Using a Description Logic to Drive Query Interfaces

Sean Bechhofer and **Carole Goble**
Department of Computer Science
University of Manchester
Oxford Road
Manchester M13 9PL
seanb@cs.man.ac.uk
http://www.cs.man.ac.uk/mig/people/seanb

1 Introduction

Description Logics have long been advocated as a suitable framework for partially structured data. A conceptual model can provide a space within which a user can navigate when constructing queries. In particular, the hierarchical compositional models provided by a Description Logic have the potential to support complex incremental manipulations of query expressions.

However, interacting with a Description Logic is not always easy. Systems generally use textual interfaces, where the user requires not only an understanding of the underlying representation but also its particular concrete syntax. We present some ideas on the use of a Description Logic with the addition of *sanctions* to drive graphical user interfaces facilitating the construction and manipulation of queries. This use of sanctions allows the constrained formulation of queries through a dialogue, with the interface providing feedback relating to both the schema and contents of the database.

We have two current prototypes. The first, developed as part of the TAMBIS project [Tam], allows molecular biologists to construct queries over concepts in biology. These description logic queries are then rewritten to database queries directed to various information sources. The second uses a small demonstration database of people and documents, and provides additional feedback about query results.

2 The GRAIL Description Logic

GRAIL[RBG+97] is a Description Logic developed by the Medical Informatics Group at Manchester University. It has a restricted set of concept formation operators and the addition of a mechanism for constraining the construction of composite concepts, known as *sanctioning*. Sanctioning plays a major part in the process which drives the construction of interfaces.

Sanctioning has two levels. So-called *grammatical* sanctions represent general relationships, while *sensible* sanctions represent those that can really be formed. To use a medical example, we may say that in general, Conditions occur in BodyParts. This is not to say that every condition can occur in every body part, but it is useful to be able to express the abstraction. At a more specific level we can now say that Fractures occur in Bones, allowing the formation of a fracture of a bone, but avoiding misnomers such as fracture of the eyebrow. Compositions which are only sanctioned at the grammatical level cannot be instantiated.

Sanctions are inherited down the subsumption hierarchy, and a grammatical sanction must be in place before a sensible sanction can be asserted.

GRAIL is implemented as a *Terminology Server* [BGA+97], providing access to a range of terminological services and operations. This separation of the core terminological services provides a clean split between the underlying representation and the client applications (such as the interfaces described here) which use it.

3 Data Entry and Query

Much work has been done on the use of GRAIL models to drive data entry interfaces. This work began with PEN & PAD, has evolved from experiments with user interface requirements and now forms part of the latest version of a major computer package for General Practioners[Kir95].

However, such work has focused primarily on data *entry* rather than *query* formulation. The data forms are driven by the sensible sanctions in the model, ensuring that the options available for input correspond to compositions that exist.

When we consider query, however, the more abstract concepts permitted due to the grammatical sanctions are important. Although the concept Condition occuring in a BodyPart is abstract in the sense that it is never directly instantiated, and is thus too general for use in a data entry context, it *does* form the basis of a valid query, as it subsumes concepts (such as Fracture occuring in Femur) which are instantiated.

The issue here is that the two questions:

- What can I *say* about a concept X?

- What can I *ask* about a concept X?

are different. The first question is concerned with the specializations which can actually be built, while the second question needs to be answered at a more abstract level, allowing the use of more general compositions.

3.1 Reasonable Sanctions

In addition to the requirement that queries can be formed at a higher level, we also wish to restrict the user from forming queries that will never be fulfilled.

Consider the case where we have a concept Person with subclasses Student Teacher and Layabout. Person is sanctioned to have the property earns Wage at the grammatical level, representing the fact that in general, person can earn wages. We can now assert sensible sanctions that Student earns Wage and Teacher earns Wage. If query was based solely on grammatical sanctioning, we could now ask the query Layabout which earns Wage, which – due to the lack of sensible sanctions – we know can have no instantiations. If query was based solely on sensible sanctions, we would be unable to form the general level query Person which earns Wage – a useful query.

This suggests that options for composition should be based on a combination of the two levels of sanction – if a composition is sanctioned at the grammatical level, but has no sensible sanctions below it allowing instantiation, the composition should not be offered as an option. This leads to the definition of *reasonable* sanctions, where a concept is reasonably sanctioned if there are sufficient sanctions in place to allow the existence of an instantion of it.

Reasonable sanctions differ from grammatical and sensible sanctions in two ways:

1. They are derived or inferred from the information present in the model
2. They are not inherited. In the discussion above, although earns Wage is reasonable for Person, it is not reasonable for Loafer.

Note also that reasonable sanctions are not part of the GRAIL language, but are a device placed on top of the language and used by applications.

4 Query Manipulation

Once an initial query has been formed, there are a variety of manipulations or reformulations we can perform on the query.

Specialization. Further criteria can be added to the description applied to the topic of the query. For example, a request for *articles about politicians*, could be specialized to a request for *articles appearing in newspapers about politicians*. Alternatively, the topic could be replaced by a more specific subclass;

Generalization. Queries can be relaxed by the removal of criteria or the replacement of the topic. For example we could move from a request for *articles appearing in newspapers about sportsmen* to *articles about sportsmen*;

As well as the *global* manipulations described above, we can perform *local* operations, where the values of criteria can be themselves specialized or generalized. When manipulating subexpressions, a further option becomes available:

Subquery Replacement. We may wish to allow replacement of subqueries with sibling concepts, say moving from *articles about politicians* to *articles about sportsmen.*

All these manipulations and replacements can be controlled by sanctioning, restricting the options presented for specialization/replacement, and ensuring that only reasonable queries are built.

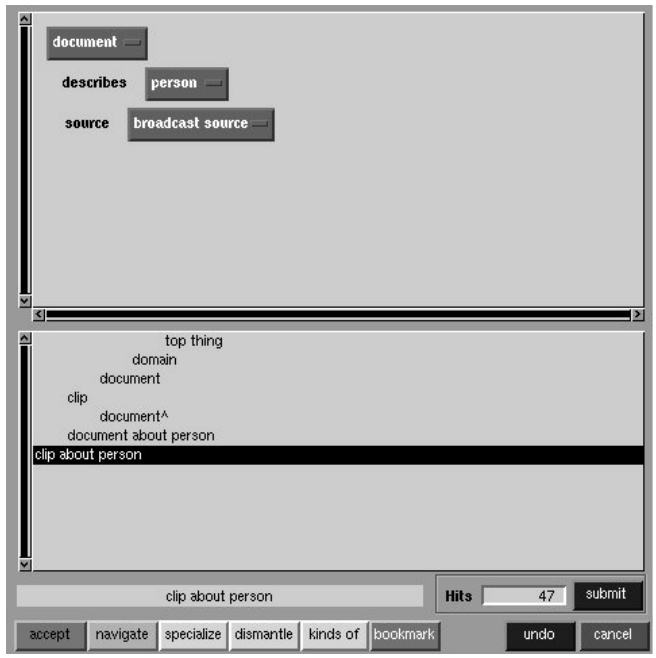

Figure 1: Initial Query

Figures 1, 2, 3 and 4 illustrate a sequence of manipulations performed while building up a query. The first shows a query asking for documents which were broadcast (i.e. were taken from TV or radio) and which describe a person. The second screen shows available options for specializing the kind of person – here we have chosen male politicians, leading to the query shown in the third screen. Finally, we show options available for replacing the source sub-query with a more specific child or sibling. The replacement screen only shows those concepts "next" to the focus. In this way the user navigates

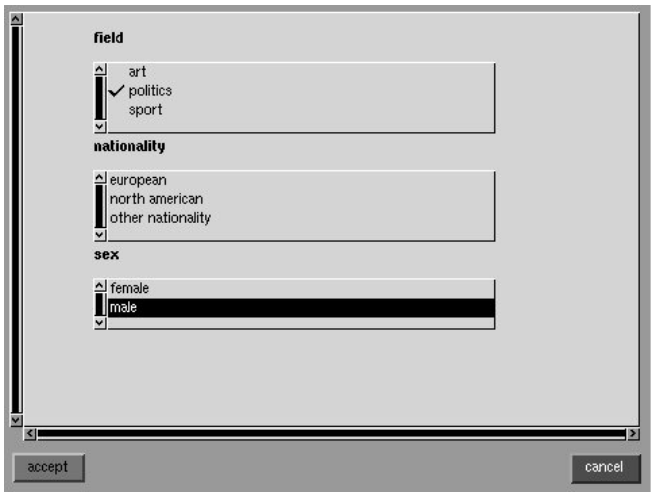

Figure 2: Specialization Options

through the model incrementally.

The interface shows the query broken down into its constituent parts, along with a natural language expression corresponding to the entire expression. A hierarchical view showing the position of the concept is also provided. In addition, the tool shows the number of "hits" found for the current query. Note that this example uses an instance space – the TAMBIS prototype relies only on concept space.

Operations such as replacement or specialization are currently focussed around the concepts involved in a query. GRAIL supports attribute hierarchies which might also be used in such manipulations. For example, we may wish to be able to generalise from a query about *articles about politicians to* articles connected with politicians, where the notion of *connected* subsumes the relationships of being the subject and the author.

5 Discussion

The approach described here provides powerful operations for the construction and manipulation of Description Logic expressions. Initial reaction from the users of our prototypes – molecular biologists with no experience of Description Logics – has been positive, and a formal evaluation of the interface will take place in the near future. However, there are still many areas in need of exploration.

An important aspect of dynamic query is the provision of *feedback* informing the user of the progress of the query and guiding toward the possible actions which can be performed. This can be separated into two levels. At the *data* level, the emphasis is on feedback concerning the answering of the query – in the example above, this feedback is of a primitive nature, simply providing the user with a count of the number of instances to be re-

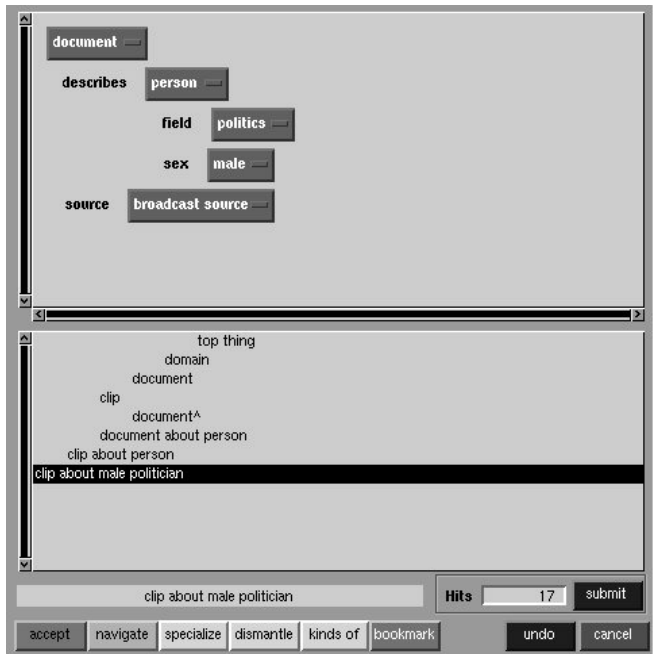

Figure 3: Specialized Query

turned. Such an approach has been used with traditional databases [EFP94]. The IMACS project [BST+93] used a CLASSIC knowledge base to support data mining and knowledge discovery, providing more sophisticated feedback.

Alternatively, we can provide feedback at a *meta* or *schema* level, constraining and guiding the user based on knowledge about the information model – for example offering suitable options for specialization of a query, while preventing the formation of queries about vegetables which have a political allegiance.

The sanctions described in this paper allow us to apply this schema level feedback, while the description logic allows us to provide data level feedback, using the classification of the instance space.

Sanctions can be seen to provide domain and range restrictions for relations, with the domain and range being defined by the disjunction of the sanctioning statements. By defining a broad and narrow domain and range for each role, we can provide more control and different constraints for different applications. The definition of a clear semantics for sanctioning is currently being investigated – [Sch96] describes a model theoretic semantics for sanctions.

The feedback provided sits well with the four maxims of Grice [Gri78]. The maxim of *Relevance* states that the interface should provide relevant contributions, while that of *Quantity* states that contributions should be as informative as is required for the current purposes of the exchange and should not be more informative than is re-

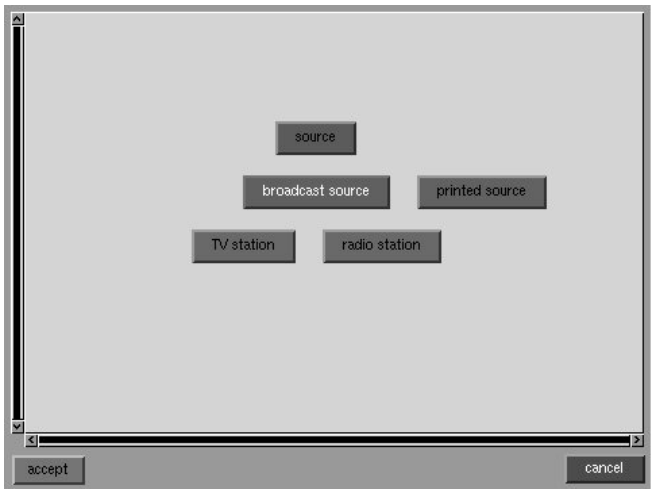

Figure 4: Replacement Options

quired. By restricting, for example, replacement options, the interface does not deluge the user with spurious options when offering alternatives.

Another common technique used for database query is that of Query-by-Example [Zlo75], or retrieval by instantiation [TWF+82], where a particular instance is presented as a representative of a class of instances which the user is interested in. In a traditional form-based approach, the properties of the example would be used to fill in the particular values of the form. By using the described instances of a description logic, we can offer a more powerful form of query by example, where the *description* applied to the instance can be used as the starting point for a query. In a sense, the description applied to the instance provides not only the values instantiating form, but also the structure of the form itself.

In tandem with the construction and manipulation of expressions, there are activities involving navigation around the conceptual model and navigation of the concept space. In addition to the manipulations described above, users may wish to "jump" to different areas of the model. In particular, entry points need to be provided. These may either be pre-defined, user-defined "bookmarks" can be supported, or a query-by-example approach can be used to gain access to starting points from which to explore.

The interaction of the reasonable sanctions along with manipulation operations poses some interesting problems. When subqueries are replaced, other parts of a query may go out of scope or become unsanctioned. Techniques are required to manage this interaction and report to the user when manipulation has side effects.

In some situations, the conceptual model may have been constructed for purposes other than driving query. In addition, users may have no experience or interest in the underlying representation. If the ontology is very deep or broad, in order to comply with the maxim of *Quantity*, there might be extra levels of abstraction or detail which we wish to hide from the end user. Such situations will vary depending on the application area. Annotation of the model with application specific metadata may help hide these abstractions and implementation details from the users' view of the model.

Our current prototype contains a naïve implementation of instances – with a large space of instances, completeness may be difficult to achieve. Can we employ a controlled amount of incompleteness in order to provide a useable system which supports aspects of both data and schema level feedback?

In summary, we feel that more intuitive methods of interacting with Description Logic systems are required. The ideas presented in this paper can go some way towards achieving this, but further investigation is required into:

- schema level user feedback during query construction, based on sanctioning information;
- data level feedback based on terminological reasoning and the completeness of that reasoning;
- the use of exemplars and query by example;
- annotation of models with metadata to aid in application specific tailoring.

References

[BGA+97] S.K Bechhofer, C.A. Goble, Rector A.L., W.D. Solomon, and W.A. Nowlan. Terminologies and Terminology Servers for Information Environments. In *to appear in: Proceedings of STEP '97 Software Technology and Engineering Practice*, 1997.

[BST+93] R.J. Brachman, P.G. Selfridge, L.G. Terveen, B. Altman, A. Borgida, F. Halper, T. Kirk, A. Lazar, D.L. McGuinness, and L.A. Renick. Integrated Support for Data Archaeology. *International Journal of Applied and Cooperative Information Systems*, 2(2):159–185, 1993.

[EFP94] G.P. Ellis, J.E. Finlay, and A.S. Pollitt. HI-BROWSE for Hotels: Bridging the Gap Between User and System Views of a Database. In P. Sawyer, editor, *Proceedings of IDS 2*. Springer-Verlag, 1994.

[Gri78] H. P. Grice. Logic and conversation. In P. Cole, editor, *Syntax and Semantics 9:Pragmatics*. Academic Press, 1978.

[Kir95] J. Kirby. PEN & PAD: The Next Generation. In *Primary Health Care Specialist Group*, Cambridge, UK, 1995.

[RBG+97] A. L. Rector, S. K. Bechhofer, C. A. Goble, I. Horrocks, W. A. Nowlan, and W. D. Solomon. The GRAIL Concept Modelling Language for Medical Terminology. *Artificial Intelligence in Medicine*, (9):139–171, 1997.

[Sch96] Dominik Schoop. Integrating qualitative spatial knowledge representation with description logics in order to reason about human anatomy. Continuation report, University of Manchester, Department of Computer Science, 1996.

[Tam] Tambis Project. WWW Home Page. `http://www.cs.man.ac.uk/mig/tambis`.

[TWF+82] Frederich N. Tou, Michael D. Williams, Richard Fikes, Austin Henderson, and Thomas Malone. RABBIT: An Intelligent Database Assistant. In *Proceedings of AAAI-82*, pages 314–318, 1982.

[Zlo75] M. M. Zloof. Query-by-Example. In *Proceedings of the National Computer Conference*, pages 431–438, Montvale, NJ, 1975. AFIPS Press.