# A knowledge-based query system for biological databases

Paolo Bresciani[1] and Paolo Fontana[2]

[1] ITC-irst, via Sommarive, 18, I-38050 Trento-Povo, Italy
bresciani@itc.it
[2] Istituto Agrario di S.Michele all'Adige, via Mach 1
S. Michele a/Adige, TN, Italy
pfontana@itc.it

**Abstract.** In more than one decade, genomic research produced a huge amount of experimental data. Although these data are usually freely available on the World Wide Web, accessing them in a consistent and fruitful way is not always an easy task. One of the main causes of this problem can be recognized in the lack of user interfaces that are sufficiently flexible and, at the same time, highly interactive and cooperative and also semantically precise.
This paper tackles this issues by presenting a semantics drive paradigm for formulating queries to genomic and protein databases. The paradigm is founded on knowledge-based reasoning capabilities, in order to provide the user with a semantics driven guide in the difficult task of building the intended query.

## 1 Introduction

Bioinformatics is the science of storing, extracting, organizing, analyzing, interpreting, and utilizing biological information. Genomic research —in particular, advances in DNA sequencing and genome mapping techniques— has provided huge amounts of data for populating several single-organism databases (e.g., [5, 26, 17]), as well as databases on many species [24, 4, 25]. Thanks to the decreasing cost of gene sequencing techniques, data on genetic sequences and the related databases are exponentially increasing.

If on the one hand, and for several years, the primary goal of genomic research and bioinformatics has been to make experimental data and resources merely available in order to allow for subsequent possible data interpretation and scientific models and theories construction, on the other hand the capability of managing such kind of data at the best —for storing, extracting, organizing, analyzing, and make them easily usable and accessible— is becoming one of the crucial factors for the next developments in bioinformatics.

Nevertheless, up to now, only few efforts have been made aimed at proposing appropriate tools for accessing distributed and syntactically heterogenous genomic and protein databases. Of course, the efficient and effective access to this kind of databases involves different challenges when various aspects of information technologies are dealt with. Although some of these topics have been tackled, such as the design of appropriate query languages [22] that allow for a better functional interoperability and

integration [21] of distributed and syntactically heterogeneous genomic databases, still only few efforts have been made in order to provide flexible, clever, and user-friendly interfaces, capable of drastically easing the task of query formulation.

Form based query interfaces like that provided with the Sequence Retrieval System (SRS) [16] are currently the most commonly used by the biologists community, although they allow only a very little flexibility, and in most of the cases are inadequate and make query formulation difficult, when not an impossible, as evidenced in [21]. On the other hand, the use of generic query languages like *SQL*, even if extended to better fit the need of accessing multiple genomic databases, like in the case of *GQL* [22], is viable only for those biologists that acquired some expertise with them.

This paper proposes a novel paradigm for the realization of easy, intuitive, and flexible user-interfaces for accessing complex and structured biological databases.

This is a crucial aspect, because the realization of intuitive interfaces that really help the user represents a necessary condition to gain user's acceptance and confidence in the provided data and knowledge resources.

The paradigm is based on the use of knowledge representation tools and ontologies for flexible and intuitive formulation of queries against relational and object-oriented biological databases.

The present paper is organized as follows. Section 2 introduces users needs in accessing databases on complex domains. Section 3 present our knowledge based paradigm for driving intelligent query interfaces. Finally, Section 4 present some technical details on the implementation of the reasoning services. Conclusions are given in Section 5.

## 2 User Needs and Query Systems

The task of building a query for accessing data stored in a database can be tackled in different ways by different kinds of user. In any case, the query must be expressed in a form suitable to be processed by the system or the specific application, and, at the same time, it must have a clear meaning for the user.

Users can be classified on the basis of their familiarity with databases, their knowledge of the application domain, the frequency of use of the application, and the heterogeneity and complexity of the queries they formulate [13, 14]. They may range from *expert* users, with a good understanding of the notions underling the data models for databases design and a good knowledge of the specific application domain, down to *naïve* users, with no background in data modeling techniques and tools, no knowledge about database design and implementation, and only a minimal knowledge of the application domain. They are likely to use the application only for simple and predefined tasks.

In the case of genomic and protein databases, users can generally be considered as domain expert with little or no expertise neither in database technologies, nor in formalisms and methods for conceptual modeling. Thus, they are placed in a particularly unfavorable position: although they need to formulate sophisticated and extemporary queries, they are likely to have no familiarity with structured query languages, and possibly they are just occasional users, unwilling to spend many efforts in trying to

understand the way the information is logically organized in the different and many databases.

The so called "Visual Query Systems" (VQS) [14] try to overcome these difficulties by providing graphical paradigms for visualizing the conceptual model of the available data, and by allowing to build queries by means of visual interactions. The VQS are, typically, addressed to all those situations in which: (i) the application domain is complex, and, therefore, a simple form-based interface, or even a QBE-like interface [27] —as, e.g., that of SRS— is insufficient; (ii) the user is eager to discover interesting information and to derive knowledge from the highly structured set of available data.

Nevertheless, most VQS still fail to adequately support the users in the task of understanding the *semantics* of the data at an intensional level. In fact, although they typically provide graphical representations of the data models, they scarcely support the users in strictly linking models with their intuitive understanding of the domain scenario. In order to better understand this connection, the users must perform a set of trials, during which they can *discover* the real meaning of the stored data by analyzing and comparing different answers to different tentative queries.

Moreover, the VQS typically require a good understanding of the diagrammatic notations used by database design methodologies, like, e.g., Entity Relationship.

In the presence of a really complex domains, like the bio-molecular ontologies, the situation is even harder. In fact, in these cases, even a user with a good understating of the data models and with a good knowledge of the scenario may find difficulties in combining a necessarily imperfect knowledge of the domain and an almost missing vision of the connection between it and the logical structure of the database, on the one hand, with the necessity of building sophisticated queries, on the other hand. For example, it can be difficult to avoid inconsistent requests, or noticing similarities between apparently different queries. This can lead to two kinds of problems:

- in her attempts of building the intended query, the user can fail several times, because reiterating slightly different, but all inconsistent, queries;
- it is possible that, after having submitted a query to a database, the answer is unexpectedly too large or too narrow: in this case, trying to modify it in a structural and relevant way could be an hard task.

For example, consider a simple case where the user initially submits a query specifying that she wants to retrieve the records on all the proteins that are expressed in the cell nucleus and have *KDEL receptor* function. Of course no record will be returned, because these proteins are only present in the endoplasmic reticulum (see, e.g., [23]). To keep our example very simple, we can imagine that, once the user somehow realizes the non-sense of her query, she tries to formulate a new query, slightly generalizing the first one by replacing the function *KDEL receptor* with a more generic function, for example *endoplasmic reticulum receptor*. A standard interface cannot suggest to the user that nothing better has been done (the query still is inconsistent), and database access time (and possibly money) is wasted again. Moreover, the only information obtained by the user, and only after having actually submitted the query, is that no record of the required type is present in the database; but nothing prevent the user from thinking that —maybe in the future or in other databases— such a record could be available. Thus, further vain searches may be done elsewhere or in other occasions.

Of course, the example sketched above is trivial for any biologist, but with slightly more specialized terms it may be easily transformed into a real problem even for expert biologists. In these cases, a tool that could exhibit an intuitive semantic description of the database content would be of great value.

For these reasons, we propose here a semi-visual paradigm, where more emphasis is put on knowledge based services, rather than on diagrammatic visualizations. In our paradigm, query validation and comparison can be carried out at the intensional level.

The idea is to consider the task of formulating the intended query as an iterative process, possibly requiring several steps of refinement and modification depending on the responses of the system. During this process, the user is assisted in *understanding* the semantics of the queries progressively built. Also, a classification of the queries with respect to the set of classes defined in the conceptual model of the database is constantly provided, giving a way for progressively understanding and learning the semantics of the model itself. The classification capability allows us to immediately verify if a query has inconsistent semantics with respect to the database conceptual model.

The resulting process of query formulation is a highly interactive one, and uses an *intensional* representation of the query. That is, the user works on the query itself, and not on the data (the *extension*) that may be in the answer to the query; only when the query is considered to be well formulated it is submitted to the database. Thus, there is a much higher chance to get the desired results. A consequence is that the process results in a faster and cheaper (in the case of pay-per-record) access, especially to remote or multiple databases.

The proposed paradigm uses, in order to convey the conceptual model and query meaning to the user, a uniform graphical presentation. But its most important characteristic is that, being based on a semantic representation of the database conceptual model, it allows us to perform some relevant inferential tasks on the semantics of the queries. In particular, it allows the user to:

1. interactively and iteratively build queries;
2. be prevented from building inconsistent queries;
3. interactively explore the semantics of the queries and of the classes involved;
4. be gradually introduced only to those parts of the conceptual model that are relevant for the query formulation;
5. be provided with simple, but effective, features for query refinement and query generalization.

## 3 The Knowledge Based Interaction

To exhibit the behavior and the features mentioned in the previous section, a system implementing our paradigm must provide a classification of the query with respect to the set of classes that form the database conceptual model. In particular, classifying the query at each step of the interaction allows us to immediately detect semantic inconsistencies.

Thus, it is then necessary that the query corresponds to an object description, possibly with nested relationships with other queries. Therefore, such kind of descriptions

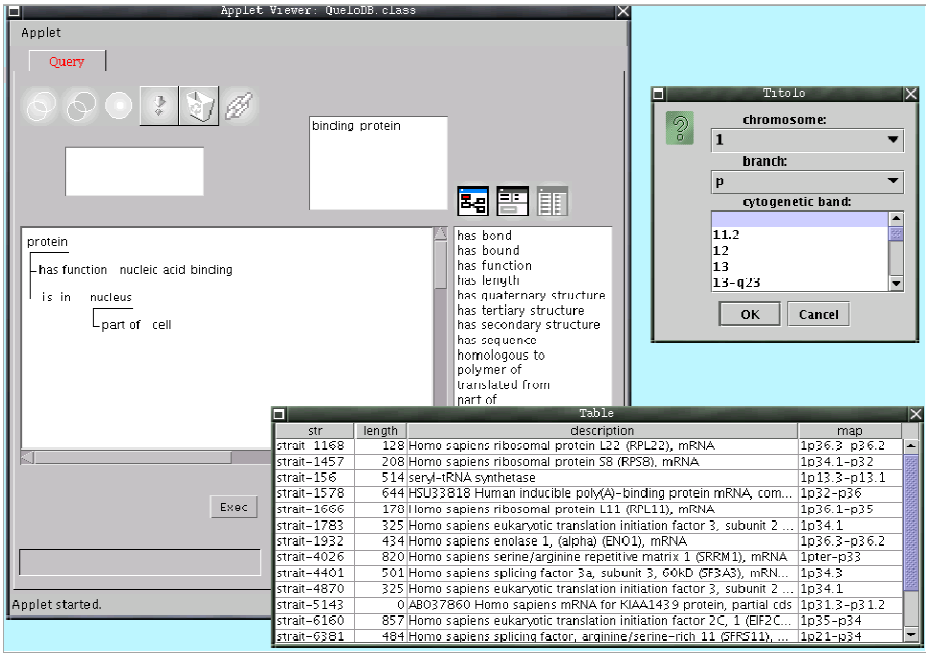


**Fig. 1.** The interface proposed by our paradigm, as appears in our implemented prototype. The picture include also the interface for answer browsing.

must be dynamically generated and managed. From the graphical point of view, a query can be represented as in Figure 1 (in the left part). The example shown in Figure 1 corresponds to the following query:

$$Q(x) \leftarrow \texttt{Protein}(x) \wedge \texttt{has-function}(x,w) \wedge \texttt{Nucleic-Acid-Binding}(w) \wedge \texttt{is-in}(x,y) \wedge \texttt{Nucleus}(y) \wedge \texttt{part-of}(y,z) \wedge \texttt{Cell}(z).$$

We believe that the graphical notation in Figure 1 is quite easily grasped by the unskilled user, because it is quite close to a natural language *Noun Phrase*, but, at the same time, avoids referential ambiguities. Moreover, the representation shows only the necessary terms of the class hierarchy, and avoids to disorientate the user with the too many details usually provided by most of the VQS interfaces.

At the same time, the user is allowed to interactively build sophisticated queries simply by choosing among the classes defined in the conceptual model, and iteratively forming, with them, complex boolean expressions of literals connected by links representing relationships. In Subsection 3.1 the main functionalities for building queries will be presented. An important aspect is that the query that is going to be built is constantly guaranteed to be consistent with respect to the conceptual model of the database. Moreover, the query is classified with respect to the conceptual model after each modification, so that the user can be informed about the position of the query in the conceptual taxonomy.

Requiring that only consistent queries are built prevents the user from uselessly querying the database. This is the minimal service that a semantics driven interface should provide. Moreover, our paradigm assumes also other constraints on the interactions allowed for building the query. Altogether, it is required that:

- Only *consistent* actions (i.e., actions that lead to a new query that still is consistent) are allowed.
- Only *relevant* modifications (i.e., query transformations that lead to new queries that are semantically not equivalent to the original one) are proposed.
- Only *close* modifications (i.e., not all the consistent and relevant modifications, but only those that lead to a new query with semantics close to the original query semantics) are proposed. This constraint is required in order to avoid to overload the user with too many options.[3]

In the following we shortly call these constraints "CRCP", that stands for "Consistent, Relevant, and Close Property".

In the next subsection the principal kinds of interactions are listed.

### 3.1 Query Building

We implemented the features illustrated above in a prototypical concept demonstrator [8], the interface of which is shown in Figure 1. The window on the left includes all the element for building the query:

1. an area where to represent the query;
2. a list of names of classes more generic than the query (let's call it a *generalization list*);
3. a list of names of classes more specific than the query (*specialization list*, hidden in the picture);
4. a list of names of classes equivalent to the query (*equivalence list*);
5. the list of relationships that can be used to build or modify the query;
6. a set of buttons that allow us to perform various actions for building the query.

The picture also shows how attributes can be restricted (window at the upper right of the screen snapshot) and a possible way to present the results (window at the bottom).

The prototype has been configured in order to access the "Muscle-TRAIT" biological DB. The Muscle-TRAIT DB [1] is part of a project carried on at CRIBI Biotech Center of the University of Padua, aimed at identifying and characterizing genes expressed in human skeletal muscle. The initial project was based on systematic sequencing of ESTs. The most relevant data are stored in the TRAIT DB (TRAnscript Integrated Table). The DB is periodically updated with cross-references to LocusLink [24], a database maintained at NCBI (National Center for Biotechnology Information) where, when possible, entries are tagged with references to the biological ontology *GeneOntology* [2]. Thus, every Muscle-TRAIT entry that has an homology with a human sequence in LocusLink inherits the association with one or more terms of GeneOntology.

[3] This point is one of the most delicate. In fact, it is difficult, in general, to give a clear notion of *close* query and *close* modification. Moreover, in some cases, some overload could be acceptable, or even desirable, if it provides for some shortcut in the interaction.

The interaction with the prototype starts with the selection of a first term among those proposed in an initial list, that, in the specific configuration, are protein, biological function, and cell part. After that, the user can iteratively modify the query by the following functionalities:

- *Query replacement:* that allows for the selection of an initial term in the taxonomy, and for the substitution of the whole query with an atomic term semantically close to it (more general, specific, or equivalent).
- *Relation selection:* that allows to restrict the query by imposing the existence of relationships with other atomic terms or complex expressions.
- *Propositional combination with other terms:* that allows for the boolean combination of selected parts of the query with other terms.
- *Negation:* that allows for the negation of literals.
- *Refinement:* that allows for the replacement of selected subexpressions, instead of the boolean combination.

The buttons in the interface give access to these function.

An example of use of the concept demonstrator and the above functionalities is described in [8]. In Subsection 4.3 a different example is introduced in abstract terms.

## 4 Reasoning Services

As mentioned, the main requirement of our paradigm is that each of the functionality listed above must comply the CRCP. To obtain this behavior some reasoning capabilities are needed. We obtain these capabilities by representing the conceptual model of the database and the query itself in terms of a Description Logic Knowledge Base.

In fact, it has been show that Description Logics (DL) offer powerful formalisms for solving several problems concerning data modeling and access [6, 12], like, for example, schemata integration and, in particular, intelligent query management [11, 7].

Below, we briefly recall some basic notions on DL, and exemplify how conceptual models and queries can be represented. Then, it will be possible to introduce the reasoning services that allow to guarantee the CRCP for the functionalities listed in section 3.1.

### 4.1 Description Logics

Description Logics are, essentially, variable-free concise reformulations of decidable restricted fragments of First Order Logic (FOL). The syntax of DL allows to express *concepts* (unary predicate symbols), *roles* (binary predicate symbols), and *individuals*[4] (constants). In different DL slightly different languages are used, with different levels of expressiveness. In all of them [15] concept expressions and role expressions can be built starting from atomic concepts and roles, using different sets of operators, like the concept-forming operators: $\sqcap, \sqcup, \neg, \forall, \exists$, and the role-forming operators: $\cdot^{-1}, \circ$.

[4] Reasoning about individuals (the *assertional reasoning*) is not relevant for our purposes; therefore, only the *terminological* aspects of DL will be considered.

| Construct | FOL Semantics |
|---|---|
| A | $A(\gamma)$ |
| $(\neg C)$ | $\neg F_C(\gamma)$ |
| $(C \sqcap D)$ | $F_C(\gamma) \wedge F_D(\gamma)$ |
| $(C \sqcup D)$ | $F_C(\gamma) \vee F_D(\gamma)$ |
| $(\forall R.C)$ | $\forall x.F_R(\gamma,x) \Rightarrow F_C(x)$ |
| $(\exists R.C)$ | $\exists x.F_R(\gamma,x) \wedge F_C(x)$ |
| $\vdots$ | $\vdots$ |
| P | $P(\beta,\alpha)$ |
| $(R^{-1})$ | $F_R(\beta,\alpha)$ |
| $(R \circ Q)$ | $\exists x.F_R(\alpha,x) \wedge F_Q(x,\beta)$ |
| $\vdots$ | $\vdots$ |

**Table 1.** FOL transformational semantics for some DL operators.

Description logics semantics can be given, for example, by mapping DL expressions into FOL formulæ [11]: an atomic concept A and an atomic role P are mapped —or *interpreted*— into the FOL open atomic formulæ $A(\gamma)$ and $P(\alpha,\beta)$. In table 1, for some DL concept expressions $C$ and $D$ the corresponding FOL open formulæ $F_C(\gamma)$ and $F_D(\gamma)$ are recursively given; similarly, for some DL role expressions $R$ and $Q$ the corresponding FOL open formulæ $F_R(\alpha,\beta)$ and $F_Q(\alpha,\beta)$ are given.

An important feature of DL is that they are equipped with a formal calculus that allows to perform some inferential tasks [15]. The most relevant, here, are (i) *consistency checking*: a concept description $C$ is said to be *consistent* iff the corresponding FOL formula, $F_C(\gamma)$, is satisfiable; (ii) *subsumption*: a concept description $C$ is said to *subsume* a concept description $D$ (written $D \sqsubset C$) iff $\forall x.F_D(x) \Rightarrow F_C(x)$ is a FOL tautology.

In DL it is possible to define *knowledge-bases* (KB) as sets of subsumption assertions. It is said that a subsumption is entailed by a knowledge base ($KB \models D \sqsubset E$) iff $F_{KB} \models \forall x.F_C(x) \Rightarrow F_D(x)$ in terms of FOL.[5] The consistency of a concept description $C$ with respect to a (consistent) KB can be expressed as: $KB \models C \not\equiv \bot$.[6]

### 4.2 Conceptual Modeling in DL

To see how a database conceptual model can be described in terms of Description Logics, let's consider the fragment of knowledge base (from now on, the "*KB*") shown in Figure 2, encoding the TRAIT DB conceptual model and domain.[7]

[5] Where $F_{KB}$ is the set of FOL axioms containing $\forall x.F_{C_1}(x) \Rightarrow F_{C_2}(x)$ for each $C_1 \sqsubset C_2$ in $KB$.

[6] $C \equiv D$ stands for $C \sqsubset D \wedge D \sqsubset C$, and the special concept $\bot$ correspond to the symbol denoting the *false* in FOL. Similarly, the special concept $\top$ corresponds to the symbol denoting the *truth* in FOL.

[7] Of course there are many different possible encodings; the one in Figure 2 tries to be the most readable, although not the shortest one.

$$Protein \sqcap Cell\text{-}Part \equiv \bot$$
$$Protein \sqcap Function \equiv \bot$$
$$Function \sqcap Cell\text{-}Part \equiv \bot$$

$$\exists has\text{-}funct.\top \sqsubset Protein$$
$$\exists has\text{-}funct^{-1}.\top \sqsubset Function$$

$$\exists is\text{-}in.\top \sqsubset Protein$$
$$\exists is\text{-}in^{-1}.\top \sqsubset Cell\text{-}Part$$

$$Protein \sqsubset \forall has\text{-}funct.Function \sqcap \exists^{\geq 1} has\text{-}funct \sqcap \forall is\text{-}in.Cell\text{-}Part \sqcap \exists^{\geq 1} is\text{-}in$$
$$Function \sqsubset \forall has\text{-}funct^{-1}.Protein \sqcap \exists^{\geq 1} has\text{-}funct^{-1}$$

$$Nucleus \sqsubset Cell\text{-}Part$$
$$Cytosol \sqsubset Cell\text{-}Part$$
$$Nucleus \sqcap Cytosol \equiv \bot$$
$$Cell\text{-}Part \sqsubset Nucleus \sqcup Cytosol$$

$$Binding \sqsubset Function$$
$$Nucleic\text{-}Acid\text{-}Binding \sqsubset Binding$$
$$Transcription\text{-}Factor\text{-}Binding \sqsubset Binding$$

$$Binding\text{-}Protein \equiv Protein \sqcap \exists has\text{-}funct.Binding$$

$$Protein \sqcap \exists has\text{-}funct.Nucleic\text{-}Acid\text{-}Binding \sqsubset \forall is\text{-}in.Nucleus$$
$$Protein \sqcap \exists has\text{-}funct.Transcription\text{-}Factor\text{-}Binding \sqsubset \forall is\text{-}in.Nucleus$$
$$Protein \sqcap \exists is\text{-}in.Nucleus \sqsubset \forall has\text{-}funct.(Nucleic\text{-}Acid\text{-}Binding \sqcup Transcription\text{-}Factor\text{-}Binding)$$

$$\vdots$$

**Fig. 2.** Translation of the conceptual model into a KB.

Using *KB* allows the system to infer, for example, that whenever a *Protein* is located (*is-in*) in the *Nucleus*, it *has-funct* only of type *Nucleic-Acid-Binding* or *Transcription-Factor-Binding*, and that in this case *Binding* is a good summarization of both, and that saying $\neg Cytosol$ as range of *is-in* is equivalent to saying *Nucleus*, and so on.

### 4.3 DL Reasoning and the CRCP

In other papers [10, 8] we motivated our approach by presenting further details and examples on the interaction. In this last section, instead, we shortly introduce the reasoning services that are implemented in our concept demonstrator.

Thus, consider that the intended query to be built is, for example:[8]

$$Q(x) \leftarrow \qquad (1)$$
$$\texttt{Protein}(x) \wedge \texttt{is-in}(x,z) \wedge \neg \texttt{Cytosol}(z) \wedge \texttt{has-funct}(x,y) \wedge \texttt{Function}(y)$$

[8] From now on, for editing reasons, we write the queries in they linear form including variables. Of course the system present them in the graphical/textual notation shown in the Figure 1.

One possible path followed for building it could first go through the formulation of the following query:

$$Q(x) \leftarrow \texttt{Protein}(x) \wedge \texttt{has-funct}(x,y) \wedge \texttt{Function}(y)$$

and then add to it the further constraint:

$$\texttt{is-in}(x,z) \wedge \neg \texttt{Cytosol}(z)$$

Once obtained (1), the user may want to refine the term `Function`. The system will then evidence that `Function` may be replaced by `Binding` without changing the semantics. The capability of showing such a kind of *contextual* equivalence is important, because this conveys to the user more knowledge about the conceptual model.

If instead the user tried to build query (1) by starting from the query:

$$Q(x) \leftarrow \texttt{Protein}(x) \wedge \texttt{is-in}(x,z) \wedge \neg \texttt{Cytosol}(z)$$

and then adding the constraint on `has-funct`, the system would immediately propose the following completion:

$$Q(x) \leftarrow \texttt{Protein}(x) \wedge \texttt{is-in}(x,z) \wedge \neg \texttt{Cytosol}(z) \wedge \texttt{has-funct}(x,y) \wedge \texttt{Binding}(y) \tag{2}$$

In fact, $\texttt{Binding}(y)$ is the most specific constraint applicable to $y$ without introducing loss of generality. Thus, form (2) of the query is also the most informative.

The behavior exhibited by the system is a consequence of the CRCP listed in section 3, considering the fact that the two queries (1) and (2) are equivalent in the context of the conceptual schema of the database.

The features illustrated above, and others, are obtained by means of some reasoning services that can be implemented in terms of the basic inferential services of *consistency checking* and *subsumption* provided by a DL reasoner. In order to do this, it is necessary to represent the query in DL. For example, query (1) is represented by:

$$Protein \sqcap \exists \textit{is-in}.\neg Cytosol \sqcap \exists \textit{has-funct}.Function \tag{3}$$

And query (2) by:

$$Protein \sqcap \exists \textit{is-in}.\neg Cytosol \sqcap \exists \textit{has-funct}.Binding \tag{4}$$

It is quite easy to verify that the equivalence among term (3) and term (4) is entailed by the knowledge base ($KB \models (1) \equiv (2)$).

Of course, this is just one example of reasoning: in particular it is relevant for the functionality of *relation selection* seen in subsection 3.1. Another example, also visual-

ized in Figure 1, is:

$$Protein \sqcap \exists is\text{-}in.(Nucleus \sqcap \exists part\text{-}of.Cell) \sqcap \exists has\text{-}funct.Nucleic\text{-}Acid\text{-}Binding$$

$$\sqsubset$$

$$Binding\text{-}Protein$$

More systematically, the kinds of checks corresponding to each of the functionalities introduced in subsection 3.1 are sketched below.

**Query replacement.** It is enough to maintain a representation of the query in term of DL, like in expression (3), and to *classify* it in the *taxonomy* implied by the knowledge base. That is, the most specific subsumers of $Q$ —$MSS(Q)$— and the most generic subsumees of $Q$ —$MGS(Q)$— must be found in the KB, in order to update the generalization and the specialization lists, respectively.

**Relation selection.** Here the task is more complex, and require two steps:

1. list all the *compatible relationships* by testing, for each relation name $R$ in the KB, if $Q \sqcap \exists R.\top$ is consistent (similarly, for the inverse relation, the test is: $Q \sqcap \exists R^{-1}.\top \not\equiv \bot$);
2. for each compatible relationship $R$ (or $R^{-1}$) determined with step 1, define the relevant range as $MSS(\exists R^{-1}.Q)$ (or $MSS(\exists R.Q)$).

**Propositional combination.** Let's consider the case of `AND` combinations. Assume that $q$ is the sub-query, selected in $Q$, for which a list of possible (i.e., CRCP compliant) terms to be conjuncted is sought. Let $Q_{q/q'}$ denote the new query obtained from $Q$ by replacing the sub-query $q$ with $q'$ in $Q$. That is, what must be checked is the CRCP of $Q_{q/(q \sqcap C_i)}$ with respect to $Q$, for all the $C_i \in KB$. It is worth noticing that only the specialization list is meaningful during this operation. Several strategies can be adopted in order to reduce the search of the $C_i$, as, for example, following a top-down search in the taxonomy, with early pruning when inconsistent cases are found, and no further search when a relevant $C_i$ is reached (in this case $Q_{q/(q \sqcap C_i)}$ will be the closest to $Q$, along the considered path, due the top-down search). The top-down strategy gives good results because, in practical cases, the conceptual model contains disjoint classes at a quite high level in the taxonomy, allowing for a good amount of pruning.

For `OR` combinations the general approach is dual, but the dual (bottom-up) strategy is not as efficient as the top-down strategy for the conjunctions.

**Negation.** The case of negation is quite easy: it is enough to check if the query ($Q_{q/\neg q}$) will still be consistent.

**Refinement.** In this case the situation is similar to the case of propositional combination, seen above, but a bidirectional search must be performed. The CRCP must be checked on all the possible $Q_{q/C_i}$. A possible strategy is to start from $MSS(q)$ and $MGS(q)$, and search upward and downward respectively, in order to find the generalization and the specialization lists. Also in this case, reaching a relevant $C_i$ or a $C_i$ that makes $Q_{q/C_i}$ inconsistent halts the search along the considered path.

Of course, the paradigm here proposed can be implemented only in presence of a DL automatic reasoner, capable of managing KB with circular definitions (or general axioms). It is also clear that the DL language required must allow to express conjunctions, disjunctions, negations, qualified quantifications and number restrictions, and inverse roles. $\mathcal{SHIQ}$ has all these characteristics. In our implementation we used the DL reasoner iFaCT for the $\mathcal{SHIQ}$ Description Logics [20]. The *MSS* and the *MGS* services, as well as all the checks and functionalities described above, have been implemented by us over iFaCT. The KB representing the conceptual model and domain of the Muscle TRAIT DB has been built by us starting from TAMBIS [3] and GeneOntology [2]. More details on these aspects and on the architecture of the prototype can be be found in [8].

## 5 Conclusion and Discussion

We have introduced a novel paradigm for providing knowledge based flexible query interfaces to complex databases. Our paradigm may be applied in several applications domains [9, 10], whenever: (i) the application domain is intrinsically complex; (ii) the queries the user wants to build are likely to be structurally complex and extemporary; (iii) the user is eager to discover, for better formulating the intended query, the structure of the conceptual model, through an interactive and iterative process. In particular, in this paper, the paradigm has been illustrated in the context of biological databases.

The main advantage of our approach lies in the fact that the user can be supported, by classification and consistency checking services, in iteratively and incrementally formulating the intended query, with a minimal cognitive effort and a minimization of malformed or unwanted queries.

In the last part of this paper we focused, in particular, on the definition of the reasoning services that must be provided so that the paradigm can be fully implemented. They are soundly based on the use of the iFaCT reasoner for the highly expressive $\mathcal{SHIQ}$ Description Logics. Reasoning with expressive Description Logics is known to be an hard problem in general [15], and, in particular, for $\mathcal{SHIQ}$ it is EXPTIME-complete. Fortunately, the pathological cases are also the most artificial. Several optimization techniques are implemented in iFaCT [18] and for some practical examples there is empirical evidence that they can provide a considerable speed-up [19].

In the case of our prototypical implementation we still have to optimize it and scale it up to verify its applicability in a more real scenario. Indeed, our future research plans include the collaborations with bio-technology institutes to thoroughly experiment our approach with the collaborations of biologists who may act as test-users.

**Acknowledgements**

Among others, a special thank goes to Giorgio Valle and Stefano Toppo for having kindly provided the TRAIT DB. As well, we thank Ian Horrocks for having made iFaCT available.

Paolo Fontana also thanks University of Trento as well as IASMAA and Fondazione Cassa di Risparmio di Trento e Rovereto for financing his research.

## References

1. http://muscle.cribi.unipd.it.
2. M. Ashburner, C. A. Ball, J. A. Blake, H. Butler, J. M. Cherry, J. Corradi, K. Dolinski, J. T. Eppig, M. Harris, D. P. Hill, S. Lewis, B. Marshall, C. Mungall, L. Reiser, S. Rhee, J. E. Richardson, J. Richter, M. Ringwald, G. M. Rubin, G. Sherlock, and J. Yoon. Creating the gene ontology resources: design and implementation. *Genome Research*, 11(8):1425–1433, Aug. 2001.
3. P. G. Baker, C. A. Goble, S. Bechhofer, N. W. Paton, R. Stevens, and A. Brass. An ontology for bioinformatics applications. *Bionformatics*, 15(6):510–520, 1999.
4. D. A. Benson, D. J. L. Ilene Karsch-Mizrachi, J. Ostell, B. A. Rapp, and D. L. Wheeler. GenBank. *Nucleic Acids Research*, 30(1):17–20, 2002.
5. J. A. Blake, J. E. Richardson, C. J. Bult, J. A. Kadin, J. T. Eppig, and T. M. G. D. Group. he mouse genome database (mgd): the model organism database for the laboratory mouse. *Nucleic Acids Research*, 30(1):113–115, Jan. 2002.
6. A. Borgida. Description logics for data management. *IEEE Transactions on Knowledge and Data Engineering*, 5(7), Oct. 1995.
7. P. Bresciani. The challenge of integrating knowledge representation and databases. *Informatica*, 20(4):443–453, Dec. 1996.
8. P. Bresciani, P. Fontana, and P. Busetta. A knowledge based interface for distributed biological databases. In D. Marra, E. Merelli, P. Romano, and G. Rossi, editors, *Proceedings of the NETTAB international workshop on Agents in Bioinformatics*. NETTAB, University of Bologna, July 2002.
9. P. Bresciani, M. Nori, and N. Pedot. A knowledge based paradigm for querying databases. In *Proceedings of the 11th International Conference and on Database and Expert Systems Applications (DEXA 2000)*, volume 1873 of *Lecture Notes in Artificial Intelligence*, pages 794–804. Springer, Sept. 2000.
10. P. Bresciani, M. Nori, and N. Pedot. QueloDB: a knowledge based visual query system. In *Proceeding of the 2000 International Conference on Artificial Intelligence (IC-AI 2000), volume III*, Las Vegas, June 2000. CSREA Press.
11. M. Buchheit, M. A. Jeusfeld, W. Nutt, and M. Staudt. Subsumption between queries to object-oriented databases. *Information Systems*, 19(1):33–54, 1994.
12. D. Calvanese, M. Lenzerini, and D. Nardi. Description logics for conceptual data modeling. In J. Chomicki and G. Saake, editors, *Logics for Databases and Information Systems*. Kluwer Academic Publisher, 1998.
13. T. Catarci, S. Chang, M. Costabile, S. Levialdi, and G. Santucci. A graph-based framework for multiparadigmatic visual access to databases. *IEEE Transactions on Knowledge and Data Engineering*, 8(3):455–475, 1996.
14. T. Catarci, M. F. Costabile, S. Levialdi, and C.Batini. Visual query systems for databases: a survey. *Journal of Visual Languages and Computing*, 8(2):215–260, Apr. 1997.

15. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proc. of the 2 [nd] International Conference on Principles of Knowledge Representation and Reasoning*, pages 151–162, Cambridge, MA, 1991.
16. T. Etzold and P. Argos. Srs —an indexing and retrieval tool for flat file data libraries. *Comput Appl Biosci*, 9(1):49–57, Feb. 1993.
17. FlyBase Consortium. The FlyBase database of drosophila genome project and community literature. *Nucleic Acids Research*, 27:85–88, 1999.
18. I. Horrocks. Reasoning with expressive description logics: Theory and practice. In A. Voronkov, editor, *Proc. of the 18th Int. Conf. on Automated Deduction (CADE-18)*, number 2392 in Lecture Notes in Artificial Intelligence, pages 1–15. Springer-Verlag, 2002.
19. I. Horrocks and U. Sattler. Optimised reasoning for $\mathcal{SHIQ}$. In *Proc. of the 15th Eur. Conf. on Artificial Intelligence (ECAI 2002)*, pages 277–281, July 2002.
20. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.
21. H. M. Jamil. Achieving interoperability of genome databases through intelligent web mediator. In *Proceedings of the IEEE International Symposium on Bio-Informatics and Biomedical Engineering (BIBE 2000)*, Washington DC, Nov. 2000. IEEE.
22. H. M. Jamil. Gql: A reasonable complex sql for genomic databases. In *Proceedings of the IEEE International Symposium on Bio-Informatics and Biomedical Engineering (BIBE 2000)*, Washington DC, Nov. 2000. IEEE.
23. B. Lewin. *GenesVII*. Oxford University Press, 2000.
24. K. D. Pruitt and D. R. Maglott. RefSeq and LocusLink: NCBI gene-centered resources. *Nucleic Acids Research*, 29(1):137–140, 2001.
25. G. Stoesser, W. Baker, A. van den Broek, E. Camon, M. Garcia-Pastor, C. Kanz, T. Kulikova, R. Leinonen, Q. Lin, V. Lombard, R. Lopez, N. Redaschi, P. Stoehr, M. A. Tuli, K. Tzouvara, and R. Vaughan. The EMBL nucleotide sequence database. *Nucleic Acids Research*, 30(1):21–26, 2002.
26. M. Westerfield, E. Dorrey, A. E. Kirkpatrik, and A. Douglas. Zebrafish informatics and ZFIN the database. *Methods Cell Biol.*, 60:339–355, 1999.
27. M. M. Zloof. Query-by-example: A database language. *IBM System Journal*, 16(4):324–343, 1977.