TR-0811

A Private Knowledge Base for Molecular Biological Research

by
H. Tanaka

October, 1992

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

A Private Knowledge Base for Molecular Biological Research

Hidetoshi Tanaka
Institute for New Generation Computer Technology (ICOT)
1-4-28, Mita, Minato-ku, Tokyo 108 Japan
htanaka@icot.or.jp

Abstract

This paper proposes a private KB system for biological information processing. It describes an experimental system developed at ICOT to evaluate the effectiveness of components of this private KB system.

A private KB system is invaluable in studying molecular biology, for example, to clarify the unknown functions of proteins empirically. Such a system would simplify the access and the update of existing public DBs and encoded biological knowledge. It would facilitate the effective sharing of various private knowledge among researchers.

A private KB system for biology should encode all the existing public biological DBs and various biological knowledge in definite semantics. The system would also require an efficient DBMS to retrieve the large volume of data in its public DBs and a KBMS to represent biological descriptions in hierarchical modules and to maintain intra-module consistency. For these requirements, we believe that the DBMS, Kappa-P, and the KRL, *Quixote*, both designed and developed at ICOT, would be useful to form such a private KB system. The nested data model and the parallel processing capability of Kappa-P provides computational efficiency while the object-oriented and deductive power of *Quixote* supports advanced query processing and KB management.

1 Introduction

Molecular biological information processing is increasing in importance, as the biological laboratories are improving in their computational capabilities and biological data are increasing much faster than our understanding. To speed up biological knowledge derived from such data, biological databases (DBs) should provide appropriate features for biological data management and query processing.

Public DBs, such as PIR, PDB, and GenBank, contain hundreds of megabytes of data and are growing rapidly, however, recent enhancement in computer allow biologists to store such data in rather small systems and to use them as part of value-added private DBs. Such personal environments also allow them to create DBs for experimental data, knowledge bases (KBs) for their private knowledge, and rule derivation among raw data, processed data, and knowledge in the future. To realize such a value-added private DB (a private KB), we employ new concepts of database management systems (DBMSs) and knowledge representation languages (KRLs).

In this paper, we focus on an integrated KB which is part of the molecular biological information processing system of the Japanese Fifth Generation Computer System (FGCS) project.

We are building a protein KB system in the framework of a deductive object oriented database (DOOD), which consists of KRL, *Quixote*, and DBMS, Kappa P. The reason why we choose protein information is due to the moderate amount of data needed to store and study. As biological applications are new, we decided to check the appropriateness of both Kappa-P and *Quixote*, and requested that several facilities be added to them.

Kappa-P and *Quixote* are developed on a parallel inference machine (PIM). The parallel DBMS, Kappa-P, employs a nested relational model and has the extensibility suitable for parallel processing and amino acid/DNA sequence retrieval. *Quixote*, a DOOD KRL, provides sophisticated query processing capability with advanced concepts, such as objects with definite identity and subsumption relations, modules with submodule relations, and the flexibility, to represent various forms of knowledge.

This paper is organized as follows. We describe the purposes of a private KB system in Section 2, the requirements for biological DB/KB systems and related works in Section 3, the suitability of Kappa-P and *Quixote* as ingredients of the system in Sections 4 and 5, and the behavior of the system in Section 6.

2 Purposes of a Private KB

Our aim in developing a private KB is to build a recursive system which consists of knowledge derivation from DBs/KBs and knowledge representation of the derived rules in KBs. We call it *knowledge discovery* system. In this section, we show our view on biological information processing, as well as on private KB.

2.1 Biological Information Processing

Scientific methods, including biological information processing, are recursive systems which consist of knowledge derivation and representation. Researchers conduct experiments, represent data in some form, derive knowledge from them, and proceed to the next experiments. We would like to add two procedures to make this process semi-automatic: the representation of knowledge in a proper KRL, and the derivation

of further knowledge from the represented knowledge and the experimental data.

Currently, the biological knowledge is processed and derived by reserchers. However, it becomes harder to process the ever-increasing amount of experimental data manually, let alone to abstract relevant knowledge out of processed data. Computer would play an important role to aid the biologists in their analysis and derivation. Right now, the final derivation must still be done by hand, and thus, knowledge presentation by graphic user interfaces (GUI) is also crucial to improve researchers' views. (see Section 3).

We consider the following to be the first steps toward a knowledge discovery system:

- integration of various data analysis tools and DBs;
- integration of public DBs, biological knowledge, private experimental results, and personal hypotheses; and
- integration of a KRL and a DBMS for a KB system.

We regard private KB as the knowledge representation module of the knowledge discovery system and will study its another component, knowledge derivation modules, in future. We believe that private KBs will simplify the sharing of various private knowledge among researchers, and among analysis tools as well.

2.2 Ingredients of a Private KB

Private biological KB should contain the following:

(1) public databases (PIR, PDB, ProSite, etc.);
(2) biological/biochemical knowledge (e.g. protein functions);
(3) private experimental results; and
(4) personal hypotheses/derived rules.

And a KBMS to support such a private KB should have the following capabilities:

(5) primitive access methods for biological use (motif search, multiple alignment, etc.);
(6) knowledge management (add, delete, and replace knowledge; inconsistency checking; etc.);
(7) inference; and
(8) presentation (e.g. GUI).

We classify that how to integrate public DBs and how to implement primitive access methods are DBMS problems, whereas how to represent domain intrinsic knowledge and experimental results, how to treat personal hypotheses, and how to implement knowledge management and inference are KBMS problems.

3 Biological DB & KB: Requirements and Related Works

In this section, we show the requirements for biological DB, DBMS and KRL as motivations for developing a private KB system, and give an overview of related works.

3.1 Integrated DBs

Most of the requirements for existing molecular biological DBs can be reduced to the problem of accessing several DBs at once. This is due to the semantic differences between the DBs – the attributes' meanings, the values' variations, and their relations – all must be understood beforehand.

Such requirements can be solved by DB integration. There are three approaches.

Standardization

Standardization is the most fundamental form of integration. It provides the simplest environment for the wide use of DBs.

CODATA (Committee on Data for Science and Technology) in ICSU (International Council of Scientific Unions) proposed the standardization of attributes to realize virtual integrated DB [10]. The schema of every public DB should be a subset of the virtual schema. NLM (National Library of Medicine) provides GenInfo Backbone Database, and according to [8], it is built as a standardized primary DB, which is assumed to be a basis for secondary, value-added DBs for the specialized interests of different biologists.

Determining a standard, however, needs a lot of manpower, discussion, and negotiation. Moreover, it is difficult to make a wide virtual schema enough to cover all the attributes of existing protein information.

Integrated User Interface

Building an integrated user interface is the fastest way of getting an integrated environment. It provides not only query processing facilities but also visual browsing facilities, which are quite attractive and useful for biologists. It used to need a lot of development, however, to remake an interface when a new DB was to be added.

GeneWorks[1] and Entrez[2] provide integrated environments to enable access to existing DNA/amino acid sequence DBs from a PC or Mac. They are, however, packaged for browsing only and are not for adding new applications or DBs.

Smith and his coworkers are developing an environment for genetic data analysis (GDE[3]) which will help access several DBs at once by providing data exchange tools between representative DBs. The first version is based on a multiple alignment editor and allows sequence analysis tools to be included. It provides many widgets to reduce the efforts required to remake interfaces.

At ICOT, we have developed an experimental GUI for protein feature descriptions of PIR and PDB. We regard it as a knowledge presentation module for our private KB system, because this level of integration is not yet sufficient for knowledge discovery purposes.

[1] IntelliGenetics, Inc., Mountain View, CA
[2] National Institute of Health, Bethesda, MD
[3] Genetic Data Environment

Knowledge Base Approach

Our approach is to build individual KBs and to integrate them. There are three stages in our KB approach. First, at the *DB stage*, every DB is built independently. Second, at the *KB stage*, intrinsic access methods for the DB are installed, and knowledge on attributes and values of the DB is collected. A DB and its intrinsic methods/knowledge are managed by a KBMS.

At the *integration stage*, it could simplify to build an integrated KB which provides data and knowledge for biologists without showing detailed information on its data/knowledge structure. It is because methods and knowledge of each DB would be ready to be retrieved and used by application programs and biologists.

In the *DB stage*, the standardization of data exchange and flexibility of data structures are important. Standardization will help us in collecting and storing plenty of biological DBs into a DBMS. Meanwhile, the DBMS should support enough capability and flexibility of data structures, because the complex data structures of biological DBs often change as research proceeds.

In the *KB stage*, expressive power of representation is important, rather than storage efficiency or processing performance. We aim to manage as various knowledge and methods as possible (see Section 3.3), though it is impossible to satisfy requirements of all biologists. We consider that every biologist should have own private KB, which contains methods and knowledge corresponding to its requirements.

3.2 Information Retrieval in DBMS

Among the requirements for DBs, we can find some that apply to DBMS, that is, the facilities of information retrieval for the sequences of DNA, RNA, and amino acids. This is partly because the concept of DBMS is rather wider for biologists than the traditional concept is for DB researchers.

Sequence retrieval is presently regarded as full text searching, except for some differences in the search criteria: similarity searching via dynamic programming (DP) or other algorithms (ex. BLAST[2]), with given similarities between characters (namely, amino acids).

Two approaches are considered: parallel processing and proper indexing. We are considering indexing to shorten the search area, but this will not get the expected results directly [1]. Thus, parallel processing is valuable when we cannot find any proper indexing for sequence similarity searches (see Section 4).

3.3 Knowledge to Represent in KRL

Knowledge on Existing Public DBs

In the *KB stage*, we should accumulate various supplementary knowledge, or intrinsic methods. It seems almost impossible to define operations common to all knowledge just as relational algebra to relational DB. Thus, new concepts should be introduced to the system, so that intrinsic methods can be defined in each item or cluster of knowledge. DOOD is a promising concept for the mechanism.

Biological/Biochemical Knowledge

It is also important to represent biological or biochemical knowledge and to use them as the supplementary knowledge mentioned above or as part of the knowledge discovery system. For example, protein functions and structures should be represented in a KRL so that we can describe relationships between them. Protein functions vary in their representation, such as chemical reactions, physical connections, and thermodynamic interactions. Thus, we need is rich expressive power rather than efficiency of a KRL for this purpose.

Derived Rules/Personal Hypotheses

The most probable usage of a private KB system is to evaluate personal hypotheses, or rules derived from a certain data set, with the KB. The hypotheses should be represented in a KRL to be evaluated through the KB. Besides, to aim at a knowledge discovery system, it should be considered how to represent discovered knowledge. These require definite semantics of KB and inference facility.

Quality Management

In treating molecular biological data, we should consider at least two kinds of errors: *experimental errors* and *identification errors*. Experimental errors are inevitable in molecular biological DBs. Experiments are repeated to reduce such errors. In reading DNA sequences, for example, the *same region* should be repeatedly read to verify the result. Identification errors possibly occur in this verification process. It is not so clear whether we can get the sequence of the *same region* because of slightly different repeating regions, or natural errors such as diseases or mutations.

Representation of relations between proteins and functions are more ambiguous than relations between loci and DNA sequences in the example above. We should always consider identification errors in both proteins and functions. As experimental facts are accumulated, for example, "cytochromes transfer electrons" may turn into relations such as "cytochromes and ubiquinone transfer electrons" (protein identification is relaxed) or "cytochromes transfer electrons to generate energy" (function identification is detailed).

Then a concept of object identity and the subsumption relations between objects are important in such cases. Results containing experimental errors should be treated as different objects to avoid integrity problems and as objects when we ask to verify them.

4 Biological Databases in Kappa-P

We use a parallel nested relational DBMS Kappa-P as a component in our experimental private protein KB system. Kappa P provides the efficient data processing in the system, using its nested relational model and parallel processing.

In this section, we show the effectiveness of Kappa-P, especially its efficiency and its congeniality with

Quixote, or with a DOOD concept, through the configuration and behavior of the DBMS module of our experimental system.

4.1 Public Protein DBs in Nested Relations

In the experimental system, PIR (Release 30), PDB (of Nov. 1991), and ProSite (Release 9) are represented in nested relations. A nested relational model is proposed by Makinouchi[7]. This model reduces the number of tables and the join operations which cost thousands of machine instructions per operation. Its efficiency for biological DBs has already been shown in a sequential DBMS, Kappa[14].

Figure 1, using feature descriptions of a protein in PIR and PDB as an example, shows an intuitive illustration of:

(1) a nested relational model;
(2) a relational model;
(3) a data model of Kappa-P, which is a nested relational model with a term-type; and
(4) an object hierarchy, which is part of the object oriented concept.

(1) In a nested relation:

id_code	features	
	position	description
CCRZ	22, 25	heme binding
	26, 88	
	10–22	helix 1

(2) In a relation:

id_code	position	description
CCRZ	22, 25	heme binding
CCRZ	26, 88	heme binding
CCRZ	10–22	helix 1

(3) With a term-type:

id_code	features	
	position	description
CCRZ	and(22,25)	heme binding
	and(26,88)	
	range(10,22)	helix 1

(4) In an object hierarchy:

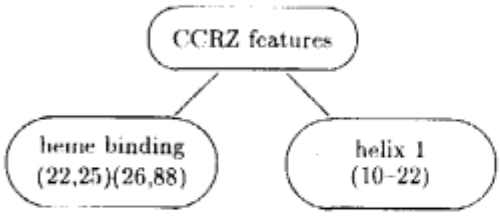

Figure 1: Data Models for Protein Features

It is represented naturally in nested relations that a "heme binding" site has two pairs of positions. Meanwhile, in the relational model, this is shown by the answers given to a proper query ((1) and (2)). Moreover, Kappa-P has broader capability of representation by allowing term-type (attribute "position" of (3)).

Object-oriented (OO) data model is also a promising concept to apply to such case. Overton et al. show the need for hierarchical representation in DNA feature description and efficiency of an OO model[9]. Gray et al. represent PDB in an OODB with a functional data model[4]. When implementing an OO model, existing DBMS is often used as a basis of the system. The nested relational model is more appropriate than the relational model for this purpose. Because, for example, the structure of (1) and (4) are so similar that proper optimization of representation and management can be expected.

4.2 Extensibility in DB Commands

Kappa-P has two levels of query languages: primitive commands and KQL, an extended relational algebra. Primitive commands are valuable for improving efficiency[6].

Kappa P also has an extensibility in its command-set. The aim of the extensibility is to reduce the interaction with applications and to customize the command interface for each application. The modules of an application which frequently use DB commands are included in the DBMS to decrease the number of communications among processes.

We designed an experimental character-pair based indexing system, especially for the motif search[1]. A motif dictionary, such as ProSite[3], could also be used as another useful index for sequence similarity searches. Extensible DBMS is so flexible that such improved indexing systems could be added to the system rather easily.

4.3 Motif Search by Parallel Processing

Figure 2 shows Kappa-P query processing. We use a special mechanism to improve parallel processing efficiency, when realizing motif searches in the experimental system. It also uses the extensibility of Kappa-P.

Parallel DBMS Kappa-P consists of a server DBMS, many local DBMSs, and an interface process. The server DBMS has a global map of local DBMSs; the interface process coordinates local DBMSs to deal with users' request; and local DBMSs hold users' data [6]. The effectiveness of parallel processing is reduced when all the answers from local DBMSs are passed to the interface process to be selected and merged into one answer. To avoid such a situation, user defined commands, motif search programs in this case, are thrown to every local DBMS. They play the role of filters between local DBs and the interface process. The filters select proteins including the given motif patterns and send them to the users through the interface process.

5 Biological Knowledge in *Quixote*

Quixote is basically a deductive system equipped with the facilities for representing and classifying var-

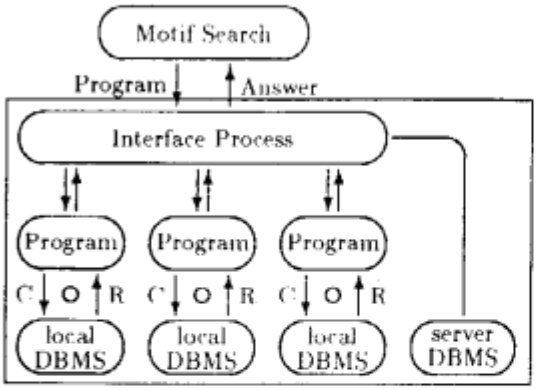

Figure 2: Kappa-P Query Processing

ious kinds of knowledge. It is as another key component of the KB system.

In this section, we give an overview of *Quixote*, and its effectiveness of knowledge representation, through some biological/biochemical representation examples. Its facilities are shown in [13] in detail.

5.1 Objects and Rules of *Quixote*

In *Quixote*, there are objects and modules. Objects have object identifiers and properties, which may be defined in rules. Subsumption relations among object identifiers define property inheritance. On the other hand, the modules are defined as sets of objects and submodule relations among modules define object inheritance.

Objects

Objects in *Quixote* are represented by extended terms called *object terms* [13]. An object term is of the following form:

$$o[l_1 = v_1, \ldots, l_n = v_n]$$

where o is called the *head* of the object term, l_i is a label, and v_i is the value of the l_i of the object term.

The labels and values of an object term represent the *properties* (intrinsic properties) of the object which are required for identification.

In this sense, object terms play the role of *object identifiers*. An object may have properties other than those specified in its object term. To represent such properties (extrinsic properties) of an object, a special form of term representation called an *attribute term* is used:

$$o[l_1 = v_1, \ldots, l_n = v_n]/[l'_1 = v'_1, \ldots, l'_m = v'_m].$$

This attribute term represents that the object identified by the object term $o[l_1 = v_1, \ldots, l_n = v_n]$ has properties $[l'_1 = v'_1, \ldots, l'_m = v'_m]$. It is important to distinguish intrinsic properties from extrinsic ones.

Simplified examples are shown in Figure 3. (1) and (2) describe the same object and their properties (values of `label2`) contradict each other while (1') and (2') represent different objects.

```
object_head [ol1 = ov1, ol2 = ov2, ...] /
            [al1 = av1, al2 = av2, ...]

      (1)  fact [label1=v1] / [label2=v2].
      (2)  fact [label1=v1] / [label2=v3].

      (1') fact [label1=v1, label2=v2].
      (2') fact [label1=v1, label2=v3].
```

Figure 3: Examples of *Quixote* objects

In addition, subsumption relations are defined among objects, and inheritances of extrinsic properties are also defined according to the relation[13].

Modules and Submodule Relations

Simplified examples for modules are shown in Figure 4.

```
(1)  module1 :: object1.

(2)  module2 >- module1.
(3)  module2 :: {{ object2. object3. }}

(4)  module3 >- module1.
(5)  module3 :: object3.
```

Figure 4: Examples of *Quixote* modules

"`object1` is an object in `module1`" is represented as (1). (2) represents a submodule relation specifying that `module2` inherits all the objects from `module1`. (3) is an abbreviation form, and (4) represents another submodule relation. Therefore, `module2` has `object1`, `object2`, and `object3`, whereas `module3` has `object1` and `object3`.

Although `object3` is in both `module2` and `module3`, it may have different properties in each module, because any relations between `module2` and `module3` are not defined. We can give different properties to the same object in different modules. Thus, we can use different modules to avoid DB inconsistency when we get conflicting results from experiments.

5.2 Knowledge Accumulation

We show how knowledge is represented and accumulated in *Quixote* through the example of the zinc finger motifs. The left-hand side of Figure 5 shows the structure of zinc finger. Zinc finger motifs are usually represented in string patterns such as "C x(2,4)-C-x(12)-H-x(3,5)-H" from the viewpoint of protein sequences.

Experimental knowledge is often accumulated hierarchically as experiments proceed. Imagine a case that a zinc finger motif was previously known as a member of DNA-protein interaction domain motifs, and by biological experiments we find that it has three types (as shown on the right-hand side of Figure 5).

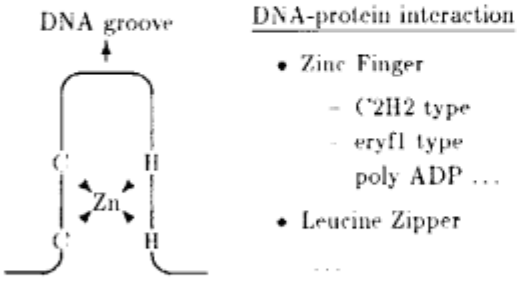

Figure 5: Zinc Finger Motif

In *Quixote*, all objects may have child objects, so we can easily add a new object as a child of an existing object. At first, we describe the zinc finger as a motif, with its function descriptions in module **zf_old**.

```
zinc_finger =< motif;;
zf_old::zinc_finger /
            [function_target = dna_groove,
             function = bind];;
```

When the following three motifs are found by experiments, they can be represented in another module **zf_new** with a submodule relation between **zf_old** and **zf_new**.

```
zf_new >- zf_old;;
zf_new::{{
 zinc_finger[name="C2H2 type"] /
   [pattern="C-x(2,4)-C-x(12)-H-x(3,5)-H"];;
 zinc_finger[name="eryf1 type"] /
   [pattern="C-x-N-C-x(4)-T-x-L-W-R-R-
                    x(3)-G-x(3)-C-N-A-C"];;
 zinc_finger[name=
             "poly ADP ribose polymerase"] /
   [pattern="C-K-x-C-x-E-x(3)-K-x(3)-R-
                x(16,18)-W-Y-H-x(2)-C"]};;
```

The attributes **function_target** and **function** are inherited by **zinc_finger** objects in **zf_new**. Thus, the following question and answers are realized.

```
(Q) ?- zf_new:X/[function_target=
                    dna_groove]||{X=<motif}.

(A) X=zinc_finger[name="C2H2 type"]
    X=zinc_finger[name="eryf1 type"]
    X=zinc_finger[name=
              "poly ADP ribose polymerase"]
```

5.3 Biological/Biochemical Knowledge

As shown in Section 3, it is important to represent biological/biochemical knowledge, especially protein functions in a KRL. For example, we show how the function of cytochromes is represented in *Quixote* (Figure 6). Other examples are shown in [11].

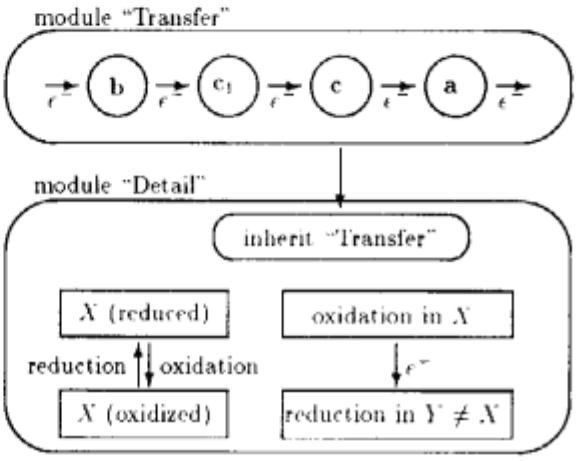

Figure 6: Mechanism of Cytochromes

This description consists of two modules ("transfer" and "detail") in order to meet two levels of queries. One module is for the global mechanism of the activities of cytochromes. To simplify, we choose a Prolog-like style example by extracting a part of the knowledge. This describes the neighborhood relationship of cytochromes and a recursive rule to represent their total mechanism.

```
transfer::{{
el_trans[from=cytochrome_b,to=cytochrome_c1];;
el_trans[from=cytochrome_c1,to=cytochrome_c];;
el_trans[from=cytochrome_c,to=cytochrome_a];;
line[from=X,to=line[from=Y,to=Z]]/
    [start=X,end=Z1] <=
      el_trans[from=X,to=Y],
      line[from=Y,to=Z]/[start=Y1,end=Z1];;
line[from=X,to=Y]/[start=X,end=Y] <=
      el_trans[from=X,to=Y]}};;
```

The other is for its detailed mechanism. It contains the knowledge on oxidation and reduction of every cytochrome.

```
detail::{{
oxidation[object=X]<= el_trans[from=X,to=Y];;
reduction[object=Y]<= el_trans[from=X,to=Y];;
redox[object=X]/[red=reduction[object=X],
                 ox=oxidation[object=X]] <=
  reduction[object=X], oxidation[object=X];;

c_line[from=oxidation[object=X],
       to=line[from=redox[object=Y],to=Z]]
    /[start=X, end=Z1] <=
  el_trans[from=X, to=Y],
  redox[object=Y]/[red=YR,ox=YO],
  c_line[from=YO,to=Z]/[start=Y1,end=Z1];;
c_line[from=oxidation[object=X],
        to=reduction[object=Y]]
    /[start=X,end=Y] <=
  el_trans[from=X,to=Y]}};;
```

Therefore, when we assert the following query, each module replies with different answers. The module "transfer" replies with an electron transfer route only while the answer from "detail" includes the mechanism for each oxidation and reduction.

```
?- transfer:line[from=X,to=Y] /
   [start=cytochrome_b,end=cytochrome_a].
?- detail:c_line[from=X,to=Y] /
   [start=cytochrome_b,end=cytochrome_a].
```

5.4 Quality Management

Flexibility Along Subsumption Relation

It is difficult to give biological data/knowledge objects the clearest identifier instantly. Identification requires the flexibility to allow trial-and-error and the facility of subsumption relation description.

An identifier sometimes has to be generalized or specialized. For example, the sentence "cytochromes have a certain feature" sometimes has to be reconsidered as "cytochromes and hemoglobins have a certain feature" or "cytochrome c has a certain feature." It seems rare to completely misidentify different objects. Most erroneous identifiers have to change only their abstraction level, and need not be altered completely.

In the process of determining the clearest identifier, we feel it is useful to have DBMS which accepts tentative identifiers which can be specialized or generalized later on. We could use trial and error to determine the proper identifier. Figure 7 is an example of how they are represented in *Quixote*.

As experiments are repeated, the identifier of the protein which has some feature ('featX') may be changed. *Quixote* expressions (**fact1**) and (**fact2**) are examples of the identification of experimental results. (**fact1**) mentions nothing about the (**fact2**) attribute "type", and (**fact2**) mentions nothing about the (**fact1**) attribute "lifename". When we consider what sort of protein has 'featX' and get

```
(fact1)cytochrome[lifename=e_coli]/
            [feature=featX].
(fact2)cytochrome[type=c]/[feature=featX].
(hyp1) cytochrome/[feature=featX].
(cf.1) cytochrome( E.Coli, _, featX )
       cytochrome(      _, c, featX )
(cf.2) cytochrome(lifename(e_coli),type(c)],
                  featX )
(hyp2) protein[name={cytochrome,hemoglobin}]/
            [feature = featX].
```

Figure 7: Proteins as objects in *Quixote*

(**fact1**) and (**fact2**), we can easily think of a hypothesis (**hyp1**). We can also get this hypothesis from *Quixote*, using the object lattice of the subsumption relation. Moreover, if we give some relations among 'cytochrome', 'hemoglobin', and 'protein', another hypothesis such as (**hyp2**) is available.

In the relational data model or Prolog, it is necessary to redesign the attributes of tables or arguments of facts to reflect such schema changes, since attributes have to be fixed. This is shown in (**cf.1**). In Prolog, we can reflect these by using lists as shown in (**cf.2**) [15][16]. However, it is necessary to support a particular unifier for the list, and users must manage the meaning of the list (e.g., connected by 'and' or 'or') carefully. *Quixote* allows set concepts with particular semantics to avoid such mismatches.

Modules for Data Management

To distinguish tentative identifiers from fixed ones, or experimental results from derived ones, a facility for making modules is desirable. This allows local integrity to be checked within the module and the global uniqueness of the labels of the identifiers to be shelved.

As well as the objects of experimental results, the verified results and public DBs have to be distinguished by modules, and be checked by different integrity checking methods.

Figure 8 shows an example of the verification process. Upper modules inherit all the facts and rules of their lower modules. '**PIR**', '**Swiss-Prot**', and '**Experimental Results**' are modules each of which allows local integrity checking. If the identifiers between modules conflict with one another, they can be reconciled in their parent module.

'**Sequence**' has some rules and cross-references between PIR and Swiss-Prot so that it can select and reply to specific sets of protein sequences in these public DBs. '**Integrated**' has some rules to verify experimental results by merging the selected sequences from public DBs. It also has cross-references between public DBs and experimental results (but they are ignored in Figure 8 to simplify the example).

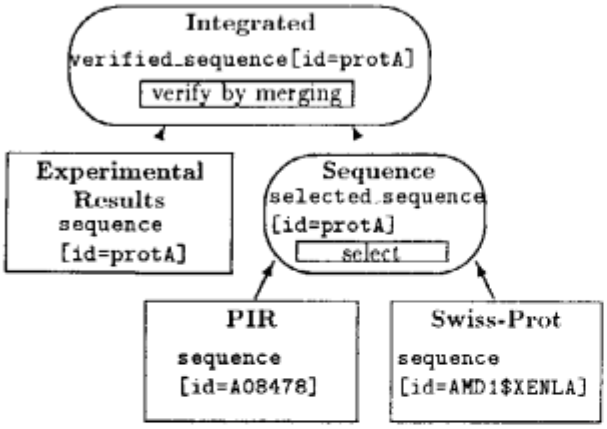

Figure 8: Modules for Verification Process

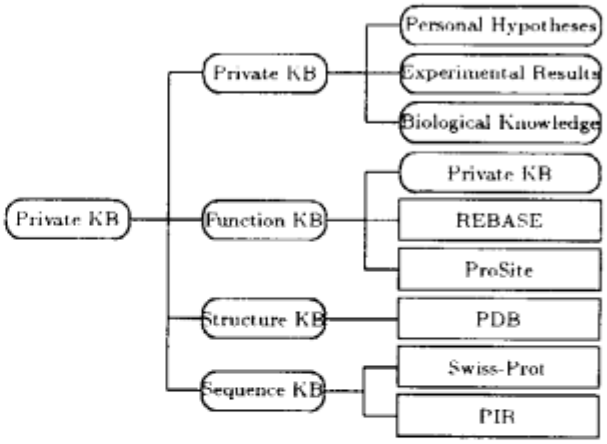

Figure 9: Integrated Knowledge Base of Proteins

6 An Integrated Private KB System

In this section, we show the configuration and the behavior of the private KB system for biological information. We can find two aspects of integration in the system. One is its contents, which are individual KBs and DBs. The other is an integration of the containers, which are KRL *Quixote* and DBMS Kappa-P We have made several experimental systems approximating to such a system[12], that is, a biological DB in Kappa-P and several experimental descriptions in *Quixote*.

6.1 Databases and Knowledge Bases

As mentioned in Section 2, the private KB should contain the following:

(1) public DBs;
(2) biological/biochemical knowledge;
(3) private experimental results; and
(4) personal hypotheses/derived rules.

Our system will hold several DBs, including public ones such as those shown at the bottom of Figure 9. An oval represents a module of rules and facts, and a rectangle represents a Kappa-P DB.

Public DBs and primitive accession methods are managed in Kappa-P. The DB part of private experimental results may also be managed in it. Meanwhile, biological and biochemical knowledge such as protein function descriptions, derived rules and other knowledge parts of private experimental results, and personal hypotheses, are represented in *Quixote*.

The experimental system in Kappa-P contains three public DBs: PIR, PDB, and ProSite. We implemented a parallel motif search algorithm and an integrated GUI for protein feature descriptions as primitive accession methods. We described some of the knowledge of biology and biochemistry in *Quixote*, which is shown in Section 5.

6.2 DBMS and KRL

In Section 2, we also mentioned to the requirements for KRL:

(5) primitive access methods for biological use;
(6) knowledge management;
(7) inference; and
(8) presentation.

Quixote system provides important facilities required for *knowledge information processing*, such as knowledge representation, management and inferences. *Quixote* also provides basic functions for constructing an integrated KBMS on top of Kappa-P. In our system, *Quixote* will interface KBs with DBs, as well as with applications. So there arethree interactions appear in the system (Figure 10).

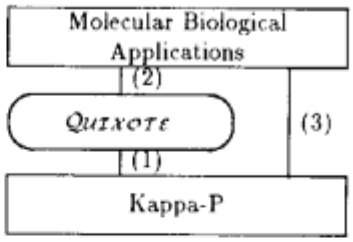

Figure 10: Integrated System of Kappa-P and *Quixote*

(1) Interactions between Kappa-P and *Quixote*

Kappa-P will act as a DB engine for a KB written in *Quixote*. All facts (non-temporal objects) in *Quixote* will be stored in Kappa-P, and Kappa-P will activate necessary objects as the result of

retrieval. In order to realize this integration, however, several refinements are needed in both systems.

(2) Applications and *Quixote*

At present, the user interface of *Quixote* is based on query commands. It is enough for writing application programs, but it is not convenient for biologists' general uses. A proper GUI is needed. The command interface should support standard forms of the molecular biological data (ASN.1, for example) to exchange data and knowledge with other systems.

(3) Applications and Kappa-P

The system should support direct access to DBs for simple queries. An experimental system in Kappa-P currently supports a GUI, to display protein feature descriptions of PIR and PDB together with their amino acid sequences, and invoke the parallel motif search program.

6.3 Toward a Knowledge Discovery System

Ishikawa et al. have developed a parallel processing algorithm of protein multiple alignment [5]. When this multiple alignment system and the KB are connected, and a new multiple alignment algorithm using motifs is developed, it becomes an integrated application and KB system. This is expected as a first step toward an automatic motif extraction and motif accumulation.

7 Concluding Remarks

Biological applications are exciting field of study for the researchers in DBMS and KRL. For such application, DBMSs and KRLs are required to have various functions: information retrieval, deduction, identification, module concepts, extensibility, and parallel processing. The existence of a private KB, consisting of various existing public DBs, would support the next phase of this research: biological knowledge discovery. Such new effective DBMS/KRL facilities should also be found and proposed by computational biologists. It is important to cooperate with them to conduct further research.

Meanwhile, the demand of information and resources would often exceed the capacity of individual DBs and KBs. Obviously, distributed DBMS is required. We believe DOOD with extensible DBMS would also play an important role for this purpose.

Acknowledgments

The author wishes to thank Kazumasa Yokota, Hideki Yasukawa, and Moto Kawamura for their valuable comments on earlier versions of this paper. The author also thanks the people in the *Quixote* project, Kappa-P project, and biological DB project for their great efforts to realize systems, and the people on the computer-biology mailing list for their suggestions from the viewpoint of biology.

References

[1] Abiru, Y. and Tanaka, H.: "Motif Searches: with an Indexing and a Parallel Processing", *(draft)*.

[2] Altshul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J.: "Basic Local Alignment Search Tool", *J. Mol. Biol.* 215, pp.403-410, (1990).

[3] Bairoch, A.: *PROSITE: A Dictionary of Protein Sites and Patterns, User's Manual*, (May 1991).

[4] Gray, P.M.D. et al.: "An Object-Oriented Database for Protein Structure Analysis", *Protein Engineering*, vol.3 no.4, pp.235-243 (1990).

[5] Ishikawa, M., Hoshida, M., Hirosawa, M., Toya, T., Onizuka, K. and Nitta, K.: "Protein Sequence Analysis by Parallel Inference Machine" *FGCS 92*, (Jun 1992).

[6] Kawamura, M., Sato, H., Naganuma, K. and Yokota, K.: "Parallel Database Management System : Kappa-P", *FGCS 92*, (Jun 1992).

[7] Makinouchi, A.: "A Consideration on Normal Form of Not-Necessarily-Normalized Relation in the Relational Data Model", *VLDB 1977*, (1977).

[8] NCBI: "GenInfo Backbone Database", Version 1.59, *Draft* (Apr 1990).

[9] Overton, G.C., Koile, K. and Pastor, J.A.: "GeneSys: A Knowledge Management System for Molecular Biology", *Computers and DNA, SFI Studies in the Science of Complexity*, vol.VII, Addison-Wesley, pp.213-239 (1990).

[10] PIR-International (JIPID): *PIR Newsletter*, No.3 (June 1990).

[11] Tanaka, H.: "Protein Function Database as a Deductive and Object-Oriented Database", *Database and Expert Systems Applications*, Springer-Verlag, pp.481-486 (Aug 1991).

[12] Tanaka, H.: "Integrated System for Protein Information Processing", *FGCS 92*, (Jun 1992).

[13] Yasukawa, H., Tsuda, H. and Yokota, K.: "Objects, Properties, and Modules in *Quixote* ", *FGCS 92*, (Jun 1992).

[14] Yokota, K. and Tanaka, H.: "GenBank in Nested Relation", *Joint Japanese-American Workshop on Future Trends in Logic Programming* (Oct 1989).

[15] Yoshida, K., Overbeek, R., Zawada, D., Cantor, C.R. and Smith, C.L.: "Prototyping a Mapping Database of Chromosome 21", *Proceedings of Genome Mapping & Sequencing Meeting*, Cold Spring Harbor Laboratory, (1991).

[16] Yoshida, K., Smith, C., Kazic, T., Michaels, G., Taylor, R., Zawada, D., Hagstrom, R. and Overbeek, R.: "Toward a Human Genome Encyclopedia", *FGCS 92*, (Jun 1992).