Building large knowledge bases in molecular biology

Schmeltzer O.[1], Médigue C.[1,2], Uvietta P.[1], Rechenmann F.[1], Dorkeld F.[3], Perrière G.[3], Gautier C.[3]

[1] INRIA Rhône-Alpes — LIFIA
46, av. Félix Viallet
F-38031 Grenoble Cedex
Phone : (+33) 76 57 45 63
Fax : (+33) 76 57 46 95
E-mail : rechenmann@imag.fr

[2] Institut Pasteur
CNRS URA 1129
Unité de Régulation de l'Expression Génétique
28, rue du docteur Roux
F-75724 Paris Cedex 15

[3] Laboratoire de Biométrie, Génétique et Biologie des Populations
CNRS URA 243
Université Claude Bernard — Lyon
43, bd. du 11 Novembre 1918
F-69622 Villeurbanne Cedex

Abstract

Large scale genome sequencing projects are now producing huge amounts of data which can be readily stored and managed within data base management systems, and analyzed using dedicated software packages. The results of these analyzes should also be stored with the input DNA sequences. The increasing complexity and size of the objects to be described and managed have led biologists to rely on advanced data models such as the object-oriented model. As a joint effort between our computer science and molecular biology research projects, the knowledge bases we have developed in molecular genetics have shown however that the basic object-oriented model is not fully adapted to the complexity of some biological situations encountered. Advanced descriptive capabilities, provided only by knowledge models originated from the AI field, are required. Composite or evolving objects, multiple viewpoints, constraints, tasks and methods, textual annotations are some examples of such capabilities. They are illustrated by biological situations for which they appeared to be necessary. Supporting powerful reasoning mechanisms (e.g. object classification, constraint propagation or qualitative simulators), they allow the development of large knowledge bases in molecular biology. These knowledge bases are expected to become the adequate support for co-operative distributed research efforts.

Introduction [1]

Large scale genome sequencing projects are producing large and rapidly expanding amounts of data. Under this pressure, computer scientists and biologists have developed the use of data base management systems (DBMS) in order to store, organize and retrieve these DNA sequences. As investigating methods were being developed, new types of objects had to be described and managed. Portions of genomic fragments had to be distinguished according to their functions, and more complex objects such as maps at different levels had to be stored together with the sequences. Simple file systems were quickly replaced by relational systems, and more recently by object-oriented data models (e.g. see (Shin et al. 1992)). The object model indeed allows the description of large complex and inter-related objects. Moreover, the possibility to attach the processing methods directly to these objects solves the integration problem. The so-called "impedance mismatch" between the data and the application programs disappears.

[1] This research is being supported by French Ministère de la Recherche et de l'Espace (grant 92.H.0904), by CNRS (Centre National de la Recherche Scientifique, GdR "Informatique et génomes") and Région Rhône-Alpes

It has to be expected that the need for more advanced modeling capabilities will develop as more knowledge is being extracted from the raw DNA sequence data. Such capabilities can be provided only by the knowledge models which emerged from artificial intelligence (AI) research work. The first section is devoted to the presentation of the basic object-oriented knowledge model. The following three sections then present and illustrate several situations for which this basic model turns out to be inadequate. Several examples are drawn from our own experiences in developing two large knowledge bases in molecular biology. The first knowledge base, ColiGene, describes the regulation mechanisms of gene expression in *E. coli*. The second one, MultiMap, allows the description of maps at different levels and for various living organisms, thus supporting comparisons and phylogenetic studies. We finally conclude on the role these large scientific knowledge bases are expected to play in co-operative distributed research efforts.

Representing knowledge with objects

Any computer program does include knowledge, but only knowledge-based systems attempt to clearly separate knowledge from its exploitation mechanisms. They rely on declarative knowledge models which indeed allow to express knowledge without anticipating the ways it will be effectively used for problem solving. Among the various available models, the object-based models currently appear as the most satisfactory to represent both data and knowledge. They draw their main characteristics from semantic networks and from some of the original ideas on frames (Minsky 1975) (Fikes & Kehler 1985).

Object-based knowledge models significantly differ from object-oriented programming languages as a consequence from their expected use. Since they have to support external inference mechanisms, their contents must be readily available ; slot descriptions and values are not encapsulated inside the objects. Moreover, incomplete instances may be created, i.e. some of their slots do not possess values. If required, these values can be determined using the knowledge in the classes, such as procedural attachments or default values. When knowledge on an instance increases, the instance can be attached to more specialized classes, i.e. sub-classes, through a classification process. Objects are here descriptions which have to be interpreted by inference mechanisms ; they are not directly executable.

The resulting object-oriented knowledge bases can be operated in several ways. First of all, they can be used to increase the knowledge on objects which are not completely known and characterized. Explanation on the results can be provided. Knowledge bases can also be simply consulted. The object model very adequately supports navigation, but requests can also be formulated to retrieve objects which satisfy some characteristics.

In this section, these common characteristics of object-based knowledge models are presented and exemplified with Shirka, a class-based knowledge model which has been conceived and implemented in INRIA Grenoble laboratory over the last eight years. Although presenting several original traits, Shirka indeed incorporates all the basic features of object-based knowledge models. It allows the development and the management of large knowledge bases, such as ColiGene and MultiMap, which have been completed in the context of a joint project involving computer scientists and molecular biologists.

Classes and instances

Shirka (Rechenmann & Uvietta 1991) relies on the distinction between classes and instances. A class describes a set of potential instances through a syntactical structure called a scheme (Fig. 1).

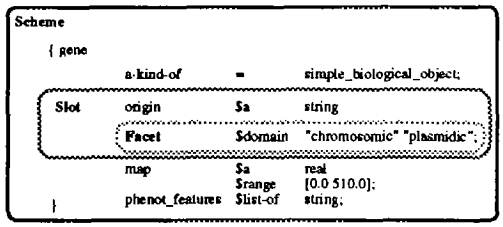

Figure 1 — Structure of a class as a knowledge description unit. Every class has a name and is composed by a list of slots. Each slot has itself a name and is described by a list of facets with their associated values.

A scheme is composed by a set of named slots which are themselves specified by facets. The role of a facet in a class is to set the type of the slot, to restrain the possible values which can be assigned to this slot in an instance and to attach computing methods and default values to be used when the value is unknown in an instance. To instantiate a class thus consists in creating an individual with the same structure as the class to which it belongs (i.e. it owns the same slots). Every slot can receive a value which must satisfy the attached constraints which have been defined in the class by the corresponding facets. Incomplete instances may be created : some of their slot values may be missing. If available in the class, methods and default values can thus be used to compute these values (see next sub-section). Since facets can only express constraints on one slot and since it is sometimes necessary to set constraints between slots of the same class or of different classes, and between instances, the system provides a means for defining constraints externally to the involved objects. These dependencies are then to be maintained whatever modification occurs in the base (Borning 81). It is up to the system to manage all the constraints declared in the base, i.e. restrict the set of values of a constrained slot, and possibly come up with a single value. The type of a slot is not bound to be a simple type like integer or string ; it can also be another class of the base. In that case, an instance of the defining class will have this slot valued by an instance of the class appearing in the description of the slot (Fig. 2). Such an object is called a complex object because it refers to other objects, the slot being a complex slot. This allows the recursive linking of objects.

```
{ protein-gene
      a-kind-of   =     gene;
      ref-num     $var-name   nseq;
      origin      $var-name   nsrc;
      type        $a          string
                  $domain     "CDS" "ORF" "URF"
                  $default    "CDS";
      rbs         $a          RBS
                  $if-needed
                        { cr-rbs
                              ref         $var<-      nseq;
                              source      $var<-      nsrc;
                              new-rbs     $var->      rbs } }
{ aceE
      is-a        =     protein-gene;
      ref-num     =     ECOACEX;
      origin      =     GenBank;
      type        =     CDS;
      rbs         =     rbs-aceE }
```

Figure 2 — Complex instance. An instance of the class `protein-gene` has its slot `rbs` valued by an instance of the class `RBS` as specified in the description of the class.

Classes are organized in specialization hierarchies (each hierarchy models a concept) where a class inherits the descriptions of its super-classes. When describing a sub-class, it is only possible to add new slots or specify the description of already existing slots.

As an example, we show in Figure 3 the hierarchical structure of ColiGene (Perrière et al. 1993). This knowledge base focuses on the study of the bacterium *E. coli*, the genome of which is still today a generic model for Prokaryotic organisms. Particularly ColiGene describes the relationships between genomic sequences and the regulation of gene expressivity. With this aim in view, it was first necessary to carefully formalize knowledge on genetic structures and to represent the large functional genomic structures which are split in genomic databases despite their functional unity: transcription and translation signals, genes coding for proteins, tRNA and rRNA genes, operons and regulons. From the biological standpoint of this knowledge base, information concerning the relationships between DNA sequences and gene expressivity in *E. coli* have been modeled.

Unknown values computation

Incomplete instances may be created, i.e. some of their slots may have no value. If required, the value can be inferred using available knowledge in the classes. Two basic mechanisms are available : default value and procedural attachment. A default value facet defines a value to be provided for the slot when no other value is available. The default facet allows thus the reasoning mechanism to be continued even in case of incomplete knowledge. Procedural attachment consists in associating to a slot in a class description a set of methods to be executed under a specific context, e.g. when the slot value is modified or removed in an instance. Procedural attachment is also an inference mechanism able to return the value of a slot in an instance in which it is unknown. The methods are then introduced with the `if-needed` facet (Fig. 2). The basic principle is to use the methods attached to the slot in the class to which the instance is known to belong.

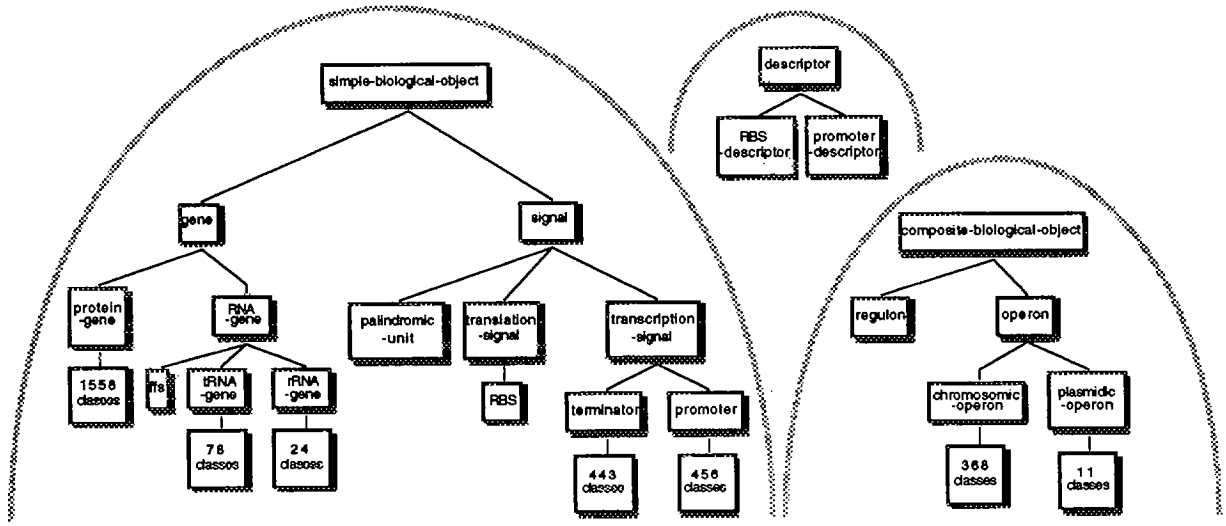

Figure 3 — Hierarchical structure of the knowledge base ColiGene. Individual subclasses and instances corresponding to individual structures are derived from the classes of upper levels. In such a way, instances take into consideration the allelic polymorphism. Three biological concepts are modeled in the base: the `simple-biological-object` concept gathers elementary information units like protein and tRNA genes and the signals involved in their regulation (promoters, operators, terminators, ...) ; the `composite-biological-object` concept contains complex regulatory systems like operons and regulons (this enables the linking of elementary objects which are completely independent in databases) ; the `descriptor` concept describes DNA features corresponding to particular regulatory signals.

If necessary, inheritance applies and methods can be looked for in the super-classes. The attached methods are generally functions written in a programming language. The example provided in Figure 2 illustrates such a mechanism : if the value of `rbs` of an instance of the class `protein-gene` is unknown, the system will execute the procedure `cr-rbs` (`$if-needed` facet), in order to provide this value.

Classification

Classification is an inference mechanism allowing an instance to migrate from its defining class to lower classes in the hierarchy ; as a consequence, the instance is compelled to stay in the same concept and is not permitted to go in another one.

When an instance has been created, it is linked to a class. This implies that the instance satisfies the constraints included in the slots of the class. Nonetheless, this instance can also satisfy constraints in the sub-classes of its class of definition. As soon as an instance has been attached to a class, all the constraints that appear in lower classes become necessary and sufficient conditions for the instance to belong to them. The classification mechanism scans recursively the sub-classes from the initial class and determines for each class if it is sure, possible or impossible. A class is *sure* if the instance satisfies all the constraints on all the slots that exist in this class, it is *possible* if no constraint in the class is violated (this happens for instance when a slot is not valued), and it is *impossible* if a constraint is not satisfied by a slot value. An instance is directly attached to a sure class whereas, if no sure class is found, a user can decide to attach it to a possible class. Then, the instance receives additional information since it can now benefit from new inference mechanisms specific to its new class. When dealing with complex objects, the classification mechanism acquires a new scope because a complex object references other objects that can be themselves classified in their hierarchies.

Navigation and query-based access

Knowledge bases include large amounts of entities to access, thus navigation facilities play an important role in such systems. Classical navigation in knowledge bases is, in some sense, a sequential one, whereas queries allow "immediate" access to relevant information.

In order to illustrate the navigation facilities we show in Figure 4 an example of a sample session under the knowledge base MultiMap. This system integrates different levels of mapping data with graphical tools allowing to switch between these levels. More precisely, MultiMap is dedicated to comparative mapping studies between man and mouse. It describes all known chromosomes and *loci* for both organisms (available in GENATLAS and EMG data-banks) and allows the location study using cytogenetic, genetic and physical maps, i.e. three different levels of description. Links between the biological objects in the base and the different maps have been carefully modeled; this way, when analyzing a human genome, the related cytogenetic maps may be accessed, and in turn, from a cytogenetic map, any of its *loci* may be accessed. A special effort has been made to develop graphic browsing and manipulation, because of the amount of objects (over 1000 *loci* for each of the two organisms).

A filter-like mechanism has been implemented as a first step solution, in our current system, in order to allow content search. Queries are built via a graphical editor, making the definition of the set of criteria rather easy. This query mechanism allows the retrieving of instances which satisfy the set of criteria ; as an example, "what are the protein genes located between minute 0 and 5, in *E. coli* chromosome?" is a typical content search, in ColiGene. Queries may be saved and modified, if needed. The resulting sets of instances may be saved too : they may, in turn, be used as input in order to refine a query. However, this solution remains rather restrictive, as it only allows content searches, but not structural queries.

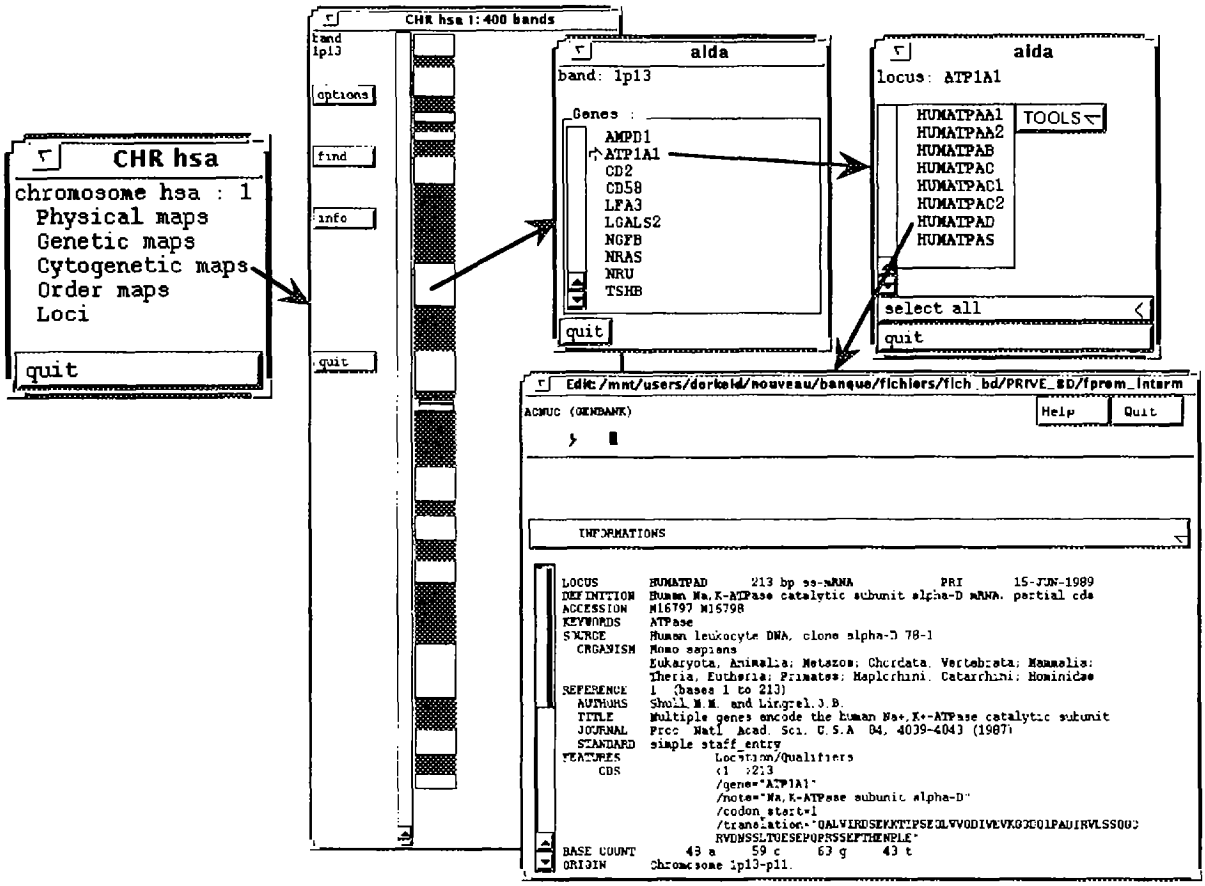

Figure 4 — Navigation facilities in the knowledge base MultiMap. Several windows obtained from successive user selections are represented here. The first one (on the left) shows the different levels of description of a human chromosome (n°1). In the second one, the cytogenetic chromosome bands obtained with G staining method are visualized. From one selected cytogenetic band, the list of its *loci* may be accessed and then, the list of the GenBank entries that describe a particular *locus* (in this example the ATP1A1 gene has been selected). Given a GenBank entry, one can also obtain the information stored in the data bank (HUMATPAD in this example).

Extracting from a knowledge base a set of entities the structure of which fulfils a given set of properties should be useful, for example when searching the relevant tasks (see the "Methodological knowledge" section) through the space of available tasks. An example of a structural query might be "what are the sequential tasks including `distance-matrix` and `multivariate-analysis` sub-tasks?" for pattern identification in DNA sequences, in ColiGene.

The design of the two knowledge bases, ColiGene and MultiMap, has led to the identification of three types of knowledge : descriptive knowledge on the biological entities involved, methodological knowledge on the ways to select and link up methods for a given task (such as DNA sequence analysis or computation of gene expressivity), and textual knowledge as annotations on the various objects. Extensions of the basic object knowledge model are required to deal with these three types of knowledge and the related increase in complexity. These extensions are described in the following three sections.

Descriptive knowledge

Several biological situations indeed appeared too complex to be described with the basic object-based knowledge model. The required extensions concern multiple perspectives or viewpoints on entities, part-whole relationships and more descriptive links between objects.

Multiple perspectives

The complexity of the biological objects involved makes a single hierarchy for the whole base very limiting. Genomic structures could be represented according to two perspectives, at least : a functional one where the hierarchy is based on the functions of the biological objects (this is the only perspective in the present state of ColiGene), and an evolutionary one allowing for the repetitive character of a sequence (i.e. whether a sequence is present in one or multiple copies in the genome and what kind of repetitive structure it contains) (Fig. 5).

On each perspective, a specific hierarchy of classes can be developed. A class in a hierarchy shares slots with other classes and owns peculiar slots that no other class has in another hierarchy. An instance belongs to one class in each hierarchy and can be viewed along the desired hierarchies by showing the relevant slots. A user can choose to look at a knowledge base along specific perspectives and dismiss information he is not interested in.

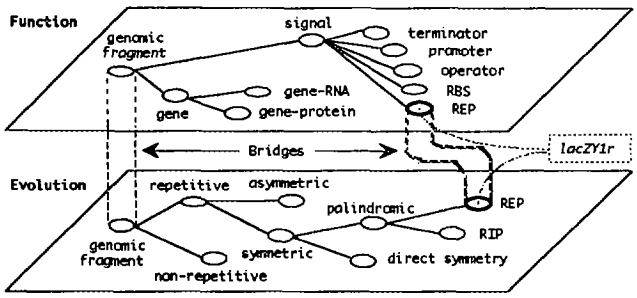

Figure 5 — Possible organization of a part of ColiGene conceptual scheme. Upper levels of both hierarchies are defined here. A genomic fragment may be considered from a functional point of view as well as from the evolutionary one. The classification of the `lacZY1r` instance in the functional perspective leads to attach it to the `REP` class; then, by using the bridge, the instance is attached to the `REP` class in the evolutionary perspective so acquiring more information on the existing slots of this class.

Classes of different perspectives can contain the same instances or, at least, there can be some inclusion between the extensions of classes. This is represented by bridges that link classes for which some inclusion relation exists. The bridges are the entities that model co-operation between different domain knowledge by connecting classes from different perspectives. An unidirectional bridge between two classes asserts that an instance of the first is also an instance of the second ; a bi-directional bridge expresses the equality between the sets of instances represented by the classes.
Bridges increase classification power (Mariño, Rechenmann, & Uvietta 1990) ; indeed, the classification takes place in each perspective, along the suitable hierarchy, and uses the bridges to skip from one class in a perspective to another in another perspective. That way, classification can label otherwise unreachable classes and consequently allows the use of new constraining facets on slots appearing in these classes.

Part-whole relation

In the basic object-based knowledge model, the link defined in a class carries the attribution meaning : an object is given properties through the slots it contains. For example, a protein gene can be described by its expressivity, its number of base pairs (these are simple slots) and the protein it codes for (a complex slot, since it takes its value in the class protein). However, other semantics, such as composition, are wrongly used in this link ; this fuzziness brings some representational problems and proves that stronger semantical relations are needed (Woods 1991).
Expressing the part-whole relation (or the composition relation) leads to introduce a new link between objects. Composition is a semantic relation that exists between an object and its components (Winston, Chaffin, & Herrmann 1987). It can be used recursively on components themselves. For instance, a cell is composed by mitochondria and ribosomes ; the latter can still be split in ribosomal RNA and ribosomal proteins. This recursive decomposition creates a hierarchy which reminds of the inheritance hierarchy. Both these hierarchies are orthogonal.
The part-whole relation is easily introduced in the basic model by keeping the scheme structure and its slots ; two kinds of slots have been defined according to the two existing semantics, attribution and composition. When representing attribution, a property slot is used, and a composite slot carries the composition semantics. Specific behaviors can be attached to composition regarding instantiation, destruction of a composite object, classification or inheritance of property slot values from a composite object to a component and vice versa (for more information, see (Kim, Bertino, & Garza 1989) and (Mariño, Rechenmann , & Uvietta 1990)).
An example of composition in molecular biology deals with the modeling of operons. An operon is a complex structure composed by genes and some regulatory parts. It enables coordinated expression of the genes contained by the operon. Either all the genes are transcribed or none is.
An operon can be modeled by an object which has both property slots and composite slots. The properties are its genomic fragment (the DNA sequence in the database), a slot expressing whether the control is positive or negative, another to assert if it is an inductible or a repressible operon. Moreover, it is composed by a set of structural genes, a regulatory gene, an operator, a promoter and terminators (Fig. 6).

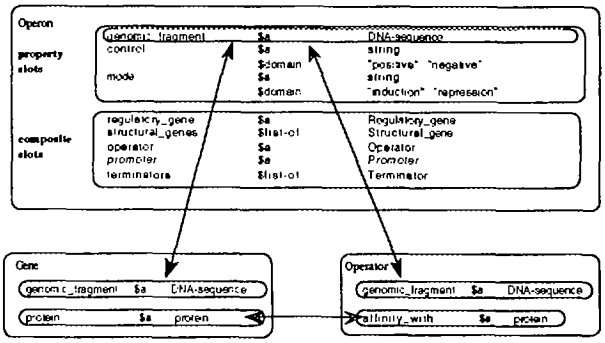

Figure 6 — Description of the class operon. It contains property slots as well as composite slots. Some values between the composite object and the components are shared: all the genes that compose the operon come from the same genomic fragment and the protein coded by the regulatory gene has affinity with the operator.

The regulatory gene codes for a protein which is either a repressor (in case of a negative control) or an apo-inducer (if the control is positive) ; this protein has much affinity with the operator of the operon and is activated or inhibited by an inducer or a co-repressor (which is in general a substract of the operon). These features show the intricacy of the biological objects and the relevance of links between them. Considering the composite slots, it would be interesting to express that the operon and the genes it contains are coming from the same genomic fragment, or that the operator has affinity with the protein coded by the regulatory gene. The operon class has been created and one of its instance (the lactose operon) is given in Figure 7.

Expressing new semantic relations

It can be useful to express specific semantic relations between objects. Since it is not possible to express them all, some systems give the user the means to define by himself the links that are going to be dealt with in the knowledge base (Fox, Wright, & Adam 1984). The drawback of such a system is the great difficulty to specify a new link.

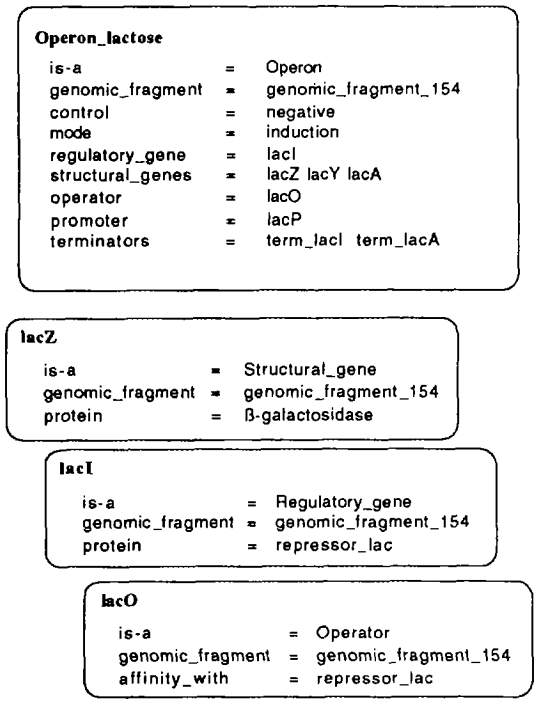

Figure 7 — Composite instance of class operon. When creating an instance of the class operon, the system automatically creates the components, provided it knows the number of each one, and the shared slots between the composite object and its components are given the same value. For example, if the lactose operon instance is created and the number of structural genes is known, three instances of structural genes, one instance of a regulatory gene, of an operator, and of a promoter will be created. Their genomic_fragment slot will be given the same value as that of the operon and the affinity_with slot of the operator will have the same value as the protein slot of the regulatory gene.

Two aspects of a link have to be represented : its semantics and its behavior. It is already an advantage to simply express a new semantics, specifying that way the relations between objects. Afterwards, it is possible to profit by this link by attaching a peculiar behavior to it.

Suppose for instance that one wants to assert that a protein activates the expression of a gene, or, on the contrary, that it inhibits it. In the simple model, one could create a concept protein with a slot describing its function. There is at once a problem if this protein is both an activator and an inhibitor for two different genes. One could then add two slots to the protein concept named activates and inhibits and whose values would be the involved genes. Then, the semantics of the activation link is in the name of the slot, not in its value, and if a protein is an activator for no gene, the slot has no value. It is not satisfactory because the protein exists regardless of the actions it is able to ensure. The best solution is to have the capability to define the link externally to the objects it connects, possibly defining inheritance mechanisms between their slots (this is part of the behavior attached to the link). This additional facility allows, for instance, to simulate the behavior of the upper mentioned protein, both as activator and a repressor.

Simulation may be either numerical, running for instance differential equations systems, or qualitative. Qualitative simulation relies on non-numerical computation. Such an approach allows simulation at an intermediate level of precision, and it makes it possible without requiring experimental data as for differential equations. (Bobrow 84) identified different qualitative reasoning tasks ; among them are simulation, envisionment (determine all possible behaviors), diagnosis and verification. Even if this approach has limited utility for large models (Forbus & Falkenhainer 1992), mainly because of combinatorics, it tends to allow the modeling of phenomena for which the numerical precision is not required, or for which laws (as in physics) are not known.

Qualitative simulation in biology has been already applied. MOLGEN allows the (qualitative) simulation of the *trp*-operon system (Friedland & Kedes 1985). (Weld 1986) has implemented a system to perform causal simulation in the molecular genetics domain : it allows the description of discrete processes to model possible enzymatic behaviors for a set of molecule types and it attempts to predict the behavior of the system. A system for the design of metabolic pathways (Mavrovouniotis et al. 1990) rests on the qualitative representation of the Boerhinger metabolic map ; it infers, from constraints on inputs and reactions, the complete set of all the possible pathways through the map.

Methodological knowledge

Methods are software modules which implement numerical, symbolic or graphic display algorithms. Domain-dependent methods, e.g. in molecular biology, give rise to various difficulties : their number and/or complexity grow regularly, they have been written in programming languages independently from the KB design, they are available in some repository(ies) elsewhere, etc. In order to make them easier to use, our solution consists in the definition of problem solving strategies or *tasks*, i.e. the identification of sub-problems and their proper ordering on the one hand, and in the selection of the (or one among a set of) convenient *method*(s) for each (sub-)problem, on the other hand. Tasks as well as methods are thus described explicitly in the KB as classes, i.e. they are considered as (particular) knowledge (Rechenmann & Rousseau 1992).

Most object-oriented languages do not allow methods to be defined outside a class (some languages, as CLOS, allow it, but with restrictions), because of the so-called "encapsulation" principle. A method in a class is selected according to the first argument of a message only. Method, selection according to multiple in put arguments is thus not allowed. Object-oriented languages are thus not adapted to the description of methodological knowledge.

Tasks, as well as methods, are described as classes with input and output slots. The set of constraints attached to input or output slots describes respectively pre- and post-conditions. A task provides additional facilities : to each complex task, a decomposition into sub-tasks is attached, unless the task is elementary ; in that case, a set of methods is directly attached to the task. A complex task can be either a selection among subtasks (e.g. extract-REPs in figure 8), or a sequence of sub-tasks (e.g. Search REP subclasses in figure 8).

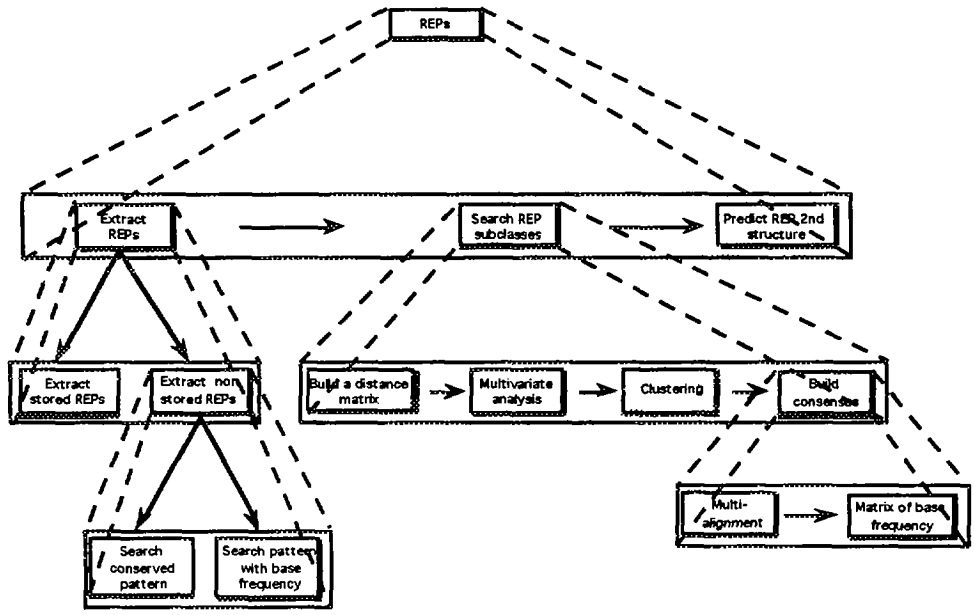

Figure 8 — Task decomposition hierarchy for Repetitive Extragenic Palindromic (REP) units identification. REPs are repetitive structures characterized by their complex secondary structures, which make them able to prevent mRNA degradation. To study REPs, a sequence of three subtasks has to be executed. First the REPs have to be extracted from the data bases. This subtask is a choice subtask: the extraction of REPs is different whether it concerns stored or non-stored REPs. In the case of the extraction of non-stored REPs, another choice is made: searching for a conserved pattern or searching a pattern with base frequency. When the complex task `extract REPs` is done, the REPs subclasses have to be identified. The complex task `Search REP subclasses` is a sequential task: first each REP sequence is aligned with the others, and the alignment score is used to build a distance matrix ; then multivariate analysis can be made followed by a clustering of the REPs. Obviously, the complex tasks `Multivariate analysis` and `Clustering` are tasks to be specialized, and a classification process must be done again. For each group defined in the clustering task, a consensus matrix has to be constructed : the task `Build consensus` is in turn a sequential task (multiple alignment in a way to establish a "consensus" sequence). Finally, when the complex task `Search REPs subclasses` is achieved, secondary structure prediction could be under study.

Since tasks are described as classes, it seems very natural to organize them into hierarchies. A task class may be a sub-class of an other task class (e.g. the task class `REPs` is a sub-class of the more general class `Palindromic-units` in figure 9).

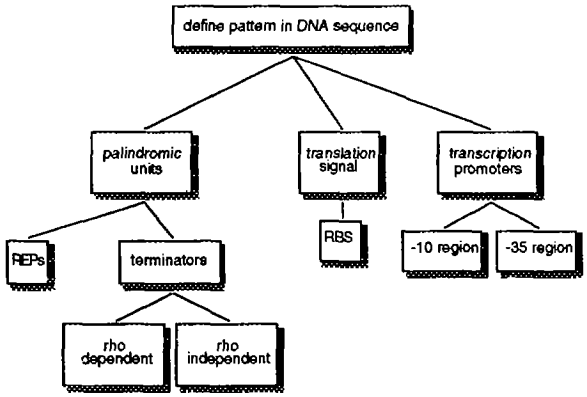

Figure 9 — Let us consider a high level task for pattern analysis in DNA sequences. The upper level task describes a general objective (`Define pattern in DNA sequences`). No specific solving strategy is associated with this task description: then using the classification process, this kind of task has to be decomposed into sub-tasks according to the control flow. At a lower abstraction level, the problem defined by a task is sufficiently precise so that a decomposition scheme can be associated to the task description. It is the case of the tasks `rho-independent`, `rho-dependent`, `REPs`, `RBS`, `-10 region` and `-35 region`.

The decomposition of a task into sub-tasks can then be different according to the sub-class retained, if required. Therefore, each time a task has to be decomposed, a classification process is executed in order to select, among the possible sub-classes, the class which matches the current context and which will provide the adequate decomposition. Thus, the overall problem solving process consists in mixing classification stages, where a task is characterized according to its input, and planning stages, during which the task is split up in sub-tasks according to the decomposition attached to the sub-class chosen during the classification stage.

Such a model supports interactively the user in choosing the methods necessary to solve his problem, and in chaining them in a proper way. It allows automatic solving as well as user-solving, if required (e.g. no available or known solving strategy, analysis of graphics, etc.).

Textual knowledge as object annotations

While developing a knowledge base, many modeling choices have to be made. Every entity of the real world may indeed be represented by several distinct object descriptions. The decisions are made by the designer according to the expected uses of the knowledge base and to his own perception. The explicit reasons which support these decisions should be accessible to the users and therefore stored in the knowledge base itself. In order to avoid indefinite loops, these reasons cannot be expressed with the knowledge model and natural language is certainly the most innate and easy to understand medium.

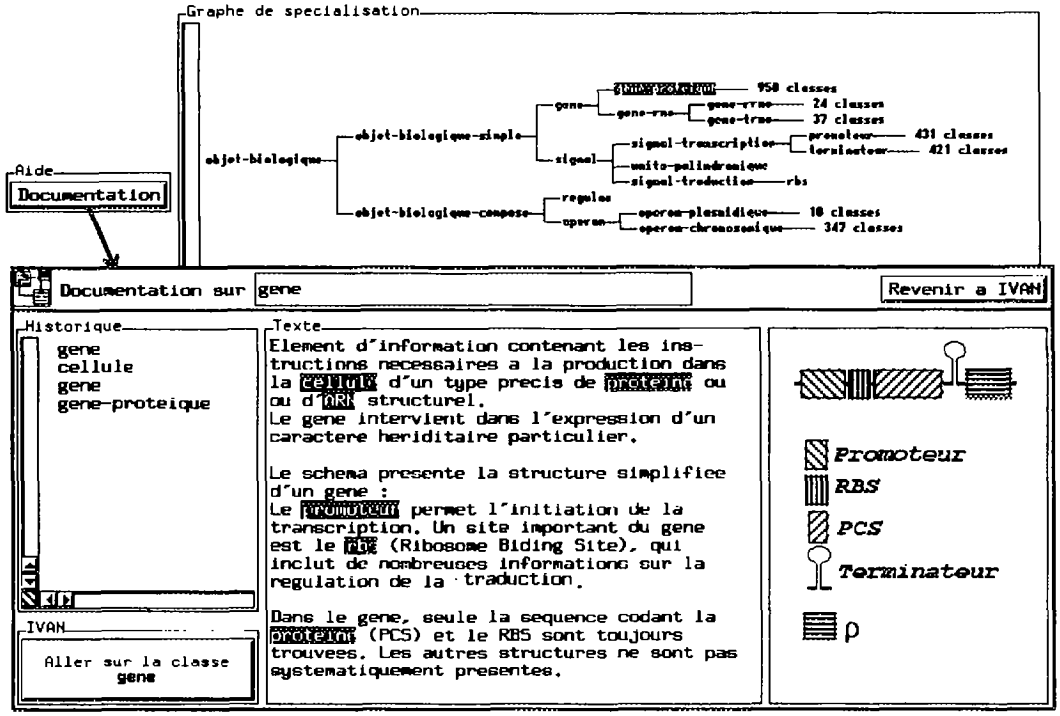

Figure 10 — The navigational interface of hypertext. The user has selected a class in the inheritance graph (upper window) and has consulted the associated node of the hypertext (lower window). Some of the words of the text are links to other nodes. From an hypertext node, it is possible to come back to the entity it is attached to. The example is drawn from the ColiGene knowledge base, in which only the highest levels of the inheritance graph have been annotated, essentially for pedagogical purposes.

Thus, pieces of text should be attached to the various object classes and to their components, such as attributes essentially. Navigating from a class to another, a user can read the associated text. But the text itself can be seen as a knowledge base, unable of course to support any inference mechanism. Arranging the various pieces of text into an hypertext (Conklin 1987) (Nielsen 1990) allows a navigation process to take place between its nodes. Moreover, the user can choose to navigate among the objects or among the hypertext nodes alternatively.
We have implemented the connection between objects and hypertext in our system. Every new piece of text is turned into an hypertext node automatically: the text is searched for words which match with class or attribute names. The node itself is named after the class or the attribute to which it is attached. Pictures can also be placed in the node and displayed together with the text at the user's request. Most of the classes and the attributes of the upper levels of the class hierarchy of ColiGene have been commented in that way (Figure 10).
What is the exact status of text in such knowledge bases? Text can be seen as simple comments as presented before. These comments can sustain pedagogical purposes: text attached to the highest levels in the class hierarchy can present fundamental notions of the scientific field the knowledge base is concerned with. Text can also be presented as the knowledge format which precedes formalization. This is the view supported for instance by (Gaines & Linster 1990) who uses hypertext for knowledge elicitation and acquisition. Even after textual knowledge has been formalized as inter-related objects, it can be conserved to play the role of comments. But text can also be seen as the most advanced form of compiled knowledge. Under this interpretation, as parts of scientific knowledge bases are progressively becoming consensual, they could be rewritten as texts and figures only, hopefully arranged in hypertexts: (hyper-)textbooks are perhaps the ultimate state in the knowledge life-cycle.

Conclusion

In the very future, the use of advanced knowledge models, together with the powerful reasoning mechanisms they support, will allow the development of large knowledge bases in molecular biology (Overton, Koile, & Pastor 1990). They will progressively include the descriptive, methodological and textual, or graphical, knowledge about large parts of this scientific domain. If such a knowledge base were allowed to be accessed by members of a scientific community which are involved in a common research project, it could become the place where consensual scientific knowledge emerges. Even if the situation is somewhat less complex than for the case of general knowledge bases which must also store and manage common sense knowledge (see for instance (Lenat & Guha 1990)), several technical problems have still to be solved in order to achieve this objective.
First of all, concurrency must be supported at the adequate level. Data base management systems do manage concurrency between multiple users, but at the data level. Users can simultaneously read or update the values of object variables and the DBMS, through appropriate protection and locking mechanisms, ensures the correctness of the transactions. But the situation for knowledge bases is quite different, since concurrency must be managed at the level of the conceptual scheme,

i.e. at the level of the descriptions of the classes in which the knowledge is described.
Then, the knowledge base management system must be able to judge the opportunity to include in the knowledge bases submissions from the different researchers. An adequate protocol must be provided, following the metaphor of paper submission to specialized scientific journals or conferences. A reviewing process should be able to decide if a submitted piece of knowledge contradicts or subsumes some existing piece of knowledge within the base and to send its comments back to the submitter. Fundamental research work must provide the necessary algorithms.
Finally, submitters should be allowed to develop their own private knowledge base and to submit parts of it to the other researchers or to the common shared knowledge base as presented above. Dedicated operations should allow them to merge knowledge pieces from several knowledge bases in order to test alternative hypotheses.
These requirements set very difficult and fundamental problems, both in the data base and the AI fields. But the resulting system would be an adequate support for co-operative distributed research efforts and should considerably accelerate consensual knowledge formation.

References

Bobrow, D.G., Qualitative Reasoning about Physical Systems : An Introduction, *Artificial Intelligence*, 24, p. 1-5, 1984

Borning A., The Programming Language Aspects of ThingLab, a Constraint-Oriented Simulation Laboratory. *ACM Transactions on Programming Languages and Systems*, 3(4):353-387,1981

Conklin J., Hypertext: an introduction and survey, *IEEE Computer*, Vol. 20, n° 9, 1987

Fikes R., Kehler, T., The Role of Frame-based Representation in Reasoning, *Communications of the ACM*, Vol. 28, n° 9, September 1985

Forbus, K.D., Falkenhainer, B., Self-Explanatory Simulations : Scaling up to large models, Proc. of the *10th National Conference on Artificial Intelligence AAAI-92*, July 12-16, San Jose, Ca, p. 685 -690, 1992

Fox M. S., Wright J. M., Adam D., Experiences with SRL: an Analysis of a Framed-Based Knowledge Representation. Proc. of the 1st International Workshop on *Expert Database Systems*, Kiawah Island, SC, 1984

Friedland P., L. Kedes, Discovering the secrets of DNA, *Computer*, November 1985, p. 49-69, 1985

Gaines B.R., Linster M., Integrating a Knowledge Acquisition Tool, an Expert System Shell and a Hypermedia System, *International Journal of Expert Systems*, Vol. 3, n° 2, p.105-129, 1990

Kim W., Bertino E., Garza J. F., Composite Objects Revisited. In Proc. of the *ACM/SIGMOD International Conference on the Management of Data*, Vol. 2, p. 337-347. Portland, 1989

Lenat D. B., Guha R.V., *Building Large Knowledge-Based Systems*, Addison-Wesley Publ., 1990

Mariño O., Rechenmann F., Uvietta P., Multiple Perspectives and Classification Mechanism in Object-Oriented Representation, *9th ECAI* (August 8-10, 1990, Stockholm), Pitman Publishing, London, 1990

Mavrovouniotis, M., *et al.*, Computer Aided Synthesis of Biochemical Pathways, *Biotechnological Bioengineering*, Vol. 36, p. 1119-1132, 1990

Minsky M., A Framework for Representing Knowledge, in P.H. Winston ed., *The Psychology of Computer Vision*, p. 211-281, McGrawHill 1975

Nielsen J., *Hypertext and Hypermedia*, Academic Press, 1990

Overton G.C., Koile K., Pastor J.A., GeneSys: A Knowledge Management System for Molecular Biology, in Bell G.I., Marr T. (eds), *Computers and DNA*, Addison-Wesley Publ., p. 213-239, 1990

Perrière G., F. Dorkeld, F. Rechenmann, & C. Gautier, Object-Oriented Knowledge Bases for the Analysis of Prokaryotic and Eukaryotic Genomes, these proceedings,1993

Rechenmann F., Uvietta P., Shirka - An object-centered knowledge base management system, in *Artificial Intelligence in Numerical and Symbolic Simulation*, ALÉAS Publ, Lyon, 1991

Rechenmann F., Rousseau B., A development shell for knowledge-based systems in scientific computing, in Houstis E.N., Rice J.R. (eds), *Expert Systems for Numerical Computing*, Elsevier Science Publishers, Amsterdam, 1992

Shin D.G., Lee C., Zhang J., Rudd K.E., Berg C.M., Redesigning, implementing and integrating *Escherichia Coli* genome software tools with object-oriented database system, CABIOS, Vol. 8, p. 227-238, 1992

Weld, D.S., The Use of Aggregation in Causal Simulation, *Artificial Intelligence*, Vol. 30, p. 1-34, 1986

Winston M. E., Chaffin R., Herrmann D., A Taxonomy of Part-Whole Relations. *Cognitive Science*, 11:417-444, 1987

Woods W. A., Understanding Subsumption and Taxonomy: A Framework for Progress. In J. F. Sowa, editor, *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, p. 45-94. Kaufmann, San Mateo, CA, 1991