In Silico Biology 3, 0019 (2003); ©2003, Bioinformation Systems e.V.

B G R S 2 0 0 2

iUDB: An object-oriented system for modelling, integration and analysis of gene controlled metabolic networks

Andreas Freier[1*], Ralf Hofestädt[1] and Matthias Lange[2]

[1] Bioinformatics/Medical Informatics Department, Bielefeld University
P.O. Box 10 01 31, D-33619 Bielefeld
Email: afreier@techfak.uni-bielefeld.de

[2] IPK Gatersleben
Corrensstraße 3, D-06466 Gatersleben

[*] corresponding author

Edited by T. Takai-Igarashi; received October 23, 2002; revised and accepted February 21, 2003; published March 17, 2003

Abstract

The long-term objective of our work is the computational construction of complex cellular systems. Therefore, input data must be prepared systematically first. In the field of biological data information is spread out ov hundreds of molecular databases. Different approaches of data integration have been already applied in the past to provide a homogeneous access to these databases. Actually, a system is required to store integrated data of different type locally and to search for networks found in the integrated dataset. Finally, the analysis of these networks can answer different questions, e. g. the role of biochemical substances in regulatory systems or the reasoning of metabolic diseases.

In this article a novel object-oriented modelling approach in the field of biochemical networks is presente Molecular objects are modelled conceptually using object classes, internally based on the object model OMG IDL. Specific object services are implemented automatically for each model and object instances are built using data integration. In combination with that, a specific view concept based on access paths has b implemented to model biochemical reaction rules automatically. Together with the application of graphica methods, pathways and cliques are computed by the system. They provide the topology of cellular proce networks. The system has been implemented using Java and CORBA.

Key words: object-oriented modelling, data integration, network analysis, Java, CORBA

Introduction

Today, more and more internet databases are published [Baxevanis, 2001] providing selected molecular data concerning metabolism, gene networks and their application, e. g. diseases. At this point several systems which implement the integration of molecular databases, e. g. *DiscoveryLink* [Haas *et al.*, 2001], *SRS* [Etzold *et al.*, 1996], *Tambis* [Stevens *et al.*, 2000] and *BioDataServer* [Freier *et al.*, 2002], are used to get efficient access to distributed databases. Likewise, information systems for modelling and visualisation of regulat molecular networks are presented, e. g. *aMaze* [van Helden *et al.*, 2000], *GeneExpress* [Kolchanov *et al.*, 1998], *Pathdb* [Waugh, 2000], *PathFinder* [Goesmann *et al.*, 2002] and *GENESYS* [Glass and Gierl, 2002], where in-house data and information from public data sources are taken and modelled within specific implemented databases or flat-files files. Even systems specialized at the same topic (e. g. metabolism)

differences in data modelling and content. On one hand there is the evolution of molecular databases and on the other one there is the problem that information systems have to be modified or rewritten to be able to be applied to new or modified molecular objects.

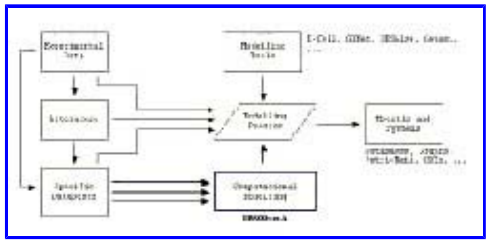

Figure 1: Support for computational modelling process.

For object-oriented, platform independent and distributed modelling of molecular data, *CORBA (Common Object Request Broker Architecture)* [OMG, 1996] has been applied in the past [Coupaye, 1999; Helgesen and Redaschi, 1998; Kemp *et al.*, 1999]. Existing systems implement single object services with statically defined data structures, representing molecular objects. Moreover, a number of molecular objects are modelled as object classes.

In this article a system of *Individually Integrated User Databases (iUDB)* is presented. The main idea of the approach is to apply objected-oriented concepts and data integration as modelling language in the field biochemical systems and to implement (multiple) specific CORBA object services automatically. These services include different capabilities, as there are typical database operations, data integration and gra algorithms. From our point of view, consistent databases are needed locally to proof the applications by the data. While the system manages different integrated databases with different molecular object classes, at the same time it enables the user to apply graphical methods to each database. In conjunction with our long-objective, which is the large-scale modelling of cellular systems, iUDB supports the modelling proces computational methods (see Figure 1).

The iUDB system

The architecture of iUDB contains three layers (see Figure 2):

- Object databases,
- Object implementations and
- CORBA services:

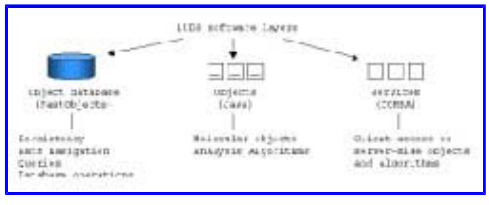

Figure 2: Architecture overview of the iUDB system.

In the centre of the approach there is the automatic implementation of molecular Java classes, according to the object models previously defined by the user. Standard database interfaces (JDBC) and the tool *BioDataServer* [Freier *et al.*, 2002] are used to retrieve Java objects from remote data sources. Object databases [Cattell and Barry, 1997] are implemented on-the-fly and used internally to handle the persistency of

complex object networks. Currently, we are using the *FastObjects* database system. Additionally, all internal algorithms have been implemented in Java. Finally, each Java object (molecular and application objects) wrapped by the CORBA environment.

Modelling object networks

The initial step in our system's workflow starts with the conceptual modelling of the real world objects. We decided to use the object-oriented modelling paradigm because of its rich semantics in comparison to, e. g., the relational database model. Object-oriented standards and modelling tools are already available an established. The user can include typical concepts, e. g. object abstraction and classification, inheritanc modularity, aggregation and complex data types (see Figure 3). For the internal specification of iUDB databases we use the *OMG IDL* language> (see OMG, 1996, for the language syntax). Using the language semantics a database scheme S of an object network consists of n interfaces i (object types):

$$S = \{i_1, i_2, \ldots, i_n\}$$

Each interface i holds an optional parent interface for inheritance i_{Inh} and a set of object attribute types A:

$$i = (i_{Inh}, A)$$

A is also called object body and holds the object properties:

$$A = \{a_1, a_2, \ldots, a_m\}$$

If $i_{Inh} \in S$, i will inherit all attributes of i_{Inh}. Attribute types can be either of literal type (support for four literal types of the OMG IDL standard have been implemented in the prototype already) or object type.

$$a \text{ } \diamond \text{ } S \text{ } \diamond \text{ } \{\text{string, double, long, boolean}\}$$

If $a \diamond S$, a is called *Reference*. A network is defined, if at least each object type refers (by *Reference*) to other object types or is referred by them. Also, we differ between unique and collection types. Collections contain a set of literals or references, while unique type are restricted to one value. In IDL notation, collections are defined as sequence type, e. g. "sequence<string>". In abstraction of the domain of *Gene Controlled Metabolic Networks (GNW)*, we model object types as "Enzyme", "Pathway", "Gene", "Promoter", "Binding Site" and other classes.

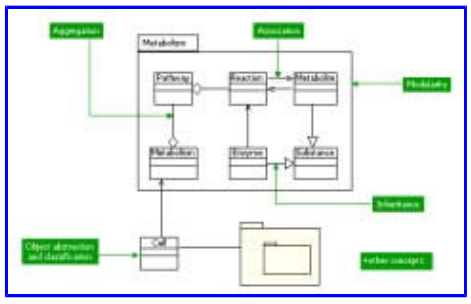

Figure 3: Modelling metabolism using UML class diagrams.

A reference example will lead through this article. The interface "Pathway" holds the names of a pathway a attribute type "string", while enzymes playing a role in a pathway are modelled as *m:n* relationship to the interface "Reaction". Thus, pathway objects contain a collection of references to reaction objects. The following example shows a part of the internal specification in IDL:

```
// Database Specification Example
```

```
// (OMG IDL)
module myDatabase {
 ...
 interface Pathway {
  attribute string           mapNumber;
  attribute sequence<string> names;
  attribute sequence<Reaction> reactions;
 };
 ...
};
```

Creating object databases

After the user has modelled his biochemical view using the object model, the resulting specification will l processed by the tool *Database Builder*. Here, an object service capable to handle objects of the users cla diagram will be implemented and activated automatically. Figure 4 shows the graphical front-end of the tool. iUDB handles multiple users and databases representing different user specific views and content. The databases including an *Application Programming Interface (API)* using CORBA is presented by a dynamically generated website. The databases are physically located where the iUDB server has been installed too. To use the system remotely, a local installation is not needed.

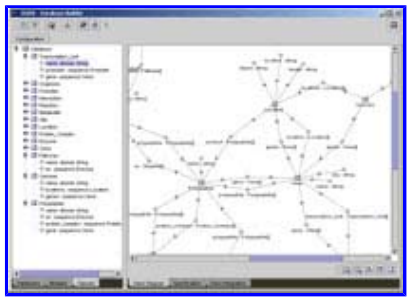

Figure 4: The Database Builder tool.

Modelling the database scheme with the Database Builder only is also possible. While the scheme is displayed as inheritance tree, object types can be edited by the user. Knowledge of the specification language is r necessary, the specification in OMG IDL will be generated by the tool automatically.

Integrating data to object networks

The previously empty databases should be filled now with data from molecular databases, representing th common domain knowledge. Here, available component data sources should be queried, e. g. BRENDA [Schomburg *et al.*, 1998] for enzyme data, KEGG [Kanehisa and Goto, 2000] for metabolic pathways, RegulonDB [Huerta *et al.*, 1998] for gene regulation (in bacteria), EMBL [Baker *et al.*, 2000] for gene annotation data and OMIM [Hamosh *et al.*, 2000] for disease information. To overcome the integration problem of heterogeneity and distribution, iUDB cooperates with the *BioDataServer* (BDS) module [Freier *et al.*, 2002], which provides a homogeneous database view to heterogeneous and distributed data sources. More precisely, for each object type in iUDB the relationships to different data sources can be modelled. A graphical user interface called *Integrated Scheme Editor* exists to exactly define the data source of each object type and its attributes. Every database can be loaded online with data using database integration. The interoperation of both modules is based on the relational JDBC interface of the BDS. The system dynamically creates SQL statements to be processed by the BDS a inserts the results into the database. Here, an object-relational transformation is carried out to create objects

from relational database content.

The integration of objects leads to object networks. Objects are stored in a database using class extensions. Here, an object database (ODBMS) is used for object persistency and database operations. A class extension e holds all objects of a class i:

$$e_i = \{o_1, o_2, \ldots, o_k\}$$

The database extension E is the set of all class extensions. Together with the database scheme, a database D includes intension (scheme) and extension (objects):

$$D = (S, E)$$

Within the integration process, iUDB ensures the referential integrity of objects. Integrity constraints can be, e. g., primary keys. Inserted and referred objects should satisfy these constraints, whereas sometimes a primary key consists of two or more attributes. The following example shows how these constraints are handled. An interface not inherited from another interface describing enzyme objects is given with the attributes EC number, organism, name and turnover number:

$$i_{\mathrm{Enz}} = (\emptyset\ , \{a_{\mathrm{EC}}, a_{\mathrm{org}}, a_{\mathrm{name}}, a_{\mathrm{turn}}\})$$

If a_{EC} and a_{org} are the primary key attributes, then e. g. the turnover number should be integrated specifically for the organism. Different enzyme objects with the same EC number and references to different organism objects should contain different turnover numbers. Thereby, the number of objects of the class "Enzyme" is at maximum the product of EC numbers and the number of organisms:

$$|e_{\mathrm{Enz}}| = |\mathrm{EC}| \times |e_{\mathrm{org}}|$$

Exploration of object networks

Database objects are accessed using class extensions. The database system (FastObjects) used by iUDB provides a standard (*Object Query Language*) OQL [Cattell and Barry, 1997] query interface to retrieve objects by their attribute properties. The user can apply this query interface to all interfaces defined in the database scheme S. As an example, the following query will return all enzyme objects with a name ending "dehydrogenase", which are all enzymes with EC numbers beginning with "1.1.1.":

```
// Example Query 1
// (OQL)
select c
from c in EnzymeExtent
where c._name like "*carboxylase*"
```

Following the object references, navigation through the object network is possible. A navigation path t between two classes i_{begin}, i_{end} can be given as a list of class interface traced. The enumeration of interfaces traced is given by the class attributes traced step by step.

$$t = (i_{\mathrm{begin}}, \{a_1, \ldots, a_n\})$$

To navigate through object networks, queries with a navigation path in the selection should be created. To retrieve, e. g. all metabolite objects whose consumption is influenced by the enzyme "6.4.1.2" (acetyl carboxylase), the following query could be used:

```
// Example Query 2
```

```
// (OQL)
select c._reaction._substrate
from c in EnzymeExtent
where c._ec= "6.4.1.2"
```

In Figure 5, an integrated object network computed with iUDB is displayed, where the navigation of quer highlighted. The data has been taken from BRENDA, EMBL, KEGG and RegulonDB.

Figure 5: Path navigation in integrated object networks.

Modelling reactions

Towards the modelling of gene networks, the very natural representation of object-oriented formalisms seems to be suitable, especially the application of concepts like classification, inheritance and complex objec relationships. The integrated object-oriented networks will be now the basis for modelling reaction rules. bioprocess can be characterized as a rule object with objects being consumed and objects being generated during the process. Additionally, there are objects not being consumed or produced, but influencing the e positively or negatively (e. g. enzymes, cofactors, feedback loops). Thus, a specific database view is n constructing reaction rules from the data. According to the fact that there are no view concepts available in most object systems, we implemented a view concept based on access paths to interlink integrated molecu objects with bioprocess objects. Furthermore, the reaction rules will be automatically derived (snapshot). framework includes a formalism to describe bioprocess patterns and a mechanism to detect predefined patterns in integrated object networks. The workflow of reaction network modelling consists of three steps:

- Interactive modelling of rule patterns
- Computational mining for rules using rule patterns
- Interactive and computational clustering of rules towards rule network s

The main element of the framework is the rule object v, specifying the topology of the bioprocess event.

$$v = (IN, OUT, INF, LOC)$$

Rule objects v interconnect database objects by referring to input objects IN_v (e. g. reaction substrates) and output objects OUT_v (e. g. reaction products). Objects influencing the event (e. g. enzymes) are referred to as influence objects INF_v. The physical location (e. g. the organism) of the bioprocess is determined by location objects LOC_v. All objects referred by rule objects are members of the database extension:

$$IN_v \times OUT_v \times INF_v \times LOC_v \subseteq E$$

The following example shows a biochemical reaction rule catalysed by the enzyme phosphofructokinase (PFK). The process is located in the glycolysis pathway of the organism yeast.

$$v = (\{ATP, F6P\}, \{ADP,F1.6P2\}, \{PFK\}, \{Glycolysis, Yeast\})$$

The objects related to the event have different object types and they are stored in different class extensions:

ATP, ADP, F6P, F1.6P2	$\in$	$e_{Metabolite}$
PFK	$\in$	e_{Enzyme}
Glycolysis	$\in$	$e_{Pathway}$
Yeast	$\in$	$e_{Organism}$

Next, the principle of rule derivation should be explained briefly, whereby the general structure of rule patterns is shown in Figure 6. We assume that objects included in one event are interconnected by a network of objects referred directly or indirectly, e. g. an enzyme object catalysing a biochemical reaction should contain one or more references to reaction objects.

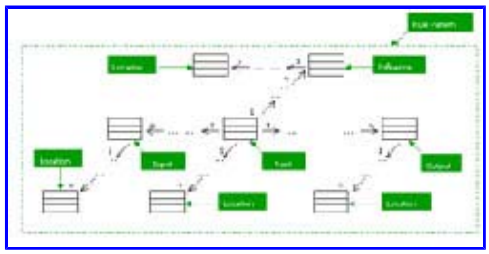

Figure 6: Schema of rule pattern with access paths.

However, there must be a path navigation possible between the event and the participant objects. Firstly, we have to identify an object root class $r \gamma S$, where there are navigation paths to every object which participates in the event. As a next step, the navigation path t to every object class must be given for IN, OUT, INF and LOC. Using our example database diagram, a rule pattern example p should be given for the biocatalysis. Starting at $i_{Reaction}$ as root class, the navigation paths to the reaction substrates p_1, the products p_2, the catalyser p_3 and the location paths p_4, p_5 are modelled (see Figure 7). The length of the navigation path is variable and also cycles are allowed.

Figure 7: Rule pattern example.

An algorithm can be informally given to insert interaction event objects based on rule patterns. For each root object $o \gamma e_r$ an event object v_o will be created. Next, the input, output, influence and location objects will be retrieved by path navigation and linked to the event object. If no objects are found by all path navigations, no rule object will be inserted in the database. A set of rule patterns P describing object interactions between database objects is called interaction model m. The framework allows to store different interaction models representing different views onto the object network. Additionally, the rules V found out by using the pattern library are stored as rule cluster.

$$m = (P, V)$$

With the definition of interaction models, iUDB databases contain the database scheme S, the extension of all database objects E and the interaction models M:

$$D = (S, E, M)$$

Figure 8 shows a screenshot of the tool *Network Designer*, a graphical user interface to design interaction models. The user is enabled to design models for each database created with iUDB. As shown in the picture, the object references in the database scheme and the structure of the rule pattern are displayed as directe graphs. Path navigations are highlighted in the database scheme by different colours.

Some advanced concepts like object aggregation have still not been discussed here. They are needed to model complex reactions structures, e. g. the binding of a signal object to a binding site object, where the complex of signal protein and binding site is originally not contained in the database. Here, surrogate objects will be constructed. Reaction properties, e. g. kinetics, are typically modelled as object attributes and should be read directly by the application where specific values are required.

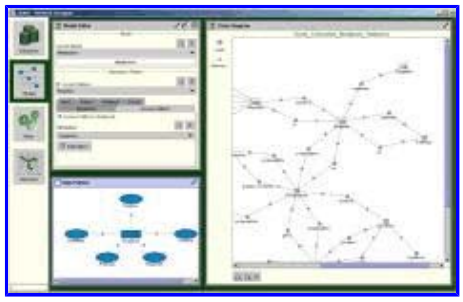

Figure 8: Network Designer tool modelling rule patterns.

Rule Networks

The analysis of integrated data using reaction patterns generates rules V_m for all patterns of the model m found in the database. They can be interconnected transitively as, e. g. pathways, by matching the identity (OID) of database objects they refer to. All rules of a network or pathway *w* found are a subset of V_m.

$$w = \{v_1, v_2, \ldots, v_n\} : v_j \in V_m$$

Such rule clusters can be stored in the system and afterwards loaded as a prepared network. In iUDB, implemented graphical analysis methods: the computation of the transitive closure and the search for pathways between given objects. Besides, the interactive pathway construction is possible and enhanced graphical methods will be implemented in the future development. Figure 9 shows the application of the system to the tricarboxylic acid cycle. For each object in the database, the user can select consuming ar producing rules to add them to the network. Computing interactivity profiles is also possible.

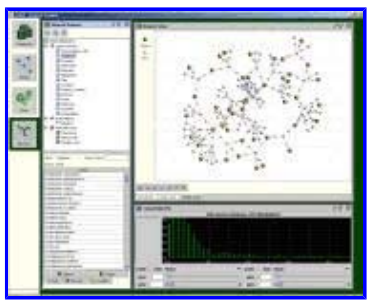

Figure 9: Network Designer tool computing tricarboxylic acid cycle rules.

Technical Issues

Currently, we are applying iUDB in the field of gene regulation and metabolism, where the number of objects and processes known is relatively small (e. g. < 4000 enzymes, < 3000 metabolites). Along with the modelling

of more complex biological systems, e. g. whole cells (*E. coli*: $4x10^6$ base pairs, [Russel, 1996]), and the multi-dimensionality of the data (organisms, tissues, compartments, chromosomes) the scalability of the approach should be discussed. Contemporarily, we tested our CORBA implementation combining different model parameters with synthetically generated data sets, which are mainly the number of different classes, the number of objects, the computation time and the memory consumption.

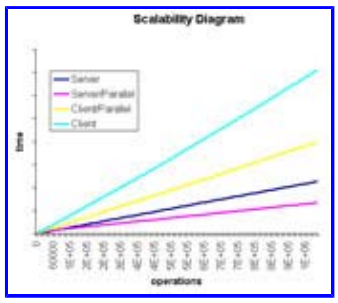

Figure 10: Scalability diagram.

First measurements with data sets containing millions of objects pointed out that the system is able to han complex object networks. To obtain higher performance, cost-intensive operations are implemented directly in the object service. The system also benefits from multiple processors. To give an illustration, Figure 10 shows the execution time for one million operations. The main result of this measurement is the approximately l behaviour of the cost function.

Conclusion

The idea of database integration applied to network analysis is not a new one. However, the growing number and the evolution of molecular databases demands adaptive systems which are able to integrate new data sources, build new integrated databases and apply suitable analysis methods dynamically. With iUDB, we have been developing a system supporting the object-oriented modelling method and the integrative analysis of bioprocess networks. As main object model, we use OMG IDL, which has been established in softv development and partially in bioinformatics. Common techniques, such as Enterprise Java Beans and SOAP could be used alternatively. However, we still do not expect any benefits from that in terms of function performance and platform independency. Regarding the modelling of complex systems, the view conc implemented allows the semi-automatically construction of network topologies. Furthermore, we intend t implement different views in the future, e. g. to support the modelling cell structures or cell states. A first version of iUDB is available under the URL: http://tunicata.techfak.uni-bielefeld.de.

Acknowledgements

This work is supported by the German Research Foundation (DFG).

References

- Baker, W., van den Broek, A., Camon, E., Hingamp, P., Sterk, P., Stoesser, G. and Tuli, M. A. (2000). The EMBL Nucleotide Sequence Database. Nucleic Acids Res. **28**, 19-23.

- Baxevanis, A. D. (2001). The Molecular Biology Database Collection: an updated compilation of biological database resources. Nucleic Acids Res. **29**, 1-10.
- Cattell, R. and Barry, D. K. (eds) (1997). The Object Database Standard: ODMG-93, Release 2.0. Morgan Kauffmann Publishers, San Francisco, CA.
- Coupaye, T. (1999). Wrapping SRS with CORBA: from textual data to distributed objects. Bioinformatics **15**, 333-338.
- Etzold, T., Ulyanov, A. and Argos P. (1996). SRS: Information Retrieval System for Molecular Biology Data Banks. Methods Enzymol. **266**, 114-128.
- Freier, A., Hofestädt, R., Lange, M. and Scholz, U. (2002). BioDataServer: A SQL-based service for the online integration of life science data. In Silico Biol. **2**, 0005.
- Glass, A and Gierl, L. (2002). A system architecture for genomic data analysis. In Silico Biology, Special Issue: GCB'01 **2**, 0019.
- Goesmann, A., Haubrock, M., Meyer, F., Kalinowski, J. and Giegerich, R. (2002). PathFinder: reconstruction and dynamic visualization of metabolic pathways. Bioinformatics **18**, 124-129.
- Haas, L. M., Schwarz, P. M., Kodali, P., Kotlar, E., Rice, J. E. and Swope, W. C. (2001). DiscoveryLink: A system for integrated access to life sciences data sources. IBM Systems Journal **40**, 489-511.
- Hamosh, A., Scott, A. F., Amberger, J., Valle, D. and McKusick, V. A. (2000). Online Mendelian Inheritance in Man (OMIM). Human Mutat. **15**, 57-61.
- Helgesen, C. and Redaschi, N. (1998). EMBL Nucleotide Sequence Database Object Model. http://corba.ebi.ac.uk/models
- Huerta, A. M., Salgado, H., Thieffry, D. and Collado-Vides, J. (1998). RegulonDB: A Database on Transcription Regulation in Escherichia coli, Nucleic Acids Res. **26**, 55-59.
- Kanehisa, M. and Goto, S. (2000). KEGG: Kyoto Encyclopedia of Genes and Genome. Nucleic Acids Res. **28**, 27-30.
- Kemp, G. J. L., Robertson, C. J. and Gray, P. M. (1999). Efficient access to biological databases using CORBA. CCP11 Newsletter, 3.1.
- Kolchanov, N. A., Ponomarenko, M. P., Kel, A. E., Kondrakhin, Y. V., Frolov, A. S., Kolpakov, F. A., Goryachkovsky, T. N., Kel, O. V., Ananko, E. A., Ignatieva, E. V., Podkolodnaya, O. A., Babenko, V. N., Stepanenk, I. L., Romashchenko, A. G., Merkulova, T. I., Vorobiev, D. G., Lavryushev, S. V., Ponomarenko, Y. V., Kochetov, A. V., Kolesov, G. B., Solovyev, V. V., Milanesi, L., Podkolodny, N. L., Wingender, E. and Heinemeyer, T. (1998). GeneExpress: a computer system for description, analysis and recognition of regulatory sequenecs in eukaryotic genome. Proc. Int. Conf. Intell. Syst. Mol. Biol. **6**, 95-104.
- OMG (1996). The Common Object Request Broker Architecture: 2.0/IIOP Specification, OMG Document Number 96.08.04. OMG (Object Management Group).
- Russel, P. J. (1996). Genetics. Fourth Edition. Harper Collins Publishers, NY.
- Schomburg, D., Schomburg, I., Chang, A. and Bänsch, C. (1999). BRENDA the information system for enzymes and metabolic information, *In:*
Proceedings of the German Conference on Bioinformatics (GCB '99), Braunschweig, pp. 226-227.
- Stevens, R., Baker, P., Bechhofer, S., Ng, G., Jacoby, A., Paton, N. W., Goble, C. A. and Brass, A. (2000). TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources. Bioinformatics **16**, 184-185.

- van Helden, J., Naim, A., Mancuso, R., Eldridge, M., Wernisch, L., Gilbert, D. and Wodak, S. J. (2000). Representing and analysing molecular and cellular function using the computer. Biol. Chem. **381**, 921-35.

- Waugh, M. (2000). Pathdb helps researchers analyze metabolism. Technical Report 1, National Center for Genome Resources.