On Reflexive Data Models*

S. Petrov

Computational Biosciences Section
Life Sciences Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831-6480

Submitted to *14th European Conference on Artificial Intelligence*, Berlin, Germany, August 20-25, 2000.

*Research was supported by the Office of Health and Environmental Research, U.S. Department of Energy under contract No. DE-AC05-96OR22464 with Lockheed Martin Energy Research Corporation.

On Reflexive Data Models

S. Petrov[1]

Abstract. An information system is reflexive if it stores a description of its current structure in the body of stored information and is acting on the base of this information. A data model is reflexive, if its language is meta-closed and can be used to build such a system. The need for reflexive data models in new areas of information technology applications is argued. An attempt to express basic notions related to information systems is made in the case when the system supports and uses meta-closed representation of the data.

1 INTRODUCTION

The information technology (IT) was initially developed for relatively simple object areas mainly related to business where distinction between information and metainformation is clear, structure and properties of data are well known. As the result, an information system is considered as an implementation of a formal theory describing general properties of represented data. States of the system (its content at a particular moment) or their sets are models of the theory.[2] There is usual trade-off between the expressive power of the language and computational properties of the expressed theory. A theory expressible in a fragment of the first order calculus corresponds to a system with queries executable in polynomial time (the case of restricted relational model). If the theory is based on some exotic logic where most algorithmic properties are unsolvable (the case of some object-oriented data models), in the corresponding system some well-expressed queries can be unsolvable. In any case, some kind of theoretical knowledge, general properties of data, should be available to a developer and their logic determines the system capabilities and efficiency.

Attempts to stretch the range of applications to object areas where formalization doesn't exist and/or is not possible were not always successful. A simple example is applications of database engines to textual data.

Application of IT to Molecular Biology is a new and extremely fascinating case. This is the case where the better an information system is designed the more it helps to make itself obsolete. Mass-production sequencing generates huge amount of data to be analyzed and stored. This task is impossible without an information system and any such system is based upon a representation, conceptual design, and logical schema. They utilize known at that particular moment system of notions and theoretical considerations. The ultimate goal of the system is to help biologists to advance the same theoretical knowledge on the subject. The better and faster it is done, the faster the system becomes obsolete. Resulting "catch 22" in Bioinformatics is extensively discussed in the section 2 below.

The text below is a first part of an attempt to investigate properties of possible reflexive data models. It serves two modest goals: to show that such data models are needed and to show that a formal description of such models is possible. Usage of meta-closed languages and its implications are considered in the section 3.

In the section 4 basic notions related to information systems (a system, its state, information equivalence, integrity, etc.) are formalized in case when the system supports and uses meta-closed representation of the data. In addition, a few properties of such systems are established.

2. THE CASE OF BIOINFORMATICS

2.2 The difference between bioinformatics and other application fields

In molecular biology, an information system serves as a necessary discovery tool used for further development of the "Theoretical Molecular Biology" whatever that means. In a contrast, information systems in Physics often deal with much greater amount of data but this data is well defined, complies with highly developed theory, and the information system has very little to do

[1] Oak Ridge National Laboratory, Life Sciences Division, Computational Biology Section, P.O.Box 2008, Bldg. 1060COM, Oak Ridge, TN 37831-6480, email: petrovs@ornl.gov

[2] For example, when time-related dependencies are supported a sequence of states represents the model and the theory is based on a temporal logic.

with farther development of the Theoretical Physics.

Amount of biological data and its nature makes manual analysis impossible. This amount results, at least partly, from our inability to perceive directly objects in question. Sequencing can be compared with scanning books written in unknown language with the goal to decipher the language. If the only available representation of Egyptian hieroglyphs would be scans of their small parts, the language wouldn't be deciphered in XIX century but only now when an information system for these scans can be constructed.

Thus, a "Molecular Biology Catch 22" exists: an information system is needed for the discovery of theoretical knowledge and theoretical knowledge is necessary for data representation in an information system. Taking dynamics into account, the same can be stated more correctly but not less painful:

> *An information system is created on the base of existing theoretical knowledge with ultimate goal to improve and, therefore, to modify it. The better the system is designed, the more it helps biologists and faster becomes obsolete – together with all other systems built at same time on the base of the same knowledge.*

2.2 "Something is rotten in the state of Denmark."

Unhappy relationships between Bioinformatics applications and existing IT illustrate if not justify the previous statement:

- The attempt to create a relational equivalent of the Genbank, GSDB [1], led to a database where the schema is changing regularly. Sometimes these changes create a cycle: one year the object "sequence" was represented as a long text, next year it was represented as a hierarchy of text pieces, the year after it was switched back to the original.
- It looks like that at almost any moment of its past existence GDB [2] had three schemata: the current one used online, the next one coming, and the one in the minds of developers. Number of people involved in the development, including logical design, was paradoxically increasing. Usage of OPM [3] didn't change the situation but harmed the performance.
- It is easier to avoid commercial products:

  > *GenBank* [3] *is an example of a Luddite approach to the technology despite NCBI background databases. Fields content is not formalized and a parsing script is required when a particular information is to be found among "features".*

  … or to develop your own:

  > *ACeDB* [4,5] *is an example of a very successful homemade tool. Despite authors' claim, it is rather hierarchical than object-oriented system. It is very popular among biologists, in particular because they can redesign their application – within rather wide supported limits – at any time and almost painlessly.*

- Computational tools add their own semantic diversity:

  > *Objects "gene" = "experimentally found gene", "GRAIL gene" = "gene found by gene predicting program GRAIL", "GenScan gene" = "gene found by the gene predicting program GenScan" are conceptually different.*

  … and information systems add more:

  > *Semantics of the object "gene" in GenBank and GDB are quite different.*

An evident problem in Bioinformatics is the handling of the metainformation, "theoretical assumptions", that are very fluid and badly defined in Molecular Biology. Commercially available DBMSs are based on the assumption that metainformation is stable within the life span of an information system. As the result, one can find more "flat file" than "true" databases in the Bionformatics. GenBank is just one of the many.

2.3 Does the object-oriented approach change the picture?

Object-oriented languages bring in three features: richer tree-like object structures, code encapsulation, and inheritance. First feature is the least important just because XML, ACeDB data model, and other not true object-oriented languages share it. It is still important because "flat table" is evident Procrustean bed for biological data. Two last features are related to expressing metainformation within a system and, therefore, are more significant. "Officially", OO languages are not meta-closed but inheritance and code encapsulation blurs the border between abstraction and meta-transition. A method inherited from a superclass can express (and support) meta-properties of a class (like database triggers). A choice between coexisting specializations of an object class can be considered as an extreme case of self-referential object modification. Extra levels of abstraction separate stable components from the conglomerate of molecular biology knowledge to be used as "theoretical foundation" of its software support. Naturally, these components will have little if any of biological semantics.

CORBA as a system integration environment creates additional meta-levels. As the result, an information transfer resembles mountain climbing. A query formulated in terms of an interface objects goes through several conversions reaching an airless level of abstraction and then slides down to, for example,

[3] Object Protocol Model – an object-oriented environment independent of a particular database engine developed at Lorenz Berkeley National Laboratory.

an SQL query(ies) to several databases. Results are coming back in the same way.

Assuming that extensive representation of highly variable meta-information is unavoidable in Bioinformatics, it seems that there are only two alternatives. First one is to use a hierarchy of representations for the hierarchy of data meta-levels as it is done in OO approach. The second one is to use a meta-closed, self-referential language representing all meta-levels of information uniformly. Just like it is done in English and all other human languages.

3 META-CLOSED LANGUAGES

Meta-closed representation implies representing and storing a description of the system properties in the same way as properties and features of an area objects. A support of such representation requires, in particular, automatic system restructuring according to a modification of the data describing the system. For information systems considered below these two properties are characteristic: a system reflects its current structure in the body of stored information and it is acting on the base of this information. According to Prof. V.A. Lefebvre, who pioneered research of formal models of reflexivity [6,7], such a system can be called reflexive [8].

The ability of an information system to handle information depends on a chosen data representation, a language. It defines the expressive power including the power to express system own properties, and therefore defines some kind of a "degree of reflexivity". The choice of the language introduces syntactic considerations and restrictions making less transparent general properties of reflexive systems and underlying data models. To avoid this, no particular information representation language is specified below and no implementation problem is considered. A leftover of this elimination of particularities is a simple algebraic construction allowing it seems to be, to express some general properties of reflexive data models. The construction itself serves as a meta-language.

Natural languages allow self-reference in phrases like "This phrase consists of six words" and, therefore, they are meta-closed. At the same time, a computer representation of data is restricted by formal computational paradigm. Within this paradigm, an information system is always an implementation of a formal theory "wired in" its software support. In particular, the theory is expressed in the conceptual and logical schemata of the database. Though meta-data is also represented and stored (for example, as a system catalog or data dictionary) it can not be modified ad hoc and used to automatically modify the database structure. Instead, they are automatically updated each time when the structure of the database is changed. Thus, a constant meta-data is used to interpret a dynamic information stored in the system. In a contrast, in human communications part of the incoming information is used to interpret the rest of it, both parts are not separated and often are barely distinct. A meta-closed data representation is supposed to treat information in more "linguistic way" in some sense diminishing the gap between human and computer representation.

Meta-closed data representation invokes the danger of paradoxes that can undermine the system integrity. On the other hand, the danger is immanent only in case of the classic logic. Limitations of computer environment as well as requirements of the real world move logic supported by a system in the intuitionistic direction. In commercial RDBMS, support for at least one null value makes it a third truth-value in logical expressions. Limited time and computer power makes complete check of consistency impossible replacing it by a para-consistency [9,10]. More or less exotic logic calculi where the danger of paradoxes is smaller are de-facto standard in information systems. Therefore, an attempt to consider meta-closed data models doesn't change the situation as much as it seems to be.

There is a definite drift of software engineering and IT in from "crystal" classic data models based on logical and algebraic constructions (e.g., relational model, object-oriented models) to amorphous software bundles blurring the distinction between object area data and data description as well as distinction between denominative and procedural semantics. Growing requirements of scale, complexity, and generality led to data representation comprising information and metainformation (e.g., XML) though quite often metainformation (including procedural one) is hidden in object class definitions. In particular, data warehousing technology based on incorporation into one system several databases having different conceptual schema requires, therefore, definition, representation, and storage of these schemata as well as transformations of data from one database into another allowing to compare it. Transformation of data representations is the base of online analytical processing (OLAP), in particular, in multidimensional data representations.

These changes actually reflect a general trend in programming toward interpreted scripts versus compiled code (e.g., PERL). Interpreted languages pose no formal restrictions on self-modification of the code during the run time. (The most interesting results in this area belong to the Theory of Supercompilation based on REFAL [11,12].)

4 REFLEXIVE DATA MODELS

A few definitions below are based on a strong version of the closed world semantics: everything relevant to the stored information is supposed to be represented within the system including any procedure, query, or a program used to process data by selection, filtering, etc. The only linguistic consideration used below is the assumption that information is somehow partitioned into *records*. Reasons for partitioning and its choice are not considered. Usually, a partition is defined by the syntax of a language used to express information – semantics is hidden under a syntactic cover. The important point is that there is no difference between records representing operational data (e.g., executable code) and any other data piece.

Definition 1. *A countable set $\mathcal{R}$ is the set of records and special element $\varphi \in \mathcal{R}$ is its faulty record.*

It is convenient to view records as arbitrary texts in an arbitrary language(s). The faulty record plays role of a universal null value.

Next two definitions express the fact that some of the records can be applied to a list of (other) records serving as arguments and the result of application is a record.

Definition 2. *The set of lists on $\mathcal{R}$ is $\mathcal{R}^+ = \{l \in \mathcal{R}^*: |l| > 0\}$, where $\mathcal{R}^*$ is free monoid generated on $\mathcal{R}$ by concatenation.*[4]

A list built from records $r', \ldots, r''$ is denoted $[r', \ldots, r'']$. If r is a record and $[r', \ldots, r'']$ is a list, the list $[r, r', \ldots, r'']$ is denoted $r \circ [r', \ldots, r'']$ where $\circ$ is concatenation. Similarly, $l' \circ l''$ is the concatenation of lists l' and l''.

Definition 3. $\mathcal{U}: \mathcal{R}^+ \rightarrow \mathcal{R}$ is the *universal mapping*.

Universal mapping can be viewed as a universal program (that itself doesn't necessary belongs to $\mathcal{R}$) or an abstract machine that can be used to model or implement any data processing described by elements of $\mathcal{R}$.

Definition 4. *The image of r on $S \subseteq \mathcal{R}$ is*

$$\mathcal{I}(r/S) = \{ \mathcal{U}([r] \circ l) : l \in S^* \}.$$

To avoid paradoxes, the assumption that a record is not applicable to itself is used:

$$\forall r \in \mathcal{R}: \mathcal{I}(r/\{r\}) = \varphi \ \& \ \mathcal{I}(r/\{\varphi\}) = \varphi.$$

For every $S \subseteq \mathcal{R}$, $\mathcal{I}$ defines is a Galois correspondence on S. It is not necessary to assume that $\mathcal{U}$ represents a virtual machine executing elements of $\mathcal{R}$ as programs. Another valid interpretation is: $\mathcal{R}$ contains formulae and their models and $\mathcal{U}$ provides interpretations. In any case, we are not concern with the complexity $\mathcal{U}$ execution.

Note that records are used as models of other ones. Two records are considered semantically equivalent ($r \approx r'$) iff $\mathcal{I}(r) = \mathcal{I}(r')$, and equivalent ($r \equiv r'$) iff $\forall l \in \mathcal{R}^+: r(l) = r'(l)$. Evidently, equivalence implies semantic equivalence but not vice versa. Stretching the scope of database terminology, we are calling r *schema* of $\mathcal{I}(r)$.

[4] For a finite set S, a free monoid S^* generated by S under operation of concatenation is the set of all finite strings (words) of elements of S including zero length string λ. The set $S^+ = S^* \setminus \{\lambda\}$ contains all strings from S^* with length bigger than 0. The same definition is valid for countable sets. As soon as the set is ordered, the order implies lexicographic order on the monoid that is, therefore, also countable. To distinguish between finite and countable sets and for some other reasons, elements of S^+ are called (finite) *lists*.

In the case of relational model, a relation is viewed as a model of its predicate. The record corresponding to the predicate is the schema of the relation. The same relations can be generated by a query, constructing it from other relations – therefore, the query-record is semantically equivalent to the schema-record.

Interpretations of mappings $\mathcal{U}$ and $\mathcal{I}$ are based on the assumption that outputs of the same procedure are somehow semantically uniform. It is possible to say, that r generates its own models that are elements of $\mathcal{I}(r)$.

Definition 5. *For $R \subseteq \mathcal{R}$, the set $[R] \subseteq \mathcal{R}$, is the closure of R iff*

1. $R \subseteq [R]$
2. $\forall r \in [R], \mathcal{I}(r/[R]) \subseteq [R]$
3. *Any set holding properties 1 and 2 contains $[R]$.*

The set $R \subseteq \mathcal{R}$ is closed iff $R = [R]$

Definition 6. *The set R, $R \subseteq R'$, is the base of R' iff $[R] = [R']$ and for any $R'' \subseteq R$, $[R''] \subset [R']$.*

If a set of records is the state of an information system, then its closure consists of all possible queries and their results. On the other hand, any of its bases represent a non-reducible subset necessary to yield the same results.

Evidently, a base represents a non-reducible subset of R that allows restoring its closure. If a closed set R represent a state of an information system, the set of explicitly present "facts" and "procedures" always contains or coincides with one of its bases.

Taking into account usual trade off between space and computational time, "stored set" can be a base of the state or the state itself. In first case, computational time is spend to find a closure that is, to answer all queries. In the second case, no derivation is needed but search time may be big.

With all information uniformly represented by records, there is no "skeleton" defining the system structure and functions. The only thing that distinguishes one system from another is its trajectory: the sequence of states it is in during its life span. It is rather difficult to agree that an arbitrary sequence of states should be considered as an information system. For a system to be modified, the modification itself should be properly expressed: there should be a record representing it.

Definition 7. *A trajectory is a sequence $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_n$,* where each $\mathcal{S}_i \subseteq \mathcal{R}$ *and*

1. $\mathcal{S}_1$ *is finite or* $\mathcal{S}_1 = \mathcal{R}$;
2. *For every $i > 1$, there is $r_i \in \mathcal{R}$ such that*

$$\mathcal{S}_{i+1} = \mathcal{I}(r_i / \mathcal{S}_i)$$

The schema and integrity constraints constitute a stable part of an information system s, in our case it is the set of records $\bigcap_{r \in I_s} r$. Note that it can be empty.

Two systems are semantically equivalent when they coexist at the same time range and at any moment can answer the same questions. It is easy to see that two information systems are semantically equivalent iff at any moment closure of their states are equal. It means, that sets of facts explicitly stored in them contain bases of the same closed set.

5 CONCLUSION

The work is "to be continued" though some implications of the above are quite evident. In particular, sets of records and their closures possess usual set of properties associated with the closure concept. Proof of these properties is very straightforward. It is also quite clear that a language for a reflexive data model should comprise all computational environment for the data. It means, in particular, that separate representation means for data and procedures contradict reflexivity.

An attempt to derive syntactic structure of records implicitly imposed on them by their procedural properties can be useful. A few interesting results were derived from the assumption that number of arguments used by a every procedure is limited.

An attempt to overcome deficiencies of current information system technology is undergoing now in Oak Ridge National Laboratory. Genome Information Warehouse under development there is a heterogeneous information system comprising several search engines. It is build around a core subsystem having some degree of reflexivity and implemented as a relational database [13].

REFERENCES

[1] C. Harger *at al.*, 'The genome sequence DataBase', *Nucleic Acids Researches,* **1**, 28(1), 31-33, (2000).

[2] A.J. Cuticchia, 'Future vision of the GDB human genome database', *Human Mutation*, 15 (1), 62-69, (2000).

[3] K. Hokamp *et al.,* 'What's new in the library? What's new in GenBank? let PubCrawler tell you', *Trends in Genetics*, 11, 471-473, (1999).

[4] R. Durbin, J. Thierry-Mieg. 'A C. elegans database', 1991. Documentation, code and data available from anonymous ftp servers at lirmm.lirmm.fr, cele.mrc-lmb.cam.ac.uk and ncbi.nlm.nih.gov.

[5] 'The record of the ACEDB Conference and Workshop Jul 27 - Aug 9, Cornell University, Ithaca, New York, USA', http://probe.nalusda.gov:8000/ace97.html (1997)

[6] V.A. Lefebvre, *The structure of awareness*, Reidel, 1977.

[7] V.A. Lefebvre, *Algebra of Conscience*, Reidel, 1982.

[8] V.A. Lefebvre. Personal communication. 1998.

[9] A.I.Arruda, 'A survey of paraconsistent logic', *Mathematical Logic in Latin America*, North Holland, New York, 1-41, 1980.

[10] L.I.Rozonoer, 'On interpretation of inconsistent theories', *Information Sciencies*,vol.47,pp.243-266, (1989)

[11] V.F. Turchin, 'An algorithm of generalization in the supercompiler', *Workshop on partial evaluation and mixed computations, Oct 1987, Denmark*, Eds. D. Bjorner, A.P. Ershov, N.D. Jones, (1987)

[12] S.A. Romanenko, 'A Compiler Generator Produced by a Self-Applicable Specializer Can Have a Surprisingly Natural and Understandable Structure', *Partial Evaluation and Mixed Computation*, Eds. D.Bjorner, A.P.Ershov and N.D.Jones, 445-463, North-Holland, (1988).

[13] S. Petrov *at al,* 'Genome Information Warehouse: Information and Databases to Support Comprehensive Genome Analysis and Annotation', *Human Genome Program, Contractor-Grantee Workshop,* March (2000).