TAMBIS - Transparent Access to Multiple Bioinformatics Information Sources.

Patricia G. Baker[a], Andy Brass[a], Sean Bechhofer[b], Carole Goble[b], Norman Paton[b], Robert Stevens[b].

[a]School of Biological Sciences,
Stopford Building,
University of Manchester,
Oxford Road,
Manchester, M13 9PT
U.K.
Telephone: 44 (161) 275 2000
Fax: 44 (161) 275 5082

pbaker@manchester.ac.uk
abrass@manchester.ac.uk
seanb@cs.man.ac.uk

[b]Department of Computer Science,
University of Manchester,
Oxford Road,
Manchester, M13 9PT
U.K.
Telephone: 44 (161) 275 6142
Fax: 44 (161) 275 6236

carole@cs.man.ac.uk
norm@cs.man.ac.uk
stevensr@cs.man.ac.uk

Abstract

The TAMBIS project aims to provide transparent access to disparate biological databases and analysis tools, enabling users to utilize a wide range of resources with the minimum of effort. A prototype system has been developed that includes a knowledge base of biological terminology (the biological Concept Model), a model of the underlying data sources (the Source Model) and a 'knowledge-driven' user interface. Biological concepts are captured in the knowledge base using a description logic called GRAIL. The Concept Model provides the user with the concepts necessary to construct a wide range of multiple-source queries, and the user interface provides a flexible means of constructing and manipulating those queries. The Source Model provides a description of the underlying sources and mappings between terms used in the sources and terms in the biological Concept Model. The Concept Model and Source Model provide a level of indirection that shields the user from source details, providing a high level of source transparency. Source independent, declarative queries formed from terms in the Concept Model are transformed into a set of source dependent, executable procedures. Query formulation, translation and execution is demonstrated using a working example.

Introduction

The biological community is a distributed one with a culture of sharing and rapid dissemination of information. Each separate area of molecular biology generates its own data and therefore its own information sources, including those for protein sequences, genome projects, DNA sequences, protein structures and motifs. Also available are a range of specialist interrogation and analysis tools, each typically associated with a particular database format. Frequently the information sources have different structures, content and query languages; the tools have no common user interface and often only work on a limited subset of the data.

When biologists need to ask questions of multiple sources they must perform the following tasks during query formulation and execution:

- identify sources and their locations
- identify the content/function of sources
- recognise components of a query and target them to appropriate sources in the optimal order
- communicate with sources
- transform data between source formats
- express syntactically complex queries and
- merge results from different sources.

Many biologists still use collections of stand-alone resources (many of which are Web-based) to formulate and execute queries. This means that all of the tasks listed above must be carried out by the user. This places a burden on biologists, most of whom are not Bioinformatics experts, and limits the use that can be made of the available information. The greater the number of the above tasks that are taken on by the system, the greater the transparency of the overall task of query formulation and execution. There are examples in the

biological community of systems which seek to relieve the user of some of this burden by easing access to multiple, heterogeneous information sources; however, these systems vary in their degree of transparency.

The Sequence Retrieval System (SRS) (Etzold 1996), for example, attempts source interoperation using *predefined* hypertext links with which the user can navigate between sources. A form-based interface allows the user to ask complex, although restricted, queries over multiple sources that are executed simultaneously. Queries composed of sub-queries, which have to be executed in a given order, must be issued separately by the user, and the results of one sub-query piped into another by hand. While SRS provides the user with transparency from *communication* (i.e. location, connection protocols and query language) with sources, it does not hide them, and provides no guidance as to which source is most appropriate for a given query.

The Collection Programming Language (CPL) is a functional programming language that allows data to be described and manipulated as complex data types such as sets, lists and records. These data types are suitable for modelling biological data and have been used to do so in the BioKliesli system (Buneman 1995) where biological sources are given a CPL driver and a set of functions to manipulate data. CPL/BioKliesli thus acts as a multidatabase language and provides ways of manipulating and piping results, allowing the user to formulate complex, *ad hoc* queries. The details of location and access to these data sources are hidden; however, the identification of which data source to use and the construction of the query in CPL is still left to the user. Comparable facilities are provided by P/FDM (Kemp 1996), which also uses a functional language, although P/FDM has a more object-oriented type system and has its own local database.

(Markowitz 1995) uses an object model, the OPM, as a common data model for the sources and a suite of OPM-based tools for exploring them. Each source either has an OPM schema or is retro-fitted with one via a view mechanism. A multidatabase directory describes how each database is linked to another. However, there is no attempt at hiding the databases from the user, who is still expected to identify them and navigate through them. Queries can be specified via a multidatabase query language OPM-QL, or using a Web interface.

The Transparent Access to Multiple Bioinformatics Information Sources (TAMBIS) project aims to provide the user with the maximum source transparency using (i) a canonical representation of biological terminology against which the user can formulate queries and (ii) mappings from terms in the representation onto terms in external sources. TAMBIS therefore provides a level of indirection between the user and the external sources which removes from the user the necessity to perform the tasks listed above. In order to do this TAMBIS adopts the three layer model of the classical mediator/wrapper architecture (Wiederhold 1992), a schematic view of which is illustrated in figure 1.

Layer 1 comprises a knowledge base (conceptual model[a]) of biological terminology and a knowledge-driven user interface. Using the interface, the user combines terms from the knowledge base to form declarative, source independent queries. Layer 2 is a mediation layer that (i) identifies the appropriate sources to satisfy a query and (ii) rewrites the query to a series of source dependent ordered procedures. Layer 3 comprises external sources wrapped with a consistent structural model, providing a common interface that affords communication and network transparency. Where possible TAMBIS exploits existing technologies. Layer 3, therefore, currently utilizes CPL/BioKliesli although the long-term intention is to use CORBA wrapped services (Roduerigez-Tome 1997).

The conceptual model is central to the TAMBIS architecture. Its use in driving query formulation and facilitating source integration, is novel in the biological domain. The emphasis on the model in this paper is, therefore, commensurate with its importance.

The Architecture

Although a detailed description of the TAMBIS architecture is outside the scope of this paper, a general overview is appropriate. The five main components of the TAMBIS architecture are:

- The biological Concept Model (knowledge base)
- The knowledge-driven graphical user interface (GUI)
- The Source Model
- The Query Transformation Module
- The Query Execution Module

The Biological Concept Model

Some bioinformatics researchers recognise that semantic schema and data matching would be greatly aided by a comprehensive thesaurus of terms (Davidson 1995) or a reference ontology of biological concepts (Karp 1995). In

[a] Because the biological knowledge base is a *conceptual* model of biological terminology, the words ‘concept’ and ‘term’ are used interchangeably in this paper.

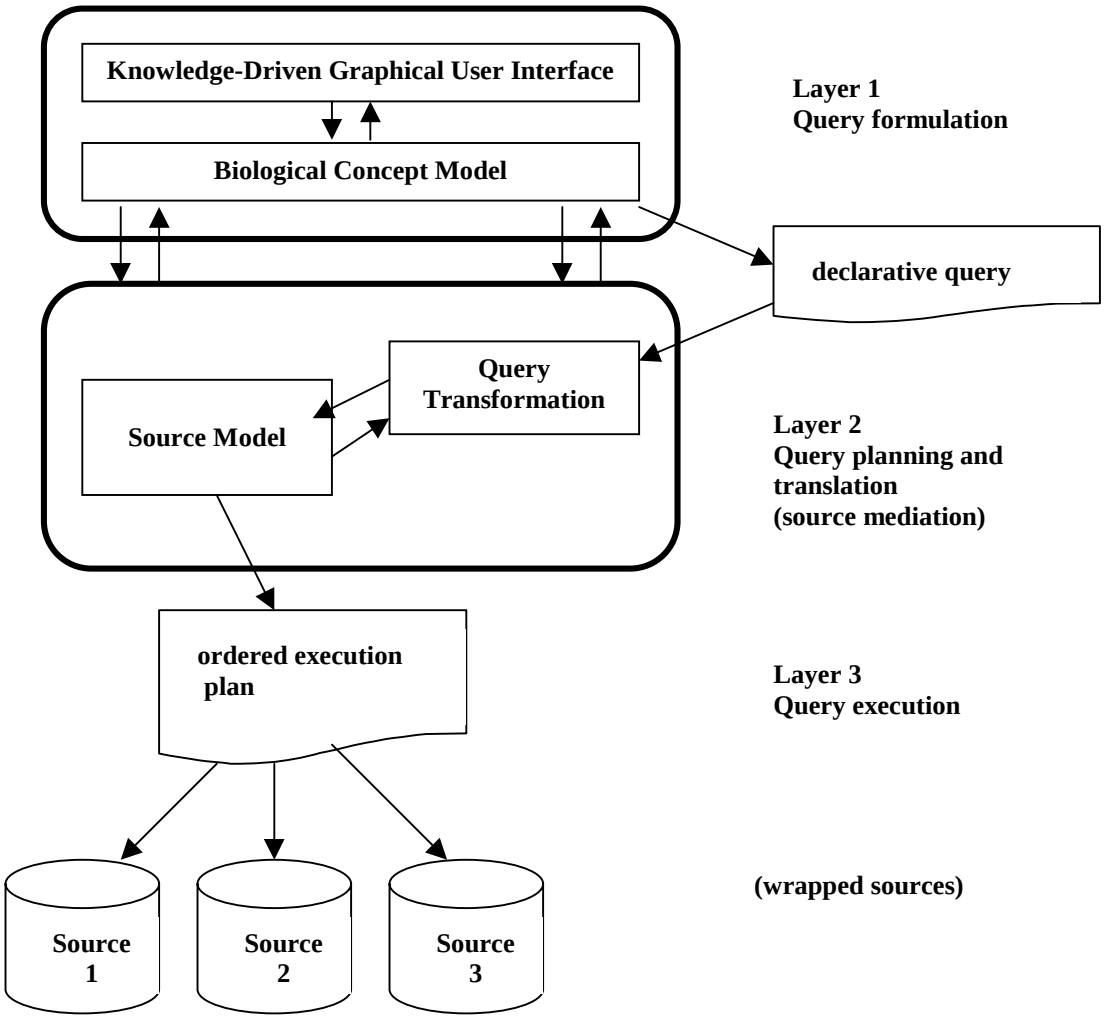


Figure 1. TAMBIS three layer, mediator/wrapper architecture.

order to share standardised and unambiguous information, controlled vocabularies, or terminologies, can be used as a framework for expressing and communicating ideas in a consistent manner. The TAMBIS biological Concept Model describes such a terminology. This knowledge base covers terms associated with proteins and nucleic acids, their component parts and their structures, biological functions and processes, tissues and taxonomy. The terminology has two key aspects:

- it is compositional, resembling a dictionary of elementary terms that are assembled according to a restricted grammar to form new complex composite terms. These composite terms can in turn be components in new compositional terms, so the terminology is *recursive*. For example, the term 'Motif' can be combined with the terms 'isComponentOf' and 'Protein', to create a new composite term 'Motif which isComponentOf Protein'. This in turn could be combined with the terms 'hasFunction' and 'Hydrolase' to form a composite term 'Motif which isComponentOf Protein and hasFunction Hydrolase'; this term is both a *concept* and a *query*.
- it is a classification scheme that organises terms into a hierarchy based on the 'isa' relationship (also known as the subsumption relationship). For example, ProteinSequence 'isa' more specialised kind of Sequence.

To be truly effective, such a terminology needs to be represented in a scheme that can reason about the inferred relationships between terms and their components, can control the formation of terms, and can automatically classify terms based on their components so that the hierarchy takes care of itself. As terms are changed the scheme should also dynamically reclassify them to ensure the hierarchy's correctness.

Description Logics (DL), also known as Terminology

Logics, are a family of logics explicitly designed to represent taxonomic and conceptual knowledge of an application domain on an abstract level; for an overview see (Borgida 1995). DLs are usually given a Tarski style declarative semantics, which allows them to be seen as sub-languages of first order predicate logic. In the TAMBIS project we use the GRAIL DL (Rector 1996), developed at Manchester. Briefly, a DL is an 'isa'-based classification system that allows a recursive, compositional model to be built from terms and binary relations. A base term can be combined with any number of relation-term pairs (or *criteria*) to create a more complex term. Any of these terms can be composite (complex) or elementary. Figure 2 gives a small fragment of the GRAIL classification, omitting the term constructors. In this example 'Motif' is the base term and 'isComponentOf Protein' is the criterion with which it is combined. GRAIL supports the automatic classification of concepts into 'isa' hierarchies by reasoning about the component descriptions of the concepts. Therefore, 'Protein Motif' would be classified automatically as a child of 'Motif' and a parent of 'Poecilia Reticulata Protein Motif' based on its definition. Only 3 of the 11 'isa' relationships shown in figure 2 have been hand-crafted by the knowledge modeller. DLs support multi-dimensional classification so that the same concept can be classified in many ways, thus allowing for the different user views of a concept. The classification is dynamic so as the description of a concept is further elaborated it is *automatically* reclassified. Description Logics therefore support the incremental description of terms.

The classification hierarchy supports imprecise and general queries and query exploration by moving around the hierarchy. The compositional nature of the representation allows for the flexible construction of queries at varying levels of complexity and abstraction. In DLs the modelling language and the query language are the same thing; to find the concept you define it and the classifier classifies it. If it is sound then it is positioned in the hierarchy and you can ask for its parent, children or the instances it describes. If it is unsound then it doesn't classify and, therefore, cannot appear in a query.

A whole family of knowledge representation systems have been built using DLs and recent work has provided a sound formal basis for several DLs along with results concerning their complexity (Donini 1991). Significantly large models are now being produced, for example the Galen-In-Use medical model (Rector 1997) expressed in GRAIL is some 10,000 concepts and relations.

DLs are expressive, and usually have complete and decidable reasoning. However, the conflict when applying any DL is between computational tractability and expressiveness; GRAILs terminological language is less expressive than most other DLs but it compensates for this by supporting a powerful set of assertion axioms and a multi-layer sanctioning mechanism. These sanctions decree whether two concepts are permitted to be related via some relationship and so constrain the construction of complex concepts. Sanctions ensure that only semantically valid concepts are formed and that a large number of complex concepts can be inferred from a sparse model. As only reasonable concepts can be inferred from the model the user is allowed to construct only those queries that it is reasonable to ask. For example, in figure 2, asserting that 'SequenceComponent isComponentOf Protein' is legal, is sufficient to infer that 'Motif isComponentOf Protein' without having to create it or position it until it is asked for. Therefore, only a small number of constraints need be asserted in order that a large number of concepts can be inferred.

In TAMBIS the biological Concept Model is used to:

- describe the metadata of the underlying data sources, representing an over-arching universal schema
- express queries in the modelling language
- drive a GUI user interface for query formulation
- mediate between the various data sources by exploiting the biological concept hierarchy to assist in the identification and resolution of equivalences or *near equivalences* – similar approaches have been taken in non-biological projects, for example SIMS [Arens93].

As (Markowitz 1995) and (Davidson 1995) suggest, integration is costly and the quest for an agreed schema futile. However, our biological terminology does not attempt to force a global schema representing a consistent integrated view of all the component databases. Instead it seeks to describe what is in the component databases and, rather than resolve conflicts, it acknowledges them and indicates possible equivalences.

The Knowledge-Driven User Interface

Queries are formulated against the biological concept model in the GRAIL language. It would be inappropriate for biologists to learn either GRAIL or the contents of the knowledge base. Instead, TAMBIS provides a forms based GUI that is driven by the terminological model. The interface supports two tasks:

- exposure of the terminological model and
- guided query formulation and manipulation.

During the query formulation process the model may be browsed to find what can sensibly be said of a concept of interest. A convenient mechanism for browsing the model without query formulation is provided by the navigation

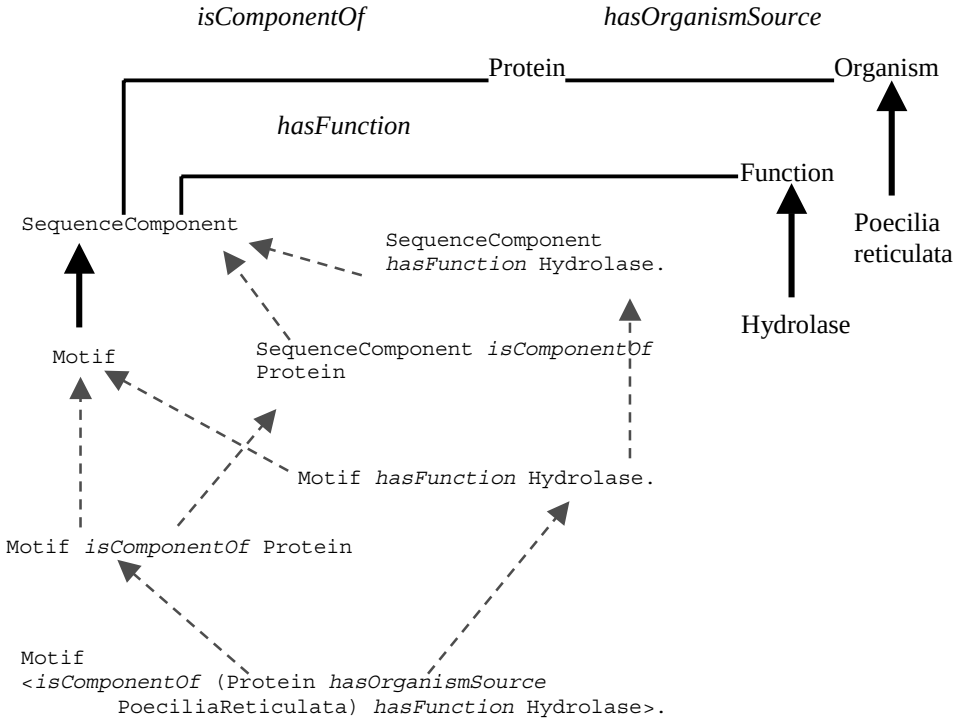


Figure 2. A simplified fragment of the TAMBIS GRAIL model showing the power of auto-classification; the only 'isa' relationships that have been 'hand-crafted' by the knowledge worker are indicated by the solid arrows. All the other terms are implied by the sanctioning scheme and automatically and dynamically classified upon request, as indicated by the broken arrows. The solid lines indicate the sanctioned relationships between terms. It is these relationships that allow the construction of all of the composite terms shown.

tool. Figure 3 shows the navigator focused on the concept 'Protein Structure'. The concept currently in focus occupies the center of the frame and related concepts from the Knowledge Base are displayed around it. The model may be browsed by promoting any of the related concepts to be the central concept. The new central concept is then surrounded by all *its* related concepts.

Having identified a concept of interest, for example 'motif', the user may want to form a query based on that concept. A Query Manipulation tool gives the user an option to add more information about the concept (or *specialise* the concept) by presenting all the legitimate criteria that can be applied to the concept 'motif' (see figure 4).

The user may choose one or many of these criteria. If they chose, for example, 'isComponentOf Protein', the query is equivalent to the English expression "find all protein motifs". Having constructed the query the user may manipulate the whole query or any of its component sub-queries by (i) the addition or removal criteria or (ii) the replacement of terms with more specialized or more general terms. Figure 5a shows a query that has been built by further specialisation of the term 'Protein' in the above query by addition of the criterion 'hasOrganismSource PoeciliaReticulata'. The query is equivalent to the English expression "find all motifs occurring in guppy proteins".

It is important to appreciate that in TAMBIS the term concept is interchangeable with the term query. Therefore, in constructing a concept (a description of what you the user wants) the user is constructing a query ("what things exist that fit the description I have just given?").

Query Planning and Translation

Queries expressed in GRAIL are declarative and source independent. GRAIL queries thus specify what information is required, but neither how it should be obtained nor from where. It is the role of the query planning and translation layer to provide this additional information. This layer takes as input a GRAIL query and generates as output an execution plan in CPL. The planning and translation process is broken into three main steps:

- *Translation into a Query Internal Form (QIF):* The GRAIL query is unnested and certain query constructs are simplified.
- *Query Planning:* A search algorithm considers alternative evaluation orders for the components of

the QIF generated at step 1, with a view to identifying both valid and efficient ways of evaluating the query.

- *Code Generation:* The query plan that results from the planning phase is converted into a CPL program for execution.

The following subsections elaborate on the above steps, both detailing what is done at each stage and outlining the auxiliary data structures that are required.

Translation into Query Internal Form (QIF). GRAIL queries are intrinsically nested structures. However, nested language structures generally imply some evaluation order, so we follow a number of earlier query planners in unnesting the source query prior to query optimisation (Paton 1990, Fegaras 1997). The QIF is a list of query components, each of which is a tuple *(Base, Variable, Criteria, Cost, Cardinality)* representing the evaluation of part of the query. *Base* is the base concept of the component, *Variable* is the name of the variable used to store values retrieved as a result of evaluation the component, *Criteria* represents the set of criteria associated with *Base*, *Cost* is an estimate of the cost of evaluating the component, and *Cardinality* is the size of the collection that it is anticipated will result from evaluating the component. Values for *Cost* and *Cardinality* are computed by the planner. Figure 5a shows an example query that is equivalent to the English query "find all motifs in Poecilia reticulata (guppy) proteins". The GRAIL representation

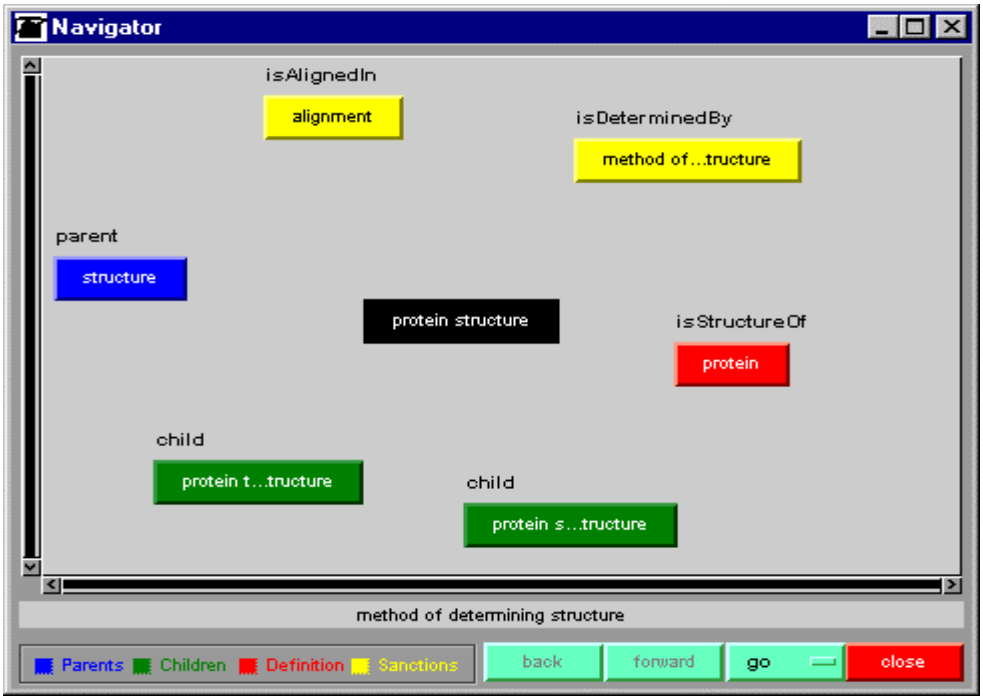


Figure 3. TAMBIS prototype user interface navigation tool showing the navigation of the concept 'Protein Structure'. The central term is surrounded by related terms. Each related term is coloured according to its relationship with the central term. There are four possible relationships: parent terms - concepts *immediately* above it in the hierarchy with which it has an 'isa' relationship e.g. 'Structure'; child terms - concepts *immediately* below it in the hierarchy which have 'isa' relationships with it e.g. 'Protein Tertiary Structure'; defining terms – relation-term pairs that form part of its definition eg. 'is structure of Protein'; sanctioned terms - concepts with which it has appropriately sanctioned relationships but which do not form part of the concept's definition eg. 'is determined by Method of Determining Structure'.

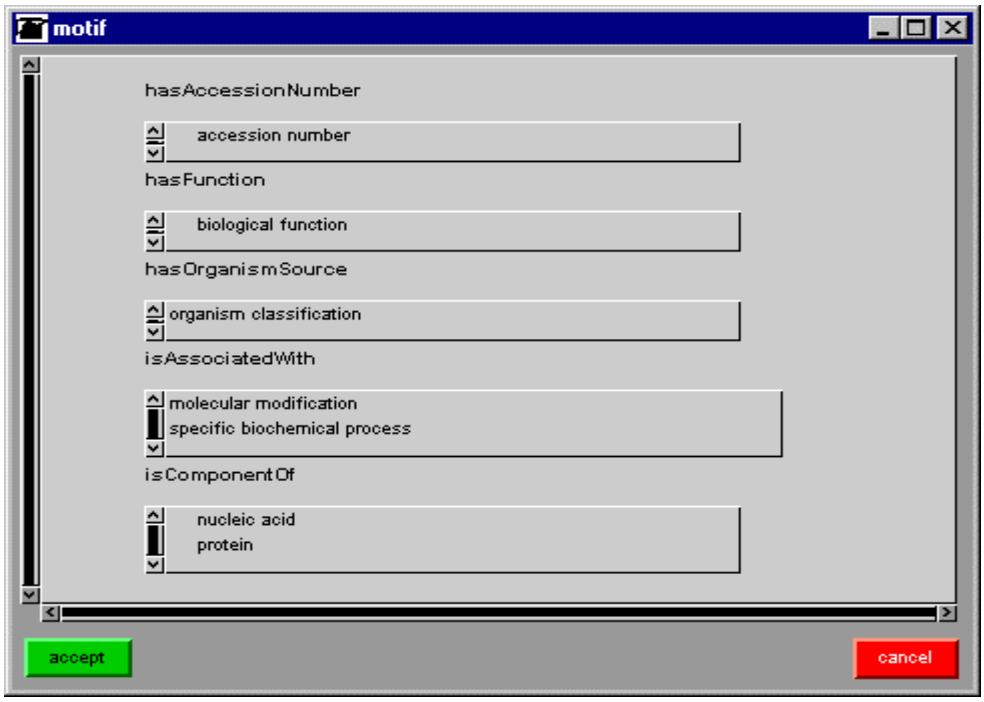


Figure 4. An example from the TAMBIS user interface prototype showing the relationships that can be used to specialise the concept of 'motif'.

of this query is "Motif which isComponentOf (Protein which hasOrganismSource PoeciliaReticulata)" (figure 5b). The initial QIF of this query is shown in figure 5c. Each term (concept) in a criterion is itself represented by a query component and is associated with the variable used to store instances that result from the evaluation of the component. The other form of mapping that takes place during the translation to QIF is the simplification of components where appropriate – for example the removal of query components which exist only to support certain modelling strategies employed by the knowledge worker. The mapping into the QIF is defined as a set of rewrite rules of the form:

```
rewrite <concept template>
as <QIF component>
if <condition>
```

The *concept template* is capable of matching concepts with specific structures in the biological Concept Model, the *QIF component* is as described above, and the *condition* makes tests involving the Concept Model and the Source Model. However, conditions never refer to the specific functions that may be used to evaluate a query, as planning is the sole preserve of the planner described below.

Query Planning. The query planner seeks to identify both legal and efficient ways of evaluating queries given the available CPL functions. The planner exploits the augmentation heuristic (Swami 1989), which essentially involves examining all the query components in a query, selecting the most promising for initial evaluation, and repeating the process for the remaining components. The Source Model is central to the planning process, as it indicates which CPL functions can be used to evaluate which query components. Lack of space prevents a detailed description of the Source Model, but the following are the principle components:

- *Concept Iteration*: Concept iteration information is a *triple (Concept, FunctionSignature, ArgumentMapping*), where *Concept* is a concept from the Concept Model, *FunctionSignature* is the signature of a CPL function, and *ArgumentMapping* is a description of how input parameters for the CPL function should be obtained.
- *Criterion Evaluation*: Criterion evaluation information indicates how the criteria of a concept can be evaluated in CPL. This is described using tuples of the form *(Concept, Criterion, FunctionSignature, ArgumentMapping*), where *Concept* is the base concept to which the criterion is applied, *Criterion* is the criterion in question and *FunctionSignature* and *ArgumentMapping* are as described for concept iteration.

- *Coercion*: CPL functions may retrieve values of different CPL types to represent the same concept. For example, retrieval of protein information from a specialist protein database such as SWISSPROT yields a complex record structure that contains significant amounts of information about the protein. Retrieval of information from a motif database such as PROSITE, however, is likely to yield only the accession numbers. This means that the query planner needs to know things like how to obtain a detailed description from an accession number and *vice versa*. Such relationships are described using tuples of the form: (*CPL_type, CPL_type, mapping_function*)
- *Costing:* Information on the anticipated cost of evaluating a CPL function and the likely cardinality of the result is stored using tuples of the form: (*FunctionSignature, Cost, Cardinality*).

The planner has two principle components, the search algorithm described at the start of this sub-section and a list of rules that indicate under which circumstances specific techniques may be used to evaluate a query component. Such rules are of the form:

```
Rewrite <QIF Component>
as <Function List, Cost, Cardinality>
if <condition>
given <variables>
```

The *QIF Component* is as described above, *the Function List* is a list of CLP functions with bound arguments, the *Cost* is an estimate of how long it will take to evaluate each of the functions in the *Function List*, and the *Cardinality* is the total number of concepts given the set of bound *variables*. The condition invariably refers to the functions that are available in the Source Model and the set of bound variables.

Code Generation. The code generator takes as input an ordered list of query components and their associated functions, and generates a single CPL program that binds together the CPL functions. The code generator is straightforward, and makes a single pass through its inputs in generating the execution plan (figure 5d).
For result presentation, TAMBIS makes use of a CPL function that transforms its data structures into HTML for display using a WWW browser (figure 5e).

Project Status

The prototype Biological Model is well populated by concepts describing those areas required for the construction of common queries, such as queries about protein structure and nucleic acid coding signals. The model currently contains around 1500 concepts and has the capability to infer many more. The biological concept model will become better populated as the prototype system is used to elicit user requirements. The prototype user interface is currently implemented in SmallTalk. It has much of the functionality that it is envisaged will be needed in the final system, although the look and behaviour of the interface is likely to change as the final implementation will be in Java to facilitate its use on the World Wide Web. We are currently eliciting general user requirements from academia and industry by means of a questionnaire. This is ensuring that the Concepts Model allows the formulation of the kinds of questions that biologists want to ask. No formal user evaluation of the prototype knowledge-driven user interface has yet been performed, although it is envisaged to play a major part in the development of the system. A Java implementation of the query transformation module is in place, although its accuracy has not yet been evaluated. A more sophisticated planner will be needed in the future. There are currently 15 wrapped sources including the BLAST suite of programs, SwissProt, Prosite, BLOCKS amd PRINTS. The Source Model has mappings between a range of CPL functions acting on these sources and the corresponding concepts in the biological model. These mappings dictate the number of queries that can be answered by TAMBIS and so the development of a more comprehensive Source Model is the next priority task. As suggested by (Davidson 1995), this approach is high cost but high benefit, and there are still many challenges to address – issues such as: tools for adding new sources; changes in sources; incorporation of CORBA sources; dynamic query optimisation based on network performance; user intervention and results attribution.

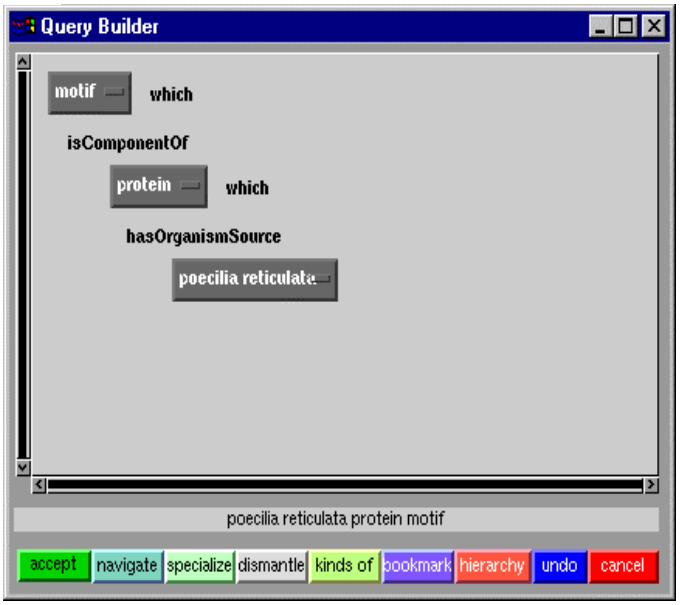
Summary

The TAMBIS project is pursuing a novel approach that will yield an integrated solution to the problem of disparate biological databases and analysis tools. The common schema (Biological Knowledge Base) is represented in a Description Logic, presenting the user with a rich description of the domain from which they may flexibly and intuitively construct and modify queries. The queries are deconstructed, rewritten into a common query language and dispatched to one or more wrapped resources. The use of a knowledge base and wrapped resources removes the need for the user to know (i) which are the appropriate resources to use and (ii) how to access them, thus greatly reducing the time taken to analyze their data.

Acknowledgements

The TAMBIS project is funded jointly by the EPSRC/BBSRC Bioinformatics Programme and by Zeneca Pharmaceuticals, whose support we are pleased to acknowledge.

a)



b)

Motif which isComponentOf (Protein which hasOrganismSource PoeciliaReticulata)

c)

[(Motif, Motif-1, [(isComponentOf Protein, Protein-1)], -1, 1),
(Protein, Protein-1, [(hasSourceOrganism PoeciliaReticulata, null)], -1, -1)]

d)

{Motif-1|
\Protein-1<-get-sp-entry-by-os("POECILIA+RETICULATA"),
Motif-1<-do-prosite-scan-by-entry-rec(Protein-1)}

e)

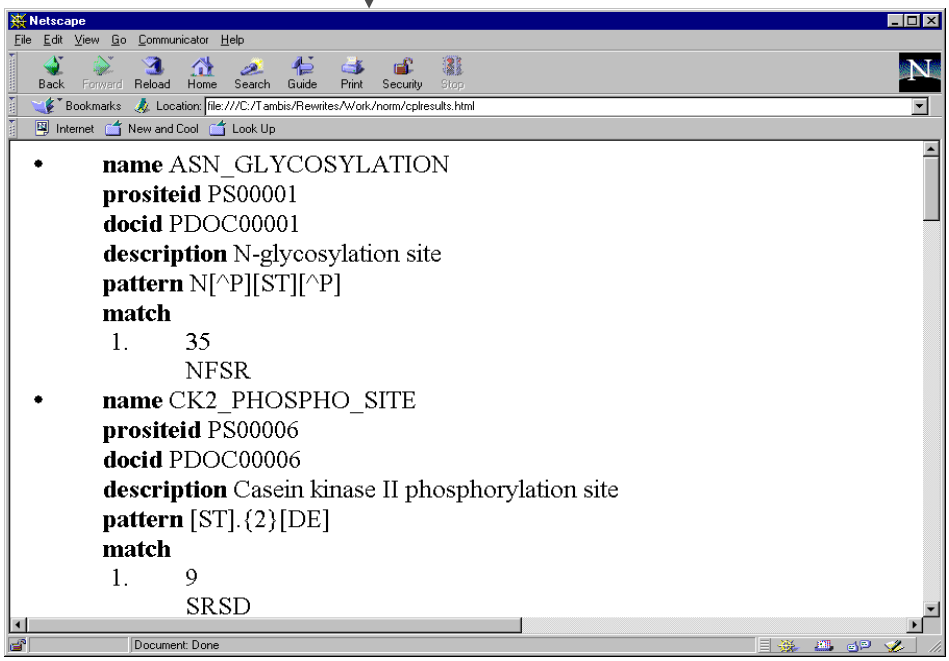


Figure 5. An example showing the stages in the information retrieval process using TAMBIS. **a)** The knowledge-driven GUI allows the user to construct a declarative, conceptual and source independent query. The query formulated at the interface is represented in GRAIL as shown in **b)**. **c)** The single GRAIL query is transformed into query internal form (QIF). **d)** The QIF is transformed into a functional, source-dependent query in CPL. **e)** The results from the CPL wrapped sources are presented to the user via a Web browser.

References

Arens Y, Chee C.Y., Hsu C-H, Knoblock C.A. Retrieving and Integrating Data from Multiple Information Sources, in Journal on Intelligent and Cooperative Information Systems, 2:127-158,1993.

Borgida A., Description Logics in Data Management. IEEE Transactions on Knowledge and Data Engineering, 7(5): 671-682, 1995.

Buneman P., Davidson S.B., Hart K., Overton C. and Wong L. A Data Transformation System for Biological Data Sources In Proceedings of VLDB, Sept. 1995 (Zurich, Switzerland).

Davidson S.B., Overton C., Buneman P., Challenges in Integrating Biological Data Sources, Journal of Computational Biology Vol 2, No 4, 1995.

Donini, F., Lenzerini, M., Nardi, D., Nutt, W., 'The Complexity of
Concept Languages', KR-91, pp151-162, 1991.

Etzold T, Ulyanov A, Argos P, SRS: information retrieval system for molecular biology data banks. Methods Enzymol. 1996, 266: 114-128.

Fegaras L. An experimental optimizer for OQL. Technical Report TR-CSE-97-007, CSE, University of Texas at Arlington, 1997.

Karp P, A Strategy for Database Interoperation, in Journal of Computational Biology, 1996.

Kemp G.J.L. and Gray P.M.G., Using the Functional Data Model to Integrate Distributed Biological Data Sources, Proc. 8th Int. Conf. on Scientific and Statistical Database Management, IEEE Press, 176-195, 1996.

Markowitz, V.M., and Ritter, O., Characterizing Heterogeneous Molecular Biology Database Systems, Journal of Computational Biology, 2(4), 1995.

Paton, N.W. and Gray, P.M.D., Optimising and Executing Daplex Queries Using Prolog, The Computer Journal, Vol 33, No 6, 547-555, 1990.

Rector A.L., Bechhofer S., Goble C.A., Horrocks I, Nowlan W.A., Solomon W.D., The GALEN modelling language for medical terminology, in AI in Medicine 1996.

Rector A. and Horrocks I. Experience building a Large, Re-usable Medical Ontology using a Description Logic with Transitivity and Concept Inclusions. AAAI Spring Symposium on Ontological Engineering, 1997.

Rodriguez-Tome P, Helgesen C, Lijnzaad P, Jungfer K, A CORBA server for the radiation hybrid database. Proceedings of the ISMB 1997, 5:250-253.

Warren D.H.D., Efficient Processing of Interactive Relational Database Queries Expressed in Logic, Proc. 7th VLDB, 272-281, 1981.

Wiederhold G. Mediators in the Architecture of future Information Systems, IEEE Computer 21(3) March 1992, pp. 38-50.