit is appropriate for the particular product. For televisions this may be the number of inches across the diagonal of the screen, for memory chips this may be in kilobytes. A philosophical logician might consider this to be a sloppy representation, but it is typical of the representations that occur in actual databases.
- Values need not have a unique denotation. A single value may be used in a database to mean different things in different tables, tuples, or columns. For example, in the product database of Figure 1b, the number 256 appears in both the size and cost columns. This is not, in itself, an inconsistency. If we want to make the units explicit, however, care must be taken to avoid attributing incompatible properties to 256.

These problems are related, and can be solved with a combination of existential quantification and systematic renaming. For example, we could write an axiom to disambiguate the product relation such as:

$$\forall\ x,y,z\ \text{product}(x, y, z)$$
$$\Leftrightarrow$$
$$(\exists\ y',z'\ \text{product-1}(x, y', z') \ \&\ \text{magnitude}(y', \text{us-dollars}) = y \ \&\ \text{magnitude}(z', \text{natural-size-units}(x)) = z)$$

That is, we could introduce a new relation `product-1` that is similar to product, except that new objects, y' and z', are introduced to represent the size and cost. The magnitude of the cost, y', must then be equal to the number in the database product table. The existential quantification is an important, but straightforward, trick. The renaming, however, is rather clumsy, and it becomes even more awkward once multiple information sources are integrated.

ist(SC1, product_type(x, y))
⇔ **ist**(IS1, product_type(x, y))

ist(SC1,
∃ y',z'
(product(x, y', z')
& magnitude(y', natural-size-units (x))=y
& magnitude(z', us-dollar) = z))
⇔ **ist**(IS1, product(x, y, z)))

ist(SC1,
natural-size-units(x) = bit*1024
⇐ product-type(x, memory-chip))

ist(SC1, natural-size-units(x) = inch
⇐ product-type(x, television))

Figure 3: These axioms lift sentences from the information source context, IS1, into the semantic context, SC1. They make the information source's implicit semantics explicit.

Context logic provides us with a more elegant and powerful mechanism than this clumsy renaming. Context logic [2,13] is an extension of first order logic in which sentences are not simply true, but are true within a context. The key extension is a modality **`ist`**, read "is true", which takes two arguments: a context and a formula.[1] It asserts that the formula is true in the specified context. Contexts are logical individuals and, as such, can be quantified over. Furthermore, it is possible to write axioms that "span" several contexts. These **lifting axioms** provide a very powerful and expressive means of shifting information from one context to another. They can be used to perform renaming, change structure, and make implicit assumptions explicit. Context logic allows us to restate the disambiguation axiom without renaming, as in the second axiom in Figure 3.

We use context logic to solve the problems of representing a single information source in logic, integrating heterogeneous information sources, and representing the semantic distinctions required by a client.

Two contexts are used to represent each information source. The **information source context** is a direct translation of a database schema into logic *without* resolving semantic conflicts, so that the translation can be done automatically. The **semantic context** holds the translation with the semantic conflicts resolved. The lifting axioms that perform the translation from the information source context into the semantic context cannot be automatically generated, because they are making the semantics that were not represented in the database schema explicit. Figure 3 shows lifting axioms to define the semantic context for the product database.

The first axiom simply lifts all `product_type` facts from the information source context into the semantic context. The second axiom lifts tuples from the `product` table into the semantic context, but it disambiguates the meaning of the numbers in the table. Every number in the cost column becomes a quantity whose magnitude, when measured in US dollars, is the original number. Translating the size column is somewhat more complicated because the unit varies with the type of the product. The last two axioms associate a product type with the unit most naturally used to measure its size.

The author of the lifting axioms must choose an appropriate way to represent the intended implicit semantics of the database. In Figure 3, the functions magnitude and natural-size-units were used along with the constants us-dollar, bit, and inch. The decisions required to construct these representations can be subtle. For instance, we have chosen to use a magnitude function that takes two arguments: a quantity and a unit. This is because the unit used to measure a quantity is not an inherent property of the quantity, whereas its dimension is (e.g., the dimension of all prices is currency, but the magnitude of a price can be measured by any unit of currency such as dollars or yen).

Using both an information souce context and a semantic context for each individual information source solves the two representational problems that we outlined.

We also use contexts to integrate multiple heterogeneous information sources. We define integrating contexts after constructing the information source context and semantic context for individual information sources. An **integrating context** contains axioms that lift sentences from several semantic (or integrating) contexts. The context logic is powerful enough to express global schema approaches, in which there is a single integrating context that combines all the semantic contexts, and the federated database approach, in which several integrating contexts are defined, each of which integrates a subset of the contexts. There are other possibilities. A query can combine multiple contexts. One information source's semantic context can be mapped directly to another semantic context without an intermediate integrating context (this is useful when integrating a new database that is similar to an existing one).

If information sources are to be integrated, the representational decisions made when constructing the semantic contexts should be as compatible as possible. This is supported by providing a number of theories or shared ontologies that can be incorporated into a context. The ontology that we used for quantities in

[1] A modality may be a modal operator, but it may also be an operator in the metatheory, or a simple predicate in a logic that reifies formulae.

this example is part of the engineering and math ontology developed in the Ontolingua project at KSL [4,5].

We use contexts to represent the semantic requirements of users. Sciore [15] presents an example in which one stock quoting service provides the latest trading price and another reports the latest closing price. For many purposes the distinction is essential and needs to be disambiguated (as the units were in the product example). For others, however, it may be irrelevant. We implement this by defining a context in which the distinction is not represented.

Think of executing a query as proving a theorem. The proof procedure uses lifting axioms as well as axioms that can be found in the individual contexts to either prove the query directly or to reduce it to a set of lemmas that can be "proven" by issuing queries to the individual information sources. Such a lemma is a sentence that does not mention any context other than that of a single information source. Clearly, there may be many possible proofs of a single query that correspond to alternate query execution plans. There is ample opportunity for optimizing the query (or equivalently finding the least expensive proof). This general approach is also taken by the SIMS project, as well as by Levy [11].

Benefits of Using Context Logic

There are several powerful consequences of using context logic to integrate information sources in the way that we have described including the ability to:

- integrate new information sources incrementally;
- share assumptions among information sources without making them explicit;
- exploit shared ontologies;
- provide a richer model of integration that goes beyond global schema or federated schema methodologies. We discuss each of these points in turn.

The single most important consequence of our approach is that it eliminates most of the up-front cost of integrating a new information source. The cost is reduced because the information source context can be automatically generated from the source's export schema. Once this has been done, it is possible to make queries in the context of the new information source as if it were a loosely coupled heterogeneous database system. The query must be expressed in the vocabulary of the new source, but the syntax and interface are consistent with the old sources. Once the information source context has been established, it is possible to incrementally add lifting axioms to populate the source's semantic context. Furthermore, this incremental integration can be performed in response to perceived and actual usage patterns, rather than expectations about usage. Our approach is in stark contrast to the global schema approach in which the ontology of the new information sources must be completely decontextualized and translated into the existing global schema.

Information sources may share common assumptions without making them explicit. For example, a new information source may exploit commonalties with an existing source in two ways. First, we can copy lifting axioms from the old source into the new source's semantic context. Second, we can write lifting axioms that map from the new source's context into the old source's context. The key point here is that the relations between contexts are much richer than theory inclusion. Lifting axioms may connect sibling contexts and exploit commonalities in numerous ways. This is similar to, but more powerful than, the federated database approach in which the schemas of subsets of the databases known to the federation are combined. Context logic enables shared implicit assumptions to be shared across information sources without the need to first disambiguate them.

The semantic contexts may exploit shared ontologies. An ontology is a set of relations and axioms that attempt to precisely characterize some domain of discourse. Ontologies for domains such as quantities, finance, product descriptions, and so on, will ease the work of integrating information sources. In our formulation, each ontology is defined in its own context. The semantic contexts of information sources may then include the ontology contexts. If the semantic contexts of several information sources share common ontologies, it is much easier to operate across them. This is one way of decomposing the otherwise daunting problem of constructing a global context.

The model we have presented is richer than global schema, federated, or loosely coupled database systems because (1) lifting axioms may be added between any pair of contexts, and (2) a query may explicitly reference several different contexts.

Acknowledgments

This research was supported by a grant from ARPA and NASA Ames Research Center (NAG 2-581). Sasa Buvac and Rupert Brauch provided valuable comments on earlier versions of this paper.

Bibliography

[1] Y. Arens & C. Knoblock. Planning and reformulating queries for semantically-modeled multidatabase systems. *Proceedings of the 1st International Conference on Information and Knowledge Management,* pages 92-101. 1992.

[2] S. Buvac & I. Mason. Propositional logic of context. *Proceedings of the eleventh national conference on artificial intelligence,* 1993.

[3] C. Collet, M. Huhns, & W.-M. Shen. Resource integration using a large knowledge base in carnot. :55-62, 1991.

[4] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In Nicola Guarino, Ed., *International Workshop on Formal Ontology,* Padova, Italy, 1993.

[5] T. R. Gruber. An ontology for engineering mathematics. In Jon Doyle, Piero Torasso, & Erik Sandewall, Ed., *Fourth International Conference on Principles of Knowledge Representation and Reasoning,* Gustav Stresemann Institut, Bonn, Germany, Morgan Kaufmann, 1994.

[6] R. V. Guha. *Contexts: A formalization and some applications.* doctoral dissertation, Stanford University, 1991.

[7] M. Huhns, N. Jacobs, T. Ksiezyk, W. M. Shen, W. Singh, & P. Cannata. Enterprise information modeling and model integration in carnot. *Enterprise Integration Modeling: Proceedings of the first international conference,* MIT Press, 1992.

[8] G. Jakobson, G. Piatetsky-Shapiro, C. Lafond, M. Rajinikanth, & J. Hernandez. CALIDA: a system for integrated retrieval from multiple heterogeneous databases. *Proceedings of the Third International Conference on Data and Knowledge Engineering,* Jerusalem, Israel, pages 3-18. 1988.

[9] T. Landers & R. Rosenberg. An overview of multibase. *Distributed Data Bases*, pages 153-183. North Holland, 1982.

[10] D. B. Lenat, R. V. Guha, K. Pittman, D. Pratt, & M. Shepherd. Cyc: Toward Programs with Common Sense. *Communications of the ACM*, **33**(8):30-49, 1990.

[11] A. Levy, Y. Sagiv, & D. Srivastava. Towards efficient information gathering agents. 1994.

[12] W. Litwin & A. Abdellatif. An overview of the multi-database manipulation language MDSL. **75**(5):621-632, 1987.

[13] J. McCarthy. Notes on formalizing context. *Proceedings of the thirteenth international joint conference on artificial intelligence,* 1993.

[14] J. McCarthy & S. Buvac. Formalizing Context (Expanded Notes). Stanford University, Technical Note STAN-CS-TN-94-13, 1994

[15] E. Sciore, M. Siegel, & A. Rosenthal. Using semantic values to facilitate interoperability among heterogeneous information sources. 1994.

[16] A. Sheth & J. Larson. Federated databases: architectures and integration. **22**(3):182-236, 1990.

[17] M. Siegel, S. Madnick, & A. Gupta. Composite information systems: resolving semantic heterogeneities. *Workshop on information technology systems,* Cambridge, MA, pages 125-140. 1991.