Integrated Access to Metabolic and Genomic Data

Peter D. Karp*, Suzanne Paley
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
pkarp@ai.sri.com
v: 415-859-6375, f: 415-859-3735

*Author to whom correspondence should be addressed.

To appear in Journal of Computational Biology, 1996.

Keywords: Metabolism, genome, knowledge representation, informatics, database

October 14, 2004

Abstract

The EcoCyc system consists of a knowledge base (KB) that describes the genes and intermediary metabolism of *E. coli*, and a graphical user interface (GUI) for accessing that knowledge. This paper addresses two problems: How can we create a GUI that provides integrated access to metabolic and genomic data? We describe the design and implementation of visual presentations that closely mimic those found in the biology literature, and that offer hypertext navigation among related entities, and multiple views of the same entity. We employ a frame knowledge representation system (FRS) called HyperTHEO to manage the EcoCyc knowledge base. Among the advantages of FRSs are an expressive data model for capturing the complexities of biological information, and schema-evolution capabilities that facilitate the constant schema changes that biological databases tend to undergo. HyperTHEO also includes rule-based inference facilities that are the foundation of expert systems, a constraint language for maintaining data integrity, and a declarative query language. A graphical KB editor and browser allows the EcoCyc developers to interactively inspect and modify this evolving KB.

1 Introduction

The EcoCyc system consists of a knowledge base (KB) that describes the genes and intermediary metabolism of *E. coli*, and a graphical user interface for accessing that knowledge.[1] EcoCyc is joint work between our group at SRI International, and a group at the Marine Biological Laboratory (MBL) led by Monica Riley. Taken together, the knowledge base and the graphical user interface (GUI) constitute an electronic encyclopedia that allows scientists to visualize and explore an integrated collection of genomic and biochemical information. By integrating genomic data with comprehensive knowledge of the metabolic functions of gene products, we greatly increase the value of the genomic data and the types of analyses that can be applied to it.

This paper describes the technical advances that underly EcoCyc. It is not a user's guide, nor does it provide a detailed description of the scientific information within EcoCyc (see (Karp et al., 1996; Karp, 1996)). Instead, the paper focuses on the technical solutions we have developed to two problems: How can we create a GUI that provides integrated access to metabolic and genomic data? And, what capabilities are required of an information-management system for metabolic and genomic data?

We present a graphical framework for integrating metabolic and genomic data that addresses issues such as: What are useful visualizations of entities such as biochemical pathways and genomic maps? How can we visualize relationships among these entities? What algorithms can compute these visualizations? What operations should be provided to support user manipulation of these visualizations? How can a user-friendly query system be designed for these data? We have designed visual presentations that closely mimic those found in the biology literature, but that offer powerful navigation capabilities within biological knowledge space, such as hypertext navigation among related entities. Our visualization tools can also present multiple computed views of the same entity that provide varying amounts of detail. Menu operations encapsulate commonly required queries without requiring users to learn the declarative query language that we also provide.

Bioinformaticians are employing a variety of information-management systems to manage electronic collections of biological information, including relational database management systems (RDBMSs), object-oriented DBMSs, and home-brewed data managers. We argue that frame knowledge representation systems (FRSs) are in many cases the best existing technology for managing biological information. Among the advantages of FRSs are an expressive data model for capturing the complexities of biological information with high fidelity. The schema-evolution capabilities of FRSs facilitate the constant schema changes that biological DBs tend to undergo. The HyperTHEO FRS used to manage EcoCyc includes rule-based inference facilities that are the foundation of expert systems. and a declarative query language. A schema driven graphical KB editor and browser allows the EcoCyc developers to interactively inspect and modify an evolving KB.

Although the techniques we describe have yielded an encyclopedia for *E. coli*, the EcoCyc software manage a collection of genomic and metabolic information for a variety of organisms.

[1]EcoCyc can be obtained by Internet FTP; for instructions see WWW URL `http://www.ai.sri.com/ecocyc/ecocyc.html`.

2 Design Requirements

Databases that integrate metabolic and genomic data can be used for a number of tasks in biology and in biotechnology. Programs will perform complex computations that involve many queries to these DBs, such as designing metabolic pathways. In a different mode of use, biologists will treat the DB as an electronic reference that consult to answer specific questions, or to seek out new relationships in an exploratory fashion. The EcoCyc GUI was created to facilitate the latter types of consultation — to provide biologists with powerful and intuitive tools for exploring and comprehending the elements of a complex information space. The design of the EcoCyc GUI was mainly influenced by the scientific tasks for which the information will be used, and the properties of the information itself.

We envision that metabolic/genomic DBs such as EcoCyc will be used for the following scientific tasks ((Karp and Mavrovouniotis, 1994) provides more details):

- Molecular biologists and biotechnologists designing a cloning project will have instant access to the best current information on map locations, identities of linked genes and their functions, lists of available clones, and locations of pertinent restriction sites.
- Molecular evolutionists could use such DBs to search out examples of duplication and divergence, and ancestral relationships among genes, enzymes, and pathways.
- The metabolic pathways present in an organism can be predicted from the DNA sequence of the organism plus its nutritional requirements, given sufficient background knowledge of metabolism (Gaasterland and Selkov, 1995).
- A scientist who has found sequence similarities between a gene of interest and one or more *E. coli* genes could use EcoCyc to obtain descriptions of the functions of the *E. coli* genes. EcoCyc contains more detailed descriptions of function than do the sequence DBs.
- Biotechnologists seek to design novel biochemical pathways that produce useful chemical products (such as flavor enhancers in food, amino acids and vitamins, or pharmaceuticals), or that catabolize unwanted chemicals such as toxins (Cameron and Tong, 1993; Bailey, 1991; Stephanopoulos and Vallino, 1991; Mavrovouniotis, 1993). Metabolic DBs can provide information about enzymes from other organisms with novel substrate specificities, kinetics, or regulatory characteristics, that can modify a metabolic network.
- Scientists who study the metabolism itself will be able to pose novel questions of broader scope than previously possible. Systematic computational studies of pathway evolution can compare many related pathways from different organisms. Those who employ numerical simulation techniques will benefit from collections of enzyme-kinetics data.
- The comprehensive characterizations of enzyme function in metabolic DBs will make possible systematic computational studies of protein structure–function relationships.

The EcoCyc GUI should assist scientists in these tasks by allowing them to quickly find relevant information, and to quickly understand that information. Scientists should be able to query objects in the EcoCyc KB according to several attributes, preferably without having to learn a complex query language. The GUI need not support every possible query that every possible user will want to formulate; it is acceptable to hardwire into the GUI a predefined class of queries that we expect to be most frequent, and to force users to pose more complex queries through means external to the GUI. Complex query languages are harder to learn, but are more expressive than the EcoCyc GUI.

The metabolic/genomic information space is highly connected — many relationships exist among the objects comprising the space. For example, a gene is related to the enzyme that it encodes; that enzyme is related to a reaction that it catalyzes; the reaction may be part of a pathway. The GUI should allow scientists to traverse these interconnections to find information.

Complex information requires complex tools for visualization and analysis. We wish to present large amounts of complex data (such as the thousands of genes on the *E. coli* chromosome) in an intelligible fashion. We can speed user comprehension of complex information (such as biochemical pathways) by using familiar graphical presentations when possible. All displays should be generated automatically by software that queries the KB and algorithmically produces a visualization. The alternative, to pre-draw displays by hand, requires large amounts of human time, yields outdated displays when the KB is updated, and may not produce as high-quality displays as can be produced by machine.

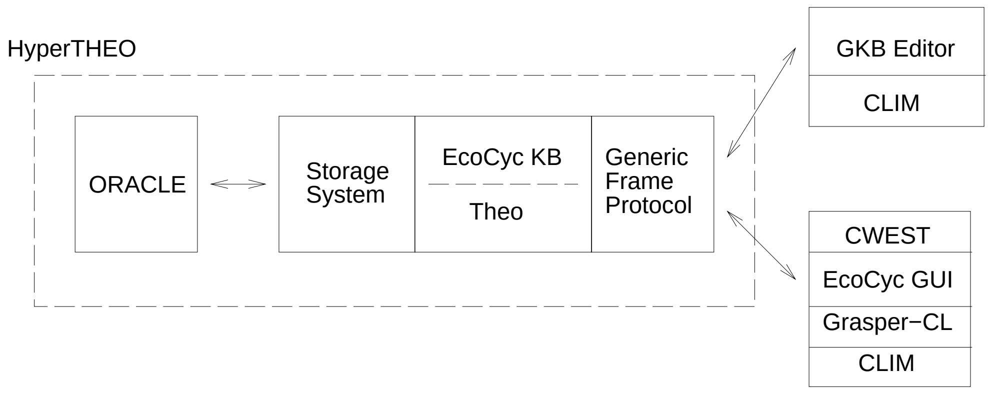


Figure 1: The architecture of the EcoCyc system.

3 Hardware and Software Architecture

EcoCyc is implemented in COMMON LISPand is developed on the SUN SPARCstation. COMMON LISP runs on the Macintosh and the PC as well as on SUN (and other UNIX) workstations. The graphics and windowing environment employed is the Common Lisp Interface Manager (CLIM). CLIM is implemented as a layer above the native windowing system (above X-Windows on UNIX workstations, above the Macintosh window system, and above Microsoft Windows for the PC). EcoCyc runs on Allegro COMMON LISP and CLIM from Franz Inc. of Berkeley, CA and on Lucid COMMON LISP and CLIM from Harlequin of Cambridge, MA.[2]

CLIM and COMMON LISP are high-level languages that facilitate fast application development. COMMON LISP has excellent debugging and incremental development facilities that yield high productivity: the entire EcoCyc GUI was developed in roughly 1.5 person years of effort.

The EcoCyc software architecture is shown in Figure 1. The major components of the system are

- A frame knowledge representation system called HyperTHEO that manages the EcoCyc KB.
- A graphical KB editor and browser called the GKB Editor.
- The EcoCyc GUI. All communication between the EcoCyc GUI and HyperTHEO flows through a well-defined KB-access library called the Generic-Frame Protocol (Karp et al., 1995), making the EcoCyc GUI a modular component that is in principle separable from HyperTHEO.
- A system for manipulating and displaying graphs, called Grasper-CL. Another reason for the quick development of EcoCyc is the fact that its pathway displays were implemented using the high-level graph display and layout capabilities within Grasper-CL.
- CWEST is a tool for retrofitting CLIM applications to run through the WWW. CWEST dynamically translates CLIM graphics to a combination of HTML and GIF images for transmission via the WWW (Paley and Karp, 1995). CWEST allows the EcoCyc GUI to operate over the WWW in a manner that is very similar to its X-windows operation.

[2]Both companies have reasonable runtime licenses that permit unlimited distribution for the Sun and for the PC. For the Sun, the developer must pay a one-time fee; for the PC there is no charge from Franz.

4 The HyperTHEO Frame Representation System

One of the first questions the designers of a new biological DB must address — and one of the most difficult — concerns the information-management technology to employ for that DB. Frame knowledge representation systems have been used in the artificial intelligence community for more than 20 years, and they have been shaped by the complex requirements of AI problems. Many of these requirements are shared by bioinformatics applications. For a growing number of bioinformatics projects, FRSs are the information management technology of choice. For example, Overton et al. use an FRS to manage genomic data for human chromosome 21 and for erythropoeitic genes (Overton et al., 1989; Haas et al., 1993). Perriere and his colleagues use an FRS to manage KBs for procaryotic and eukaryotic genomes (Perrerie et al., 1994; Perrerie et al., 1993; Schmeltzer et al., 1993). Brutlag et al. used an FRS to construct a KB describing DNA metabolism (Brutlag et al., 1991).

An FRS called HyperTHEO is used to manage the information within the EcoCyc KB. Our group at SRI produced HyperTHEO by extending the THEO system (Mitchell et al., 1989). In this section, we describe the architecture and capabilities of HyperTHEO, explain why HyperTHEO is well suited to representing biological information, and compare HyperTHEO to traditional database systems. We also note how HyperTHEO satisfies the desiderata that Letovsky and Berlyn identified for information-management technologies used to build complex scientific DBs (Letovsky and Berlyn, 1994). See (Karp, 1994) for detailed comparisons of the capabilities of FRSs and relational DBMSs.

4.1 The Frame Data Model

HyperTHEO employs the frame data model, which is very similar to the object data model. (Historically, the frame data model has existed since the mid 1970s, and was developed in parallel with the object data model, which was adopted by the object-oriented database community in the late 1980s.) The frame data model comprises the following components.

4.1.1 Frames

A *frame* is an object with which facts are associated. Each frame has a unique name. Frames are of two types: *classes* and *instances*. A class frame represents a semantically related collection of entities in the world. Each individual entity is represented by an instance frame. A frame can be an instance of many classes, which are called its *types*, and a class can be a type of many instances.

For example, in EcoCyc, instance frames represent individual compounds, genes, and reactions. Class frames represent such things as the class of all amino acids, and the class of all genes that code for tRNAs. Figure 2 show two frames that respectively represent a polypeptide and a protein complex in which that polypeptide is a component.

A *knowledge base*, or KB, is a collection of frames and their associated slots, values, facets, and annotations. Many FRS KBs also include production rules, which this paper does not emphasize.

```
--- Instance AROH-MONOMER ---
    Types:  Polypeptides

  CITATIONS:      "[89053867]"
  COMMENT:        "The aroH gene has two promoters..."
  COMPONENT-OF:   AROH-CPLX
  CREATION-DATE: "22-Aug-1993  14:36:13"
  CREATOR:        MRILEY
  GENE:           EG10080
  ISOZYME-SEQUENCE-SIMILARITY: (AROG-MONOMER YES), (AROF-MONOMER YES)
  MOLECULAR-WEIGHT-SEQ: 38.721
  SPECIES:        "E. coli"


--- Instance AROH-CPLX ---
    Types:  Protein-Complexes

  CATALYZES:      DAHPSYNTRP-ENZRXN
  COMMON-NAME:    "2-dehydro-3-deoxyphosphoheptonate aldolase"
  COMPONENTS:
    AROH-MONOMER
    ---COEFFICIENT: 2
  CREATION-DATE: "22-Aug-1993  14:36:13"
  CREATOR: MRILEY
  SPECIES:        "E. coli"
  SYNONYMS:       "phospho-2-keto-3-deoxyheptonate aldolase",
                  "DHAP synthase", "DHAPS", "KDPH synthetase",
                  "tryptophan sensitive 3-deoxy-D arabino-heptulosonate 7-phosphate synthase",
                 "3-deoxy-D-arabinoheptulosonate-7-phosphate synthetase (trp)"
```

Figure 2: This figure shows a printed representation of two frames; the names of the frames are `aroh-monomer` and `aroh-cplx`. The former is an instance of the class `Polypeptides` and the latter is an instance of the class `Protein-Complexes`. For each frame we list the name of a slot within the frame, followed by a colon, followed by the zero or more values of that slot. An annotation records the coefficient of the `aroh-monomer` component within the `aroh-cplx`.

4.1.2 Slots

Slots encode the attributes or properties of a frame, and also represent relationships between frames. A slot is a mapping from a frame and a slot name to a collection of values. Each value can be any Lisp object (such as a symbol, list, number, or string). For example, the slot called `components` in the `aroh-cplx` frame in Figure 2 lists the frames that represent the polypeptide subunits of the enzyme complex. HyperTHEO supports three collection types for slot values: sets, multisets, and lists.

4.1.3 Facets

Facets encode information about slots. Facets are uniquely identified by a facet name, a slot name, and a frame. A facet has as its values a set of LISP objects.

For example, facets allow comments or citations to be associated with an entire slot, and they allow access to schema information such as the datatype of a slot and the constraints attached to a slot (schema information is usually not stored in each slot, but is inherited from slotunits (see Section 4.2)).

4.1.4 Annotations

To store information about individual slot values, we use annotations. An annotation associates a labeled set of data objects with a particular slot value. Annotations are used in EcoCyc to attach comments and citations to slot values. In addition, as shown in Figures 2, an annotation called `coefficient` stores the coefficient of each subunit within a protein complex.

4.2 HyperTHEO Operations

HyperTHEO supports a number of operations that access and modify KB structures. A well-documented function-call interface to HyperTHEO called the Generic Frame Protocol provides an elegant and succinct set of operations such as (1) retrieving the values of a slot, (2) altering the values of a slot, (3) deleting a frame, (4) and enumerating the instances of a class:

```
1. (get-slot-value 'aroh-cplx 'synonyms)
2. (replace-slot-value 'aroh-monomer 'molecular-weight-seq 38.721 38.712)
3. (delete-frame 'aroh-monomer)
4. (get-class-all-instances '|Polypeptides|)
```

Another distinguishing characteristic of FRSs is that they provide logical inference. The KB designer can cause HyperTHEO inference capabilities to be triggered when the user retrieves the value of a slot. If a requested slot value is not found by HyperTHEO, it can transparently invoke several possible inference capabilities (as determined by the KB designer on a per-slot basis) to derive a value for the slot (the derived value is cached so that it need not be recomputed). Slot values can be derived by:

- **Attached procedures** — HyperTHEO calls a LISP function associated with the slot to compute a value for the slot. Letovsky advocates this operation (which he calls indirect fields) (Letovsky and Berlyn, 1994). For example, the products and reactants of each EcoCyc reaction frame are stored in slots called `left` and `right`. We define a virtual slot called `substrates` whose value is computed through an attached procedure as the union of the values of the slots `left` and `right`.
- **Inheritance of default values** — HyperTHEO searches parent classes of the frame from which the slot value was requested in an attempt to locate a default value for the slot. Such defaults can be changed at run time, unlike in most object-oriented database management systems (OODBs). Letovsky notes the importance of inheritance from multiple parents, which HyperTHEO supports (Letovsky and Berlyn, 1994).
- **Backward chaining** — HyperTHEO invokes PROLOG rules that are attached to the slot, and can infer a value through backward-chaining inference.
- **Inverse maintenance** — the EcoCyc KB declares that the slots `gene` and `polypeptide` are inverses. When the user modifies the value of the `polypeptide` slot in the *fumA* frame to record that product is the fumarase A monomer, HyperTHEO automatically updates the `gene` slot in the frame for the fumarase A monomer to point to *fumA*. Inverse maintenance is also advocated by Letovsky (Letovsky and Berlyn, 1994).

```
--- Instance GENE ---
    Types:  SLOT
  MAXIMUM-CARDINALITY: 1
  VALUE-TYPE: |Genes|
  PUBLIC?: T
  INVERSE: POLYPEPTIDE
  DOMAIN: |Polypeptides|
```

Figure 3: A slotunit is a frame that describes how a slot may be used throughout a KB. The slot `gene` is applicable to instance frames in the class `Polypeptides`. The slot can have at most one value; that value must be the name of a frame in the class `Genes`.

4.3 Constraints

HyperTHEO has a constraint system, developed by our group at SRI. Integrity constraints can be specified for each slot. Among the constraints that can be recorded are

- The *domain* — the class(es) of frames in which this slot may appear (for example, the `left` slot may appear in frames in the `Reactions` class).
- The *range* — the types of values this slot may take. Range constraints can include COMMON LISP types such integer and string, and KB classes. The range of the `left` slot is the class `Compounds`, meaning the values of the slot can be any instance frame in the class `Compounds`.
- The *cardinality* — the minimum and maximum number of allowable values for the slot. For example, the `left` slot can have any number of values, whereas the `keq` slot is constrained to have at most one value since a reaction has only one equilibrium constant.
- The *bounds* — this constraint specifies upper and/or lower bounds on the values of numeric slots.
- An *enumeration* — a finite set of symbolic values that this slot may take. The `location` slot in protein frames specifies the cellular locations(s) of a protein; it is constrained to take values from the set `{cytoplasm, inner-membrane, outer-membrane, periplasm}`.

Constraints are defined for each slot on a KB-wide basis. For each slot, the KB contains a frame called a *slotunit* that defines the properties of that slot. Constraints are stored in slots of the slotunit, as shown in Figure 3. Currently, our implementation of constraint checking is lazy in the sense that constraints are only evaluated with respect to some set of frames when requested by the user. We concur with Schweigert et al that for a variety of reasons, it is often desirable to allow noncomformant information to enter the KB (Schweigert et al., 1995), although it must be brought to the user's attention by the constraint system. Exceptions to the rule are common in biology, so HyperTHEO tolerates (but reports) constraint violations within a KB. We plan to integrate constraint checking into the GKB Editor so that constraints are verified interactively during editing sessions.

4.4 Declarative Query Language

The EcoCyc GUI provides a set of built-in queries to the user via menu commands. But users will wish to pose many more queries to EcoCyc than are feasible to encode within menus. Therefore, we have designed and implemented a query processor for HyperTHEO that accepts queries in a declarative language, and evaluates the queries with respect to the EcoCyc KB. A declarative query language allows the user to declare a question, without specifying a procedure for answering the question. Questions are posed in the KIF language (Genesereth and Fikes, 1992), which is equivalent to first-order logic (as is SQL). KIF is an emerging standard in the AI community.

For example, the following KIF query returns the set of all reactions whose left sides (reactants) include pyruvate:

```
(setofall r (exists c
   (and (reaction r) (left r c) (member c (pyruvate)))))
```

4.5 Knowledge Editing and Browsing Tools

Our group has developed a suite of graphical tools for interactive KB browsing and editing, called the Generic Knowledge Base Editor (GKB Editor). These tools allow the developers of the EcoCyc KB at SRI and at the Marine Biological Laboratory to interactively inspect and modify the EcoCyc KB using a menu-driven graphical interface. One tool presents the class/subclass/instance hierarchy as a graph, with incremental expansion of nodes to support exploration of large KBs. Another tool graphs arbitrary relationships among frames as a semantic network. The third tool displays the slots and facets of an individual frame, and allows the user to edit the values, facets, and annotations of the frame through mouse operations. For example, the user can easily add or delete values to or from a slot, or modify a slot value (such as a long comment).

The GKB Editor is schema driven in the sense that it can be used with any KB without reprogramming. The editor automatically queries the KB schema to determine the set of defined classes, the generalization relationships among classes, the slots defined for each class, and the attributes of each slot (such as the constraints defined for slot values). It uses the results of the schema queries to dynamically construct interactive forms for editing a frame selected by the user. This approach makes the editor extremely versatile since it can be reused with any KB in any application domain.

The price of that generality is that the graphical presentations of the GKB Editor are not biological — a reaction frame is not presented as a graphical reaction equation as in the EcoCyc GUI. We have therefore coupled the two GUIs: the user can browse the EcoCyc KB using the EcoCyc GUI; when the user sees a KB error, a mouse click will invoke the GKB editor on the appropriate frame so that the user can correct the error.

Documentation and sample screen snapshots for the GKB Editor are available through the WWW at URL `http://www.ai.sri.com/~gkb/`.

4.6 Knowledge Server Capabilities

Like all other FRSs, THEO KBs reside in main memory during processing, but can be saved to and restored from disk files. When we distribute EcoCyc to users, we create one large binary executable file that contains both the EcoCyc software and the EcoCyc KB. (That executable file is created by saving the virtual memory of an executing LISP process to a file, which is a standard LISP operation.)

This approach would be cumbersome for the EcoCyc developers to use to update the KB, because they must load it in its entirety at the start of a work session, and save the KB in its entirety at the end of a session. These operations take time proportional to the size of the KB. We would prefer a situation where the time required to load KB information is proportional to the amount of information accessed in a session rather than to the size of the KB, and where the time required to save updates is proportional to the number of updates rather than to the size of the KB. Another drawback of the file approach is that it provides no way for multiple users to concurrently update a KB.

Our group has extended THEO to store its KBs via a storage subsystem that is implemented using a commercial relational DBMS, in a fashion transparent to the HyperTHEO user (Karp and Paley, 1995b). HyperTHEO uses demand faulting to incrementally retrieve frames from the DBMS. It also tracks modified frames, which can be saved to the DBMS. This *knowledge server* approach has the desired properties that frame loading time is proportional to the number of frames referenced, and that the time to save updates is proportional to the number of updates. This approach also allows multiuser access to HyperTHEO KBs, since communication with the DBMS can flow over a network.

We further enhanced the performance of the HyperTHEO storage subsystem by designing a multiprocess architecture that allows HyperTHEO and DBMS processing to occur in parallel, and that prefetches frames from the DBMS during user idle time. The prefetcher retrieves frames that are linked to previously fetched frames. The scheme performs well in practice — we achieve frame prefetching rates of 50 frames/second over a local-area network using a SPARCstation-20.

These improvements to THEO have yielded two practical benefits to the EcoCyc project. First, they facilitate the distributed development of the EcoCyc KB by our group at SRI and by Monica Riley's group at the MBL. Both groups can update the same knowledge server via the Internet (however, these updates are currently uncontrolled; the next phase of our computer-science research is to develop new methods of controlling concurrent KB updates). Second, the knowledge-server approach facilitates user access to up-to-date information in EcoCyc. Although the entire EcoCyc KB is currently bundled within the EcoCyc program that is distributed to users, in the future we will employ a distributed model that decouples the GUI from the KB, allowing the GUI client to fault portions of the latest version of the KB across the Internet from an SRI server.

4.7 Discussion

The frame data model has a number of benefits. Its strong similarities to the object-oriented model give it the same advantages over the relational model as has the object-oriented model, such as that objects are often a more compact and natural representational device than are tables. For example, in the `aroh-cplx` frame in Figure 2, the `synonyms` slot is multivalued. This situation is handled trivially by the frame data model, but relational normalization would split every multivalued attribute out into a separate table, which complicates both the schema and user queries. The frame data model differs from the object-oriented model in several respects. Facets make schema information easily available to the application at run time, facilitating the development of generic tools such as schema-driven knowledge editors (see Section 4.5). Annotations allow us to attach structured information to any fact in the KB — in an object-oriented or relational DBMS, this ability would require explicit declarations in every attribute of every object, which would horrendously complicate the schema and consume large amounts of storage for fields that are only sparsely populated. Letovsky and Berlyn identified the need for annotations, and the need for supporting complex schemas (Letovsky and Berlyn, 1994).

The accepted wisdom for the DB-design process is to first design the schema for the application, and then populate the DB and write application code. This wisdom does not work in the biological domain because of its complexity. The first released design of a schema for a biological DB can virtually never be set in stone because its designers did not understand the full complexity of the domain, or because they did not attempt to encode its full complexity, or because the requirements of the DB expanded (perhaps to encompass a new experimental technique), or because of revisions to biological theory. Schema evolution has caused substantial delays in the introduction of new versions of bioinformatics software, and in bioinformatics we should view schema evaluation as the rule, not the exception (see also (Letovsky and Berlyn, 1994)).

Commercial RDBMS and OODB products provide meager support for schema evolution. Adding or renaming attributes is relatively simple, but deleting or splitting attributes, or changing attribute types, can be a major undertaking. For OODBs based on C++, each object class is specified as a declaration to the C++ compiler. A schema change therefore requires the application to be recompiled. Moreover, the new C++ objects are incompatible with the old objects, in the sense that the binary encoding of instances of the new class differs from the binary encoding of instances of the old class. Most OODB products contain no utilities for transitioning a database through schema changes. For example, Goodman et al. were forced to develop special mechanisms for schema migration (Goodman, 1994) for their OODB-based laboratory database. In contrast, HyperTHEO frames are not datatypes within the Common Lisp language, and are therefore not subject to compilation. HyperTHEO frames are implemented using a tagged data structure that essentially packs slot names into each HyperTHEO instance. Therefore, if a class definition changes, the instance still has a meaningful interpretation. The KB administrator typically writes a tiny program to coerce instances from the old schema to the new. For example, if a slot were to be deleted from the KB, an administrator program could simply iterate across all frames in the class to which that slot applied, and call a function that removed all values for that slot from those frames. If one slot is reconceptualized as two slots, a short program could conditionally transfer values of the old slot into one of the two new slots for all affected frames.

The HyperTHEO constraint language is an improvement over OODBs because of its succinctness, compared with the C++ methods that must be written for OODBs.

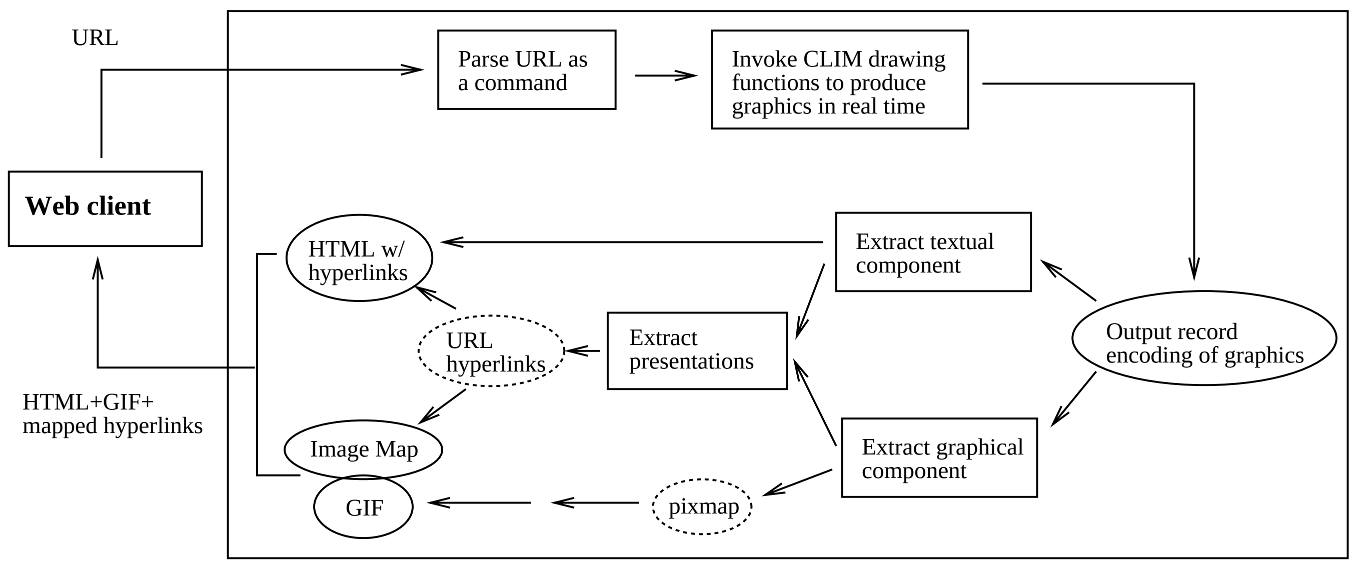

URL
Parse URL as a command
Invoke CLIM drawing functions to produce graphics in real time
Web client
HTML w/ hyperlinks
Extract textual component
URL hyperlinks
Extract presentations
Output record encoding of graphics
HTML+GIF+ mapped hyperlinks
Image Map
Extract graphical component
GIF
pixmap

Figure 4: Dataflow diagram illustrating CWEST operation

5 The EcoCyc Graphical User Interface

The Hypermedia approach produces displays that combine both text and graphics, based on the information stored in the slots, facets, and annotations of one frame, or a combination of frames. In EcoCyc, these displays are generated by LISP functions that dynamically query the KB as they construct a display. All EcoCyc displays shown herein were generated by EcoCyc and were captured using the Postscript-generation capabilities of CLIM.

The remainder of this section describes common properties of all the object display windows; the following section describes each type of object display in detail.

5.1 Retrofitting EcoCyc for the WWW

Because of the advantages of making EcoCyc available through the WWW in addition to our initial CLIM/X-windows distribution, we developed a generic tool called CWEST for retrofitting CLIM applications to run through the WWW (Paley and Karp, 1995). When running as a WWW server, EcoCyc listens for incoming URLs on its own TCP/IP port (see Figure 4). It interprets a URL as a command to display a metabolic or genomic object. The display is generated using the existing EcoCyc CLIM procedures. CLIM produces both X-window graphics, and an internal data structure called *output records*. CWEST converts the information in the output records into a combination of HTML plus GIF files plus an image map that defines active regions within the GIF file.[3]

CWEST extends the functionality of EcoCyc at the cost of extremely minor modifications to the EcoCyc source code. Our future extensions to the EcoCyc GUI will work with both X-windows and the WWW. CWEST can therefore be viewed both as a tool for retrofitting existing applications to run through the WWW, and as a high-level WWW authoring tool.

[3]The EcoCyc WWW server can be found at `http://www.ai.sri.com/ecocyc/browser.html`.

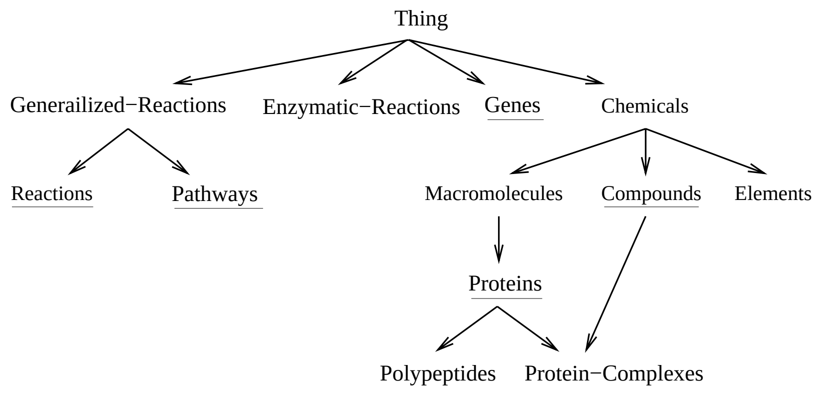


Figure 5: The top of the class hierarchy for the EcoCyc KB.

5.2 Object Interconnection

When displaying an EcoCyc frame, the GUI extracts information from slots of the frame to create each line in the textual portion of the display window. Some slots represent a relationship to other frames in the KB. Because HyperTHEO slots are typed (using slotunits), the GUI can recognize the relationship slots, and present them as hyperlinks in the display. This mechanism is the basis for hypertext navigation in the GUI. Some relationships among objects in the KB are not explicitly encoded. For example, although a slot in a gene frame records the product of the gene, no slot in the gene frame records the pathways in which that product plays a role, or the reactions catalyzed by the product. The GUI derives these implicit links dynamically during display of an object, using HyperTHEO attached procedures.

5.3 Object Queries

The user can set the GUI in six possible modes—one mode for each bold-face object class in Figure 5 (plus genomic maps). Within each mode, the user can select from a class-specific menu of predefined queries. For example, in compound mode the user can query compounds by any of these criteria:

- An exact compound name
- A substring within a compound name
- A compound class chosen from a menu-based classification hierarchy of compounds (such as amino acids, carbohydrates, and nucleotides)
- A chemical substructure specified using the SMILES chemical notation (Weininger, 1988) (all compounds containing that substructure will be returned as the results of the query)

As a second example, in reaction mode the user can call up a reaction based on its internal identifier, its name, its EC number, and pathways that contain the reaction. The user can freely mix mode-based queries with navigation using object links.

The intent of the different modes is to provide the user with the set of hardwired queries that we anticipate will be the most common. The GUI provides no way of issuing complex boolean queries.

Such queries would be issued through some other means external to the GUI, such as by writing a (usually simple) LISP program, or a KIF query (see Section 4.4).

5.4 Query Processing

EcoCyc computes answers to the menu-based queries using several processing strategies. Queries based on exact object names are computed using a hash table that maps both the common name and the synonyms stored for each frame to one or more internal frame identifiers. (That hash table is stored in virtual memory, along with the rest of the EcoCyc KB, as described in Section 4.) A different hash table is created for each of the five major classes in Figure 5 whose names are in bold face. The KB is organized such that the common-name field is unique for each frame (it is a key), whereas the synonyms for each frame are not necessarily unique. Therefore, the hash table is structured such that either a common name for a frame, or a synonym that happens to be unique for a frame, hashes to that unique frame. A non-unique synonym hashes to a list of the frames that all share that synonym. A user who enters an ambiguous name is asked to disambiguate by selecting from a menu of the matching objects. This approach yields fast searches across all recorded names, and allows interactive disambiguation of non-unique names (a similar approach is used in (Letovsky and Berlyn, 1994)).

Substring searches are computed using an exhaustive search of the appropriate hash table, which therefore covers both common names and synonyms.

Queries based on classification hierarchies are computed using hierarchies that are stored in the EcoCyc KB. Several of the object classes in Figure 5 is the root of a classification hierarchy ranging in size from a dozen (in the case of `Pathways`) to several hundred subclasses (the entire Enzyme Nomenclature (Webb, 1992; **?**) system is a class hierarchy under the class `Reactions`).

Queries based on chemical substructures are computed using a substructure matcher we implemented in COMMON LISP based on algorithms in the computational chemistry literature (Attias and Dubois, 1990). The algorithm converts the query structure entered by the user in SMILES form (a convenient ASCII representation for user entry of structures) into a graph representation where each atom is a node, and each bond is an edge. It then determines if that graph is a subgraph of the graphs corresponding to each compound in the EcoCyc KB.

When any EcoCyc query returns a list of objects as its answer (such as the query to find all compounds, one of whose names contains the substring “pyruvate”) the GUI presents the list in a multiple choice menu; the chosen objects are stored on a stack called the *answer list*, and the first object on the answer list is displayed using the normal EcoCyc display window. The user can step through the answer list using a main-menu command that displays the next item on the list.

5.5 Gene–Reaction Schematics

The many-to-many relationships among genes, enzymes, and reactions can be complex. An enzyme composed of several subunits might catalyze more than one reaction, and a given reaction might be catalyzed by multiple enzymes. The *Gene–Reaction Schematic* depicts the relationships among a set of genes, enzymes, and reactions (see Figure 6). It is drawn in reaction windows, enzyme windows, and gene windows. It is generated by starting with the object that is the focus of the current window (which is highlighted in the schematic), and then recursively traversing KB relationships from that object to related objects, such as from a gene to its product, or from a reaction to the enzyme(s) that catalyzes it. The schematic summarizes these complex relationships succinctly, and also constitutes a navigational aid: when the user clicks on an object in the schematic, EcoCyc displays that object.

5.6 Citations and Comments

Comments exist both as slot values, and as attachments to slots and to individual values within slots. Attachments to slots are stored in facets, and attachments to values are stored in annotations. We use the term *scope* to describe whether a comment (or citation) is located in a slot value, facet, or annotation. The scope of a comment or citation indicates the entity to which it refers: a citation within a slot value refers to the frame containing that slot; within a comment text the citation refers to the comment text; within a facet the citation refers to *all* values of the slot; within an annotation the citation refers to the exact value to which it is attached.

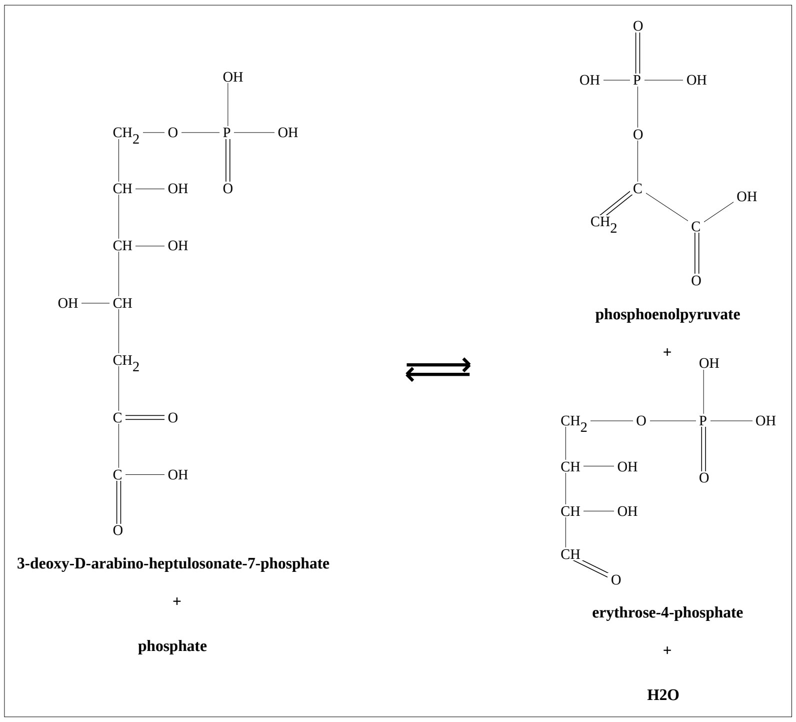
EcoCyc presents both comments and citations as bracketed numbers (such as "[1]") within object displays, as in Figure 7. The position in which the bracketed number is drawn indicates the scope of the comment or citation. For example, citation 1 refers to the preceding text, whereas citation 3 is the source of the information about the similarity of the AroG polypeptide to its isozymes. If the user clicks on a bracketed number, EcoCyc creates a pop-up window containing either the comment text, or the literature reference plus (in many cases) the abstract of the publication. Many references are obtained dynamically from Medline using the NCBI toolbox software to query the NCBI Entrez server via Internet. Those references not present in Medline are stored in the EcoCyc KB.

Reaction: 4.1.2.15

Enzymes and Genes:

2-dehydro-3-deoxyphosphoheptonate aldolase: **aroH**,
2-dehydro-3-deoxyphosphoheptonate aldolase: **aroF**,
2-dehydro-3-deoxyphosphoheptonate aldolase: **aroG**

In pathway: **chorismate biosynthesis**



Comment: The first committed step in the biosynthesis of aromatic compounds

This reaction occurs in E. coli.

Gene-Reaction Schematic:

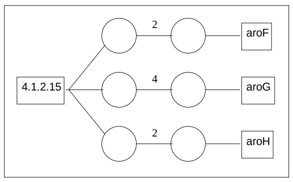
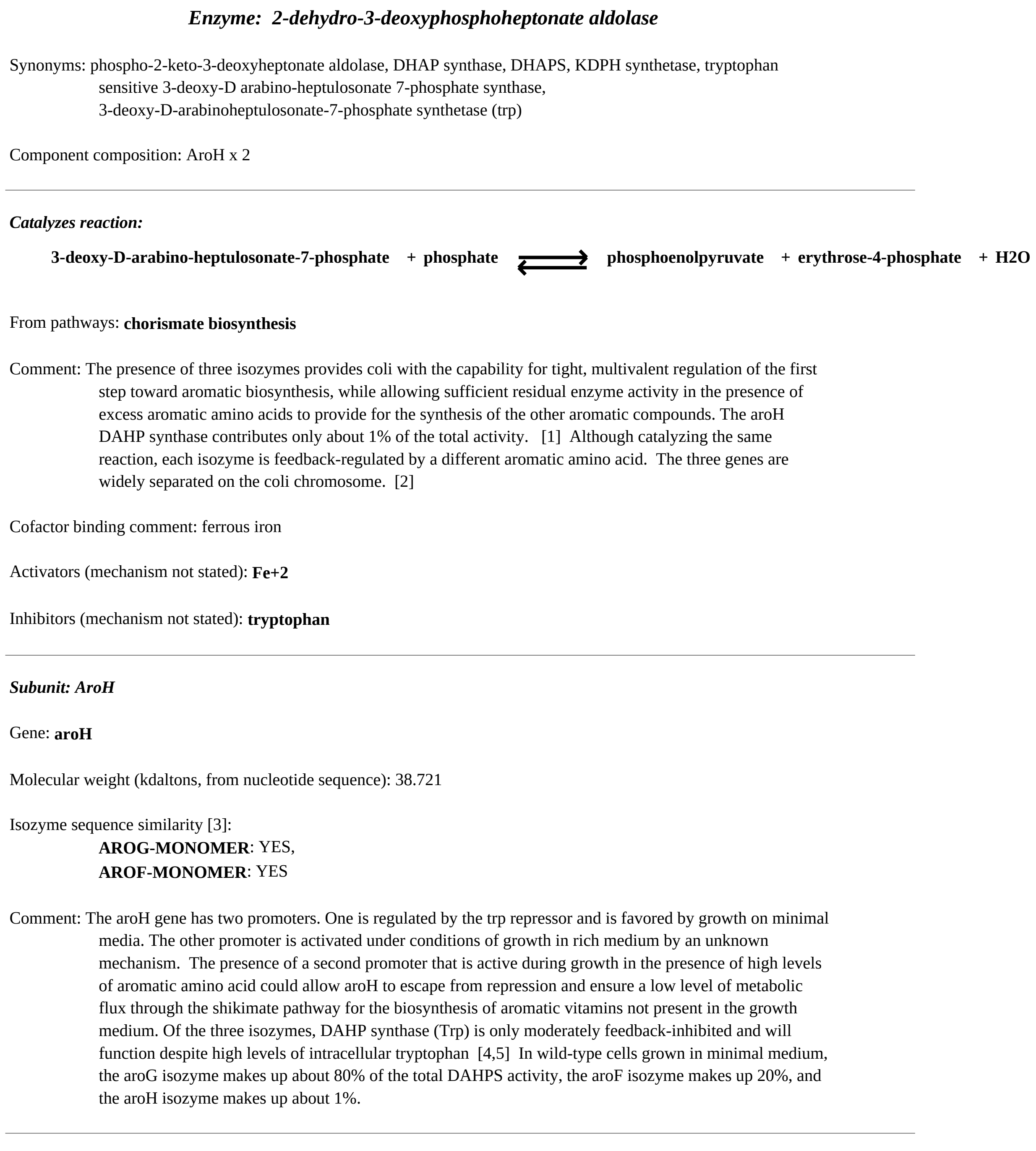


Figure 6: A reaction display window for the first reaction in the pathway for chorismate biosynthesis. The display shows properties of the reaction, and lists related objects, such as the three enzymes (in this case) that catalyze the reaction, and the genes that code for those enzymes. The Gene-Reaction schematic indicates that the product of the aroH gene acts as a homodimer.



Enzyme: 2-dehydro-3-deoxyphosphoheptonate aldolase

Synonyms: phospho-2-keto-3-deoxyheptonate aldolase, DHAP synthase, DHAPS, KDPH synthetase, tryptophan sensitive 3-deoxy-D arabino-heptulosonate 7-phosphate synthase, 3-deoxy-D-arabinoheptulosonate-7-phosphate synthetase (trp)

Component composition: AroH x 2

Catalyzes reaction:

3-deoxy-D-arabino-heptulosonate-7-phosphate + phosphate $\rightleftharpoons$ **phosphoenolpyruvate + erythrose-4-phosphate + H2O**

From pathways: **chorismate biosynthesis**

Comment: The presence of three isozymes provides coli with the capability for tight, multivalent regulation of the first step toward aromatic biosynthesis, while allowing sufficient residual enzyme activity in the presence of excess aromatic amino acids to provide for the synthesis of the other aromatic compounds. The aroH DAHP synthase contributes only about 1% of the total activity. [1] Although catalyzing the same reaction, each isozyme is feedback-regulated by a different aromatic amino acid. The three genes are widely separated on the coli chromosome. [2]

Cofactor binding comment: ferrous iron

Activators (mechanism not stated): **Fe+2**

Inhibitors (mechanism not stated): **tryptophan**

Subunit: AroH

Gene: **aroH**

Molecular weight (kdaltons, from nucleotide sequence): 38.721

Isozyme sequence similarity [3]:

AROG-MONOMER: YES,
AROF-MONOMER: YES

Comment: The aroH gene has two promoters. One is regulated by the trp repressor and is favored by growth on minimal media. The other promoter is activated under conditions of growth in rich medium by an unknown mechanism. The presence of a second promoter that is active during growth in the presence of high levels of aromatic amino acid could allow aroH to escape from repression and ensure a low level of metabolic flux through the shikimate pathway for the biosynthesis of aromatic vitamins not present in the growth medium. Of the three isozymes, DAHP synthase (Trp) is only moderately feedback-inhibited and will function despite high levels of intracellular tryptophan [4,5] In wild-type cells grown in minimal medium, the aroG isozyme makes up about 80% of the total DAHPS activity, the aroF isozyme makes up 20%, and the aroH isozyme makes up about 1%.

Figure 7: Enzyme display window for 2-dehydro-3-deoxyphosphoheptonate aldolase.

6 EcoCyc Object Displays

This section discusses issues that arise in generating each type of display window.

6.1 Reactions

The drawing of a reaction equation (see Figure 6) should make it easy for the user to visualize the chemical transformation taking place. In most cases, cofactors or carrier molecules that are only peripherally involved in the reaction can actually obscure the real transformations. Therefore, we maintain a list of molecules for which chemical structures are normally not drawn. This list includes small common molecules for which structures are unnecessary, such as water or carbon dioxide, as well as large common molecules for which structures would usually be distracting to their function in the reaction, such as ATP or Coenzyme A.

In what direction should we draw a reaction? Some reactions proceed in a single direction in the cell, whereas other reactions run in different directions under different physiological conditions. Furthermore, the system of nomenclature established by the Enzyme Commission (Webb, 1992) specifies a preferred direction for writing each reaction — a direction that often conflicts with its physiological direction, and thus looks wrong to a biologist. EcoCyc allows the user to specify that reactions are drawn in the systematic (Enzyme Commission) direction or the physiological direction. The former is easy to implement since all reactions are stored within the KB in the systematic direction. To implement the latter, EcoCyc infers the direction of the reaction within one of the pathways in which it occurs (Karp and Paley, 1994).

6.2 Enzymes

The complexity in drawing enzyme windows results from the many-to-many relationship between enzymes and reactions—one enzyme can catalyze multiple reactions, and each catalytic activity of an enzyme can be influenced by different sets of cofactors, activators, and inhibitors. We represent enzymes and reactions as distinct entities (Karp and Riley, 1993), and we use a third type of entity, called an enzymatic reaction, to represent the pairing of an enzyme with a reaction. These distinctions are important because some information is specific to the reaction (such as its equilibrium constant), some information is specific to the protein (such as its molecular weight), and some information is specific to the pairing of an enzyme and a reaction (such as the inhibitors that modulate the reaction).

Consequently, the enzyme window is divided into several sections (see Figure 7). Some sections describe the structure of the enzyme and its components; other sections describe each activity of the enzyme.

6.3 Compounds

Compound structures are drawn from a representation of each compound stored within the KB. Each structure is encoded as a list of atoms, a list of bonds between these atoms, a list of atom charges, and a list of 2-D display coordinates for each atom (Karp, 1992). Although the positions of each atom within the structure are given, each structure drawing is scaled just before it is drawn.

Every compound is cross referenced to the reactions and the pathways in which it appears.

6.4 Pathways

Our initial goal for automating the drawing of biochemical pathways was to produce drawings that are familiar to biochemists—drawings that mimic those found in biochemistry textbooks. We developed our algorithms for the automated drawing of metabolic pathways after studying pathway drawings in a variety of biochemistry textbooks. Our second goal was to move beyond textbook drawings by allowing the user to see the same pathway from multiple perspectives by viewing the pathway through different information filters, and by giving the user tools for seeing how different metabolic pathways interconnect. A third goal was to make pathways easy to specify. We developed a simple and compact representation of pathways that can yield very detailed drawings.

Our study of the pathway drawings published by biochemists yielded several observations. First, biochemical pathways are labeled, directed graphs. In a pathway graph, nodes are chemical compounds and enzymes, edges indicate chemical reactions, and both nodes and edges are often labeled (such as with a compound name). We can view the pathway drawing problem as a graph layout problem — a valuable perspective given the long history of computer science research on graph layout.

Second, biochemists use different styles of graph layout for drawing pathways of different topologies. For example, they use the "snake-layout" style for linear pathways (see Figure 8) and they use a tree-layout style for branching pathways (see Figure 9).

Third, the chemical substrates in a pathway are drawn in two ways. The compounds that are shared between subsequent reactions (the *mains*) are drawn along the backbone of the pathway. *Side* compounds are adjacent to the reaction arrow; a curved arrow shows whether they are consumed or produced by the reaction. Enzyme names are drawn to the other side of the arrow. We can therefore define a pathway graph as consisting of three types of nodes: mains, sides, and enzymes. The edges of the graph connect mains, and show the role of sides (reactant or product) in each reaction.

Pathway-Drawing Algorithms Our pathway-drawing algorithms accept as input a pathway graph in which each node and edge is annotated with its type (such as side-product, or enzyme). As output, the algorithm assigns positions to each node in the graph, and produces a CLIM drawing of the resulting pathway. The algorithm consists of three steps:

1. Determine the topology of the input pathway as either linear, cyclic, branched, or complex.
2. Apply a graph layout algorithm that is appropriate to the topology of the pathway to the main nodes of the pathway, thus assigning positions to the main nodes.
3. Assign positions to the side nodes and the enzyme nodes of the pathway. (The same algorithm is applied in this step regardless of which layout algorithm was used in Step 2.)

The actual algorithms for Steps 1–3 are described in detail in (Karp and Paley, 1995a; Karp et al., 1994). The second publication describes a general system for managing and creating graphs, called Grasper-CL. It provides general data structures for manipulating graphs, a graphics toolkit for drawing nodes and edges of different shapes, and a variety of graph-layout algorithms.

Figures 8 and 9 illustrate the linear and branched layout algorithms, respectively. We define the topology of a pathway as complex if it is neither linear nor branched nor circular — if it contains

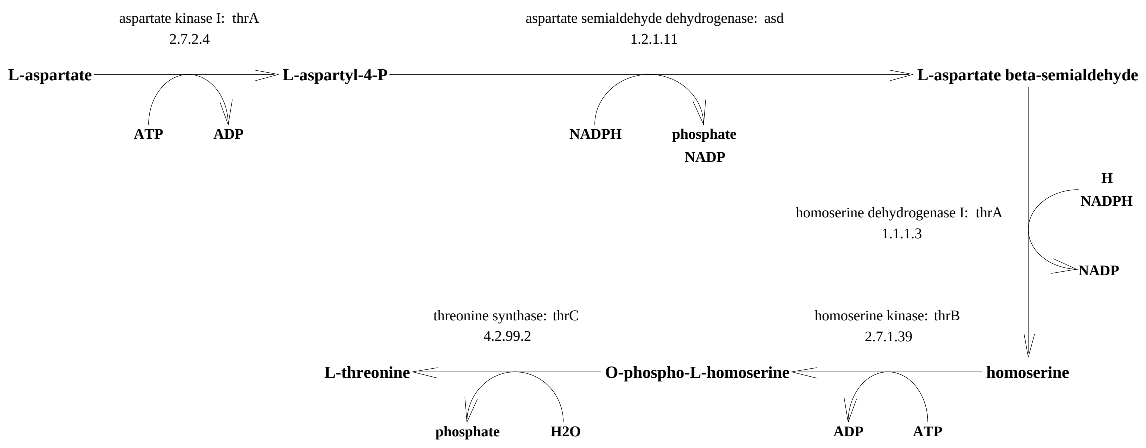
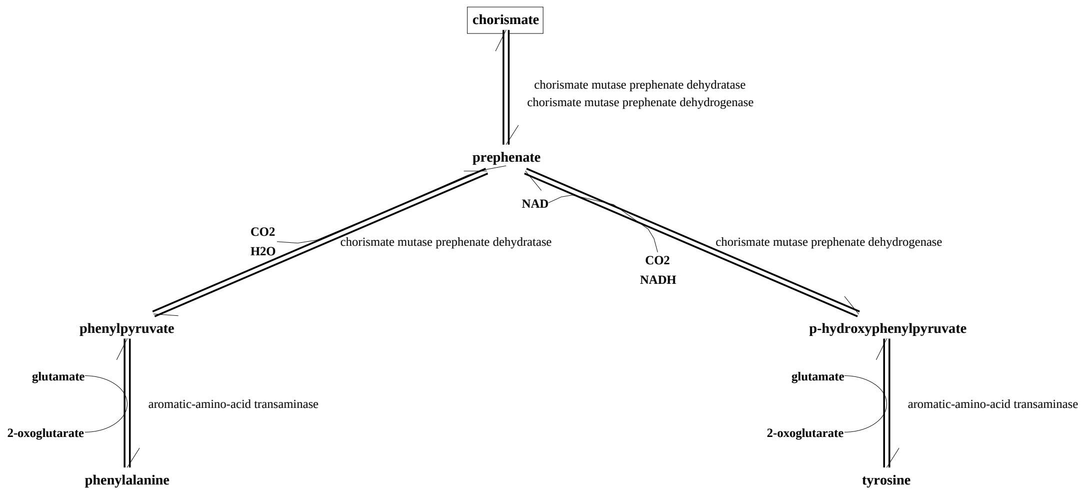


Figure 8: The biosynthetic pathway for threonine.



chorismate
chorismate mutase prephenate dehydratase
chorismate mutase prephenate dehydrogenase
prephenate
NAD
CO2
H2O
chorismate mutase prephenate dehydratase
CO2
NADH
chorismate mutase prephenate dehydrogenase
phenylpyruvate
p-hydroxyphenylpyruvate
glutamate
2-oxoglutarate
aromatic-amino-acid transaminase
phenylalanine
glutamate
2-oxoglutarate
aromatic-amino-acid transaminase
tyrosine

Figure 9: The biosynthetic pathway for phenylalanine and tyrosine.

one cycle plus additional nodes that are not part of the cycle. In (Karp and Paley, 1995a) we advocate the use of hierarchical layout algorithms for drawing complex pathways.

We allow the user to apply different filters to a pathway drawing by allowing the user to alter the parameters that control the behaviors of the drawing algorithms:

- Main compounds can be drawn as chemical names, chemical structures, or both. Or, mains can be drawn only at branch points within a pathway and at the exterior of the pathway. This skeletal view provides a useful summary of complex pathways.
- Side compounds can be drawn as names, structures, or both, or they can be omitted.
- Enzyme names, EC numbers, and gene names can be present or absent from the drawing.

These parameters allow the user to see as little or as much detail as they wish in a given pathway. EcoCyc always draws the skeletal view first when drawing a branching or complex pathway, and allows the user to request additional detail.

Metabolic pathways are interconnected to form a large and complicated network. EcoCyc provides the user with three mechanisms for exploring the structure of that network. Given a pathway display, the user can navigate via hyperlinks to the object display for any visible component of the pathway, such as a compound, reaction, or enzyme. By right-clicking on a compound in a pathway display, the user can view and select from a menu of known pathways involving that compound.

The third form of navigation relies on the notion of a superpathway–subpathway relationship. We say that a pathway P_1 is a superpathway of a pathway P_2 if the pathway graph of P_1 is a supergraph of the pathway graph of P_2. We define superpathways explicitly in the EcoCyc KB as aggregations of smaller, related pathways. For example, we define a superpathway for aromatic amino acid biosynthesis that contains as subpathways the individual pathways for biosynthesis of tryptophan, tyrosine, and phenylalanine. Pathway display windows contain hyperlinks to the superpathways and subpathways of the pathway, allowing navigation from one pathway to those that are closely related.

Finally, some pathway frames contain explicit links to or from other pathways that feed directly in to or out from the current pathway. These links are drawn as part of the pathway graph, and can be used to navigate among sequentially connected pathways.

Representation of Metabolic Pathways We require a representation of pathways within the EcoCyc KB from which we can generate drawings such as Figure 8. One obvious representation is the pathway graph itself, that is, we would store in the KB a list of the nodes (mains, sides, and enzymes) and the edges within a given pathway.

The problem with this approach is that the pathway graph is redundant with respect to information stored elsewhere in the EcoCyc KB. It is redundant with respect to reaction frames (which describe the reactants and products of a reaction) and with respect to enzymatic-reaction frames (which associate an enzyme with a reaction). Elimination of redundancy is a basic principle of database theory, and motivates database normalization. Redundancy is problematic because increased effort is required to create and update redundant information. Furthermore, if a user is not aware of a redundancy, all copies of a given datum may not be changed during an update, leading to an inconsistency.

Therefore, since the KB already describes individual reactions, we can avoid redundancy by describing a pathway in terms of reaction frames. Intuitively, a pathway is a set of linked reactions. We specify the linkages by recording, for each reaction R in a pathway, the reaction(s) preceding R in that pathway. That is, we encode pathways as a set of reaction–predecessor pairs. This *predecessor-list* representation is overkill for a simple linear pathway, but it is needed for cyclic and branched pathways, where the reactions do not form a total order.

We have developed algorithms for inferring the graph representation of a pathway from its predecessor-list representation. This inference is fairly complex because it involves inferring the main and side compounds of a pathway, and the direction of each reaction in pathway (Karp and Paley, 1994). The result is that a pathway as complex as that in Figure 9 can be described with the list structure

```
((r2 r1) (r3 r1) (r4 r3) (r5 r2))
```

where `r1` is the unique identifier for the chorismate mutase reaction, and `r2` is the id of the prephenate dehydratase reaction, and `(r2 r1)` means that the predecessor of `r2` is `r1`.

6.5 Genes

The gene window lists information such as the physical map position of the gene as derived by Rudd (Rudd et al., 1992), the functional class assigned by Riley (Riley, 1993), and the protein product (when known). When the product is an enzyme known to EcoCyc, the display shows the equation(s) of the reaction(s) catalyzed by the enzyme, and the pathways that contain those reactions.

The displayed map position is mouse-sensitive: clicking on this number creates a full-map display (explained in the next section) zoomed in on the region of the chromosome containing the gene so that the gene is visible.

6.6 Genomic Map Displays

We have developed several methods for a scientist to visualize relationships among the roughly 1,500 mapped *E. coli* genes in the EcoCyc KB. These methods are based on displays that allow the user to view the distributions of genes on a genomic map at multiple resolutions.

6.6.1 The Full-Map Display

One method is to draw mapped genes at their chromosomal positions as shown in Figure 10. When first drawn, the full-map display shows only the left-most map in Figure 10, which is a low-resolution view of the entire *E. coli* chromosome. A predefined set of genes are drawn (the genes shown can be changed by the user). The *child maps* to the right show successive magnifications of the region around the *aroC* gene. A given row in a magnified map shows all genes within the region of the chromosome opposite that row. Genes are ordered along each row from left to right according to increasing map position. We limit the horizontal extent of the drawing to allow zooming ("..." at the end of a row indicates that the row was truncated). We also provide a circular map viewer.

Both map viewers allow the user to request a child map that shows a fixed magnification of a region of the chromosome by clicking on a gene name, or on the chromosome, or by querying a gene by name. The user can also request a variable magnification. Arbitrarily many levels of child maps are supported. Since the *E. coli* chromosome is circular, all magnified views of the region around the origin wrap around properly. That is, the user can zoom in on the region of the chromosome of length 2 centisomes that lies between 99 and 1 centisome.

6.6.2 The Partial-Map Display

The partial map allows an investigator to examine map relationships among subsets of *E. coli* genes by beginning with a display of the chromosome with no genes present, and selectively adding genes to the drawing. The user can specify which genes to add in several ways: they can enter an exact gene name or a substring to be matched against all gene names. The user can also request the addition of all genes within one of the classes defined by Riley (Riley, 1993) (the classification is based on the function of the gene products, such as membrane components or tRNAs, or the set

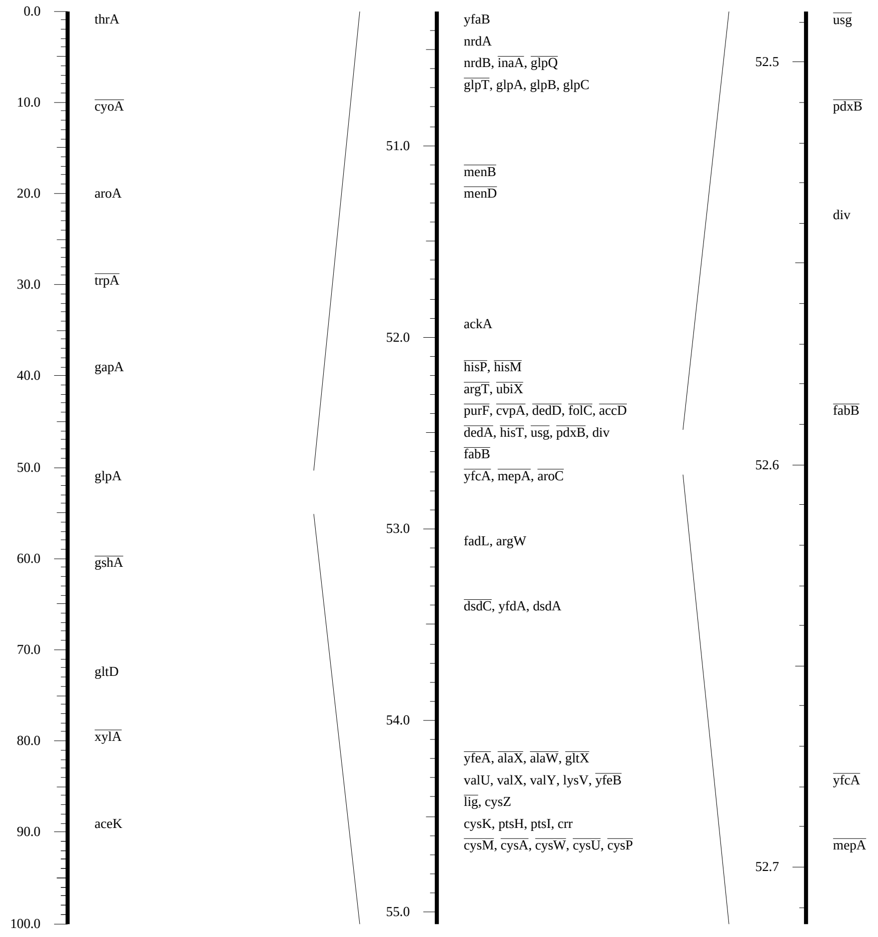


Figure 10: The full-map display that shows two levels of magnification of the region around 52.66 centisomes. A gene with a horizontal line above its name is translated in the counterclockwise direction.

of gene coding for enzymes in a chosen biochemical pathway), and they can request the addition of all genes that code for an enzyme within a given biochemical pathway.

The user can consecutively add sets of genes according to these criteria, and can undo previous add operations in reverse order. For example, the user might add all genes in the biosynthetic pathways for tryptophan, tyrosine, and phenylalanine, and then add all genes coding for membrane proteins. Each set of genes is drawn in a different color. The same zoom capabilities described in conjunction with the full map also work for the partial map.

7 Related Work

A number of projects are now under way to create metabolic databases, and in some cases to integrate metabolic information with genomic information. We summarize the information-management technology used in each project, and the graphical user interface developed for each project. Earlier work includes (Ochs and Conrow, 1991; Seressiotis and Bailey, 1988; Barcza et al., 1990).

Bairoch entered the system of enzyme nomenclature developed by the International Union of Biochemistry and Molecular Biology into the ENZYME DB (?). It describes enzyme-catalyzed reactions across a range of species, but does not identify the species in which a reaction occurs. ENZYME does not use an information management system; it is encoded as a flat file. It also lacks a GUI. A similar encoding of the enzyme-nomenclature system was developed by Suyama et al (Suyama et al., 1993).

Flat files are the most primitive mechanism for managing information. They provide none of the services of a DBMS (no data model, no query language, no consistency constraints, and so forth). The burden is on the DB developer to develop these tools as external programs. The burden is also on the DB developer to define a formal specification of the syntax of the flat file. Few developers create such tools, decreasing the accessibility of the resulting DB.

Letovsky augmented the AceDB system to display metabolic pathways. Imsande and colleagues at Iowa State University are using that software to encode information about the genome and metabolic pathways of the soybean. Letovsky has developed algorithms for automatic drawing of metabolic pathways that are similar to the algorithms in EcoCyc. Letovsky's software displays pathways of linear, tree, and circular topologies, and some combinations thereof. Its drawings are not as close to textbook-style drawings as are those of EcoCyc, and it provides a smaller range of information-filtering capabilities, for example, there is no snake layout of linear pathways. Letovsky has not developed displays of reactions, compounds, or gene-reaction schematics. AceDB is a custom data manager that shares some properties of FRSs: it employs an object data model with arbitrary nesting that is similar to facets and annotations. AceDB contains no constraint system, and thus cannot automatically perform data consistency checking. AceDB has no deductive capabilities, nor attached procedures, nor inheritance of default values. It also lacks the concurrency control and atomic-update capabilities of DBMSs.

Selkov's EMP contains more than 10,000 records describing enzymes from many species (Selkov et al., 1989). Each record encodes information from one journal article. The DB captures information about the function, kinetics, physical properties, and purification of enzymes. EMP is managed as flat files, but a relational implementation is under development. Users search the DB with an information retrieval system that supports fast text searches across the DB, but does not provide graphical drawings of metabolic information. The PUMA system embeds a subset of the EMP data within a PROLOG-based WWW server.[4] PUMA includes simple pathway-display capabilities, and combines multiple alignments of enzyme sequences with the pathway data.

The Metalgen DB combines information about the genes and metabolism of *E. coli* (Rouxel et al., 1993). The DB is implemented using a relational DBMS. Metalgen uses hand drawings of pathways, but does not include drawings of reactions nor compounds, nor a chromosome browser. The manually drawn graphics may be superior to algorithmic drawings in some complex cases where an algorithm does not produce as aesthetically pleasing drawings as a person can. But we find that

[4]See WWW URL `http://www.mcs.anl.gov/home/compbio/PUMA/Production/puma_graphics.html`

our pathway layout algorithms produce acceptable results in the vast majority of cases, follow a more consistent style than manual drawings, reduce the effort required to enter new pathways, and allow updates to the underlying DB to be automatically reflected in drawings.

Kazic developed detailed representations for compounds, and for reasoning about the mechanisms of individual reactions (Kazic, 1993b; Kazic, 1993a), and constructed a DB of compounds (see WWW URL `http://ibc.wustl.edu/klotho/`). Her WWW interface provides several high-quality visualizations of compound structures, including Fischer projections and 3-D wire diagrams. She uses PROLOG to manage the information. PROLOG contains sophisticated query and logical inference capabilities. Its data model can be viewed as both first-order logic and as relations. However, since PROLOG is not a DBMS, it has no built-in notion of a schema, nor of data consistency constraints — PROLOG has no notion of conformant versus nonconformant data.

8 Conclusions

Genome databases should strive to correlate metabolic information with genomic information because these data are synergistic. EcoCyc demonstrates both a software architecture and a graphical framework for accomplishing that integration.

The software architecture includes several components that are implemented in COMMON LISP. EcoCyc demonstrates the viability of COMMON LISP as both a development and a delivery environment. The relative cost of computer hardware and computer programmers has reached the point where we can deliver COMMON LISP applications with acceptable performance on workstations that are affordable to biologists, while minimizing development costs because of COMMON LISP's high productivity. Furthermore, the portability of the COMMON LISP language and the CLIM windowing system provide an excellent environment for developing GUIs that run on multiple platforms, and via the WWW.

The HyperTHEO FRS has proven to be a valuable tool for managing metabolic and genomic information. Its frame data model provides powerful constructs for representing the complexity of biological information, such as the use of annotations to attach literature references to individual data items. HyperTHEO also includes deductive capabilities based on PROLOG, inheritance of default values, and a constraint language. The complexity of biological information ensures that schema evolution is ongoing in bioinformatics projects; schema evolution within HyperTHEO is considerably easier than in most DBMSs. HyperTHEO also includes a declarative query language, schema-driven KB browsing and editing tools, and a client-server architecture that supports demand fetching of frames over a network by multiple users.

The contributions of our work on the EcoCyc GUI include the following. We defined various problems and issues that arise in displaying metabolic information, such as what classes of pathway layouts biologists are familiar with from biochemistry texts, and how to represent a pathway minimally. We designed visualizations of genes, genomic maps, enzymes, biochemical reactions, and metabolic pathways, and described algorithms for computing those visualizations. We also illustrated the value of putting intelligence (inference) in a GUI to allow that GUI to construct a complex visualization (a pathway drawing) from an input that is relatively simple for a user to construct (our predecessor representation of pathways). In addition, we show that we can communicate complex information effectively by allowing the user to apply different filters to that information to view it from multiple perspectives, such as by showing a subset of the loci on a genomic map, or a skeletal view of a metabolic pathway.

EcoCyc provides two types of query facilities. The first type allows the user to choose from a small number of commonly used queries on a menu. These queries allow objects to be retrieved by name (with extensive synonym lists provided), by substring search against names and synonyms, and according to several comprehensive classification hierarchies that are supplied with EcoCyc. Classification hierarchies are part of the frame data model, and are therefore easy to encode within HyperTHEO. Additional class-specific queries are usually supported, such as querying reactions by EC number, or compounds by chemical structure. The second query facility is hypertext navigation.

One lesson learned in this project is that bioinformatics can benefit from computer-science research in a short time frame. The research our group performed on extending the storage capabilities of HyperTHEO was implemented and tuned to produce a useful substrate for managing the EcoCyc KB within two years. Some bioinformaticians have argued for the use of long-established, reliable technology within bioinformatics projects (Robbins, 1994). But existing technology does not solve many of the hard computational problems within bioinformatics. Focused research projects can produce practical results within a limited time.

Acknowledgments

The authors are grateful to our collaborator Monica Riley for extensive discussions. Programming contributions to EcoCyc were made by Rezal Rahman, Evgeni Shevelev, and Adarsh Bhat. This work was supported by grant R29-LM-05413-01A1 from the National Library of Medicine, and by grant 1-R01-RR07861-01 from the National Center for Research Resources. The contents of this article are solely the responsibility of the authors and do not necessarily represent the official views of the National Institutes of Health.

References

Attias, R. and Dubois, J. (1990). Substructure systems: Concepts and classifications. *J. Chem. Inf. Comput. Sci.*, 30:2–7.

Bailey, J. (1991). Toward a science of metabolic engineering. *Science*, 252:1668–1675.

Barcza, S., Kelly, L., and Lenz, C. (1990). Computerized retrieval of information on biosynthesis and metabolic pathways. *J. Chem. Inf. Comput. Sci.*, 30:243–251.

Brutlag, D., Galper, A., and Millis, D. (1991). Knowledge-based simulation of DNA metabolism: prediction of enzyme action. *Computer Applications in the Biosciences*, 7(1):9–19.

Cameron, C. and Tong, I. (1993). Cellular and metabolic engineering: An overview. *Applied Biochemistry and Biotechnology*, 38:105–140.

Gaasterland, T. and Selkov, E. (1995). Reconstruction of metabolic networks using incomplete information. In Rawlings, C., Clark, D., Altman, R., Hunter, L., Lengauer, T., and Wodak, S., editors, *Proc. of the Third International Conference on Intelligent Systems for Molecular Biology*, pages 127–135, Menlo Park, CA. AAAI Press.

Genesereth, M. R. and Fikes, R. E. (1992). Knowledge Interchange Format, Version 3.0 Reference Manual. Technical Report Logic-92-1, Computer Science Department, Stanford University.

Goodman, N. (1994). An object oriented DBMS war story: Developing a genome mapping database in C++. In *Modern Database Management: Object-Oriented and Multidatabase Technologies.* ACM Press.

Haas, J., Aaronson, J., and Overton, G. C. (1993). Using analogical reasoning for knowledge discovery in a molecular biology database. In *Proc. of the Second International Conference on Information and Knowledge Management*, pages 554–564, New York, NY. Association for Computing Machinery.

Hunter, L., Searls, D., and Shavlik, J., editors (1993). *Proc. of the First International Conference on Intelligent Systems for Molecular Biology.* AAAI Press, Menlo Park, CA.

Karp, P. (1992). A knowledge base of the chemical compounds of intermediary metabolism. *Computer Applications in the Biosciences*, 8(4):347–357.

Karp, P. (1994). Distinguishing knowledge bases and data bases: Who's on first and what's on second. Technical Report 546, SRI International AI Center.

Karp, P. (1996). The EcoCyc user's guide. unpublished; see URL `http://ecocyc.PangeaSystems.com/ecocyc/doc/ecocyc-uguide/paper.html`.

Karp, P., Lowrance, J., Strat, T., and Wilkins, D. (1994). The Grasper-CL graph management system. *LISP and Symbolic Computation*, 7:245–282.

Karp, P. and Mavrovouniotis, M. (1994). Representing, analyzing, and synthesizing biochemical pathways. *IEEE Expert*, 9(2):11–21.

Karp, P., Myers, K., and Gruber, T. (1995). The Generic Frame Protocol. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*, pages 768–774. See also URL `ftp://ftp.ai.sri.com/pub/papers/karp-gfp95.ps.Z`.

Karp, P. and Paley, S. (1994). Representations of metabolic knowledge: Pathways. In Altman, R., Brutlag, D., Karp, P., Lathrop, R., and Searls, D., editors, *Proc. of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 203–211, Menlo Park, CA. AAAI Press.

Karp, P. and Paley, S. (1995a). Automated drawing of metabolic pathways. In Lim, H., Cantor, C., and Robbins, R., editors, *Proc. of the Third International Conference on Bioinformatics and Genome Research*, pages 225–238. World Scientific Publishing Co. See also URL `ftp://ftp.ai.sri.com/pub/papers/karp-bigr94.ps.Z`.

Karp, P. and Paley, S. (1995b). Knowledge representation in the large. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence*, pages 751–758. See also URL `ftp://ftp.ai.sri.com/pub/papers/karp-perkobj95.ps.Z`.

Karp, P. and Riley, M. (1993). Representations of metabolic knowledge. In (Hunter et al., 1993), pages 207–215.

Karp, P., Riley, M., Paley, S., and Pellegrini-Toole, A. (1996). EcoCyc: Electronic encyclopedia of *E. coli* genes and metabolism. *Nuc. Acids Res.*, 24(1):32–40.

Kazic, T. (1993a). Reasoning about biochemical compounds and processes. In Lim, H. W., Fickett, J. W., Cantor, C. R., and Robbins, R. J., editors, *Second International Conference on Bioinformatics, Supercomputing and the Human Genome Project*, pages 35–49, Singapore. World Scientific.

Kazic, T. (1993b). Representation, reasoning, and the intermediary metabolism of *Escherichia coli*. In *Proc. of the 26th Annual Hawaii International Conference on System Sciences*, volume I, pages 853–862. IEEE Computer Society Press.

Letovsky, S. and Berlyn, M. (1994). Issues in the design of complex scientific databases. In *Proceedings of the 1994 Hawaii International Conference on System Sciences*, pages 5–14.

Mavrovouniotis, M. L. (1993). Identification of qualitatively feasible metabolic pathways. In Hunter, L., editor, *Artificial Intelligence and Molecular Biology*. AAAI Press / MIT Press.

Mitchell, T., Allen, J., Chalasani, P., Cheng, J., Etzioni, E., Ringuette, M., and Schlimmer, J. (1989). Theo: A framework for self-improving systems. In *Architectures for Intelligence*. Erlbaum.

Ochs, R. and Conrow, K. (1991). A computerized metabolic map. *J. Chem. Inf. Comput. Sci.*, 31:132–137.

Overton, G. C., Koile, K., and Pastor, J. A. (1989). GeneSys: A knowledge management system for molecular biology. In Bell, G. and Marr, T., editors, *Computers and DNA*, pages 213–240, Reading, MA. Addison-Wesley.

Paley, S. and Karp, P. (1995). Adapting CLIM applications for use on the World Wide Web. In *Proc. of the Association of Lisp Users Meeting and Workshop*, pages 1–9. See URL `ftp://ftp.ai.sri.com/pub/papers/paley-luv95.ps.Z`.

Perrerie, G., Chevenet, F., Dorkeld, F., Vermat, T., and Gautier, C. (1994). Building integrated systems for data representation and analysis in molecular biology. In *Proceedings of the 1994 Hawaii International Conference on System Sciences*, pages 89–97.

Perrerie, G., Dorkeld, F., and Gautier, C. (1993). Object-oriented knowledge bases for the analysis of prokaryotic and eukaryotic genomes. In (Hunter et al., 1993), pages 319–327.

Riley, M. (1993). Functions of the gene products of *Escherichia coli*. *Microbiological Reviews*, 57:862–952.

Robbins, R. (1994). Report of the invitational DOE workshop on genome informatics, 26–27 April 1993; Genome informatics I: Community databases. *Journal of Computational Biology*, 1(3):173–190.

Rouxel, T., Danchin, A., and Henaut, A. (1993). METALGEN.DB: Metabolism linked to the genome of *Escherichia coli*, graphics oriented database. *Computer Applications in the Biosciences*, 9(3):315–324.

Rudd, K. E., Bouffard, G., and Miller, W. (1992). Computer analysis of *E. coli* restriction maps. In Davies, K. E. and Tilghman, S. M., editors, *Genome Analysis, Vol.4: Strategies for Physical Mapping*, pages 1–38. Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York.

Schmeltzer, O., Medigue, C., Uvietta, P., Rechenmann, F., Dorkeld, F., Perrerie, G., and Gautier, C. (1993). Building large knowledge bases in molecular biology. In (Hunter et al., 1993), pages 345–353.

Schweigert, S., Herd, P., and Sibbald, P. (1995). Issues in incorporation semantic integrity in molecular biological object-oriented databases. *Computer Applications in the Biosciences*, 11(4):339–348.

Selkov, E., Goryanin, I., Kaimatchnikov, N., Shevelev, E., and Yunus, I. (1989). Factographic data bank on enzymes and metabolic pathways. *Studia Biophysica*, 129(2–3):155–164.

Seressiotis, A. and Bailey, J. (1988). MPS: An artificially intelligent software system for the analysis and synthesis of metabolic pathways. *Biotechnology and Bioengineering*, 31:587–602.

Stephanopoulos, G. and Vallino, J. (1991). Network rigidity and metabolic engineering in metabolite overproduction. *Science*, 252:1675–1681.

Suyama, M., Ogiwara, A., Nishioka, T., and Oda, J. (1993). Searching for amino acid sequence motifs among enzymes: the enzyme-reaction database. *Computer Applications in the Biosciences*, 9(1):9–15.

Webb, E. C. (1992). *Enzyme Nomenclature, 1992: Recommendations of the nomenclature committee of the International Union of Biochemistry and Molecular Biology on the nomenclature and classification of enzymes*. Academic Press.

Weininger, D. (1988). SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.*, 28:31–36.