Proc. Int. Conf. Intell. Syst.
Mol. Biol., 1995, 3: 127-135

Reconstruction of Metabolic Networks Using Incomplete Information

Terry Gaasterland*
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60432
gaasterland@mcs.anl.gov

Evgeni Selkov†
Institute for Theoretical and Experimental Biophysics
The Russian Academy of Science
142292 Pushchino, Russia
evgeni@mcs.anl.gov

Abstract

This paper describes an approach that uses methods for automated sequence analysis (Gaasterland *et al.* August 1994) and multiple databases accessed through an object+attribute view of the data (Baehr *et al.* 1992), together with metabolic pathways, reaction equations, and compounds parsed into a logical representation from the Enzyme and Metabolic Pathway Database (Selkov, Yunus, & *et.al.* 1994), as the sources of data for automatically reconstructing a weighted partial metabolic network for a prokaryotic organism. Additional information can be provided interactively by the expert user to guide reconstruction.

Introduction

As available genome sequence data for microbial organisms increases both in the amount of data for individual organisms and in the number of organisms with data, we ask: how much of an organism's metabolic structure can be pieced together using sequence evidence, knowledge about metabolism, and encoded metabolic pathways? How much of this process can be automated? How can the resulting tool be used to help people investigate an organism? This paper describes a prototyped methodology for automatically reconstructing partial metabolic networks. Our goal is to describe the modules of knowledge necessary for this endeavor, to present novel methods to use them, and to describe a prototype system written in Prolog with examples from Mycoplasma capricolum. The methodology is grounded in theory of logic programming, annotated logic programming, reasoning with incomplete information, and handling user confidences. The work has unfolded in close concert with the manual reconstruction of the metabolism of Mycoplasma capricolum by domain experts with the intention of building a tool that will work rapidly and reliably for procaryotic organisms, in particular for archaea (Doolittle 1992).

One critical component for carrying out reconstruction of metabolism from genome sequence data is a system that produces a ranked list of potential proteins that appear in the sequence data. The AutoSeq system at Argonne (Gaasterland & Lobo September 1994; Gaasterland *et al.* August 1994) provides this facility. It takes assembled sequence fragments as input and produces an interpretation of proteins with genomic evidence at varying levels of confidence. The AutoSeq system is written in Prolog and produces an analysis as Prolog facts as well as in a WWW browseable form. So the interpretation is accessible for answering queries about putative proteins in an analyzed organism.

Another necessary component is encoded knowledge about metabolic pathways and reaction equations. The EMP database (Selkov, Yunus, & *et.al.* 1994) provides this information. Over 1000 pathways from EMP have been parsed into Prolog facts as lists of connected reaction equations with direction, reversibility, and other supporting information.

Also needed is an environment for answering queries about individual sequences, phylogenetically related organisms, and sets of sequences interrelated by homology. The GenoBase integrated database system (Baehr *et al.* 1992; Yoshida *et al.* 1992) provides this environment. It treats each body of data as a set of objects with attributes. Each object-attribute structure is encoded as one or more Prolog facts. Prolog rules capture operators over the objects. Together the rules and facts form a knowledge base that can be used to answer queries during the reconstruction.

The next section provides background information. Next we describe how reconstruction is carried out using direct evidence and give the basic reconstruction algorithm. Then we give examples of how the basic reconstruction algorithm applies to a selected body of evidence from a procaryotic organism, Mycoplasma capricolum. Finally we show how to use additional information about the environment that an organism requires for its survival. This facility becomes important for organisms that import substances instead of

*This work was supported in part by the DOE Office of Scientific Computing, contract W-31-109-Eng-38. Dr. Gaasterland has a joint appointment with the Department of Computer Science at the University of Chicago.

†Joint with the Mathematics and Computer Science Department, Argonne National Laboratory

— or as well as — manufacturing them.

Background

Procaryotic Organisms With the eventual goal of carrying out sequence analysis for complex organisms, we seek to build and understand analysis tools for the simplest organisms: procaryotes. Procaryotic organisms have fewer complications than other organisms. In their genome sequences, they rarely have introns; a sequence that translates into a protein usually has a starting point marked by a start codon and continues until a stopping point marked by a stop codon is reached. A prokaryote nuclear region has no nuclear membrane and consists of a single DNA molecule that divides non-miotically. It lacks histones and nucleolus. The cell's plasma membrane usually lacks sterols, and internal membranes are limited to specific groups. Ribosomes are 70S in size, smaller than in other organisms. Procaryotic cells have no microtubules and are generally small, on the order of $<2\mu$m.

With their simplicity and small size, procaryotes almost have an enumerable set of metabolic functions. One can estimate the order of the number of genes in the genome from a functional layout of such an organism's necessary components. This knowledge greatly facilitates the process of reconstructing an procaryote's metabolism. Given just a few protein parts involved in a module of the metabolic machine of an organism in question and given a limited but crucial amount of specific knowledge about the pathways that appear in similar organisms, we can infer much abut the module. The examples section gives illustrates this with Mycoplasma capricolum, a procaryotic organism whose genome is 30-40% sequenced.

Prolog and Logic Programming Much knowledge about a domain can be captured in logical expressions. Logic programming provides a means to compute new information based on captured information. Prolog is a programming language that carries computation over logical expressions. Logic programs have two types of logical expressions: *facts* and *rules*. Facts are atomic expressions formed from a predicate with arguments — the arguments contain terms. A term is either a variable, a constant, or a function applied to a term. So, if p is a predicate with two arguments; a and b are constants; f is a function; and X and Y are variables, $p(a,b)$ is a fact, as are $p(f(a),b)$, $p(X,X)$, $p(f(Y),a)$ (and more). A fact may be referred to as an *atom*.

A rule has a body and a head: $Head \leftarrow Body$. It expresses the notion that $Head$ is true **if** $Body$ is true. The head of a rule is an atom. The body of a rule is a (possibly empty) conjunction of atoms. A rule with an empty body is a fact. So, if we add the predicates r and s each with one argument to our collection above, $p(X,X) \leftarrow s(X), q(X)$. (meaning "p(X,X) is true if s(X) is true and q(X) is true") is a rule as are $r(X) \leftarrow s(X), q(f(X))$. and $s(b) \leftarrow$. Rules can have negated atoms in their bodies. So, $p(X,Y) \leftarrow q(Y), NOTs(Y)$. is a rule meaning "$p(X,Y)$ is true if $q(Y)$ is true and $s(Y)$ is not true." We could also say "$p(X,Y)$ holds if $q(Y)$ holds and $s(Y)$ does not hold." Deduction allows us to derive new facts from a set of rules and facts. For example, if we have the facts $s(a)$ and $q(b)$ and the rule $p(X,Y) \leftarrow s(X), q(Y)$., we can derive $p(a,b)$.

Representing Preference and Confidence The theory of annotated logic programming supplemented with user constraints provides a means to consider levels of confidence and preferences while reasoning about knowledge expressed in logic. In a logic program, a fact, say *young(kim)* (meaning "kim is young") is either true or false. In an annotated logic program, a fact may have an annotation, and the annotated fact, say *young(kim):very* (where *very* is the annotation) is considered true or false. The values used in annotations are distinct from the values used in the facts. They also have a partial order, for example, *very* > *somewhat* > *slightly*. An annotated rule uses a combination function to combine the annotations of the atoms in the body into an annotation for the head. Suppose our combination function is to take the maximum of the annotation values. Then, the rule $innocent(X) : max(A,B) \leftarrow young(X) : A, sheltered(X) : B$. allows us to derive $innocent(kim) : very$ from $young(kim) : very$ and $sheltered(kim) : somewhat$. When multiple rules define the same atom, some combination function can be used to find a consensus annotation for a derived fact. For example, suppose a second rule says $innocent(X) : A \leftarrow naive(X) : A$. and we have a fact $naive(kim) : slightly$. With a combination function that takes the minimum of the annotations, we obtain $innocent(kim) : slightly$ in the context of the two rules. For more details on annotated logic programming, see (Gaasterland & Lobo September 1994; Kifer & Subrahmanian 1993).

Automatic Sequence Analysis The AutoSeq automatic sequence analysis system is composed of three separate modules. The first is a data collection module that accepts DNA sequence, sends it out to an array of analysis tools (including blastx, blastn, tblastn, blaize, fasta, genmark, and blocks (Altschul *et al.* 1990; Henikoff & Henikoff 1993; Collins & Coulson 1990)), and parses the output into prolog facts. Each input sequence has a unique identifier. Each fact gleaned about a sequence expresses a property about some range within the sequence. For example, output from tools that find homologies between a query sequence and protein database sequences is encoded into facts of the form: **sim(QID, QBegin, QEnd, DBID, DBegin, DEnd, Score, Tool).** meaning that the query sequence QID between QBegin and QEnd aligned with the database sequence DBID between DBegin and DEnd. Output from tools that locate patterns in a query sequence produce facts of the form: **pattern(QID, QBegin, QEnd, PatternID, Score, Tool).** Meaning the pattern associated with PatternID occurs in the QID sequence between QBegin and QEnd. Output from tools that find other properties such as codon usage is encoded in facts of a similar nature.In summary, the data collection phase produces a body of facts about input sequences where each fact expresses a feature associated with some interval of the input sequence.

The second module of AutoSeq analyzes the facts about features of the input sequence and associates regions that are likely to code for some protein with a set of possible proteins. Whenever possible, an attempt is made to find a generalization that characterizes the actual protein encoded in each coding regions. This data analysis module produces facts of the form: **putative_cds(QID, [B1+E1, B2+E2, ..., B_n+E_n], Protein, Score).** where the second argument listing beginning and ending points captures discontinuities (e.g. frameshifts and sequencing errors). The third module of AutoSeq presents putative_cds regions together with supporting evidence in a WWW browseable form. More information on AutoSeq can be found in (Gaasterland & Lobo September 1994; Gaasterland *et al.* August 1994).

Metabolic Information: the EMP Database

The reconstruction process takes advantage of the Enzyme and Metabolic Pathway database (EMP). EMP includes data on both enzymology and metabolism. The approach described in this paper relies on metabolic information that is encoded in over 1000 records, each giving a different pathway instance in some organism. For each pathway, an EMP metabolic record contains an equational representation, a graphical portrayal, and regulatory information.

Records in the database — both enzymological and metabolic — are structured distillations of the factual content of published research articles. The database attempts to capture all relevant facts from each article (there are over 300 distinct fields used to encode data). Encoded articles have been selected from the leading international journals on biochemistry. A journal article is encoded into one or more EMP records (averaging 2 articles per record) with a current total of over 14,000 records. The enzymological section of EMP includes data on over 70% of enzymes classified by EC numbers. In addition, it includes records on over 600 new, unclassified enzymes (that have not yet been assigned EC numbers). The database now includes data on approximately 3000 distinct enzymes. Enzymes without official EC numbers are assigned temporary "partial" EC numbers.

For the purposes of reconstruction, we need to know about substrates, products, cofactors and enzymes (if any) for each reaction in a pathway. To reason automatically about the pathways through logic (e.g. for the purposes of metabolic reconstruction), they must be represented in logical expressions. Thus, over 1100 pathways have been extracted from EMP records and stored as facts in the following logical form: **pathway(PathID, [Reaction_1, ..., Reaction_n]).** where PathID is a unique identifier for the pathway and each Reaction_i is a term of the form: **[Enzyme, Substrates, Cofactors, Products, Reversible, Direction].**

An enzyme and a pathway are connected when an enzyme appears in one of the reactions in the pathway. The following rule expresses this notion.

```
enzyme_to_pathway(Enzyme,PathID) ←
pathway(PathID, ReactionList),
member(Reaction, ReactionList),
Reaction == [Enzyme | _].
```

One may consider a metabolic pathway to be a judgement of proximity of reaction equations. At its simplest level, a pathway is a set of connected reactions equations that form a group. For the purposes of reconstruction, each pathway is considered to be a distinct set of connected reaction equations. By this definition, even if two sets of reactions differ by only one reaction, they comprise two distinct pathways. One may refer to a pathway at an abstract level according to its general name, for example, "glycolysis" and at a specific level according to the set of connected reaction equations and the organism(s) associated with the pathway. Facts containing common names of specific pathways have also been parsed from EMP into the form **pathway_name(PathID,CommonName).**

We also extracted connections between instances of pathways and the organism in which they were determined from the EMP records. The connections are stored as facts of the form: **pathway_to_organism(PathID,Organism).** Knowledge about which organism goes with a pathway instance becomes important when we want a suitable pathway in a nearest neighbor to the organism in question.

Phylogenetic Information Phylogenetic relationships between organisms can be computed from multiple sequence alignments of molecules common to each. The ribosomal RNA Database Project (RDP) has built multiple sequence alignments of the small subunit ribosomal RNA (SSU rRNA) for over 3200 taxa. A single tree was computed (Olsen, Woese, & Overbeek 1994) for these taxa by constructing thousands of smaller trees using a maximum likelihood method (Olsen *et al.* 1994; Felsenstein 1981) and then carrying out an optimized assembly. Figure 1 shows an excerpt from the SSU rRNA phylogenetic tree centering on Mycoplasma.

The relative positions of taxa in the phylogenetic tree reflect changes over time in the sequences of the molecules used in the alignment. Each organism appears in a leaf node of the tree. Each pair of organisms

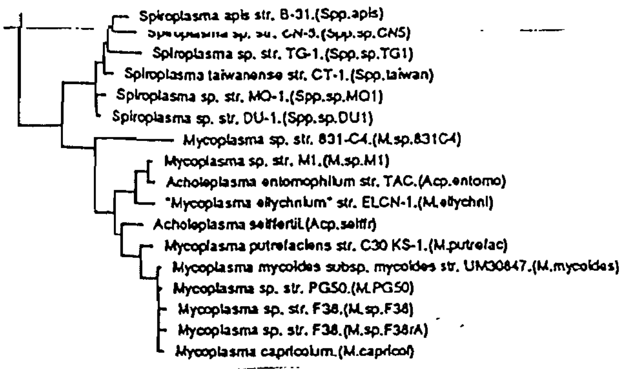
Spiroplasma apis str. B-31.(Spp.apis)
Spiroplasma sp. str. CN-5.(Spp.sp.CN5)
Spiroplasma sp. str. TG-1.(Spp.sp.TG1)
Spiroplasma taiwanense str. CT-1.(Spp.taiwan)
Spiroplasma sp. str. MQ-1.(Spp.sp.MQ1)
Spiroplasma sp. str. DU-1.(Spp.sp.DU1)
Mycoplasma sp. str. 831-C4.(M.sp.831C4)
Mycoplasma sp. str. M1.(M.sp.M1)
Acholeplasma entomophilum str. TAC.(Acp.entomo)
"Mycoplasma ellychnium" str. ELCN-1.(M.ellychni)
Acholeplasma seiffertii.(Acp.seiffr)
Mycoplasma putrefaciens str. C30 KS-1.(M.putrefac)
Mycoplasma mycoides subsp. mycoides str. UM30847.(M.mycoides)
Mycoplasma sp. str. PG50.(M.PG50)
Mycoplasma sp. str. F38.(M.sp.F38)
Mycoplasma sp. str. F38.(M.sp.F38rA)
Mycoplasma capricolum.(M.capricol)

Figure 1: Excerpt from the SSU rRNA Phylogenetic Tree: Mycoplasma Region

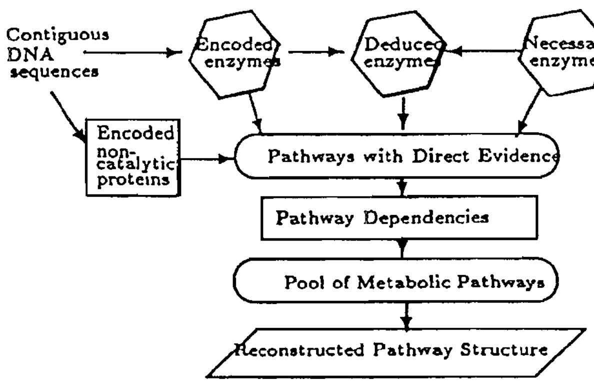
Contiguous DNA sequences
Encoded enzymes
Deduced enzymes
Necessary enzymes
Encoded non-catalytic proteins
Pathways with Direct Evidence
Pathway Dependencies
Pool of Metabolic Pathways
Reconstructed Pathway Structure

Figure 2: Flow of Information in Reconstruction

has a closest common ancestor — which may or may not have actually occurred at some time in evolutionary history. The distance between organisms is the sum of their distances to their closest common ancestor. Thus, the phylogenetic tree provides a measurement of proximity between organisms for use in query relaxation (Gaasterland, Godfrey, & Minker 1992).

Reconstruction from Direct Sequence Evidence

The problem of reconstructing metabolism automatically for an organism can be partitioned into a series of reasoning steps. The starting point is a list of proteins in the organism for which there is evidence. Evidence comes from two sources: automatic interpretations of genome sequence data from AutoSeq and observations in the biochemical literature about the existence or absence of particular proteins in an organism. From this evidence, we want to reason forward about what pathways are potentially present in the organism. Then, we want to reason further about pathways that must be present either because they complement pathways with direct evidence or because they are mandatory (in some form) in any organism. Next, we want to identify conflicts and inconsistencies in the set of candidate pathways and resolve them, when possible, using confidence about the direct evidence. The output is a partial metabolic framework for the input organism. Figure 2 gives an overview of the process. In this section, we describe these steps in detail.

Utilizing Direct Evidence AutoSeq produces a set of proteins for which there is direct evidence in the genome sequence data. Those proteins are represented as facts of the following form that relate confidence (preference) to protein identifiers and descriptions. For convenience, we take the SwissProt ID as the identifier for a protein when available. The annotation Level captures the calibrated score for the protein: **putative_protein(ProteinID):ConfidenceLevel.**

Next, we want to find a set of reaction equations that are associated with the putative proteins. To do so, we ask, "Which of the putative proteins is an enzyme?" and "What enzyme number is associated with that enzyme?" The resulting pool of enzymes becomes candidates for which we have direct evidence from the organism's genome sequence. They are annotated with levels of confidence (or preference) and represented in the form: **putative_enzyme(EnzymeNumber):ConfidenceLevel.** For the annotation value in Level, we take the maximum of the confidence values associated with the set of instances of putative proteins with the enzyme number.[1]

Given a set of putative enzymes for an organism and a database of pathways as defined earlier, one may ask "Which metabolic pathways correspond to the putative enzymes and in what organism is this instance of a pathway reported (in the literature)?" Answers are represented as facts of the form: **putative_pathway(PID, Organism):ConfidenceLevel.** where **PID** is a unique ID for an instance of a pathway and Organism is the name of the organism in which that instance of a pathway was encountered.

Assigning a confidence level to putative pathway differs from assigning level to a putative enzyme. With putative enzymes, many instances of protein sequences for the same enzyme may align with a single region of genome sequence. Since the sequences tend to align with each other, each subsequent sequence provides new evidence for the enzyme only if it covers a larger portion of the genome sequence. On the other hand,

[1]We use *maximum* for ease of illustration. Because we use meta-interpreters to handle supporting evidence, this choice of combination function could easily be replaced with another, for example, least upper bound (lub) or a probabilistic combination. The final choice in a reliable working system will likely be tuned as more is learned about the methodology. We leave this to be worked out with time and experience.

a pathway may contain multiple enzymes (that is, it may contain multiple reaction equations catalyzed by different enzymes) one or more of which appears in the evidence list. Each evident enzyme adds confidence to the presence of the pathway, so the annotation value should reflect this state. We capture this phenomenon abstractly as a combination function θ over the annotation values of the set of putative enzymes connected to a pathway[2].

Thus, one may start with a set of putative proteins for which there is direct evidence in an organism and proceed to gather a set of pathways. Annotated logic programming provides the theory for using confidence in the initial evidence to determine confidence in each pathway. We shall refer to pathways determined in this manner as *directly supported pathways* and annotate them as **direct**: **putative_pathway(PID, Organism):ConfidenceLevel, direct.**

Utilizing Default Information The previous section laid out a method to gather a set of directly supported pathways from genome sequence evidence. The genome sequence interpretation is inherently incomplete — we are limited to identifying coding regions that correspond to sequenced proteins with known function. One way to supplement the reconstruction method is to infer possible pathways from default information about pathways that *must* be present in an organism according to some criteria. Organisms must have the ability to carry out certain functions if they are to survive. We consider the abstract characterization of these functions as default information.

Each pathway in our database of encoded pathways is assigned to one or more of these default modules. Further, each pathway is associated with a *common name* which serves as an abstraction for the pathway. For example, energy metabolism includes glycolysis, and the pathway may be refered to by "glycolysis" rather than a particular set of reaction equations. We can use the abstract pathway name to identify a candidate set of default pathways.

Of interest is the question: for each default pathway, how specific can we be? Should we select an instance of that pathway from a phylogenetically close neighbor or should we leave the pathway at an abstract level. For example, should we simply state that the organism has glycolysis or should we borrow an instance of a glycolytic pathway from the closest available organism? Again with annotated logic programming, we are able to select a close neighbor if it is available and keep track of the fact that it is not yet exact. As before, we represent a selected pathway as a putative pathway, but now, we annotate it with the phylogenetic distance (in the ribosomal RNA tree) to the neighboring organism and note that it is a default (or necessary) pathway: **putative_pathway(PID, Organism):distance, default.**

[2]As with the exact combination function for putative enzyme, we expect appropriate combination functions for putative pathways to emerge with time. In our prototype, we use *least upper bound* (lub) defined as follows for a lattice over integers: [lub(i,j) = i if i > j, lub(i,j) = j if i < j, lub(i,j) = i+1 if i = j]. So, for example, if two pathway enzymes are putative enzymes at level 2, the pathway receives level 1; if two pathway enzymes are putative enzymes at levels 1 and 4, the pathway receives level 1.

Connecting Disjoint Pathways Once a set of directly evident pathways and a set of default pathways are determined, we can ask a series of queries about them: What compounds and cofactors are used but not produced? What pathways are known (encoded) that produce these substances? What compounds and cofactors are produced but not used?

These dangling inputs and outputs to the current set of pathways must be resolved in any final reconstruction. So the query is used to identify alternative connections from the pathways encoded in the database:

?- produced(C), NOT used(C),
pathway_to_substrate(PID,C).
?- used(C), NOT produced(C),
pathway_to_product(PID,C).

These pathways are annotated as **connective**: **putative_pathway(PID, Organism):connective.**

Handling Conflicts and Inconsistencies A (possibly disjoint) metabolic network is considered inconsistent if it violates certain truths. We limit consistency checking to the notion that all compounds that are produced must be either used or exported and all compounds that are used must be either produced or imported. Pathways that have missing inputs or unconsumed outputs even after the search for connections are annotated as **dangling**.

Redundancy is a potential source of conflict. But redundancy in organisms is common, so redundant pathways for producing the same compounds are not necessarily conflicts. We limit conflict resolution to handling cases in which alternative pathways are mutually exclusive. Since the system cannot choose between such pathways without further information, they are represented as a disjunction, indicating that one or more may be present[3].

Reconstruction Algorithm The following algorithm encapsulates the previously described steps.
INPUT: an empty set of metabolic pathways.
OUTPUT: an annotated set of metabolic pathways; a set C of dangling compounds (compounds that are produced or used by some pathway but not used or supplied by some other pathway).

Let P and C be empty sets:

[3]This approach generalizes to *n* mutually exclusive pathways, but we shall not expand on this point here.

1. Add to P each putative pathway with direct evidence. Annotate each with confidence level based on the evidence level and with the label *supported.*
2. For each metabolic module, if an abstract pathway in the module does not have a corresponding supported pathway in P, add to P a default pathway for the abstract pathway, where the default pathway is selected from phylogenetic neighbors when available. Annotate the pathway with phylogenetic distance from the organism in question and label it **default**.
3. For each supported pathway in P that has substrates with no source, search for a connecting pathway that could supply the substrate. If it exists, add it to the set of pathways P with the annotation **connective**.
4. For each pathway in P that has substrate S (or a product D) that is not supplied (consumed) by some other pathway:
   (a) Add each dangling substrate to C in the form substrate(S).
   (b) Add each dangling product to C in the form product(S).
   (c) Annotate P further with the label *dangling*.
5. For each pair of conflicting pathways in P, p1 and p2, replace p1 and p2 with the disjunction (p1 ∨ p2) until no more conflicts exist.

The resulting set P contains a collection of (partially) connected metabolic pathways, some of which have direct evidence, some of which have indirect evidence, and some of which are hypothesized. All pathways in P are annotated with their respective properties.

Utilizing Experimental Data

The previous section described how to carry out by reasoning from proteins with direct evidence. Other sources of knowledge about proteins in the organism allow us to augment the reconstruction process. Those sources are (1) experimental information about reactions that are known to take place inside the organism, (2) experimental information about reactions that are known NOT to take place inside the organism, (3) experimental information about substances that the organism requires from outside the cell, and (4) experimental information about substances that the organism is able to utilize from outside the cell.

In an ideal world, each of these bodies of information would be available in encoded form and could be used to answer queries during the reconstruction process. In reality, we must supply expert users with a means to supply additional information that they want to be part of the reasoning process. This section describes a knowledge representation for capturing the information described above and shows how they may be used in the reconstruction process.

Additional and Absent Reactions If some reaction that is catalyzed by some enzyme, say E, is known to take place in the organism, the user can assert its presence by adding a annotated fact of the form: **reported_enzyme(E):ConfidenceLevel.** In this case, confidence level is assigned by the user and reflects confidence in the relevant literature. A single rule allows the reported enzymes to be used in the generation of putative pathways: **putative_enzyme(E):Level ← reported_enzyme(E):Level.**[4]

There are several alternatives for handling negative information. For now, we have chosen to handle it at the consistency checking phase. As with the reported_enzymes, the user expresses absent reactions according to their enzymes through asserted facts, now of the form: **absent_enzyme(E):Level.** Again, Level reflects the user's confidence in the source of information. An integrity constraint is a logical statement that must be true of the represented data. Here, two integrity constraints express unacceptable states :

← **absent_enzyme(E):LevelA,**
    putative_enzyme(E):LevelB.
← **absent_enzyme(E):LevelA,**
    reported_enzyme(E):LevelB.

We can check for conflicts by asking the integrity constraint as a query. If it is *true* in the data, then a conflict exists, and any pathway that depends on the putative or reported enzyme should be suspect. Again, this is handled through a single rule:

suspicious_pathway(P,O):Annotation ←
    putative_pathway(P,O):Annotation,
    pathway_to_enzyme(P,E),
    absent_enzyme(E).

External Medium Information about external sources of compounds provides possible resolutions for dangling compounds. If a compound in the set *C* produced by the reconstruction algorithm is known to be necessary in the external growth medium for the organism, we can hypothesize that it is imported. Until a database of growth conditions for organisms is available and integrated into the system, we depend on the user to provide growth medium compounds in asserted facts of the form: **growth_medium(C)**. A single rule allows the **growth_medium** fact to be used to resolve dangling compounds: **produced(C) ← growth_medium(C)** Furthermore, for any compound that is necessary in the growth medium and is not dangling, we can infer that pathways are present to consume it. The connection step is used together with

[4]We have chosen to consider the level of the derived putative_enzyme fact to be the same as the level of the reported_enzyme fact. However, a user who wants to express a more refined opinion about the domain could easily change the rule to contain a function *f* on the head annotation as in: **putative_enzyme(E):*f*(Level) ← reported_enzyme(E):Level.**

the rule above to augment the current set of pathways with additional connected pathways.

Intuitively, a faithful representation of the growth medium completely would require that we be able to capture information about mutually exclusive growth medium compounds or alternative compounds as in "Organism O requires compound C or compound D, but not both." However, we intend the reconstructed pathways to represent the full collection of pathways that may be present in an organism — i.e. the pathways that are available to be invoked when an substrate compound is present in sufficient concentration. So it is sufficient to represent external growth medium compounds as definite facts without qualifications about mutual exclusion or alternatives.

Interactive Reconstruction Algorithm To recapitulate briefly, we augment our reconstruction algorithm by allowing the user to assert the following types of facts: **reported_enzyme(E):Level. absent_enzyme(E):Level. growth_medium(C).** From these facts, we can augment the set of pathways with additional pathways and identify new inconsistencies. The algorithm is modified as follows:

Let P and C be the initially empty sets of pathways and compounds.

- (Replace Step 1) Generate the putative_pathways in P using putative_enzyme facts that are derived both from putative_protein facts and from reported_enzyme facts.
- (Replace Step 4) For each pathway in P that has substrate (or a product) S that is not supplied (consumed) by some other pathway or contained in a fact of the form **growth_medium**(S):

  1. Add each dangling substrate to C in the form substrate(S).
  2. Add each dangling product to C in the form product(S).
  3. Annotate P further with the label *dangling*.
- (New step) For each compound in growth_medium, search for a connecting pathway that could consume the compound. If it exists, add it to the set of pathways P with the annotation *connecting*.
- (New step) Identify suspicious pathways as defined above.

This augmented version of the metabolic reconstruction algorithm produces a set of (partially) connected annotated metabolic pathways.

Examples from Mycoplasma capricolum

In the background section, we noted that one can lay out the basic parts of the metabolic machine for procaryotic organisms. In this section, we consider Mycoplasma capricolum as a specific example and use it to illustrate the steps in the reconstruction algorithm. In an interpretation of the available Mycoplasma capricolum genome sequence data, our group at Argonne and an independent group at EMBL have identified just over 300 regions that code for proteins.

First, we lay out the modules: an organism like Mycoplasma capricolum has DNA replication, maintenance, and repair. It manufactures proteins through folding, refolding, assembly, degradation, maintenance. It assembles, maintains, repairs, uses, and translocates ribosomes. Its energy metabolism includes amino acid synthesis and degradation, carbohydrate synthesis and degradation, lipid (and membrane) synthesis and degradation. It synthesizes, degrades, and uses nucleotides. It has a means to do membrane transport and signal transduction. It must carry out cell division and maintain a cell clock. Optional modules include cell wall synthesis and degradation, motility mechanism synthesis and maintenance (e.g. cilia and flagella) and secondary metabolic functions. In the interpretation of Mycoplasma capricolum, we have evidence of proteins from metabolite transport and activation, amino acid metabolism, nucleotide metabolism, lipid metabolism, carbohydrate metabolism, DNA replication and recombination, DNA repair, cell division, transcription, translation, protein biosynthesis, ribosomal proteins, internal transport and translocation as well as RNAs and several unclassified proteins.

To illustrate the reconstruction steps, we turn to three examples. We reason about arginine biosynthesis as follows. We have weak genome sequence evidence for the enzyme that catalyzes arginine hydrolysis, 3.5.3.6. We have strong evidence for 2.1.3.3, ornithine carbomoyl transferase. The literature tells us that arginine hydrolysis occurs in several Mycoplasma strains(Pollack 1986). So from the putative enzymes 3.5.3.6 and 2.1.3.3, we infer that the pathway between arginine and L-ornithine is present. Through connectivity and default reasoning, we suppose that carbamoyl phosphate is taken to CO_2 by enzyme 2.7.2.2. There is evidence in the literature that Mycoplasma mycoides is dependent on arginine in the growth medium. Since M. mycoides is a close neighbor of M. capricolum, it gives additional evidence that the organism can use arginine to make ornithase and carbamoyl phosphate. Figure 3a gives the reconstructed pathway.

Our next example shows how we use default reasoning to construct a module from scant evidence. We have a strong genome sequence hit on arabinose permease, a transport protein that brings arabinose into the cell. In addition, we have the evidence described above that carbamoyl phosphate and ornithine are produced by the arginine deaminase pathway. Further, we have weak sequence evidence for xylose permease. From this evidence, we infer strongly a putative pathway that takes L-arabinose to D-xylulose-5-phosphate. Further, we infer weakly that D-arabitol is taken through D-xylulose to D-xylulose-5-phosphate. From the existence of these two parts of the pentose

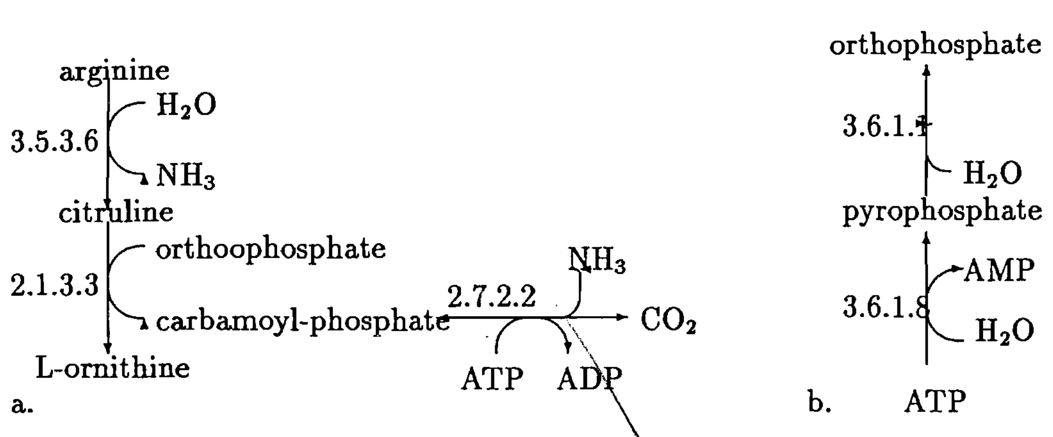

Figure 3: Arginine Hydrolysis and Phosphate Metabolism Pathways

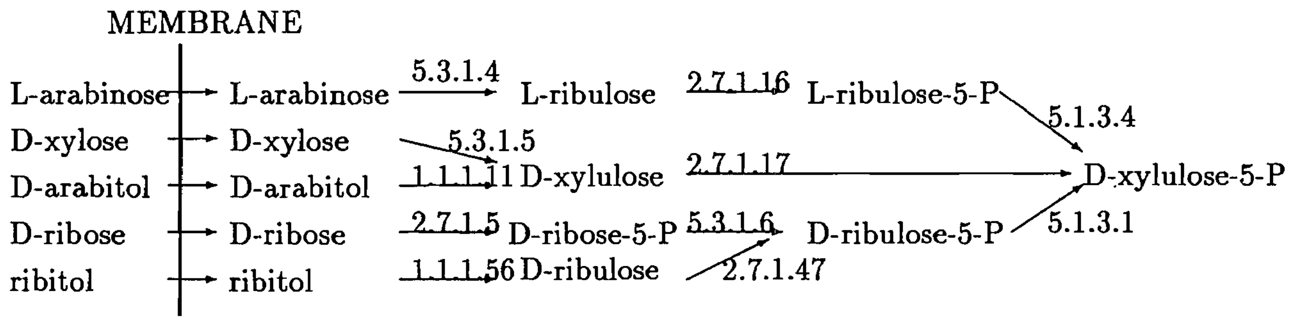

Figure 4: Pentose Pathway

pathway, we infer through connectivity that the remaining branches of the pathway that take D-xylose, D-ribose, and ribitol to D-xylulose-5-phosphate. All of these pentoses are very common in the host of M. capricolum (the gut of a goat), so we can postulate that M. capricolum takes advantage of it.

Finally, we reason forward through connectivity that D-xylulose-5-phosphate enters the glycolytic system via the truncated hexose-monophosphate shunt (Desants, Tryon, & Pollack 1989). There is strong direct evidence for transketylase, 2.2.1.1, which takes D-xylulose-5-phosphate to D-fructose-6-phosphate, which like carbamoyl phosphate leads us into the glycolytic system.[5] Figure 4 shows the reconstruction.

Our third example carries the unfolding pathway topology in the direction of energy metabolism. We have strong sequence evidence for 3.6.1.8 and 3.6.1.1 The first of these mediates between ATP and pyrophosphate using H_2O and producing AMP. The second takes pyrophosphate to orthophosphate using H_2O. So we conclude that the ATP pyrophosphate hydrolase dependent pathway is present. Figure 3b shows this pathway.

Conclusions

The methods for reconstruction described in this paper emerged from the manual reconstruction of a topology of metabolism for Mycoplasma capricolum. The methods depend on the completeness of encoded pathways and the domain expert's knowledge of the literature for supplementary information. An important feature of the system is ease of of assertion of new information from the literature during the reconstruction process. M. capricolum provides a particularly interesting test case because it is dependent on its environment for particular compounds. An open question centers on whether particular pathways with sequence evidence are in truth used in the living organism or whether they are relics of ancestral organisms, leaving the modern organism to import pathway products from the outside instead of producing them.

One may think of the reconstruction process as protein driven or as compound driven. The approach we describe here uses both types of information to con-

[5]It is worthwhile to note that the shunt pathway can work backwards as an additional source of D-ribose-5-phosphate, and thus ribose-1-phosphate, to lead into synthesis of purine and pyrimidine. For space reasons, we forgo further description of the glycolytic pathway topology.

clude the presence of pathways. Evidence for compounds and evidence for catalysts both play roles in the process. The resulting topology can be considered as a graph with weighted edges; each weight indicates the strength of confidence or preference about an arc in the graph. We are exploring tools for visualizing reconstructed pathways so that the domain expert user can easily see the ongoing process and turn quickly to the literature for additional evidence.

Another facet of the work is to use the reconstructed pathways to feed backward into the interpretation. This is a straightforward process: if there is strong evidence for one enzyme in a pathway, we reason forward that the other proteins in the pathway are present and then turn back to the sequence analysis in AutoSeq to look for other supporting evidence for those proteins.

It is clear that by limiting the implementation to enzymes only and by characterizing individual reaction equations according to the enzymes that catalyze them, we limit the search space for reconstruction to catalyzed reactions and reactions that appear in encoded pathways. However, we also recognize that defining the methodology is a substantial step forward. Once we show that the approach works for organisms beyond Mycoplasma, we shall explore how it may be generalized to handle individual non-enzymatic reactions in a general manner.

References

Altschul, S. F.; Gish, W.; Miller, W.; Myers, E. W.; and Lipman, D. J. 1990. Basic local alignment search tool. *Journal of Molecular Biology* 215:403–410.

Baehr, A.; Dunham, G.; Ginsburg, A.; Hagstrom, R.; Joerg, D.; Kazic, T.; Matsuda, H.; Michaels, G.; Overbeek, R.; Rudd, K.; Smith, C.; Taylor, R.; Yoshida, K.; and Zawada, D. 1992. An integrated database to support research on escherichia coli. Technical Report ANL-91/1, Argonne National Laboratory.

Collins, J., and Coulson, A. 1990. Significance of protein sequence similarities. In Doolittle, R., ed., *Methods in Enzymology*, volume 183. Academic Press. 474–486.

Desants; Tryon; and Pollack. 1989. Metabolism of mollicutes. *Journal of General Microbiology* 135:683–691.

Doolittle, W. 1992. What are the Archaebacteria and Why are They Important? In Danson, M.; Hough, D.; and Lunt, G., eds., *The Archaebacteria. Biochemistry and Biotechnology.* 1–6.

Felsenstein, J. 1981. Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal Of Molecular Evolution* 17:363–376.

Gaasterland, T., and Lobo, J. September, 1994. Qualified Answers That Reflect User Needs and Preferences. In *Proceedings of the International Conference on Very Large Databa ses.*

Gaasterland, T.; Lobo, J.; Maltsev, N.; and Chen, G. August, 1994. Assigning Function to CDS Through Qualified Query Answering. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology.*

Gaasterland, T.; Godfrey, P.; and Minker, J. 1992. Relaxation as a Platform for Cooperative Answering. *Journal of Intelligent Information Systems.*

Henikoff, S., and Henikoff, J. 1993. Protein family classification based on searching a database of blocks (document: blockman.ps). obtained from anonymous ftp to *sparky.fhcrc.org* in */blocks.*

Karp, P., and Riley, M. 1993. Representations of metabolic knowledge. In *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology.* AAAI Press.

Kifer, M., and Subrahmanian, V. 1993. Theory of generalized annotated logic programming and its applications. *Journal of Logic Programming.* accepted for publication.

Olsen, G.; Matsuda, H.; Hagstrom, R.; and Overbeek, R. 1994. fastdnaml: A tool for construction of phylogenetic trees of dna sequences using maximum likelihood. *CABIOS.*

Olsen, G.; Woese, C.; and Overbeek, R. 1994. The winds of (evolutionary) change: Breathing new life into bicrobiology. *Journal Of Bacteriology* 176(1):1–6.

Pollack. 1986. *Journal of Syst. Bact.* 36(1):60–65.

Selkov, E.; Yunus, I.; and *et.al.*, S. B. 1994. Database on enzymes and metabolic pathways. Technical report, Institute of Theoretical and Experimental Biophysics, Russian Academy of Sciences, 142292 Pushchino, Russia.

Yoshida, K.; Smith, C.; Kazic, T.; Michaels, G.; Taylor, R.; Zawada, D.; Hagstrom, R.; and Overbeek, R. 1992. Towards a human genome encyclopedia. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, 307–320.