Knowledge Transformations between Frame Systems and RDB Systems

John H. Gennari
Biomedical & Health Informatics
University of Washington
gennari@u.washington.edu

Peter Mork
Computer Science Dep't
University of Washington
pmork@cs.washington.edu

Hao Li
Biomedical & Health Informatics
University of Washington
haoli@u.washington.edu

ABSTRACT
For decades, researchers in knowledge representation (KR) have argued for and against various choices in KR formalisms, such as Rules, Frames, Semantic nets, and Formal logic. In this paper, we present a set of transformations that can be used to move knowledge across two fundamentally different KR formalisms: Frame-based systems and Relational database systems (RDBs). We also describe partial implementations of these transformations for a specific pair of such systems: Protégé and the Postgres RDB system.

Categories and Subject Descriptors
I.2.4 Knowledge Representation Formalisms and Methods – *representation languages*

General Terms
Algorithms, Performance, Design.

Keywords
Knowledge interoperation, knowledge transformation.

INTRODUCTION
Researchers in knowledge representation (KR) have long understood that there is a tradeoff between the representational expressivity of a KR versus the efficiency of reasoning with that KR.[1] However, the great majority of researchers have used this tradeoff simply to explain how their preferred choice along the expressivity continuum was "ideal", or at least "appropriate" for their particular class of problems. In today's world of heterogeneous knowledge sources, it is impractical to suggest that all researchers converge on some "best" or "optimal" uniform KR formalism. It is also quite unsatisfactory to simply say that one KR is more expressive and less tractable than another. Instead, what researchers need are methods to combine and synthesize data and knowledge across KR systems, even in the face of fundamental representational differences.

We have developed a paired set of *transformation rules* that describe how to move data back and forth between a frame-based KR system and a relational database (RDB) system. These rules are a specific example of a more general architecture for defining how to transform data from one KR formalism to another. We use the term "knowledge representation formalism" very broadly, so as to include choices such as XML and relational databases.

We focus our work on a particular pair of KR systems: Protégé, a frame-based system, and Postgres, a relational database system. Protégé is a very well-used Frame-based system that subscribes to the Open Knowledge Base Connectivity (OKBC) meta-model.[2] Postgres is an open source relational database system. Unlike MySQL (a more popular open-source RDB), Postgres provides a fairly complete implementation of the SQL meta-model.

Intuitively, these two sorts of KR systems may seem quite dissimilar; they were certainly designed with very different motivations.[3] Yet, as we will show, one can map at least a portion of data and knowledge from one to the other. Indeed, our belief is that one can transfer *enough*—enough to make it worthwhile for collaborators to communicate even though they have chosen very different KR systems.

TRANSFORMATIONS, PROTÉGÉ → RDBs
At a coarse level, the transformation from Protégé to RDBs is simple: Classes become tables, slots become attributes (columns), and individuals become rows. However, this simple view omits a great deal from the source Protégé model. In fact, because the expressivity of frame-based systems is greater than that of relational databases, transformations in this direction must necessarily lose some information. (A complete and formal specification of these transformations can be found at the Seedpod web site.[4])

Most obviously, our transformations do not fully capture inheritance across classes. If the KB includes parent and child classes, these simply appear as separate tables, and there is no explicit statement that attributes of the parent class should also be attributes of its descendants.

In addition, the OKBC model blurs the strong distinction that a database makes between tables (classes) and tuples (individuals). Thus, in Protégé, one can have "own-slots" that effectively provide arbitrary attribute-value pairs for classes (rather than individuals). In an RDB, one cannot easily annotate the definition of a table in this manner. Similarly, there is no notion of a user-defined metaclass in an RDB. (Although most RDB systems, including Postgres, use a special table that lists all tables in the DB.)

K-CAP'05, October 2–5, 2005, Banff, Alberta, Canada.
ACM 1-59593-163-5/05/0010.

TRANSFORMATIONS, RDBs → PROTÉGÉ

When transforming data from the less-expressive relational formalism to Protégé, we face a different set of challenges. In a sense, when going in this direction, the task is to "reverse-engineer" the underlying logical model, given only the physical model of the relational schema of tables, attributes, and constraints. Our rules in this direction operate roughly in reverse: Tables become classes, attributes become slots, and rows becomes individuals (again, details are available at the Seedpod web site [4]).

Although an RDB system is generally less expressive than an OKBC system, there are some exceptions. For example, modern RDB systems (including Postgres) allow for a number of different sorts of integrity constraints and triggers that may be added to particular columns. For example, if one column (such as a body-mass-index) is dependant on other columns (such as height and weight), this relationship can be expressed as a constraint. In general, such a constraint would be expressed as an *axiom* in an OKBC system. However, although Protégé includes special classes for expressing such axioms, it does not include any built-in way to interpret or enforce axioms. Therefore we have not tried to transform this sort of information.

IMPLEMENTATIONS

There are a variety of ways that our rules could be used by transformation systems. For example, the Protégé-to-RDBs rules could be implemented as an export function within Protégé, or as an import function in some relational DB system. To date, we have developed partial implementations of our transformation rules in both directions, as extensions to the Protégé system.

A portion of our Protégé-to-RDB rules are implemented as part of the Seedpod system, which exports database tables from Protégé.[4] The Seedpod system has been tested with an experiment management system for brain researchers. Seedpod currently does not export any individuals—all of the tables it provides for export are empty.

Likewise, some of our RDB-to-Protégé rules are implemented in the DataGenie Protégé plug-in. (See the Protégé plug-in library at protege.stanford.edu.) In part, the DataGenie plug-in was built in response to the demand for data transformations. Protégé users wanted to build a frame-based ontology and then populate parts of this ontology with individuals from a pre-existing database. DataGenie is an extension (a plug-in) to the base Protégé system that implements most of our RDB-to-Protégé transformation rules so that users can import data from RDBs.

The Protégé system also includes a built-in "convert-project-to-DB" functionality. However, this function is quite different from Seedpod or any application of our transformation rules. In particular, this conversion function saves the entire Protégé KB as an entity-attribute-value (EAV) database. The advantage of this choice is that there is no loss of information. However, the disadvantage is that there is only a single table, and therefore the system cannot provide any efficient indexing or DB querying. In contrast, the tables generated by Seedpod are designed to more closely match how a DB designer would choose to model the information in the Protégé KB.

SUMMARY

Choices in knowledge representation are important. Our architecture and our transformation rules should not be interpreted to mean that researchers can be cavalier about their choice of a KR system. However, our claim is that differences in expressivity can be overcome: Even if lossless transformation across different KRs is impossible, enough information can be retained to make data transformations worthwhile. Our plan is to test this hypothesis in the bio-informatics domain, where there is a strong need and motivation to collaborate and share data, and where there is a wide variety of different KR systems in use.

Our current implementations are not satisfactory. One of the significant advantages of a formal set of rules is that they can be used to deduce what information is lost or altered when moved from one KR to another. Thus, systems like Seedpod or DataGenie that implement these transformations should include user interfaces that make explicit the information that will be lost. Users can then better understand the cost of a particular transformation and they may be able to adjust or modify information so that it can be better captured by the target KR.

More generally, we hope that our sets of rules represent a first step toward a larger library of transformations among a variety of meta-models. Our next steps will be to work with bio-informatics researchers and real-world knowledge and data stores, to validate that our transformations can help make knowledge sharing easier and more transparent.

Acknowledgements

Thanks to Adam Silberfein for work with DataGenie, and additional ideas for RDBs → Protégé transformations. This work was partially funded by a training grant (#1T15LM07441-01) and a Bisti planning grant (#P20 LM007714) from the NLM.

REFERENCES

[1] Levesque HJ and Brachman RJ. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence,* 1987; **3**:78-93.

[2] Gennari JH, Musen MA, Fergerson RW, et al. The evolution of Protégé: An environment for knowledge-based systems development. *International Journal of Human-Computer Studies,* 2003; **58**(1):89-123.

[3] Uschold M and Gruninger M. Ontologies and Semantics for Seamless Connectivity. *SIGMOD Record,* 2004; **33**(4):58-64.

[4] Project Seedpod, at http://sig.biostr.washington.edu/projects/seedpod/