BIOINFORMATICS ONTOLOGY

Vol. 16 no. 3 2000
Pages 269–285

An ontology for biological function based on molecular interactions

Peter D. Karp

Bioinformatics Research Group, SRI International, EK223, 333 Ravenswood Avenue, Menlo Park, CA 94025, USA

Received on March 28,1999; revised on September 9, 1999; accepted on September 30, 1999

Abstract

Motivations: *A number of important bioinformatics computations involve* computing with function: *executing computational operations whose inputs or outputs are descriptions of the functions of biomolecules. Examples include performing functional queries to sequence and pathway databases, and determining functional equality to evaluate algorithms that predict function from sequence. A prerequisite to computing with function is the existence of an ontology that provides a structured semantic encoding of function.* Functional bioinformatics *is an emerging subfield of bioinformatics that is concerned with developing ontologies and algorithms for computing with biological function.*

Results: *The article explores the notion of computing with function, and explains the importance of ontologies of function to bioinformatics. The functional ontology developed for the EcoCyc database is presented. This ontology can encode a diverse array of biochemical processes, including enzymatic reactions involving small-molecule substrates and macromolecular substrates, signal-transduction processes, transport events, and mechanisms of regulation of gene expression. The ontology is validated through its use to express complex functional queries for the EcoCyc DB.*

Contact: *pkarp@ai.sri.com*

Introduction

Molecular-biology databases (DBs) have become an essential complement to the molecular-biology literature. Over time, more and more of the world's biological knowledge will be deposited into these DBs. How can we assure that all deposited knowledge can be extracted in a meaningful form? The hypothesis of this paper is that declarative representations of biological function are essential for maximizing the utility of biological DBs because they increase our ability to encode functional information in biological DBs in a meaningful form. Declarative representations are structured semantic encodings of information that can be queried and manipulated by computer programs in a reliable fashion (this definition will be refined later in the article).

Representations (or ontologies) of biological function are essential for encoding functional descriptions of the nucleic-acid and amino-acid sequences that are entering the public DBs at an ever-increasing pace, so that functional information can be queried and manipulated by bioinformatics programs in addition to being interpreted by scientists. Most biological DBs assume that the functional descriptions of their entries are intended to be read by humans, rather than processed by computer—but forms of analysis that require a person to read thousands of DB entries do not scale with the rapidly increasing size of most DBs.

This paper begins with a detailed explanation of what is meant by the notion of 'computing with function,' and of why this notion is so important. It then dissects biological function into two subconcepts: local function and integrated function. The paper articulates general principles of representing local biological function, and presents the ontology of function developed by the EcoCyc project (Karp *et al.*, 1999). EcoCyc is an integrated pathway/genome database that describes the *Escherichia coli* genome, its metabolic pathways, and other aspects of its cell function. EcoCyc encodes biological function at the molecular level by capturing information about the molecular components of the *E.coli* cell, and about the interactions of those components.

An ontology is a specification of a conceptualization that is designed for reuse across multiple applications and implementations (Gruber, 1993, 1995; Guarino, 1996; Guarino and Giaretta, 1995). Put another way, a specification of a conceptualization is a written, formal description of a set of concepts and relationships in a domain of interest. For example, a DB schema is a specification of a conceptualization. We present examples of how EcoCyc employs its ontology (its schema) to encode the functions of metabolic enzymes, signal-transduction proteins, transporters, and DNA-binding repressor proteins. However, the ontology generalizes well beyond *E.coli* alone; at Pangea Systems our group used the ontology to create pathway/genome DBs for a variety of microorganisms

(Karp, 1999), and to create a general pathway DB called MetaCyc.[†]

Functional bioinformatics: Computing with function

Functional bioinformatics is an emerging subfield of bioinformatics that is concerned with ontologies and algorithms for computing with biological function. Functional bioinformatics is the computational counterpart of functional genomics. We define functional genomics as large-scale phenotypic analysis of a sequenced genome.

Functional bioinformatics is concerned with managing and analyzing functional-genomics data, such as gene-expression experiments and large-scale knock-out experiments. Although functional bioinformatics is concerned with representing, visualizing, and computing with functional descriptions of individual genes and gene products, like functional genomics, functional bioinformatics emphasizes large-scale computational problems, such as problems involving complete metabolic networks and genetic networks. Functional bioinformatics research areas include: prediction of metabolic networks and genetic networks from genomics data; simulation of metabolic and genetic networks; design of ontologies of biological function; computational prediction of the phenotype of a knock-out mutant; and computational prediction of the growth-media requirements of an organism from its metabolic network.

The term 'computing with biological function' refers to algorithms whose inputs or outputs are descriptions of biological functions. For example, an algorithm that predicts the metabolic network of an organism from its genome produces an output (a metabolic network) that is a description of a biological function.

Problems outside the realm of functional bioinformatics include base calling, genome assembly, gene finding, prediction of protein secondary and tertiary structure, and visualization of genomic maps. The boundary is a fuzzy one since a gene-finding algorithm is of course predicting the function of a region of the genome. We also call sequence-similarity algorithms outside the domain of functional bioinformatics because these algorithms are concerned with detecting sequence similarity, and not with inferring function *per se*. The inputs and outputs of these algorithms are not descriptions of biological functions—the decision as to whether to infer a shared function based on a sequence similarity is left to the user of the algorithm.

This section provides several examples of computing with function to illustrate this concept and its importance in bioinformatics and biology. For some of the problems listed in this section, this article demonstrates concrete ways in which our functional ontology provides solutions to these problems. The first example—deciding if two biomolecules share the same function—is conceptually trivial, but sadly, the functional descriptions in most of the public sequence DBs do not allow this important type of functional computation to be performed reliably. By 'performed reliably' I mean that although some rough approximations to such a query might be possible, we would not have confidence in their results.

The second set of examples are function-based queries to bioinformatics DBs. These examples are more significant biologically, and they are also queries that existing sequence DBs cannot answer, but that EcoCyc can answer because of its ontology.

The third section provides several more complicated examples of computing with function, such as searching for novel pathways in a metabolic network, and predicting the phenotype of a knock-out mutant. Many of these examples are unsolved problems in functional bioinformatics.

Testing functional equality

The first functional computation we consider is the operation `functions-equal(F1,F2)`, which we define to be a comparison of two functional descriptions F_1 and F_2 of two macromolecules. The operation returns True if and only if F_1 and F_2 describe the same function.

The `functions-equal` operation would be extremely useful for the evaluation of automated programs for prediction of protein function from sequence, such as GeneQuiz (Scharf *et al.*, 1994), Magpie (Gaasterland and Sensen, 1996), and GeneWorld (a product of Pangea Systems). To evaluate such programs we would like to compare the functional predictions that they make across thousands of test cases, such as all proteins in all fully sequenced microbial genomes. We could use `functions-equal` to ask whether the function F_1 predicted for a protein sequence by GeneQuiz is the same as function F_2 predicted by the team who annotated the genome, for each protein in the test set. Such evaluation is critical for assessing the accuracy of existing programs and for improving procedures for functional annotation.

The reason most sequence DBs do not support this computation is that they use natural-language descriptions (uncontrolled text) of function that programs cannot compute with. GeneQuiz and Magpie assign such natural-language descriptions to the sequences that they process. Although natural-language descriptions clearly have some structure, that structure is too complex to be reliably queried and dissected by computer programs. For example, the descriptions 'transporter for tryptophan,' 'tryptophan transporter,' 'L-tryptophan transporter,' and 'TnaB tryptophan ArAAP transporter' are equivalent functions, but for a program to compare these phrases and infer they are the same, the program must understand that

[†] See URL http://ecocyc.PangeaSystems.com/ecocyc/metacyc.html.

'tryptophan' and 'L-tryptophan' are probably synonyms, that word order can vary, and that 'TnaB' is a gene name and 'ArAAP' is a protein-family name.

Constructing a program that could be relied upon to compare two such functional descriptions and decide whether they are equivalent would be a difficult if not impossible task when we consider the range of natural-language descriptions used in DBs such as Genbank and SwissProt. One could argue that rule-driven parsing programs could be written to interpret and compare these descriptions of transporters, but our claim is that more accurate results can be obtained with less effort by using declarative representations in the first place, particularly since we could have little confidence that our rules correctly anticipated *all* natural-language descriptions that any scientist might ever write (not to mention the infinite variety of spelling errors that have found their way into the public DBs).

Controlled vocabularies will not suffice. Another approach to describing biological function is to employ controlled vocabularies of functional terms. Use of controlled vocabularies would clearly be preferable to the natural-language descriptions of function described in the preceding section. Although controlled vocabularies could be used to solve the `functions-equal` problem, they do have significant limitations. For example, a simple list of controlled terms could not be used to evaluate the `function-subsumes` operation described in Section **Function-based database queries**—a taxonomy of functional terms would be required for that operation.

Nor could a controlled vocabulary be used to evaluate the functional DB queries described in Section **Function-based database queries**, nor to solve the more complex functional-bioinformatics problems described in Section **Additional functional computations**. The reason is that controlled vocabularies constitute a simple class of ontologies consisting of a list of terms. Controlled vocabularies by definition do not define the web of relationships for those terms that provide semantics for the terms. We might define a controlled term such as 'pyruvate kinase,' but that term in itself does not even specify the substrates of the enzyme, and therefore could not be used to answer a DB query such as: find all enzymes whose substrates include phosphoenolpyruvate (which is a substrate of pyruvate kinase). Nor does a single term capture information about the cofactors or modulators of the enzyme, nor all of the other information that our ontology captures. Because controlled vocabularies capture so little domain semantics, their power is extremely limited.

Function-based database queries

Function-based DB queries can be used to address a number of biological questions, such as to enhance our understanding of sequence–function and structure–function relationships. An additional operation, `function-subsumes(F1,F2)`, would allow us to retrieve all proteins whose function F_2 is subsumed by some more general user-defined function, F_1. For example, the function 'alcohol dehydrogenase' is subsumed by the function 'oxidoreductase.'

A number of examples of functional queries are presented here, along with an actual query that answers each question for the EcoCyc DB. Each EcoCyc query is written in the LISP language using operations in the Generic Frame Protocol (Karp *et al.*, 1995), and has been evaluated with respect to the EcoCyc DB.

Some of the key LISP operations used in these examples are as follows.

- `get-class-all-instances(class)`—Returns all objects that are instances of `class`.
- `get-slot-value(frame,slot)`—Returns the first value of `slot` of `frame`.
- `get-slot-values(frame,slot)`—Returns the set of values of `slot` of `frame`.
- `member-slot-value-p(frame,slot,value)`—Tests whether `value` is one of the current values of `slot` of `frame`.
- `coercible-to-frame-p(thing)`—Tests whether `thing` is a frame in the current KB.
- `instance-all-instance-of-p(frame, class)`—Tests whether `frame` is a child of `class`.
- `enzymes-of-reaction(reaction)`—Returns all enzymes that catalyze `reaction`.

1. Find all oxidoreductases

```
(loop for x in (get-class-all-instances 'EC-1)
      append (enzymes-of-reaction x) )
```

The query iterates through all instances of the EcoCyc class for the Enzyme Commission class #1, which is oxidoreductases, and uses a built-in function that returns all enzymes that catalyze each of those reactions, and appends together the results.

2. Find all enzymes for which pyruvate is a substrate

```
(loop for x in (get-class-all-instances
           '|Reactions|)
      when (member-slot-value-p x 'substrates
           'pyruvate)
      append (enzymes-of-reaction x) )
```

The query iterates through all instances of the class Reactions; for those instances whose `substrates` *slot contains pyruvate, the query appends together the enzymes that catalyzes those reactions.*

3. Find all enzymes for which ATP is an inhibitor

```
(loop for x in (get-class-all-instances '|Enzymatic-Reactions|)
      when (member-slot-value-p x 'inhibitors-all 'atp)
      collect (get-slot-value x 'enzyme) )
```

4. Find all enzymes for which ATP is a competitive inhibitor

```
(loop for x in (get-class-all-instances '|Enzymatic-Reactions|)
      when (member-slot-value-p x 'inhibitors-competitive 'atp)
      collect (get-slot-value x 'enzyme) )
```

5. Find all enzymes for which pyridoxal phosphate is a prosthetic group

```
(loop for x in (get-class-all-instances '|Enzymatic-Reactions|)
      when (member-slot-value-p x 'prosthetic-groups 'Pyridoxal_Phosphate)
      collect (get-slot-value x 'enzyme) )
```

6. Find all enzymes in the TCA cycle

```
(loop for x in (get-slot-values 'TCA 'reaction-list)
      append (enzymes-of-reaction x) )
```

7. Find all proteins that autophosphorylate

```
(loop for x in (get-class-all-instances '|Reactions|)
      for left = (get-slot-values x 'left)
      when (and (= 2 (length left))
                (setq protein
                      (loop for reactant in left
                            thereis
                            (and (coercible-to-frame-p reactant)
                                 (instance-all-instance-of-p reactant '|Proteins|)
                                 reactant) ) )
                (member-slot-value-p x 'left 'ATP)
                (= 2 (length (get-slot-values x 'right)))
                (member-slot-value-p x 'right 'ADP)
                )
      collect protein)
```

The query searches for reactions with ATP and a protein as reactants, and with two products, one of which is ADP.

8. Find all permeases for l-lysine

```
(loop for x in (get-class-all-instances '|Transport-Reactions|)
      when (member-slot-value-p x 'substrates 'Lys)
      append (enzymes-of-reaction x) )
```

9. Find all enzymes that are regulated by phosphorylation, i.e. the enzyme activity changes as a result of phosphorylation. (This query finds reactions whose reactants are a protein plus ATP, and whose products are a protein plus ADP, where the two proteins catalyze different sets of enzymatic reactions.)

```
(loop for x in (get-class-all-instances '|Reactions|)
      do (loop for (side1 side2) in '((left right) (right left))
               for cpds1 = (get-slot-values x side1)
               for cpds2 = (get-slot-values x side2)
               when (and (= (length cpds1 2))
                         (= (length cpds2 2))
```

```
                    (member-slot-value-p x side1 'ATP)
                    (setq enz1 (first (set-difference cpds1 '(ATP))))
                    (coercible-to-frame-p enz1)
                    (instance-all-instance-of-p enz1 '|Proteins|)
                    (member-slot-value-p x side2 'ADP)
                    (setq enz2 (first (set-difference cpds2 '(ATP))))
                    (coercible-to-frame-p enz2)
                    (instance-all-instance-of-p enz2 '|Proteins|)
                    (not (equal (get-slot-values enz1 'catalyzes)
                                (get-slot-values enz2 'catalyzes) ))
                    )
               do (push (list enz1 enz2) *enzymes*)
               ) )
```

10. Find all enzymes that accept glutamine and ammonia as alternative substrates

```
(loop for x in (get-class-all-instances '|Enzymatic-Reactions|)
      do (loop for item in (get-slot-values x 'alternative-substrates)
               when (or (and (fequal 'AMMONIA (first item))
                             (find 'GLN (rest item)) )
                        (and (fequal 'GLN (first item))
                             (find 'AMMONIA (rest item)) )
                        )
               (push (get-slot-value x 'enzyme) *enzymes*)
          ) )
```

Additional functional computations

There is a class of more complex functional computations that involve causal reasoning about chains of functional interactions, such as:

1. Search for novel pathways with specified characteristics in a metabolic network, e.g. find a series of enzymatic steps that convert metabolite X into metabolite Y (Mavrovouniotis, 1989).
2. Predict the growth-media requirements of a cell based on a description of the full metabolic network and transporter complement of the cell.
3. Predict the phenotype of a knock-out mutant. For example, when a given function is removed from the cell, how are its growth-media requirements changed?
4. Quantitative simulation of the actions of one or more metabolic pathways.
5. Trace radioactive label through a metabolic pathway. Given a pathway input compound containing a radioactive atom, predict which output compound(s) from the pathway will contain that label (Cohen and Bergman, 1994).

The ontology presented herein subsumes the ontology that Mavrovouniotis used to solve problem 1. We believe the ontology will suffice to solve 2 and 3, but no proof yet exists. The ontology will require further development to solve 4 and 5.

Biological function

Biological function is a complex concept. Here we examine the notion of function in more detail, and find that the word 'function' encompasses two very different concepts in biology.

Consider the following two definitions of 'Function' from Websters.

1. Function: The action for which a person or thing is specially fitted or used or for which a thing exists.
2. Function: The normal and specific contribution of a bodily part to the economy of a living organism.

How can we interpret these definitions with respect to biological function? When applied to a molecular component of the cell, definition 1 asks what *action* that component performs within the cell. That is, what other cellular components does it interact with, such as proteins or small molecules? What are the nature of those interactions: Does it bind to a protein? Does it catalyze chemical transformations among small molecules? Does it activate or inhibit an enzyme, or serve as an enzyme cofactor?

Definition 2 refers to the role played by the molecule within the operation of the cell as a whole. If we understood the function of a protein, we would know how that protein contributed to the behavior of the cell. That is, we would understand which *behavioral component* of the cell (which cellular subsystem) the protein is associated with. Put another way: if we understand the function of a protein, then we know how the cell would behave when that protein is absent, and why.

Definitions 1 and 2 refer to two very different notions of function that we will call *local function* and *integrated function*, respectively. Questions of local function concern the individual interactions that can occur between the biological entity of interest, and other entities. If we are interested in the local function of an enzyme, then we are interested in the substrates that it can act on, and in the ligands that activate or inhibit the enzyme.

Questions of integrated function concern the role that a biological entity plays in some larger system of which it is a part. For example, an enzyme in the pathway for lysine biosynthesis might have several integrated functions, including 'biosynthesis of lysine,' 'amino-acid biosynthesis,' and 'growth on lysine-poor media.' This enzyme has multiple integrated functions because it participates in multiple biological systems of hierarchical, nested scope. In this example, the system 'lysine biosynthesis' is a subsystem of the system 'amino-acid biosynthesis.' When we refer to a system such as lysine biosynthesis, we are describing a subprocess of the whole behavior of the cell. Each of these cellular systems can be identified by decomposing the cellular growth process into smaller linked chains of events. That decomposition can be performed in many alternative ways, and it is likely that different scientists will have different intuitions about which decomposition is best. My belief is that multiple alternative decompositions will be scientifically useful because it will prove impossible to define a single best set of biologically acceptable rules for performing this decomposition.

One illustration of the fact that local function and integrated function are truly different concepts is the fact that the exact same protein could have identical local functions in two different organisms, but different integrated functions. Consider an enzyme E that catalyzes the reaction $X + Y = Z$ that is present with the exact same amino-acid sequence in two different bacteria, B_1 and B_2. However, imagine that E is used in glycolysis in B_1, but is used in gluconeogenesis in B_2. The enzyme might be used in two different pathways either because it is expressed under different conditions in the two organisms, or because the glycolytic pathway does not occur in B_2, and the gluconeogenesis pathway does not occur in B_1. E clearly catalyzes the same molecular interaction in both organisms, therefore it has the same local function in each, but it just as clearly operates in the context of two different cellular processes in the two organisms, and therefore has different integrated functions.

Note that local functions are not necessarily the primitive functions in a functional decomposition of the biochemical network of the cell. For example, a local function such as enzyme activity can usually be decomposed into more elementary reaction steps that comprise the enzyme mechanism.

In addition, biological function has both quantitative aspects and qualitative aspects. For example, the fact that an enzyme catalyzes a particular reaction, and the fact that a given compound inhibits that enzyme, are qualitative aspects of function. The rate at which the enzyme catalyzes the reaction, the degree of inhibition that is provided by a given concentration of inhibitor, and the number of copies of the enzyme that are typically present in the cell, are quantitative aspects of function.

Biological function can also be addressed at many levels of organization, from interactions between molecules, to interactions between cells, to interactions between organs and tissues, to interactions between organisms. Function depends on the environment, the ontogeny, and the phylogeny of the organism.

Declarative representations

We define *declarative representations* by describing both what they are, and what they are not. Declarative representations are structured representations: they break information down into atomic components, and define relationships among those atomic components. Structured representations allow us to capture different facets of biological function, and allow us to encode biological function with high fidelity in the sense that we can represent exactly what is known about the function of a biological entity, without encoding in a DB either more or less than is actually known experimentally about the entity. Declarative representations allow us to query and to process information in a reliable fashion, therefore, declarative representations of function allow us to compute with function.

One type of representation that is not declarative is the natural-language representation that most biological DBs use to describe function. Natural language has a complex and ambiguous structure that cannot be reliably queried by machine, as described in Section **Testing functional equality**.

A second type of representation that is not declarative is a procedural representation, such as that embodied in most simulation programs. For example, imagine writing a C program that simulates the TCA cycle of *E.coli*. Although this program clearly represents information about the enzymes of the TCA cycle, the information is almost certainly embedded within the C program in such a

convoluted fashion that it becomes impossible to compute with *for other purposes besides simulation*. If we wished to write a second program that somehow queried the first program (by analyzing its source code) to determine which enzyme(s) in the TCA cycle consume citrate, we would have an extremely difficult programming task ahead of us. Of course, if the simulation program stored its data in clearly defined tables or other data structures, those structures could be used for other purposes. But the more the data becomes disentangled from the procedures of the simulation program, the more it takes on a declarative flavor. Therefore, an advantage of declarative representations is that, if designed well, they can be employed for multiple purposes.

The OCELOT frame representation system

To fully understand the EcoCyc ontology, one must first understand the data model for the EcoCyc DB. A data model is the primitive data structuring mechanism that underlies a DB management system (DBMS). The most common data models are the relational model used by relational DBMSs, and the object model that underlies object-oriented DBMSs.

Frame knowledge representation systems (FRSs) are an information-management technology developed in the Artificial Intelligence community that use a variant of the object data model (Karp, 1992). The EcoCyc data are encoded within an FRS called OCELOT (Karp and Paley, 1996; Karp *et al.*, in press). In FRS terminology, objects are called frames. Frames come in two varieties: classes and instances. Class frames represent generic types of objects, such as the class of all genes and the class of all proteins. Classes can be arranged in class–subclass hierarchies, for example, the class of all proteins can be partitioned into the subclass of polypeptides (single gene products), and the subclass of protein complexes (a collection of chemically associated polypeptides). Instance frames represent specific entities, such as a specific gene or a specific polypeptide. Every frame has an identifier that is unique within the DB.

Associated with each class are a set of slots, which define attributes and relationships of the class. Slots can have one or more values, of several possible primitive datatypes. For example, the EcoCyc class called `Proteins` has a slot called `pI` that takes a single value that is a real number: the isoelectric point of the protein. The EcoCyc class called `Protein-Complexes` has a slot called `Components` that takes multiple values, which can be the identifiers of instances of the class `Polypeptides`—these values are the polypeptide subunits of the protein complex. A variety of constraints can be attached to slots to define acceptable values for the slot, for example, the `pI` slot is constrained to take at most one value that must be a real number between 0 and 14. The `Components` slot is constrained such that its values can only be identifiers of instances of the class `Chemicals` (a parent of the class `Proteins`). Generally, a class frame or an instance inherits the slots defined for its parent classes.

In February of 1998, the EcoCyc DB contained 742 class frames and 14 959 instance frames.

The EcoCyc ontology of function

The ontology described in this paper is concerned with local function, not with integrated function. It is concerned with qualitative rather than quantitative aspects of function. And the ontology addresses function at the molecular level, but not function at the cellular or organismal level.

The principles that guide the representation of function in EcoCyc are:

1. Represent the local function of an entity by describing that entity as a distinct molecular species.
2. Describe each distinct local function of that species in a declarative fashion, using a distinct DB object to encode the function.
3. In all of its molecular interactions, an entity must act as some combination of a substrate, a catalyst, a modulator,[‡] or a cofactor.[§]

The remainder of the paper will describe these principles in more detail, and will illustrate how they are applied within EcoCyc.

To describe all of the distinct molecular species in a pathway, we create frames in the EcoCyc DB for every species. In EcoCyc, a frame exists for every substrate in a pathway, and for every enzyme in the pathway. EcoCyc defines objects for every enzyme cofactor, activator, and inhibitor compound. We take very seriously the notion of creating a distinct object for every molecular species. For example, the *E.coli* isocitrate dehydrogenase exists in both an inactive (phosphorylated) and active (unphosphorylated) form. EcoCyc contains a distinct object for each form. The DB contains distinct objects for different modified forms of a protein, and for an enzyme complex as well as for its component polypeptides.

[‡] A modulator is an activator or an inhibitor of a bioreaction that interacts directly with a catalyst or a substrate of that bioreaction, such as through an allosteric interaction. We do not use this term to mean any indirect type of activation or inhibition, such as via regulation of transcription.

[§] We define cofactors and prosthetic groups to be compounds that are required for an enzyme to catalyze a reaction, but that are unchanged by the reaction. Thus, cofactors and prosthetic groups are (loosely speaking) activators of an enzyme in the sense that the enzyme is not active when these compounds are absent. However, cofactors and prosthetic groups have an infinite 'activation degree,' thus distinguishing them from those compounds that will be listed in the `Activators` slots (whose definitions were given earlier in this section). When a cofactor is absent, an enzyme does not function; when an activator is missing, the enzyme still functions, but at a lower rate.

The representation of a molecular interaction of an entity depends on the role of that entity in the molecular interaction. We posit that all local biological functions can be expressed as one of the following roles: reaction substrate, reaction catalyst (enzyme), reaction modulator, or enzyme cofactor.

These roles are encoded as follows. Every reaction is represented as a distinct object in the EcoCyc DB. The reaction object lists the substrates (reactants and products) of the reaction. For each reaction, EcoCyc describes what enzyme(s) catalyze that reaction by creating an intermediary object called an *enzymatic reaction* that serves to link a reaction to an enzyme that catalyzes it (Karp and Riley, 1993). The enzymatic-reaction object lists the cofactors, activators, and inhibitors for that enzyme with respect to that activity of the enzyme.

The idea of encoding molecular species, reactions, and enzymatic reactions as separate objects has proven to be an important insight: decoupling the representation of a biological entity from the representation of the function of that entity yields significant leverage and clarity. There is a many-to-many mapping between entities and functions: one entity can be multifunctional, and one function can be carried out by multiple entities. By decoupling an entity from its function(s), we have a more normalized and therefore less redundant representation. By using a separate object to encode a function, we can tease apart the different elements of function into different attributes of the object, allowing us to describe function with high fidelity. When encoding the function of an enzyme, we encode attributes specific to the protein in the protein object (such as its pI, molecular weight, and subunit structure). We encode attributes specific to the reaction in the reaction object (attributes such as the K_{eq} and the substrates are independent of what enzyme catalyzes the reaction). The enzymatic-reaction object encodes attributes that are specific to the pairing of an enzyme with a reaction, such as the cofactors, activators, and inhibitors that modulate a particular activity of a (potentially multifunctional) enzyme.

The remainder of this section describes the overall EcoCyc class hierarchy and the slots of a set of key EcoCyc classes.

The EcoCyc class hierarchy

The structure of the EcoCyc class hierarchy is important for several reasons. First, one can think of the class hierarchy as providing definitions of biological terms. They are also definitions in the sense that the slots defined for each class, and the constraints defined for each slot, define what can be said about the instances of those classes. For example, if the class `Polypeptides` did not have a slot called `Gene`, then for the purposes of the biological reality modeled by this DB, the definition of a polypeptide would not include the notion that we can specify the gene that encodes the polypeptide.

Second, the correctness and the consistency of those definitions have a direct influence on the accuracy and precision with which we can represent biological information because the slots defined on each class define what information can be associated with instances of the class, and the classes are used to specify constraints on slot values. For example, constraints on the `Products` slot for the class `Reactions` ensure that values of this slot are instances of the class `Chemicals`. Because `Chemicals` include subclasses as diverse as `Ions`, `Promoters`, and `Proteins`, we can create a definition of reactions that has a diverse (and biologically realistic) set of products.

Third, the class hierarchy is important because it influences the ease with which users can query the EcoCyc data, and the correctness of those queries, because user queries often refer to the class hierarchy. For example, a query to find all reactions involving amino-acid substrates would query all instances of the class `Reactions`, and would test which of its instances were members of the class `Amino-Acids`. A class whose definition is not clear to the user will lead to erroneous query results.

Figure 1 shows only the tip of the iceberg in the sense that some of the classes in this figure have many subclasses (sometimes hundreds) that are not shown. For example, under the class `Reactions` is the entire EC classification system for enzyme-catalyzed reactions—about 250 classes (Webb, 1992). Under the class `Genes` is the gene-classification system developed by Riley (Riley, 1993).

The `Chemicals` hierarchy in Figure 1 is concerned with representing the molecular building blocks of the cell. `Chemicals` is partitioned into small molecules and elements, whose slots allow us to encode the exact chemical bonding structure of the molecule, and `Macromolecules`, for which the exact bonding structure is either not known, or is not relevant to the EcoCyc project.

The class `Small-Molecules` is divided into a number of different compound types to facilitate retrieval operations for specific types of compounds, and to provide a semantic grouping of compounds. The ‘All–’ classes serve to group a set of chemically related compounds, such as grouping the 20 amino acids found in proteins with other amino acids, and with compounds that are chemically close to amino acids.

The `Macromolecules` class represents both entire macromolecules, such as `Chromosomes, Plasmids,` and `tRNAs,` and regions of a macromolecule, such as `Promoters, Operons,` and `Genes` (a gene in EcoCyc corresponds exactly to a coding region of DNA).

`Reactions` and `Pathways` share a common parent that captures the dual aspects of reactions and pathways: a

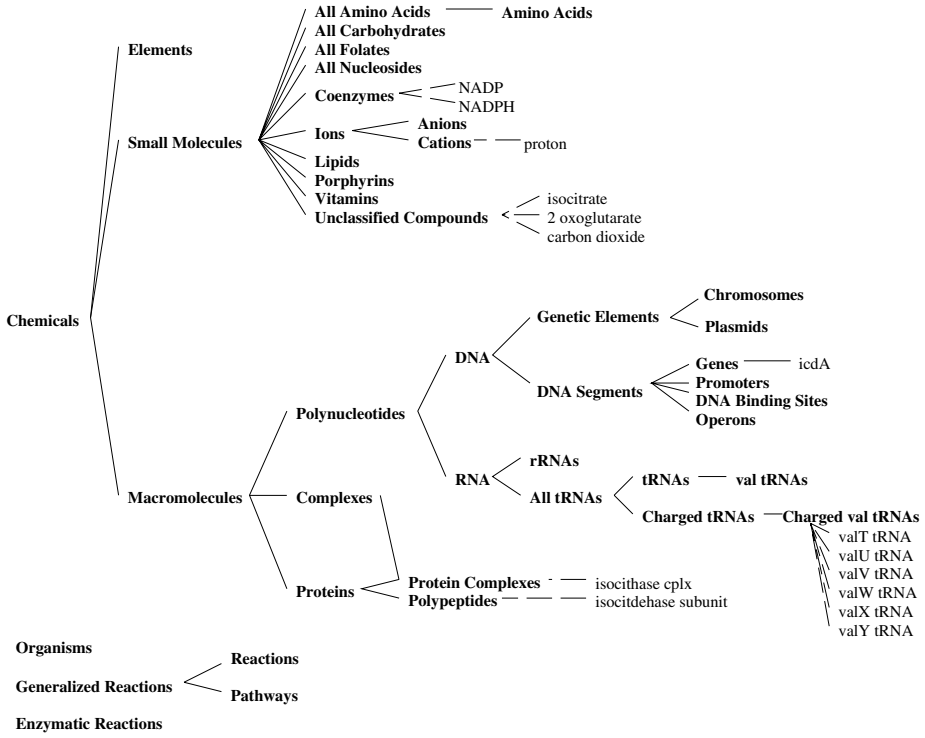

Fig. 1. This figure shows a number of the most important EcoCyc classes (bold), and their relationships within the EcoCyc class hierarchy. Some instances are also shown (roman). Class–subclass links are drawn as solid lines; class–instance links are drawn as dashed lines.

reaction can be viewed as a pathway since a reaction mechanism constitutes a pathway of steps that when composed yield the reaction. Conversely, a pathway can be viewed as having a net transformation and can be treated as a single reaction step when embedded within a larger network of pathways.

Despite its name, the class `Enzymatic-Reactions` is not simply a specialization of class `Reactions` (which is why it is not a subclass of `Reactions`). The `Enzymatic-Reactions` class defines an association between an enzyme and a reaction that the enzyme catalyzes.

Instances of the `Organisms` class encode a strain of a given species (e.g. *E.coli* K12); the `genome` slot lists the genetic elements of that organism (one or more `Chromosomes` and zero or more `Plasmids`).

Definitions of key classes

Table 1 lists slots that are present in all of the EcoCyc classes. The `common-name` slot lists the most commonly used name for an object, and the `synonyms` slot lists additional names for the object. One or more comments about the object are stored in the `comments` slot; a commentary about changes made to the object over time are stored in the `history` slot. References to the literature about this object are stored in the `citations` slot, and links to objects in other DBs are stored in the `dblinks` slot.

The slots used in the `Reactions` class (see Table 2) describe the reactants (slot `left`) and products (slot `right`) of the reaction, as well as the full set of re-

Table 1. Slots inherited by all EcoCyc classes

Slot name	Inverse	Value-type	Max card
`common-name`		string	1
`synonyms`		string	
`comment`		string	
`history`		string	
`citations`		string	
`dblinks`		string	

For each slot we list its name, its inverse if any (i.e. the name of the slot that encodes the inverse relationship), its value type (i.e. the datatype of the values of the slot, such as 'the value must be a string' or 'the value must be the object identifier for an instance of the class Chemicals'), and the maximum cardinality of the slot, i.e. the maximum number of values that the slot may have. For example, the `common-name` slot can take a single value that is a string. The `synonyms` slot can take any number of values, each of which must be a string.

action substrates (the values of the slot substrates is computed by an attached procedure as the union of the left and right slots). The names left and right were chosen to be more neutral with respect to reaction direction, since a given reaction can be favored in different directions under different conditions. For reactions that can occur spontaneously under physiological conditions, the spontaneous? slot is set to True. The ec-number slot holds the EC number for those reactions that have been assigned an EC number by the enzyme commission (Webb, 1992).[¶] The $\Delta G'_0$ and K_{eq} of the reaction are stored in slots deltag0 and keq, respectively. Slot enzymatic-reaction stores links to the enzymatic-reaction objects for the one or more enzymes that catalyze the reaction. Links to pathways containing the reaction are stored in slot in-pathway. The species-distribution slot stores the names of species in which this reaction is known to occur, to represent those cases where a reaction is known to be catalyzed by a given species but the enzyme may not yet have been identified.

The slots used in the Enzymatic-Reactions class (see Table 3) include links to the enzyme and reaction that the enzymatic-reaction associates (slots enzyme and reaction), as well as a description of the typical direction of this reaction when catalyzed by this enzyme (slot reaction-direction). If the enzyme can accept alternative substrates for the substrates listed in the left and right slots of the associated reaction, the alternatives are listed in the alternative-substrates slot (each slot value consists of a list whose first element is a substrate in the original reaction and whose second element is the compound that can substitute for that substrate). Prosthetic groups and cofactors required by the enzyme are listed in the prosthetic-groups and cofactors slots; alternative acceptable cofactors are listed in the alternative-cofactors slot, whose values are analogous to the values of the alternative-substrates slot. A variety of slots list the known activators and inhibitors of the enzyme, broken down by mechanism, when known (for example, the -mechnotstated slots are for modulators whose mechanism of action is not known). The values of slots inhibitors-all and activators-all are computed by attached procedures as the union of all inhibitors and activators, respectively. The slot physiologically-relevant lists the subset of all activators and inhibitors from the other slots whose action is relevant physiologically (as opposed to those compounds whose effects have been studied *in vitro*).

Slots of the Proteins class describe physical properties of the protein such as its molecular weight as derived from sequence or from experimentation (molecular-weight-seq and molecular-weight-exp, respectively). The one or more known cellular locations of the protein are described in the locations slot. The slot modified-form links a protein to chemically modified forms of that protein; slot unmodified-form links a modified protein back to the unmodified form. The slot components links a protein complex to its subunits, which may be either monomers or smaller complexes; slot component-of links a polypeptide or a protein complex to a larger complex that contains it. Slot catalyzes links a protein to enzymatic-reaction frames that describe one or more catalytic activities of the protein. Slots appears-in-left-side-of and appears-in-right-side-of link a protein to reactions in which it is a reactant or a product, respectively.

[¶] Note that the EC number is properly an attribute of reactions, not enzymes. Two enzymes that catalyze the same reaction receive the same EC number. A multifunctional enzyme that catalyzes two reactions receives two EC numbers.

Example representations of function

This section illustrates the surprisingly diverse array of biological functions that can be encoded by the EcoCyc ontology. Although some readers may consider it awkward to use the single underlying concept of a reaction to model functions ranging from enzyme catalysis to gene regulation, we argue that this unity is an advantage. Reactions are the underlying physical mechanisms of all these events. Also the use of a single underlying concept means the same algorithms can easily operate across many biological functions.

Enzyme function

The EcoCyc representations of the function of the enzyme isocitrate dehydrogenase capture the reaction catalyzed by the enzyme; the activators, inhibitors, and cofactors of the enzyme; and the phosphorylation of the enzyme to an inactive form.

The reaction catalyzed by isocitrate dehydrogenase is

$$\text{isocitrate} + \text{NADP} = \text{2-oxoglutarate} + \text{NADPH} + \text{H}^+ + \text{CO}_2$$

The first principle of EcoCyc representations is to encode each molecular species of the reaction as a separate object. Therefore, the EcoCyc DB contains distinct objects for isocitrate, NADP, 2-oxoglutarate, NADPH, H^+, CO_2, for the isocitrate dehydrogenase complex (a homodimer), and for the monomer subunit of the enzyme complex. Yet another object represents the *icdA* gene that encodes this monomer. The *E.coli* chromosome is represented by another object. Figure 1 shows each of these objects linked to the EcoCyc class of which it is an instance.

The reaction the enzyme catalyzes is represented by another object, shown in Figure 2. The slots of this reaction frame encode the reactants and products of the

Table 2. Slots of the class `Reactions`

Slot name	Inverse	Value-type	Max card
`left`		`Chemicals`	
`right`		`Chemicals`	
`substrates`		`Chemicals`	
`spontaneous?`		boolean	1
`ec-number`		string	1
`deltag0`		number	1
`keq`		number	1
`enzymatic-reaction`	`reaction`	`Enzymatic-Reactions`	
`in-pathway`	`reaction-list`	`Pathways`	
`species-distribution`		string	

Note that the value-types other than the built-in types (such as boolean, string, and number) all refer to classes in the EcoCyc ontology, such as those in Figure 1. Two slots that are inverses encode links of opposite directionality within the DB, e.g. slot `enzymatic-reaction` encodes a link from a reaction R to an enzymatic reaction E, and slot `reaction` encodes a link from E to R.

Table 3. Slots in the class `Enzymatic-Reactions`

Slot name	Inverse	Value-type	Max card
`reaction`	`enzymatic-reaction`	`Reactions`	1
`enzyme`	`catalyzes`	`Proteins`	1
`reaction-direction`		*One-of:* **reversible, irreversible physiologically-unidirectional**	1
`alternative-substrates`		(substrate alternate)	
`prosthetic-groups`		`Chemicals`	
`cofactors`		`Chemicals`	
`alternative-cofactors`		(cofactor alternate)	
`inhibitors-all`		`Chemicals`	
`inhibitors-allosteric`		`Chemicals`	
`inhibitors-competitive`		`Chemicals`	
`inhibitors-neither`		`Chemicals`	
`inhibitors-mechnotstated`		`Chemicals`	
`activators-all`		`Chemicals`	
`activators-nonallosteric`		`Chemicals`	
`activators-allosteric`		`Chemicals`	
`activators-mechnotstated`		`Chemicals`	
`physiologically-relevant`		`Chemicals`	

reaction in the slots `left` and `right`, respectively. The slot `in-pathway` contains the identifiers of the frames that represent the pathway(s) that contain this reaction. The `enzymatic-reaction` slot contains the identifier of the enzymatic-reaction frame that links this reaction to the enzyme. The $\Delta G'_0$ and EC number of the reaction are encoded in two additional slots.

These frames are linked together into a network as shown in Figure 3. The links between objects are encoded by slots on the objects. For example, a slot called `components` on the *E.coli* chromosome object lists every gene within the chromosome, and serves to link the chromosome to each of those genes. A gene in turn is linked to its product by the slot `product`. A polypeptide is linked to the protein complex of which it is a part by a slot called `component-of`. An enzyme (in this case an enzyme complex) is linked to the reaction(s) that it catalyzes via an intermediary enzymatic-reaction object.

The enzymatic-reaction frame for isocitrate dehydrogenase. Figure 4 shows the enzymatic-reaction frame that links the isocitrate dehydrogenase enzyme frame to the reaction frame `isocitdeh-rxn`. These two frames together define the catalytic activity of the enzyme, and describe the requirements and sensitivities of that activity by listing the cofactors and inhibitors of the enzyme. The inhibitors are of known and unknown mechanisms; no enzyme activators are known. The enzymatic reaction

Table 4. Slots of the class `Proteins`

Slot name	Inverse	Value-type	Max card
`molecular-weight-seq`		number	1
`molecular-weight-exp`		number	1
`pI`		number	1
`locations`		*One-of:* **cytoplasm, periplasm, membrane, inner-membrane, outer-membrane**	
`modified-form`	`unmodified-form`	`Proteins`	
`unmodified-form`	`modified-form`	Proteins	1
`component-of`	`components`	`Chemicals`	
`components`	`components-of`	`Chemicals`	
`catalyzes`	`enzyme`	`Enzymatic-Reactions`	
`appears-in-left-side-of`		`Reactions`	
`appears-in-right-side-of`		`Reactions`	

```
--- Instance ISOCITDEH-RXN ---
    Types:  EC-1.1.1

          EC-NUMBER : "1.1.1.42"
               LEFT : isocitrate, nadp
              RIGHT : nadph, proton, 2-oxoglutarate, carbon-dioxide
  ENZYMATIC-REACTION : isocitdeh-enzrxn
            DELTAGO : -5.0
         IN-PATHWAY : tca, anaresp1-pwy
```

Fig. 2. The EcoCyc frame whose internal identifier is `isocitdeh-rxn` represents the reaction catalyzed by isocitrate dehydrogenase. The identifier of the parent class of this instance frame is EC-1.1.1, which is the enzyme commission category to which this reaction is assigned. The remainder of the figure lists the slots of `isocitdeh-rxn` and their values, for example the `left` slot has the two values `isocitrate` and `nadp`, which are the identifiers of the two frames representing the compounds isocitrate and NADP, respectively.

frame describes the catalytic function of isocitrate dehydrogenase, the function of Mn^{+2} as a cofactor, and the function of glyoxylate as an inhibitor.

Inactivation of isocitrate dehydrogenase. One local function of isocitrate dehydrogenase is to act as a catalyst, however, isocitrate dehydrogenase undergoes an additional molecular interaction: it can be phosphorylated by isocitrate-dehydrogenase kinase and converted to an inactive form. Isocitrate-dehydrogenase kinase is an enzyme, one of whose substrates is isocitrate dehydrogenase. The reaction catalyzed by the enzyme is encoded using the reaction frame shown in Figure 5. The reaction is linked to a set of objects representing isocitrate-dehydrogenase kinase and its encoding gene, *icdA,* as shown in Figure 6. In keeping with our practice of creating a distinct frame for every molecular species, an additional frame is created to represent isocitrate dehydrogenase phosphate. The fact that the unphosphorylated form of isocitrate dehydrogenase is the active form is conveyed by the fact that it is the isocitrate dehydrogenase frame, rather than the isocitrate dehydrogenase phosphate frame, that is the enzyme (that is linked to the enzymatic-reaction frame) in Figure 3. The isocitrate dehydrogenase phosphate frame is not linked to any enzymatic reaction because it has no catalytic function.

Enzyme function: tRNA charging

The tRNA-charging reactions of the cell are enzyme-catalyzed reactions like any other, therefore the rules used to encode tRNA-charging reactions in EcoCyc are the same as for other types of reactions. A tRNA and its charged analog are distinct molecular species, therefore EcoCyc frames are created for each. The tRNA synthetase enzyme is represented as one or more frames (depending on its subunit structure), and is linked via an enzymatic-reaction frame to a frame that represents the tRNA-charging reaction. Such a reaction frame for charging of valyl tRNAs is shown in Figure 7.

This reaction differs from the reaction of isocitrate

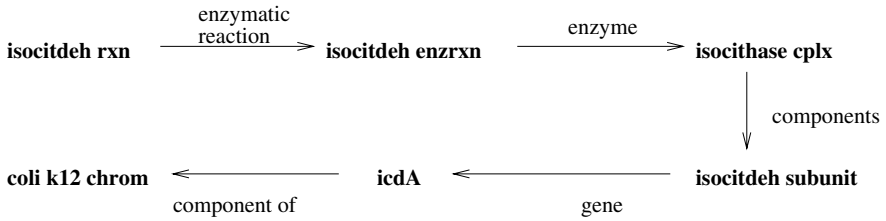

Fig. 3. A semantic-network depiction of the relationships among the frames that encode isocitrate dehydrogenase, its function, and its gene. Each arrow represents a single relationship encoded by the slot named next to the arrow. All relationships shown here are bidirectional, meaning that each is encoded by two slots that encode inverse relationships. For example, frame `isocitdeh-enzrxn` has a slot called `enzyme` that has the value `isocithase-cplx`, and frame `isocithase-cplx` has a slot called `catalyzes` that has the value `isocitdeh-enzrxn`.

```
--- Instance ISOCITDEH-ENZRXN ---
    Types:  |Enzymatic-Reactions|

            COMMON-NAME : "isocitrate dehydrogenase"
               SYNONYMS : "oxalosuccinate decarboxylase"
                 ENZYME : isocithase-cplx
               REACTION : isocitdeh-rxn
              CITATIONS : "[84236175]"
              COFACTORS : mn+2
  ALTERNATIVE-COFACTORS : (mn+2 mg+2)
                COMMENT : "Isocitrate dehydrogenase is the first bacterial enzyme
shown to be regulated by phosphorylation/dephosphorylation. The
modulation of this key enzyme activity enables E. coli
to make rapid shifts between TCA and glyoxalate bypass pathways.
Fluxes and intracellular concentrations for this junction have been
determined. The state of phosphorylation of isocitrate dehydrogenase determines
its activity.  [85234559], [89374109],[85054851]"
             INHIBITORS : phenylglyoxal
 INHIBITORS-COMPETITIVE : oxalomalate, oxalacetic_acid, glyox
```

Fig. 4. An enzymatic-reaction frame. The bracketed numbers are citations to Medline entries.

dehydrogenase in that the enzyme valyl-tRNA synthetase can charge a family of six different valyl-tRNAs in *E.coli*. One way to represent this situation would be to create six different reaction frames that describe the charging of each different tRNA species. A more succinct approach allows us to represent the entire family of reactions using a single reaction frame. The substrates `val-tRNAs` and `charged-val-tRNAs` listed in Figure 7 are both class frames whose instances are the six uncharged tRNAs, and the six charged tRNAs, respectively. This reaction is interpreted as describing a family of transformations whose substrates include any instances of the chemical-compound classes linked in the `left` and `right` slots.

Transport function

We represent the function of transport proteins by viewing these proteins as enzymes, where the reactions catalyzed by these proteins encode transport events. We generalize the representation of reactions to allow any of the reaction substrates to be annotated with the name of the compartmental pools from which those substrates are drawn. When no compartment is specified, we treat the default as the cytoplasmic pool.

```
--- Instance PHOSICITDEHASE-RXN ---
    Types:  EC-2.7.1

          EC-NUMBER : "2.7.1.116"
               LEFT : atp, isocithase-cplx
              RIGHT : adp, isocithase-p
  ENZYMATIC-REACTION : icitdehkinase-enzrxn
```

Fig. 5. A reaction frame representing the phosphorylation of isocitrate dehydrogenase (`isocithase-cplx`) to form isocitrate dehydrogenase phosphate (`isocithase-p`).

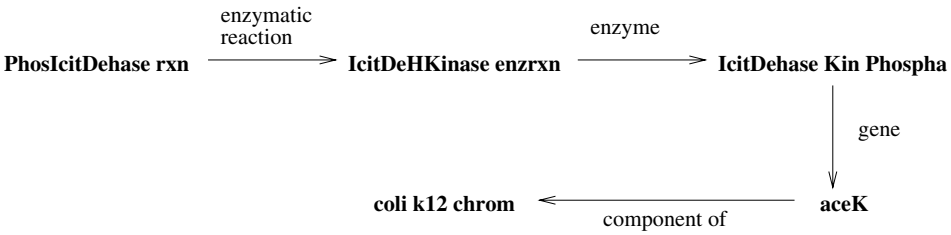

Fig. 6. A semantic-network depiction of the network of frames that represent isocitrate dehydrogenase phosphate and its function.

```
--- Instance VALINE--TRNA-LIGASE-RXN ---
    Types:  EC-6.1.1, tRNA-Reactions

          EC-NUMBER : "6.1.1.9"
               LEFT : val-tRNAs, val, atp
              RIGHT : charged-val-tRNAs, ppi, amp
  ENZYMATIC-REACTION : vals-enzrxn
```

Fig. 7. Reaction encoding a family of tRNA charging reactions for valyl tRNAs.

```
--- Instance TRANS-RXN-126A ---
    Types:  TR-12

                LEFT :
                  na+
                  ---COMPARTMENT: periplasm
                  val
                  ---COMPARTMENT: periplasm
               RIGHT : na+, val
  ENZYMATIC-REACTION : trans-enzrxn-126a
```

Fig. 8. Reaction frame encoding symport transport of valine. The values `val` and `na+` in the `left` slot are annotated with the compartment `periplasm`, indicating that valine and sodium both start in the periplasm, and move to the cytoplasm (the default compartment assumed for the products listed in the `right` slot).

This representation generalizes to eukaryotic cells with their large number of compartments (mitochondria, nucleus, etc). It handles the whole spectrum of transport processes, including diffusion-mediated channels, ATP-driven transport, symport, and antiport. For example, the function of a valine symport transporter that transports valine from the periplasm to the cytoplasm, driven by the simultaneous diffusion of sodium ions from the periplasm to the cytoplasm, is shown in Figure 8. The frame `trans-enzrxn-126a` is an enzymatic-reaction frame that links the transport reaction to a frame that represents the transport protein.

Signal-transduction functions

Signal-transduction functions of the two-component regulation proteins of *E.coli* are also represented as reactions in EcoCyc. For example, Figure 9 shows a reaction in which the phosphorylated NRII protein transfers a phosphate group to the NRI protein. All four species in this reaction are represented as distinct frames in EcoCyc. However, no enzymatic-reaction frame is linked to `nriphos-rxn` reaction frame because conceptually it is not clear which, if any, of these two proteins to label as the enzyme—is NRII–P the enzyme, or is the NRI the enzyme? We concluded that since there is no clear answer to this question, it is most appropriate to not choose either protein as the enzyme, but to treat them symmetrically as substrates in a reaction that occurs spontaneously. Similarly, in the reaction in which NRI–P autocatalytically dephosphorylates, we simply treat NRI–P and NRI as substrates of the reaction without assigning either species as the enzyme.

Genetic-regulation functions

Bacterial gene expression is controlled by a variety of mechanisms, such as by DNA-binding proteins that activate and inhibit the initiation of transcription by RNA polymerase. The *E.coli* tryptophan operon, for example, is controlled by the TrpR repressor protein. When bound to tryptophan, this protein can bind to the operator region of the tryptophan operon, thus inhibiting the initiation of transcription. High concentrations of tryptophan thereby inhibit expression of genes that encode the pathway for tryptophan biosynthesis (Yanofsky, 1981).

EcoCyc represents the functions of TrpR with the following three reactions. The notation `A.B` means ‘a complex formed by `A` and `B`’.

1. `apo-TrpR + tryptophan`
   `= tryptophan.TrpR`
2. `tryptophan.TrpR + trp-operator`
   `= tryptophan.TrpR.trp-operator`
3. `RNA-polymerase + trp-promoter`
   `= RNA-polymerase.trp-promoter`

```
--- Instance NRIPHOS-RXN ---
    Types:  2Comp-Reactions

          LEFT : protein-nriip, protein-nri
         RIGHT : protein-nrii, protein-nrip
     CITATIONS : "[95214091]", "[93128858]", "[90206041]", "[93272330]", "[95158695
       COMMENT : "In this reaction the nitrogen regulator protein I (NRI) becomes
                  phosphorylated through interaction with the kinase-n
                  phosphotransferase nitrogen regulator protein II-his-P (NRII-P).
                  Nitrogen regulator protein I (NRI) is phosphorylated through
                  interaction with the phosphorylated form of NRII.  The phosphoryl
                  group is transferred from a histidine residue on NRII to an
                  aspartyl residue on NRI.  The phosphorylated NRI then activates
                  glnA transcription. [95050613] [92279271]"
    IN-PATHWAY : nri-pwy
```

Fig. 9. Signal-transduction reaction involving transfer of a phosphate group from the NRII protein to the NRI protein.

Following our principle of representing each distinct chemical species as a separate object, we use different frames to represent the apo-repressor and the complex of tryptophan with TrpR. Reaction 1 describes binding of tryptophan to the aporepressor. Reaction 2 describes binding of that complex to the trp-operator region of DNA. Reaction 3 describes initiation of transcription at the trp-promoter region of DNA. We also specify that the product of Reaction 2 can inhibit Reaction 3, to express the fact that binding of the repressor protein at the operator region influences the rate of transcription initiation. None of these three reactions are linked to an enzymatic reaction because for none of these reactions is there an enzyme catalyst. Therefore, we encode the inhibition of Reaction 3 using the slot `inhibitor` in Reaction 3. This representation does not address the attenuation mechanism; see Karp (1993) for a model of attenuation.

This approach to representing gene regulation may seem overly complex, but the complexity is needed to faithfully capture complex regulatory mechanisms. Two alternative representations and their limitations are as follows. Approach: For each regulatory protein, simply list the genes it regulates. Limitation: Does not distinguish induced genes from repressed genes. Approach: For each regulatory protein, list induced and repressed genes separately. Limitations: Does not indicate the relevant promoter and operator regions, and does not describe what ligands activate or inhibit the regulatory protein.

Discussion

In the EcoCyc ontology, we can compute with function by computing with reaction frames and enzymatic-reaction frames. Recall the predicate `functions-equal(F1,F2)`, which compares the local functions of two macromolecules. The arguments to this function are a pair of reaction frames that describe the functions of the macromolecules. The definition of the predicate is extremely simple: it returns true if and only if the reaction frames are identical. Section **Function-based database queries** illustrated how the ontology can be used to express a range of DB queries.

Related research falls into two categories. First are the ontologies developed by other metabolic-DB projects, such as the MPW/WIT/PUMA/EMP projects and the KEGG project. The WIT project (Selkov *et al.*, 1997) has not published its ontology.

The KEGG authors describe their ontology as being organized around the concept of a binary relation (Goto *et al.*, 1997) although in fact, some of the examples in Goto *et al.* (1997, p.178) employ trinary relations—relations involving three-element tuples. The KEGG ontology defines a handful of relationships between DB objects, such as the relationship between reactions, substrates, and products; the relationship between an enzyme and its location in a metabolic pathway; the relationship between an enzyme and a protein super family to which it belongs. The KEGG ontology does not include the notion of an enzymatic-reaction, nor does it include the wider range of attributes and relationships that allow the EcoCyc ontology to model subtle aspects of biological function. For example, the KEGG ontology does not capture the concepts of enzyme activators, inhibitors, or cofactors.

The second category of related work is that on developing functional role categories by Riley (1993),[‖] by Ashburner and colleagues as part of the Gene Ontology (GO) project,[**] and by the MIPS project.[††] Riley and the MIPS project are addressing the problem of encoding integrated

[‖] See URL http://ecocyc.PangeaSystems.com:1555/ECOLI/class-subs?object=Genes.

[**] See URL http://www.ebi.ac.uk/~ashburn/GO/.

[††] See URL http://www.mips.biochem.mpg.de/proj/yeast/catalogues/funcat/index.html.

function, not local function, because their role categories are organized according to an hierarchical decomposition of cellular processes. The GO project is addressing integrated function, and is developing a controlled vocabulary for local function (see Section **Controlled vocabularies will not suffice** for a discussion of controlled vocabularies). These efforts have led to extremely useful systems for categorizing and retrieving lists of genes and gene products, and for evaluating what fraction of the cellular machinery of an organism is devoted to different types of integrated function (Ouzounis *et al.*, 1996).

Descriptions of integrated function and of local function are complimentary since each contains information that the other lacks. The functional role categories typically provide coarser descriptions of function than do descriptions of local function. On the other hand, they provide higher-level conceptualizations of biological systems than do descriptions of local function. It is worth noting that some integrated functions can be computed automatically from local functions. For example, any protein whose local function involves a reaction whose reactants and products occupy different compartments can automatically be classified as being part of the transport machinery of the cell.

The coarseness of descriptions of integrated function implies that they may ultimately be less versatile than local functions. Although descriptions of integrated function tell us that a given gene product is involved in arginine biosynthesis, they do not tell us *how* the gene product is involved. Their representations do not distinguish among the 10 gene products that are involved in arginine biosynthesis, whose local functions are very different. The local functions range from a variety of enzymes to regulatory proteins. Current role categories do not even distinguish the polarity of a regulatory protein because they do not distinguish activator from repressor proteins. Because role categories describe function with so little precision, they could not be used to answer the types of queries listed in Sections **Function-based database queries** and **Additional functional computations**. Role categories will also not lead to simulations of biological processes, as representations of local function will.

Conclusions

The biological function of a molecule can be described in terms of the role of that macromolecule in one or more cellular processes, which we call its integrated function.

The local biological function of a molecule is defined by its biochemical interactions with other molecules. To describe the local function of a molecule, we describe the molecule as a frame that is distinct from frames that represent other chemically modified forms of the molecule. We then model each of its interactions in turn by considering whether the role of the molecule in a given interaction is that of a substrate, catalyst, modulator, or cofactor.

When the role is that of a substrate, we encode the molecule as a reactant or product in a reaction frame. When the role is that of a catalyst, we link the frame describing the molecule to an enzymatic-reaction frame that describes its catalytic activity through a second linkage to a reaction frame. When the role is that of a modulator or cofactor, we encode the molecule as an activator, inhibitor, or cofactor of the appropriate mechanism using a slot of an enzymatic-reaction frame. That frame is linked to a catalyst whose activity the molecule modulates.

The EcoCyc ontology of biological function consists of an hierarchy of classes such as `Reactions`, `Polypeptides`, `Genes`, and `Enzymatic-Reactions`, plus a set of slot definitions for those classes that define their attributes and relationships.

This functional ontology can encode a surprisingly diverse array of biochemical processes, including enzymatic reactions involving small-molecule substrates and macromolecular substrates, signal-transduction processes, transport events, and mechanisms of regulation of gene expression. The ontology has been validated and refined through its use in the EcoCyc project to encode the functions of hundreds of *E.coli* proteins as described in the biochemical literature. A second form of validation is the ability of the ontology to express the set of functional queries in Section **Function-based database queries**, which cannot be reliably answered by existing sequence DBs.

Acknowledgements

Monica Riley has contributed greatly to the ideas in this paper in the course of our long collaboration on EcoCyc. Julio Collado contributed to the design of the ontology for regulatory events. Milton Saier and Ian Paulsen contributed to the design of the ontology for transport. Fidel Salas and Christos Ouzounis made valuable comments on this manuscript. This work was supported by grant 1-R01-RR07861-01 from the National Center for Research Resources.

References

Cohen,D. and Bergman,R. (1994) Prediction of positional isotopomers of the citric acid cycle: the syntactic approach. *Am. J. Physiol.*, **266**, 341–350.

Gaasterland,T. and Sensen,C. (1996) Fully automated genome analysis that reflects user needs and preferences. A detailed introduction to the MAGPIE system architecture. *Biochimie*, **78**, 302–310.

Goto,S., Bono,H., Ogata,H., Fujibuchi,W., Nishioka,T., Sato,K. and Kanehisa,M. (1997) Organizing and computing metabolic pathway data in terms of binary relations. In *Pacific Symposium on Biocomputing '97,* pp. 175–186.

Gruber,T. (1993) A translation approach to portable ontology specifications. *Knowl. Acquis.*, **5**, 199–220. URL for

Ontolingua is http://www-ksl.stanford.edu/knowledge-sharing/ontolingua/README.html.

Gruber,T. (1995) Toward principles for the design of ontologies for knowledge sharing. *Int. J. Human-comput. Studies*, **43**, 907–928. See WWW URL http://ksl-web.stanford.edu/KSL_Abstracts/KSL-93-04.html.

Guarino,N. (1996) Understanding, building, and using ontologies. In *Proceedings of Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop.* See WWW URL http://ksi.cpsc.ucalgary.ca/KAW/KAW96/KAW96Proc.html.

Guarino,N. and Giaretta,P. (1995) Ontologies and knowledge bases: towards a terminological clarification. *Towards Very Large Knowledge Bases.* IOS Press, Amsterdam, pp. 25–32.

Karp,P. (1992) The design space of frame knowledge representation systems. *Technical Report 520.* SRI International AI Center, URL ftp://www.ai.sri.com/pub/papers/karp-freview.ps.Z.

Karp,P. (1999) Integrated pathway/genome databases and their role in drug discovery. *Trends Biotech.*, **17**, 275–281.

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

Karp,P. and Paley,S. (1996) Integrated access to metabolic and genomic data. *J. Comput. Biol.*, **3**, 191–212.

Karp,P. and Riley,M. (1993) Representations of metabolic knowledge. In Hunter,L., Searls,D. and Shavlik,J. (eds), *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology.* AAAI Press, Menlo Park, CA, pp. 207–215.

Karp,P., Riley,M., Paley,S., Pellegrini-Toole,A. and Krummenacker,M. (1999) EcoCyc: Electronic encyclopedia of *E.coli* genes and metabolism. *Nucl. Acids Res.*, **27**, 55–58.

Karp,P.D. (1993) A qualitative biochemistry and its application to the regulation of the tryptophan operon. In Hunter,L. (ed.), *Artificial Intelligence and Molecular Biology.* AAAI Press, Menlo Park, CA.

Karp,P.D., Chaudhri,V.K. and Paley,S.M. A collaborative environment for authoring large knowledge bases. *J. Intell. Inform. Syst.*, in press.

Mavrovouniotis,M. (1989) Computer-aided design of biochemical pathways, PhD thesis, Massachusetts Institute of Technology.

Ouzounis,C., Casari,G., Sander,C., Tamames,J. and Valencia,A. (1996) Computational comparisons of model genomes. *Trends Biotechnol.*, **14**, 280–285.

Riley,M. (1993) Functions of the gene products of *Escherichia coli*. *Microbiol. Rev.*, **57**, 862–952.

Scharf,M., Schneider,R., Casari,G., Bork,P., Valencia,A., Ouzounis,C. and Sander,C. (1994) GeneQuiz: a workbench for sequence analysis. In Altman,R., Brutlag,D., Karp,P., Lathrop,R. and Searls,D. (eds), *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology.* AAAI Press, Menlo Park, CA, pp. 348–353.

Selkov,E., Galimova,M., Goryanin,I., Gretchkin,Y., Ivanova,N., Komarov,Y., Maltsev,N., Mikhailova,N., Nenashev,V., Overbeek,R., Panyushkina,E., Pronevitch,L. and Jr,E.S. (1997) The metabolic pathway collection: an update. *Nucl. Acids Res.*, **25**, 37–38.

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, London.

Yanofsky,C. (1981) Attenuation in the control of expression of bacterial operons. *Nature*, **289**, 751–758.