VERSION 9.5

User's Guide

PATHWAY TOOLS

Volume II: Editors and PathoLogic

Pathway Tools User's Guide, Version 9.5

We gratefully acknowledge contributions to Pathway Tools, used by permission, from:
Jeremy Zucker, Harvard Medical School
The Laboratory of Christos Ouzounis, European Bioinformatics Institute

SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025 U.S.A.
ptools-support@ai.sri.com

CONTENTS

PREFACE

This document will familiarize you with the Pathway Tools software and the Pathway/Genome Databases (PGDBs) that are available from SRI International. Pathway Tools operates across one or more PGDBs, including the EcoCyc *E. coli* DB, and the MetaCyc DB. Each database describes the biochemical pathways and genome of a single organism. All PGDBs are managed by an object-oriented database system called 'Ocelot'.

Pathway Tools contains several component software modules including:

- Pathway/Genome Navigator, the primary tool by which users visualize, query, and analyze the information contained within a PGDB.
- PathoLogic Pathway Predictor, which creates new PGDBs from annotated genomes.
- Pathway/Genome Editors, which modify the information contained within PGDBs, such as allowing users to add in-house proprietary data about a pathway or gene of interest. The Editors include a general tool for database browsing and editing, called the GKB Editor.

Pathway Tools is most commonly provided in two configurations:

- BioCyc Configuration. Includes the Pathway/Genome Navigator, and one or more PGDBs, such as EcoCyc and MetaCyc. This configuration is available for the Sun workstation running Unix, for the PC running Linux, and for the PC running Windows 2000, XP, or higher. BioCyc can run as both an X-windows or Windows application and as a Web server for the user's intranet. In this configuration, PGDBs are embedded inside the binary executable program, and are available for read-only access. In addition, PGDBs that have been created using the full Pathway Tools configuration can be imported into the BioCyc configuration.
- Full Pathway Tools Configuration. Includes the Navigator, PathoLogic, and Editors, as well as one or more PGDBs. This configuration is available for the Sun workstation running Unix, for the PC running Linux, and for the PC running Windows 2000, XP, or higher. The Navigator functionality can run as an X-windows application and as a Web server. This configuration allows the creation of new PGDBs, which can be stored either as disk files or in an Oracle or MySQL database server. Newly created PGDBs can be updated by users. Use of an Oracle or MySQL database is recommended if multiple users will be updating a given PGDB.

Volume I of the Pathway Tools User's Guide describes the Navigator. Volume II describes PathoLogic and the Pathway/Genome Editors.

In Volume II of this user's guide

- Chapter 1 describes how to use the PathoLogic Pathway Predictor to create new Pathway/ Genome Databases.
- Chapter 2 describes how to use the Pathway/Genome Editors to interactively update Pathway/Genome Databases.

The Preface of Volume I of the Pathway Tools User's Guide lists other information resources about Pathway Tools.

Chapter 6 of Volume I outlines what to do if you encounter problems with the software or with information contained within any of the provided databases.

1 PATHOLOGIC: AUTOMATED CREATION OF PATHWAY/GENOME DATABASES

The PathoLogic Pathway Predictor software supports the creation of new pathway/genome databases (PGDBs) from the annotated genome of an organism. The program assumes that the positions of genes within the genome have already been identified, and that gene functions have already been predicted. The program infers metabolic pathways by analyzing the genome annotation with respect to a reference database of metabolic pathways, MetaCyc. PathoLogic also generates reports that summarize the evidence used for deducing the pathways inferred and the results of comparing the reactions carried out in the subject organism versus those in the reference database.

We define the term *metabolic pathways* in this document to mean pathways involved in small molecule metabolism. Therefore, the PGDB created with the Predictor does not represent pathways involved in macromolecular metabolism, nor transport pathways, nor signal-transduction pathways.

The process of inferring the metabolic network of a organism from its genome, called pathway analysis, extends the paradigm of genome analysis by producing interpretations of a genome sequence that are biologically more informative than the annotated genome alone. For example, predictions of individual enzymatic gene function(s) for a given organism may be viewed from the perspective of the entire predicted metabolic network for the organism. From this "whole-organism" perspective, one may determine whether the given functional assignment fits into the predicted metabolic network in a coherent way, e.g., does it help form a model of the organism's metabolism that is consistent with known physiological and biochemical properties such as experimentally isolated enzymatic activities, growth media substrate requirements, and/or substrate utilization patterns?

PathoLogic performs most of its work automatically, but some interactive assistance from the user is required. This chapter describes the steps carried out by PathoLogic and how it is used. "Overview of PathoLogic Execution" in section 1.1 describes the events that take place during the execution of the Predictor. In "Batch Mode" in section 1.6 the format of each of the two input files expected by the Predictor is described. "Creating a Pathway/Genome Database" in section 1.3 gives a step-by-step guide to how to use PathoLogic. "Output from the PathoLogic Pathway Predictor" in section 1.5 describes the reports generated by the Predictor. Finally, "Error Sources in PathoLogic Predictions" in section 1.7, discusses some possible sources of error the user should be aware of before using the Predictor.

1.1 OVERVIEW OF PATHOLOGIC EXECUTION

In this section, we describe the processing performed by PathoLogic. The first part of this section is a rather brief technical description of what transpires at the database level. The second part describes, chronologically, the phases of the Predictor execution.

1.1.1 Database Generation Perspective

PathoLogic generates a PGDB representation of the genome and metabolic pathways of the subject organism. More specifically, the principal goal of PathoLogic is to create the appropriate set of instance frames, to populate them with appropriate slot values, and to interconnect these frames in a manner that accurately reflects their semantic relationships. Two types of information are encoded in the new PGDB:

- A set of class frames that encodes the database schema. These frames are copied directly from MetaCyc.
- A set of instance frames that encode the chromosomes, genes, proteins, reactions, pathways, and substrates of the subject organism.

The key to understanding how PathoLogic operates is to understand how it creates, populates, and interconnects these frames. PathoLogic creates one frame for each genetic element (e.g., chromosome or plasmid) in the organism, and populates its slots with data extracted from the annotation and sequence files that PathoLogic accepts as input. It creates one frame for each gene described in the annotation file. Some of the gene information in the input file is used to populate slots in the gene frames (such as the base-pair position of the gene and the name). Other attributes populate slot values in the frame describing the protein or RNA product of the gene the (e.g., product name). PathoLogic also creates a link from the gene to the genetic element that contains it, based on the association of the gene to a file for a particular genetic element.

PathoLogic first initializes the new PGDB with the MetaCyc schema. It then reads the genetic-elements.dat file, and creates a frame for each genetic element defined in that file. Next, it reads the input file for each chromosome, and creates a gene frame for each gene in the input file. It creates one polypeptide frame for each gene whose product is a protein, and creates a frame in the rRNAs or tRNAs class for genes whose products are rRNAs or tRNAs. Then, PathoLogic creates connections between polypeptide frames and the appropriate reaction frames for those proteins that are enzymes. The connections are made based on the EC number assigned to a protein in the annotation file or using a name-matching tool (for a more detailed explanation, see "PathoLogic Operation", following). The actual connection is made using an intermediary frame called an enzymatic reaction that describes the pairing of an enzyme and a reaction; see [11] for a detailed explanation of the role of enzymatic-reaction frames. The next step is to match the reactions now known to be catalyzed by the organism against the reactions in each MetaCyc pathway. PathoLogic initially imports every pathway containing a reaction in the subject organism (and imports all reactions and compounds in those pathways), but it then prunes out some pathways for which it decides insufficient evidence exists for their presence.

1.1.2 PathoLogic Operation

PathoLogic accepts the following inputs:

A file called **`genetic-elements.dat`** defines each genetic element in the organism, and provides pointers to each of the following files:

- A FASTA file containing the DNA sequence for each genetic element in the subject organism (that is, each chromosome or plasmid).

- A file containing the annotation for the corresponding genetic element (e.g., features, chromosomal location, gene function). This file can be in Genbank format or PathoLogic format.
- The MetaCyc DB, which comes with the Pathway Tools.

PathoLogic generates the following output:

- A new PGDB for the subject organism

The major steps in using PathoLogic are as follows.

- Enter defining information for the new PGDB, such as the name of the organism and the names of the DB authors
- Perform one or more trial parsing operations on the input data files to (a) ensure that they are in the proper format -- this step might be repeated multiple times if the files are not initially in the proper format, and (b) ensure that as many enzyme names as possible are recognized by the PathoLogic enzyme name matcher -- this step might be repeated multiple times after research on unrecognized enzyme names
- Build the new PGDB, which automatically creates the genes, proteins, reactions, pathways, etc., within the PGDB
- Refine the PGDB, using several different interactive dialogs to define protein complexes within the PGDB, identify protein substrates, and create the Overview diagram

1.1.2.1 Matching of Enzymes to Reactions

An operation of central importance for PathoLogic is the matching of enzymes defined in the input file(s) to reactions and pathways defined in the MetaCyc DB. By establishing a correspondence between an enzyme and the reaction it catalyzes, PathoLogic infers a structured description of the function of the enzyme, and immediately is able to place the enzyme in one or more pathways, since each reaction in MetaCyc is connected directly to those pathways that contain it. PathoLogic performs this enzyme matching during both the trial parse phase and the automated build phase of its operation.

If an EC (Enzyme Commission) number is provided for an enzyme in an input file, PathoLogic prefers to use the EC number to match the enzyme to its corresponding reaction, because EC numbers are relatively unambiguous. If no EC number is provided, then PathoLogic attempts to match using the enzyme name. Note that the protein sequence of the enzyme is not used for matching. Our approach is to perform matching at the functional level rather than the sequence level because we prefer to utilize existing assignments of gene function that may have been made by expert sequence analysts rather than to attempt to infer new enzyme functions for each gene in an automated fashion.

PathoLogic matches enzyme names against a dictionary of enzyme names that is constructed from the following four sources.

1. All of the enzyme names contained in the MetaCyc DB (which includes all *E. coli* enzymes from the EcoCyc DB).
2. All the enzymes found in the ENZYME database [1]; those names have been imported into reaction frames in MetaCyc.

3. A file that maps enzyme names not found in MetaCyc to MetaCyc reactions. The name of this file is aic-export/ecocyc/pathologic/9.5/data/pangea-enzyme-mappings.dat.
4. An optional user-provided file that maps enzyme names not found in MetaCyc to MetaCyc reactions. The name of this file is aic-export/ecocyc/pathologic/9.5 /data/local-enzyme-mappings.dat.

The results of the matching process are summarized in the PathoLogic window for the user. A more detailed list of all matching and non-matching enzyme names are saved to a file called **`aic-export/ecocyc/ORGIDcyc/VERSION/reports/name-matching-report.txt`**, which is the *Enzyme Name to Reaction Mapping Report*. The name-matching process is applied to each gene product in the input annotation file(s): The name-matching outcome for a given gene product places it into one of the following sections of the file:

- Unambiguous Matches: There is an enzyme name in the lookup table that matches the gene product name string exactly. In this case, PathoLogic automatically creates the Enzymatic-Reaction (see Appendix A, "Guide to the Pathway Tools Schema" for definition of this concept) connection between the appropriate enzyme and reaction frames in the PGDB.
- Ambiguous Matches: The gene product name string is ambiguous because it is associated with more than one reaction in the PathoLogic lookup table. Since PathoLogic cannot make a decision as to the correct enzyme name to reaction mapping, it is up to the user to resolve the ambiguity as described later in this section.
- Probable Metabolic Enzymes: The gene product name string has not matched any name in our lookup table, but is considered a probable metabolic enzyme because the following conditions are true: (a) one of the words in the gene product name ends in the string “ase,” and (b) the name does not contain words that would indicate it to be a non-metabolic enzyme, such as “protein kinase,” “peptidase,” etc.
- No Match: The gene product name string has not matched any name in the PathoLogic lookup table, and is not considered a probable metabolic enzyme.

After matching has completed, the user can use the name matching report to drive a manual phase of enzyme name matching performed by the user. The Possible Metabolic Enzymes section of the report contains the gene product names most likely to be unrecognized metabolic enzymes, so the user should focus their efforts on that section. The user should take care to ignore product names that are not metabolic enzymes (such as protein kinases involved in cell signaling pathways), and to ignore product names that represent general, non-specific enzymatic activities that do not correspond to a single reaction, such as “oxidoreductase” or “transaminase”.

For example, imagine that the enzyme name matching report contains the name “carboxymuconolactone decarboxylase.” The user can follow several strategies to identify what reaction this enzyme name corresponds to:

- In the Navigator interface, select the MetaCyc DB, enter protein mode, and perform a substring query on the phrase “carbox muc lact”. This query will search all MetaCyc enzymes and reactions for names that contain all three of these substrings. A variant of this name might be present in MetaCyc, allowing the user to identify the reaction. When the reaction is found, right-click on the reaction to find the MetaCyc frame name for the reaction (the frame name is printed at the very top of the right-button pop-up menu).

- Search for the enzyme name in SwissProt, and try to discern the reaction that the enzyme catalyzes from the SwissProt entry, either based on its EC number (if present), or the Comment section, which sometimes describes the catalytic activity of the enzyme.
- Search for the enzyme name in PubMed, and try to discern the reaction the enzyme catalyzes from the literature, and then determine whether MetaCyc describes that reaction, such as by entering compound mode, searching for one of the substrates, and then examining the list of all reactions containing that substrate.
- Search for the enzyme name in other metabolic DBs such as KEGG, and again try to discern what reaction the enzyme catalyzes, and identify the MetaCyc entry for that reaction.

When the MetaCyc reaction corresponding to that enzyme name has been identified, the user can assign the reaction to the enzyme using the **Refine -> Assign Probable Enzymes** command after completion of the automated build procedure. This is the simplest and, in most cases, the preferred method of ensuring the reaction is assigned to the enzyme. Alternatively, the user can inform PathoLogic of this new enzyme to reaction correspondence by entering the enzyme name and the reaction frame name in a new line in file **`aic-export/ecocyc/pathologic/9.5/data/local-enzyme-mappings.dat`**. This alternative is useful if the name is likely to appear in future annotation files for other PGDBs, as the local enzyme mapping file will be reused for later PGDBs. A third approach is to change the enzyme name in the input file to a name that you know PathoLogic will recognize because that name is defined in MetaCyc, such as if there was a typographical error in the original name. For the latter two approaches, the new assignment will take effect the next time the enzyme name matching procedure is invoked (e.g. if the PGDB is rebuilt or if the **Refine -> Re-Run Name Matcher** command is invoked.)

If the reaction corresponding to that enzyme cannot be identified, nothing further can be done for that enzyme. If the reaction can be identified, but it does not exist in MetaCyc, the user should manually create the reaction in the new PGDB after completion of the automated build process.

1.1.2.2 Assigning Evidence Scores to Predicted Pathways

This section describes the process by which PathoLogic determines whether a pathway exists in the subject organism. For each pathway predicted to be present in the subject organism, the Predictor computes a score that reflects an approximate measure of the likelihood that the pathway is present. It then combines the score with other criteria to decide whether to copy the pathway from MetaCyc to the new PGDB.

The first step in assessing the evidence that pathway P occurs in organism O is to consider the fraction of reaction steps in P that can be catalyzed by enzymes in O. For example, the reference pathway for pyrimidine biosynthesis is comprised of the set of reactions represented by EC numbers 6.3.5.5, 2.1.3.2, 3.5.2.3, 1.3.3.1, 4.1.1.23, and 2.4.2.10. Sequence analysis of the *H. pylori* genome predicted the existence of enzymes that catalyze all of the steps in pyrimidine biosynthesis except for 2.4.2.10. Therefore, we say we have evidence that 5 of the 6 steps of *E. coli* pyrimidine biosynthesis can be catalyzed by *H. pylori*.

The method also considers evidence against the occurrence of P in O. If a reaction R occurs in two pathways, P1 and P2, we should exercise caution in counting the occurrence of an enzyme that catalyzes R as evidence for the presence of P1 in O, when the enzyme might in fact be present in the genome only because of its role in P2. In particular, if *every* reaction step that is present in pathway P is also present in some other pathway, the presence of P in organism O should be treated as very hypothetical. The user may choose to later remove from the database those pathways for which no steps unique to that pathway are present.

We use the notation *X|Y|Z* to denote the score for a pathway P in O, meaning that P consists of X total reactions, that enzymes for Y of the reactions are present in O, and that Z of the Y reactions are used in other known pathways. For example, the score computed for *H. pylori* for the reference pathway for valine biosynthesis is 4|2|1. This means that 2 of the 4 reactions required for this pathway can be catalyzed by *H. pylori* enzymes; one of those two enzymes catalyzes a step in a different pathway (alanine biosynthesis).

Pathways are excluded from the new PGDB based on the following conditions. These conditions are primarily designed to address the fact that many different pathways share reactions in common, and more specifically, that MetaCyc contains a number of metabolic pathways that are highly related to one another, which we term "variant pathways." For example, MetaCyc contains 10 variants of the TCA cycle that share many reactions in common. PathoLogic intelligently selects among variant pathways by preferring those pathways containing a unique enzyme -- an enzyme catalyzing a reaction step found in only a single pathway. A pathway is pruned from the PGDB if one of the following conditions holds:

- No reactions contained in the pathway are known to be catalyzed by the organism (X = 0), OR
- None of the reactions in the pathway that are catalyzed by the organism are unique to that pathway, AND
- The pathway consisted of more than two reactions but contained evidence for only a single reaction, OR
    - The set of reactions for which there was evidence was the same as the set of reactions for which there was evidence in some other pathway, but the pathway contains additional reactions (for which there is no evidence) not contained in the other pathway, OR
    - The set of reactions for which there was evidence was a proper subset of the set of reactions for which there was evidence in some other pathway, AND
        - Both the pathway and the other pathway are members of the same variant class, OR
        - The pathway was classified as a biosynthetic pathway and was missing at least the last two steps from the end of the pathway. The rationale for this criterion is that if the final steps are missing, then the pathway is not producing its intended target, so the other pathway(s) containing the common reactions are more likely to reflect the actual role of those reactions in the organism. OR
        - The pathway was classified as a degradation pathway and was missing one or more steps from its beginning. The rationale for this criterion is that if the

initial steps are missing, then the pathway is not degrading its intended substrate, so the other pathway(s) containing the common reactions are more likely to reflect the actual role of those reactions in the organism. OR

- The pathway was classified as an energy metabolism pathway and was missing at least half its reactions. The rationale for this criterion is that there is a great deal of overlap of reactions among the energy metabolism, with many reactions appearing in many pathways. Thus, many false positives are likely to be inferred from among these pathways. An arbitrary but reasonable cutoff for excluding some of these false positives is 50%.

Our approach is conservative in the sense that it attempts to bring more potential pathways to the attention of the user, to allow the user to prune out those pathways rather than having PathoLogic prune them out. That is, PathoLogic attempts to err on the side of more false-positive pathway predictions than on the side of more false-negative predictions.

The scores assigned to the computationally predicted pathways of an organism should be interpreted with due caution. Firstly, the user should evaluate pathway predictions and corresponding scores by reference to relevant experimental data, e.g., growth media requirements, substrate utilization patterns, experimentally isolated enzymatic activities, relevant metabolic pathway studies and gene expression analysis. For example, there may be experimentally demonstrated enzymatic activities even though the corresponding gene(s) for this (these) enzyme(s) may not have been identified in the genome. This situation would result in an artificially low score for any pathway that utilized this (these) enzymatic activities). Secondly, the user should consider the limitations of sequence analysis as a tool for understanding metabolic function. For example, genes within a genome may have been incorrectly assigned a given enzymatic function, resulting in an artificially high score for pathways that used that enzyme.

1.2 PATHOLOGIC INPUT FILE FORMATS

This section describes the PathoLogic input files in more detail.

1.2.1 File genetic-elements.dat

This "master file" is essentially an index to a larger set of files provided for the organism. These files are designed to accommodate both fully assembled genomes, and partially assembled genomes. Therefore, each entry in the **`genetic-elements.dat`** file describes one genetic element (meaning a chromosome or plasmid), or one contig. Each entry specifies the data files provided for each genetic element or contig, and specifies properties of the genetic element or contig (properties such as is a genetic element circular or linear; what genetic element is a given contig part of).

A sample genetic-elements.dat file is provided at URL **`http://bioinformatics.ai.sri.com/ ptools/sample-genetic-elements.dat`**. The comments at the start of the sample file define the syntax of the file, the allowable fields, and the values that should be provided for each field.

For each genetic element or contig, two additional input files must be provided:

- A sequence file in FASTA format, containing the full nucleic-acid sequence of the genetic element or contig. This file is identified by the suffix fsa or fna (e.g., tpal.fna is the FASTA file containing the Treponema pallidum complete genomic sequence).
- An annotation file describing the predicted genes for that genetic element or contig. The annotation file should be in either PathoLogic format or Genbank format. PathoLogic relies upon the file suffix to determine which format the file is in: suffix gbk indicates Genbank format (e.g., tpal.gbk is the file containing the annotation in GenBank flat file format), and suffix pf indicates PathoLogic format.

1.2.2 The PathoLogic File Format

PathoLogic format is a simple and easy-to-parse attribute-value based file format. Each gene record starts with a line containing the **ID** attribute, and ends with a line containing two slashes ("//"). One attribute-value pair is allowed per line, and the value is separated from the attribute name by one tab character. The location of the gene in the corresponding FASTA file is specified by the attributes **STARTBASE** and **ENDBASE**. These locations refer to the start and end of transcription for a given gene and thus serve to indicate the direction of transcription along the chromosome. Lines starting with ";" are comment lines and are ignored by the parser (see examples in file format below).

The valid attributes are:

- ID. The unique identifier for the gene. Should be the same as the unique ID used by the sequencing effort. If supplied, it will become the frame ID for the gene. Highly Recommended.
- NAME. The mnemonic shorthand name used for the gene. It will become the common name of the gene. Required.
- STARTBASE. An integer. Location in the corresponding FASTA file of the translation start of the gene. The location corresponds to the first nucleotide of the initial triplet, typically ATG. Required.
- ENDBASE. An integer. Location in the corresponding FASTA file of the translation end of the gene. The location corresponds to the last nucleotide of the last translated codon. When the gene is on the minus strand, the endbase will be a smaller number than the startbase. Required.
- FUNCTION. The assigned function of the product of the gene. It will become the common name of the protein. The word ORF should be used if the function is unknown. For multifunctional proteins, supply multiple FUNCTION lines in the file. Required.
- PRODUCT-TYPE. The type of gene product, as chosen from the following controlled vocabulary of terms. P = protein (or a hypothetical ORF); PSEUDO = pseudogene; TRNA = tRNA; RRNA = ribosomal RNA; MISC-RNA = some other RNA, such as can be part of various ribonuclear protein complexes. Required.
- SYNONYM. Additional synonyms under which a gene may be known, can be indicated herewith, one value per line. Optional.
- EC. If the annotation effort has yielded an enzyme assignment with EC number(s), it (they) should be indicated in the standard way of using four numbers (and/or dashes),

separated by three dots. Recommended.

- DBLINK. Used for linking the gene object to other databases. String should be of the form <DB:Accession>. If available, the SwissProt accession number which corresponds to the gene is specified here (see the Predictor format example below for how to specify a link). The following are the databases for which links may be created (Optional):
    - ENTREZ
    - CGSC
    - SWISSPROT
    - PDB
    - MetaCyc
    - SWISSMODEL
    - PID
    - LIGAND
    - ENZYME-DB
    - LIGAND-MAP
- GENE-COMMENT. An additional comment may be placed here. It will end up in the comment slot of the gene frame. Optional.
- FUNCTION-COMMENT. An additional comment may be placed here. It will end up in the comment slot of the protein frame or the enzymatic-reaction frame. If there are multiple functions for the gene product, this comment will be assigned to the function immediately preceeding. Optional.
- PRODUCT-ID. A unique identifier for the object that encodes the gene product. Optional.
- FUNCTION-SYNONYM. Other names by which the gene product is known. If there are multiple functions for the gene product, this synonym will be assigned to the function immediately preceding. Optional.
- INTRON. The start and end positions, in absolute base pair numbers with a hyphen (-) in between, of an intron in the gene. Each intron should appear on a separate line. Optional.

The attributes **FUNCTION**, **SYNONYM**, **EC**, **DBLINK, FUNCTION-COMMENT** and **FUNCTION-SYNONYM** may be used several times per gene record.

Here is a short example PathoLogic Format file. For a longer example file please see URL http://brg.ai.sri.com/ptools/ tpal.pf.

```
;;; The PF file format.
;;; This is a comment in front of an imaginary example record. It
    starts
;;; with a semicolon. Each record starts with an "ID" line, and
    is
;;; terminated by a "//" line.
;;;
```

```
ID              b1262
NAME            trpC
STARTBASE       1317812
ENDBASE         1316451
DBLINK          SP:P00909
PRODUCT-TYPE    P
SYNONYM         foo
SYNONYM         foo2
GENE-COMMENT    f453; 99 pct identical to TRPC_ECOLI
                     SW: P00909

;;  The following shows how information about multiple functions
    of a
;;  protein is supplied:

FUNCTION          N-(5-phosphoribosyl) anthranilate isomerase
EC                5.3.1.24
FUNCTION-SYNONYM  phosphoribosyl anthranilate isomerase
FUNCTION-COMMENT  Amino acid biosynthesis: Tryptophan (3rd step)
FUNCTION          indole-3-glycerolphosphate synthetase
EC                4.1.1.48
FUNCTION-COMMENT  Amino acid biosynthesis: Tryptophan (4th step)
//
```

1.2.3 GenBank File Format

The information the Predictor extracts from the GenBank file comes from its feature table, which is fully defined in the document *The DDBJ/EMBL/GenBank Feature Table Definition* [3]. To allow the user to provide additional information not currently supported by Genbank format, PathoLogic supports a few additional feature qualifiers: *product_comment*, *EC_number*, and *alt_name*; see below for their descriptions.

A common omission from GenBank files is the following line which must precede the genes and other features in each GenBank file. Be sure to include it in yours. Note that 13 spaces follow the word "FEATURES".

- FEATURES[13 spaces]Location/Qualifiers

In our experience, most genome centers do not prepare their Genbank files in a manner that is fully consistent with the specification of Genbank file format. For example, they sometimes put a given piece of information in the wrong field. We now describe the fields within the Genbank feature table that PathoLogic uses, and what information PathoLogic expects to find in those fields. We also indicate the degree to which these fields are required by PathoLogic.
The accepted features are:

- CDS. A gene that codes for a protein (or which represents a hypothetical ORF).
- tRNA. A gene that codes for a tRNA.
- rRNA. A gene that codes for a ribosomal RNA.
- misc_RNA. A gene that codes for some other RNA, such as can be part of various

ribonuclear protein complexes.

The accepted qualifiers for any of the above features are:

- /gene. The official gene symbol used for the gene. In the absence of a gene symbol, a mnemonic shorthand name is suggested. The value of this qualifier will become the common name of the gene. Some annotation files contain the value desired here in the /product qualifier, enclosed in parentheses, at the end of the string (e.g., /product="ATP-dependent protease LA (lon-1)").
  Required.
- /EC_number. If the annotation effort has yielded an enzyme assignment with EC number(s), it (they) should be indicated in the standard way of using four numbers (and/or dashes), separated by three dots.
  Recommended.
- /product_comment *. An additional comment can be put here. It will end up in the comment slot of the protein frame.
  Optional.
- /locus_tag. The unique identifier for this gene. Its recommended value is the unique ID assigned by the group that sequenced the genome. For example, The Institute for Genomic Research typically assigns IDs of the form "GSnnnn," where G and S are the first letters of the genus and species names, respectively, and nnnn is a number unique to each gene. An example unique ID assigned by TIGR to Helicobacter pylori is "HP0023" and to Caulobacter crescentus is "CC1087". If a value for the /locus_tag qualifier is supplied, it will become the frame ID for the gene. If no /locus_tag qualifier is supplied, Pathway Tools will use the value of the /label qualifier if present, and otherwise Pathway Tools will generate an arbitrary unique ID for each gene.
  Highly Recommended.
- /gene_comment *. An additional comment can be put here. It will end up in the comment slot of the gene frame.
  Optional.
- /alt_name *. Additional synonyms under which a gene may be known can be indicated herewith.
  Optional
- /product. The assigned function of this protein. It will become the common name of the protein. The string "ORF" should be used if the function is unknown. Only real functional assignments should be entered here. This means that values such as "similar to..." or "putative..." should be changed to "ORF".
  Required.
- /pseudo. Indicates that the CDS is a pseudogene.
  Optional

Some of the feature qualifiers above (e.g., *alt_name* and *EC_number*) may be used several times per gene record (a single value per line), if several values need to be specified. The "*" indicates that the qualifier is not an official GenBank/EMBL/DDBJ feature qualifier.

Here is an example:

```
FEATURES     Location/Qualifiers

CDS          complement(3480761..3481768)
             /gene="gap"
             /EC_number="1.2.1.12"
             /EC_number="1.2.1.13"
             /product_comment="glycolysis"
             /label="b1779"
             /gene_comment="o331; 100 pct identical to
             G3P1_ECOLI SW: P06977;
             CG Site No. 718; alternate name gad
             /alt_name="gad"
             /alt_name="foo"
             /product="glyceraldehyde-3-phosphate
             dehydro-genase"
                    . . .
```

1.2.4 Directory Structure for a PGDB

During the course of PGDB construction, PathoLogic creates a directory tree within the **aic-export** directory tree of your Pathway Tools distribution to store all information about that PGDB. For example, this directory tree contains reports created by the PathoLogic parser and the PathoLogic enzyme name matcher, it contains the sequence files for the organism, and, for PGDBs stored in files, it contains a file containing the complete PGDB.

Upon completion of the build of the new PGDB, the directory tree created by PathoLogic has the following structure and files:

aic-export/ecocyc/ORGIDcyc/
- default-version
- VERSION/
  - input/
    - genetic-elements.dat
    - organism.dat
    - organism-init.dat
    - <chromosome.fsa>
    - <chromosome.{pf,gbk}>
  - reports/
    - name-matching-report.txt
    - trial-parse-report.txt
  - data/
    - <orgid_CHROMOSOMEID.fsa>
    - overview.graph
  - kb/
    - ORGIDbase.ocelot

where ORGID and VERSION are the Organism ID and version number the user entered into the *Input Project Information Window* (see Figure 1-2). The files denoted by <chromosome.fsa> and <chromosome.pf,gbk> are the sequence and annotation files, respectively. The files denoted by <orgid_CHROMOSOMEID.fsa> are copies of the sequence files. The file ORGIDbase.ocelot is the file containing the complete PGDB for PGDBs stored in files.

PathoLogic creates the file "default-version", putting inside it only the version number that you enterted into the *Input Project Information Window*. If you later create any additional versions of the PGDB, PathoLogic updates the "default-version" file each time. If you decide to abandon the latest version of a database, then you can manually edit the "default-version" file to specify the version you want Pathway Tools to use.

1.3 CREATING A PATHWAY/GENOME DATABASE

This section describes each PathoLogic operation in detail. The user will typically execute the PathoLogic operations described in this section in the order we describe, although there is some flexibility in this order.

1.3.1 Invoke PathoLogic

Invoke PathoLogic through the Pathway/Genome Navigator menu using the command **Tools -> PathoLogic**. PathoLogic creates a separate new window that runs in a separate process so that PathoLogic operations can run in parallel with Navigator operations. The PathoLogic window is shown in Figure 1-1. PathoLogic commands are organized into several menus that roughly correspond to the order in which the commands are used.

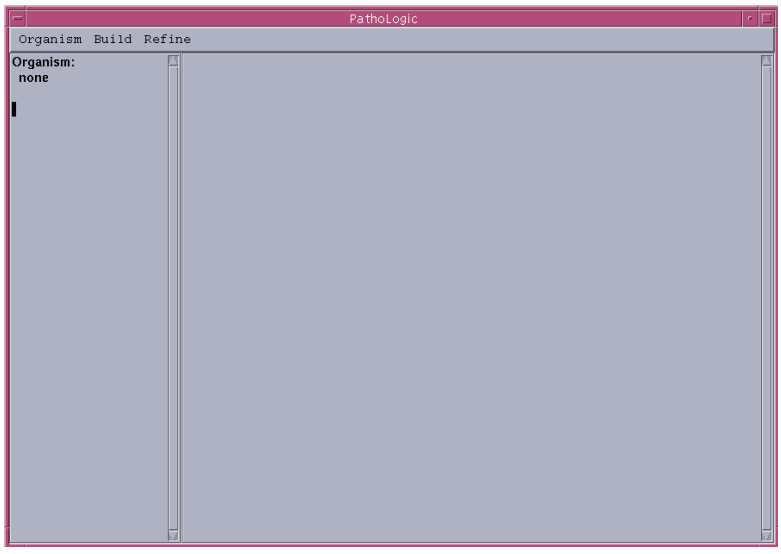


Figure 1-1 PathoLogic Window

1.3.2 Create New Organism

Begin creation of a new PGDB with the command **Organism -> Create New**. This command creates the PathoLogic Project Information dialog shown in Figure 1-2. Enter information about the organism for which you are creating a new PGDB. The Organism ID should be a short unique identifier for the PGDB itself (unique with respect to the other organism identifiers in directory **`aic-export/ecocyc/`**). Most of the information requested in this dialog should be self-explanatory, but we provide descriptions of some of these fields here. Most of the information requested in this dialog can be altered at a later time (such as the PGDB authors), but this dialog is the most convenient place to enter the information.

The information that should be supplied in the fields in this dialog is as follows:

- Organism/Project ID: A short mnemonic for the name of the organism that will be used as a unique identifier for the organism. This ID should be unique with respect to all other PGDBs. This ID will be used to construct the name of the directory tree containing the files for this PGDB. For example, "bsub" might be used as the identifier for B. subtilis.
- Database Name: The name to use for this DB when communicating with the user. Examples: "BsubCyc", "BsubtilisDB".
- Full Species Name: The full name of the organism.
- Abbreviated Species Name: An abbreviated name for the organism.
- Strain: The name of the strain of the organism that the PGDB describes.
- NCBI taxonomy ID: The ID for the organism from the NCBI taxonomy database, http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=Taxonomy
- Phylogenetic Classification: Specifies which of several main evolutionary branches the organism resides in. Of the three domains, Archaea, Bacteria, and Eukaryota, two of them, Bacteria and Eukaryota, are subdivided into several important subdomains. These are not complete subdivisions. If your organism is a bacterium, but does not fall under any of the 10 branches listed, you should classify it as simply Bacteria. To determine which branch your organism falls under, search for it in the NCBI taxonomy database listed above.
- Default Codon Table: Specifies which genetic code this organism uses.
- DB Storage Type: Specifies where the new PGDB will reside. A storage type of "file" means the PGDB will reside in a flat file. This approach is much easier to use, but saving of DB updates will be slower, and multiple users cannot edit a file PGDB simultaneously. A storage type of "oracle" or "mysql" means the PGDB will reside in an Oracle or MySQL database, respectively. This approach requires configuration of the Oracle or MySQL DB. It is always possible to transition a PGDB from one storage type to another. We recommend starting with a storage type of file and transitioning to Oracle or MySQL later if necessary.
- Authors: A list of the authors of the PGDB, and their institutions.
- Citations: A list of Medline UIDs for citations for the PGDB, such as the publication of the full genome sequence of the organism, if available.
- Project Home Page URL: The URL of a home page for this PGDB, if any.
- Project Primary Contact Email: The email address of a primary contact person for this PGDB, to whom requested DB corrections should be sent.
- Copyright String: A copyright notice (in HTML format) for this PGDB, if any is desired.

- Footer citation for web pages: A string (in HTML format) specifying a publication that users of the PGDB should cite.

The data collected in this dialog is saved in two files, called **`organism.dat`** and **`organism-init.dat, in directory aic-export/ecocyc/ORGIDcyc/VERSION/input/`**. In order to run PathoLogic, the user must have write access to the **`aic-export`** directory tree. Comments in these two files describe the conditions under which the files may be edited by the user.

The remainder of this section describes the process of creating a new database.

The user is required to enter values for *Organism ID* and *Full Species Name*. The value entered for *Organism ID* is automatically converted to upper case by the Predictor and is restricted to contain any combination of alphanumeric characters and underscores ("_"). This restriction is due to the fact that the Predictor uses the *Organism ID* to generate the name of the top level directory. It is recommended that the user choose a short (single word) mnemonic name as the organism name (e.g., "ctra" for *Chlamydia trachomatis*, "bsub" for *Bacillus subtilis*). The full species name (e.g., *Homo sapiens*) is required since it is needed in order for the GenBank queries from the gene mode to work.

This operation may take a few minutes to finish because it is initializing the schema (class hierarchy) of the new PGDB, and saving this initialized form of the PGDB.

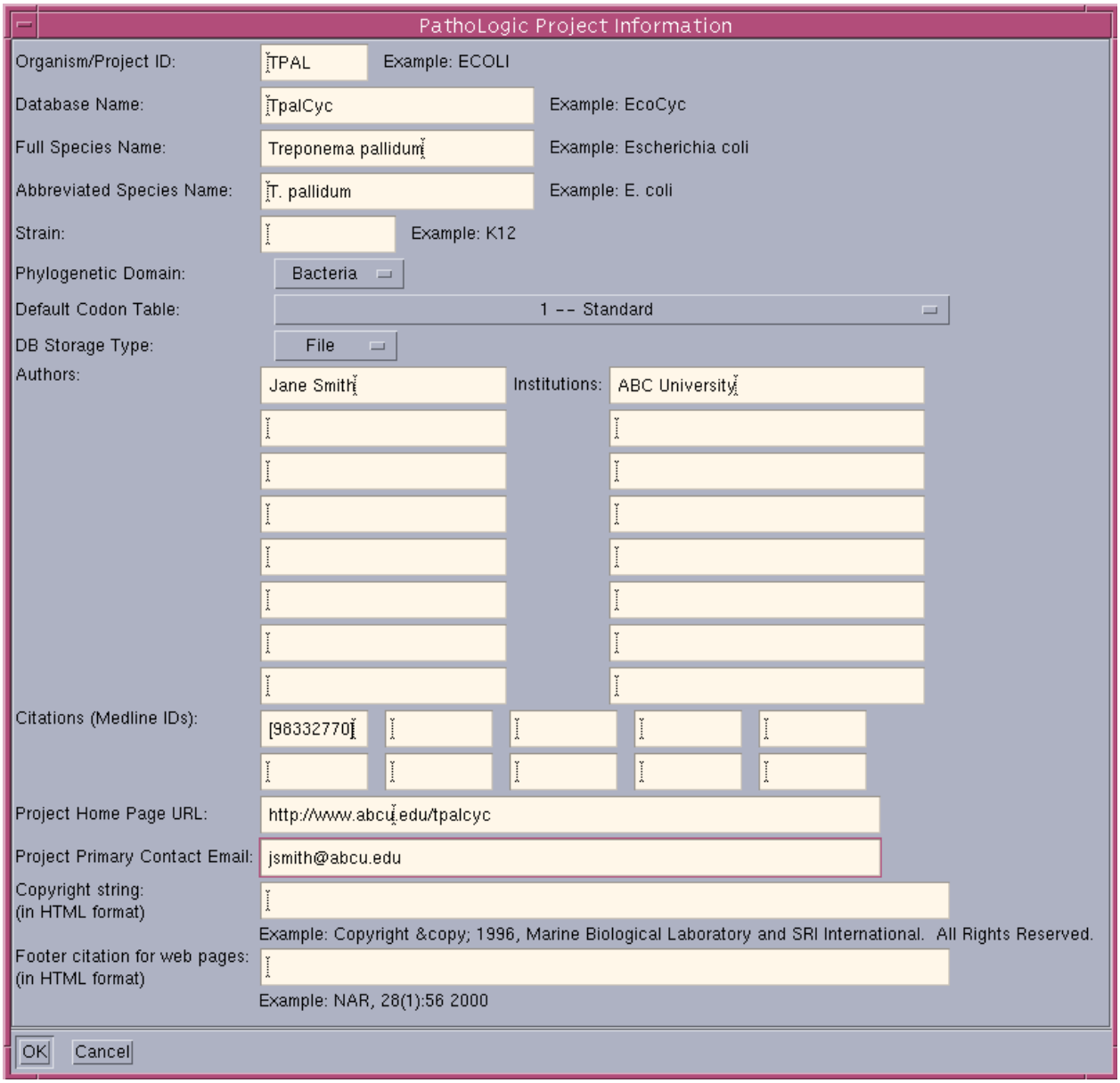


Figure 1-2 Input Project Information Window (after user input)

1.3.3 Create genetic-elements.dat File

Once a new PGDB has been initialized, you must create a file called **`aic-export/ecocyc/ORGIDcyc/VERSION/input/genetic-elements.dat`** to describe the one or more chromosomes or plasmids of this organism. PathoLogic will have created a file called **`sample-genetic-elements.dat`** in this same directory, which you can copy and edit. The format of this file is defined in "File genetic-elements.dat" in section 1.2.1. Creation of this file is essential because it provides pointers to all of the other input files required by PathoLogic. You cannot proceed further with the operation of PathoLogic without creating this file.

1.3.4 Specify Reference PGDB

In some cases, you may have already created or downloaded a PGDB for a different, related organism. This other PGDB, if it has been curated, may contain reactions or pathways not present in MetaCyc, and you may wish to see if these additional pathways or reactions can be predicted in your new PGDB. To specify one or more PGDBs to use as references for pathway prediction, in addition to MetaCyc, invoke the command **Organism->Specify Reference PGDB(s).** You will be presented with a list of available organism PGDBs. Select any you wish to use as a reference and click OK. Once you save the database, this selection will remain in force for all subsequent attempts to infer reactions and pathways until you change it again.

When a PGDB is specified as a reference, all pathways and reactions that are in that PGDB and are a) not already in MetaCyc, and b) supported by evidence other than purely computational evidence will be imported into MetaCyc before enzyme-to-reaction mapping and pathway prediction is performed. The changes will remain a part of MetaCyc for the remainder of the user's session, but will not persist for future sessions (though they will be repeated again if further PathoLogic work is done in those sessions). In addition, any enzyme names from the reference PGDB will supplement the enzyme names in MetaCyc for the purpose of matching enzyme names to reactions.

Note that this step is entirely optional – most users will find that they do not require any other reference PGDB besides MetaCyc. None of the organism PGDBs that SRI distributes as part of Pathway Tools contain additional reactions or pathways not in MetaCyc, so users would find little benefit in specifying them as reference PGDBs. This functionality is designed for users who have access to PGDBs that have been curated outside of SRI, whose pathways have not yet been incorporated back into MetaCyc.

1.3.5 Trial Parse

The **Build -> Trial Parse** operation allows the user to test whether the input files (see "Batch Mode" in section 1.6) can be properly parsed by PathoLogic. The purpose of this phase is to allow the user to detect and correct errors in the input file before the file has been used to populate the new PGDB. Malformed input files are much easier to correct at this early stage in processing than after manual refinement of the PGDB has begun.

When this command is invoked, a dialog window is created that allows the user to choose one or more files to be parsed. Click the **Parse** button to initiate parsing. You can run the parser as many times as you like. Click **Done** when you do not wish to perform any more parsing operations.

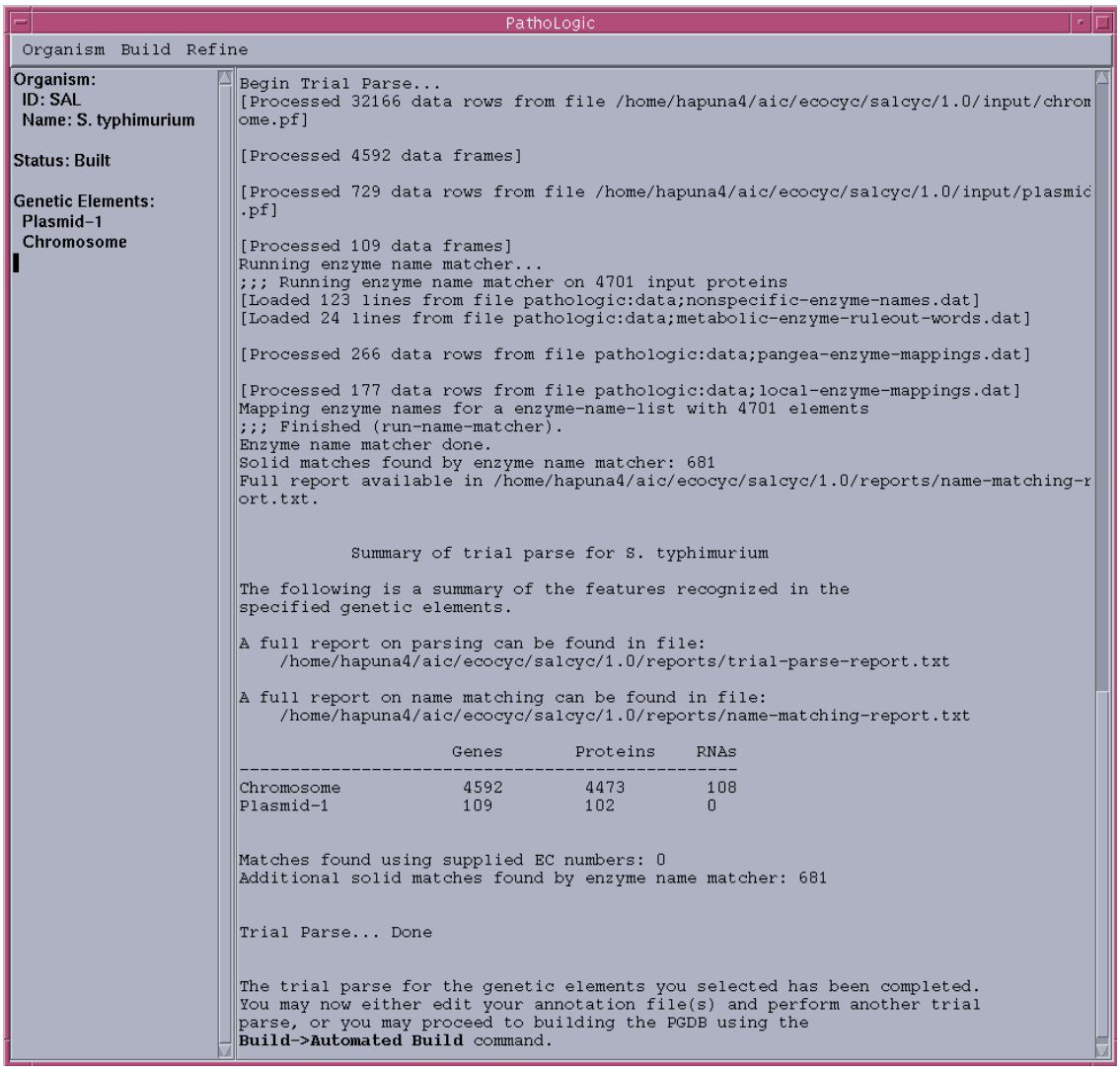


Figure 1-3 Trial Parse Output

The parser sends output to the main PathoLogic window that lists, for each genetic element, the number of genes and gene-products found in the corresponding annotation file by the parser. A more detailed summary of the data extracted by the parser will be written to a file whose name is given in the output. Sample output is shown in Figure 1-3.

Carefully compare the statistics for each file with the values you expect. Does the number of gene-IDs found in each file match the number of genes you know to be in each file? Does the number of startbase and endbase values match the number of genes? The number of gene-product-types should be the sum of gene-Proteins, gene-tRNAs, gene-rRNAs, gene-snRNAs, and gene-miscRNAs.

Anomalies in these statistics are probably due to formatting problems with the files. Has the Tab character been used as the separator between the attribute names and values in every line? Has the correct attribute name been used in every case? If errors in the input file are found, correct the files and repeat the Trial Parse until no more anomalies are found.

1.3.6 Build Pathway/Genome Database

By the time of the automated build, you should have already resolved all syntactic problems with the input files, and should have manually added as many enzyme-reaction correspondences as possible for unmatched enzyme names, as described in "Matching of Enzymes to Reactants" in section 1.1.2.1.

The command **Build -> Automated Build** is the main phase of PathoLogic operation. During this step, PathoLogic re-parses the input files, and it creates DB objects for each of the chromosome, genes, and gene products of the subject organism. It also links the products of those genes to reaction objects for as many enzyme-reaction associations as can be inferred automatically.

Note: This step takes several minutes to complete.

The parser again generates output that lists, for each genetic element, the number of genes and gene-products found in the corresponding annotation file by the parser. A more detailed summary of the data extracted by the parser is written to a file whose name is provided.

If you do not notice any obvious errors during the automated build, then save the newly built PGDB with the command **Organism -> Save DB**. Saving of the database will take some minutes.

Should you wish to not save the results of the build so that you can modify the input files and rerun the automated build, select **Organism -> Revert DB**, which will result in erasing from memory everything that took place since the PGDB was initialized. The build and revert operations may be performed as often as deemed necessary.

1.3.7 Select Organism

If you are creating several new PGDBs simultaneously, you can switch between them using the command **Organism -> Select**.

1.3.8 PGDB Housekeeping Tasks

The following commands are found in the **Organism** menu.

1.3.8.1 Reinitialize DB

If you want to delete a PGDB and then reinitialize it so that another automated build can be performed, use the command **Organism -> Reinitialize DB**. This command should be used with care since it will delete not only the results of the automated build, but also any manual changes you made to the PGDB during the manual polishing phase.

1.3.8.2 Convert File DB to Oracle or MySQL DB

These two commands converts a PGDB from one using a file for storage to one using an Oracle or MySQL database for storage. In order to use Oracle or MySQL, you must have an Oracle or MySQL RDBMS properly installed and configured at your site, and you must have created a database that has been initialized with the Pathway Tools schema. Instructions for setting up Oracle or MySQL are beyond the scope of this document. For information on installing the Pathway Tools schema, contact ptools-support@ai.sri.com.

1.3.8.3 Backup DB to File

For PGDBs that use an Oracle or MySQL database for storage, this command can be used to create a file version of the PGDB as a backup.

1.3.8.4 New Version

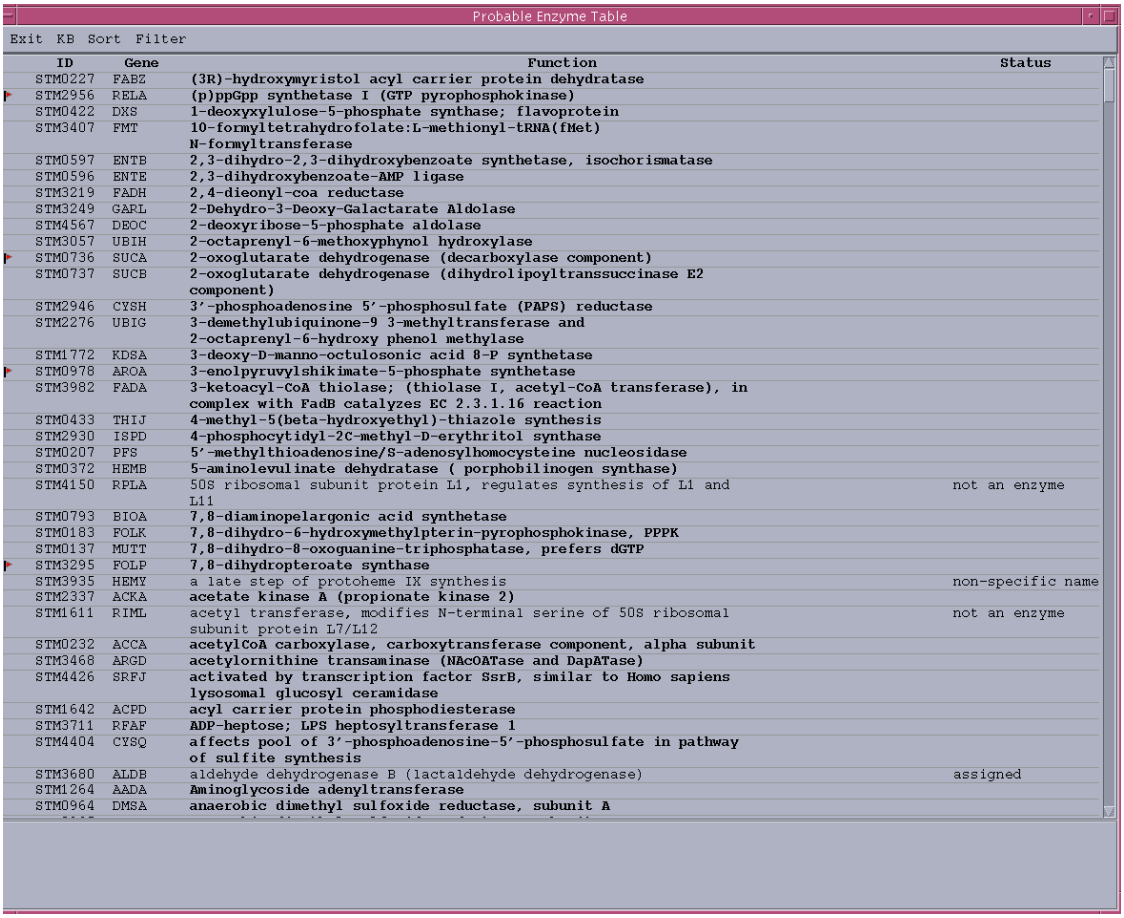
You may wish to freeze a stable version of a PGDB for public use while you continue to edit a development version. This command freezes the current development version by making a copy of the current version. You get to designate the new version number.

1.4 REFINING THE PGDB

This section describes the *Refinement* phase of PathoLogic operation, which has several parts. Refinement involves manual creation of relationships within the PGDB.

1.4.1 Assign Probable Enzymes

The command **Refine -> Assign Probable Enzymes** is used to make manual enzyme-to-reaction assignments that the automated name matching procedure failed to find. It brings up a table of genes and their function names that were classed as probable enzymes during the name-matching process but for which matches were not found. An example table is shown in Figure 1-4. Not all of the function names listed will be able to be assigned to reactions. Some may not be metabolic enzymes at all. Others may be too non-specific to be able to be assigned to a particular reaction or set of reactions. In addition to making assignments to reactions, this table allows the user to mark certain gene products as unassignable so as to avoid wasting time considering these proteins in the future (these proteins will continue to appear in the table for the time being, with an appropriate notation in the Status field, but the next time this command is invoked they will no longer be considered probable enzymes, so will no longer appear in the table). The user can also flag certain proteins for future consideration, and record comments describing, for example, progress made toward identifying the reaction, problems encountered, and the rationale behind particular decisions. If a comment has been recorded for a particular protein, then passing the mouse over that row in the table will cause the comment to be displayed in the lower pane. This enables the user to assess at a glance the status of any particular protein.



ID	Gene	Function	Status
STM0227	FABZ	(3R)-hydroxymyristol acyl carrier protein dehydratase	
STM2956	RELA	(p)ppGpp synthetase I (GTP pyrophosphokinase)	
STM0422	DXS	1-deoxyxylulose-5-phosphate synthase; flavoprotein	
STM3407	FMT	10-formyltetrahydrofolate:L-methionyl-tRNA(fMet) N-formyltransferase	
STM0597	ENTB	2,3-dihydro-2,3-dihydroxybenzoate synthetase, isochorismatase	
STM0596	ENTE	2,3-dihydroxybenzoate-AMP ligase	
STM3219	FADH	2,4-dieonyl-coa reductase	
STM3249	GARL	2-Dehydro-3-Deoxy-Galactarate Aldolase	
STM4567	DEOC	2-deoxyribose-5-phosphate aldolase	
STM3057	UBIH	2-octaprenyl-6-methoxyphynol hydroxylase	
STM0736	SUCA	2-oxoglutarate dehydrogenase (decarboxylase component)	
STM0737	SUCB	2-oxoglutarate dehydrogenase (dihydrolipoyltranssuccinase E2 component)	
STM2946	CYSH	3'-phosphoadenosine 5'-phosphosulfate (PAPS) reductase	
STM2276	UBIG	3-demethylubiquinone-9 3-methyltransferase and 2-octaprenyl-6-hydroxy phenol methylase	
STM1772	KDSA	3-deoxy-D-manno-octulosonic acid 8-P synthetase	
STM0978	AROA	3-enolpyruvylshikimate-5-phosphate synthetase	
STM3982	FADA	3-ketoacyl-CoA thiolase; (thiolase I, acetyl-CoA transferase), in complex with FadB catalyzes EC 2.3.1.16 reaction	
STM0433	THIJ	4-methyl-5(beta-hydroxyethyl)-thiazole synthesis	
STM2930	ISPD	4-phosphocytidyl-2C-methyl-D-erythritol synthase	
STM0207	PFS	5'-methylthioadenosine/S-adenosylhomocysteine nucleosidase	
STM0372	HEMB	5-aminolevulinate dehydratase (porphobilinogen synthase)	
STM4150	RPLA	50S ribosomal subunit protein L1, regulates synthesis of L1 and L11	not an enzyme
STM0793	BIOA	7,8-diaminopelargonic acid synthetase	
STM0183	FOLK	7,8-dihydro-6-hydroxymethylpterin-pyrophosphokinase, PPPK	
STM0137	MUTT	7,8-dihydro-8-oxoguanine-triphosphatase, prefers dGTP	
STM3295	FOLP	7,8-dihydropteroate synthase	
STM3935	HEMY	a late step of protoheme IX synthesis	non-specific name
STM2337	ACKA	acetate kinase A (propionate kinase 2)	
STM1611	RIML	acetyl transferase, modifies N-terminal serine of 50S ribosomal subunit protein L7/L12	not an enzyme
STM0232	ACCA	acetylCoA carboxylase, carboxytransferase component, alpha subunit	
STM3468	ARGD	acetylornithine transaminase (NAcOATase and DapATase)	
STM4426	SRFJ	activated by transcription factor SsrB, similar to Homo sapiens lysosomal glucosyl ceramidase	
STM1642	ACPD	acyl carrier protein phosphodiesterase	
STM3711	RFAF	ADP-heptose; LPS heptosyltransferase 1	
STM4404	CYSQ	affects pool of 3'-phosphoadenosine-5'-phosphosulfate in pathway of sulfite synthesis	
STM3680	ALDB	aldehyde dehydrogenase B (lactaldehyde dehydrogenase)	assigned
STM1264	AADA	Aminoglycoside adenyltransferase	
STM0964	DMSA	anaerobic dimethyl sulfoxide reductase, subunit A	

Figure 1-4 Table of Unresolved Probable Enzymes

While working with this table, the user can save the PGDB or revert to the previously saved version at any time by invoking the **DB -> Save** or **DB -> Revert** commands. The user can change the order in which the rows are presented by specifying a different **Sort** criterion. By default, the rows are ordered by function name, but they can also be ordered by Gene ID, name, or chromosomal location. The user can also choose whether flagged items should be listed at the beginning or not. If there are many proteins that have been classified as probable enzymes, the table can be very large. Using the **Filter** command, the user can choose to include, for example, only flagged items, or only function names that contain a particular substring. Alternatively, if the list of proteins in the table is not inclusive enough (e.g. if some of the proteins not identified as probable enzymes can in fact be linked to reactions), the user can use the **Filter** command to select from the full list of unassigned proteins.

To assign a protein to a reaction, or record the other types of information described above, the user should select the row of the table corresponding to the particular function. This will bring up the Probable Enzyme Status Dialog, illustrated in Figure 1-5. This dialog lists the function name and synonyms, if any. The user can edit these or supply additional synonyms by clicking on the button. Besides assigning the protein to one or more reactions, the user can also enter a comment, as described in the previous paragraph, can flag (or unflag) the protein, or remove it from future consideration for one of the reasons provided.

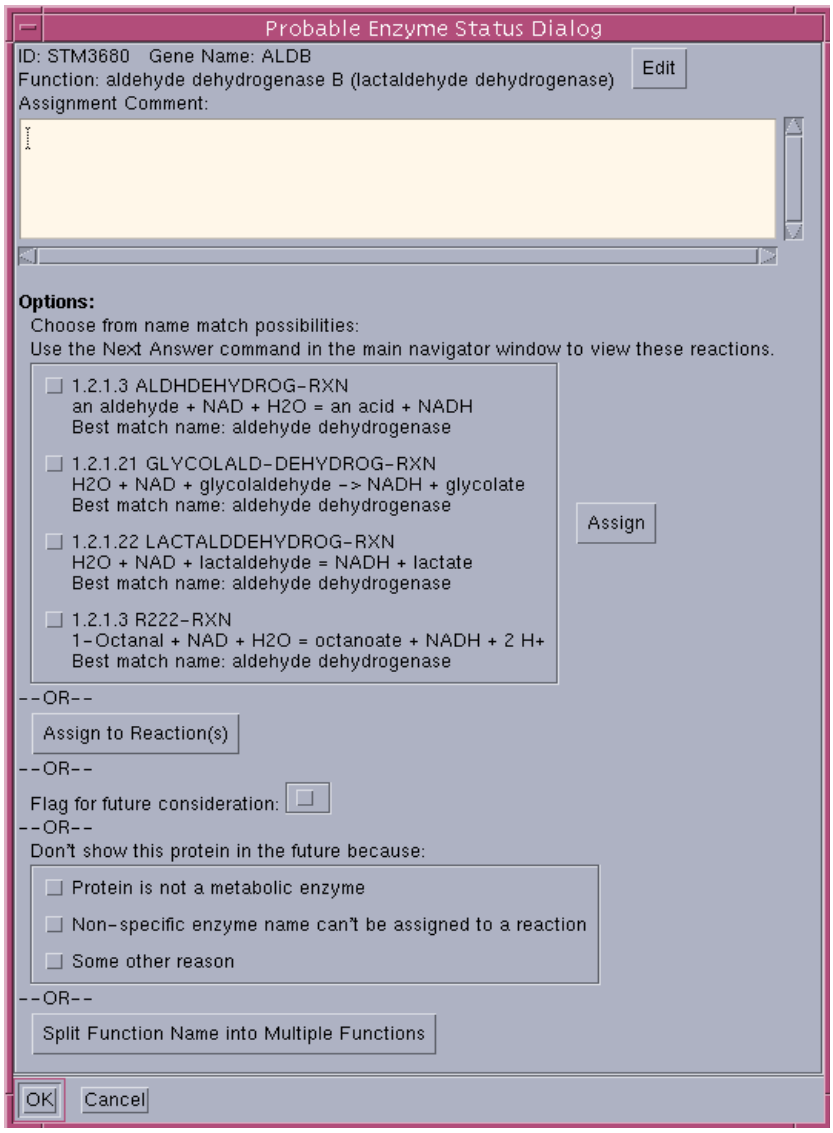


Figure 1-5 Dialog for assigning or classifying a probable enzyme

For some proteins, including the one in the figure, a list of suggested reactions is provided. These are generally reactions whose enzyme names bear some similarity to the supplied function name (or one of its synonyms), but not enough for the name matcher to have made a definitive assignment. In cases where there are no such reactions, or too many such reactions (which is often the case when a function name is non-specific), this option is not provided. The user should conduct research, as described in "Matching of Enzymes to Reactants" in section 1.1.2.1, to attempt to discern which reaction or reactions correspond to the function name. Such reactions could be among those suggested, in which case the user should select them and click the **Assign** button, or they could be completely different reactions, in which case the user should click the **Assign to Reaction(s)** button. Either of these options will bring up a new dialog for assigning a protein to one or more reactions, shown in Figure 1-6. If reactions were selected from the list provided, then they will already be filled in this dialog, otherwise the user must type in the reaction ID or EC number. If the reaction to be assigned to does not yet exist in MetaCyc, the user should first create it using the normal editing tools from the Pathway/Genome Navigator

window, and then supply the new ID here. Multiple reactions can be assigned using a comma-separated list, as shown in the figure. Additional information, such as synonyms, a comment, and citations can also be provided as part of this dialogue.

Some enzymes catalyze multiple reactions. If all such reactions go by the same function name (for example, if the enzyme is non-specific and can accept multiple substrates), then the reactions should be specified together in the above dialogue, as illustrated. Often, however, each function has its own name and refers to a different catalytic activity. An example is the trpC gene product in *E. coli*, which performs both the phosphoribosyl anthranilate isomerase and the indole-3-glycerol phosphate synthase functions. The PathoLogic file format specifies that each of these functions should be specified on its own line (see "The PathoLogic File Format" in section 1.2.2, but some annotators may disregard this and supply a single function name that refers to both functions. When this happens, use the **Split Function Name into Multiple Functions** button in the Probable Enzyme Status Dialog to specify multiple function names for the protein. Each function name can then be assigned to a reaction individually.

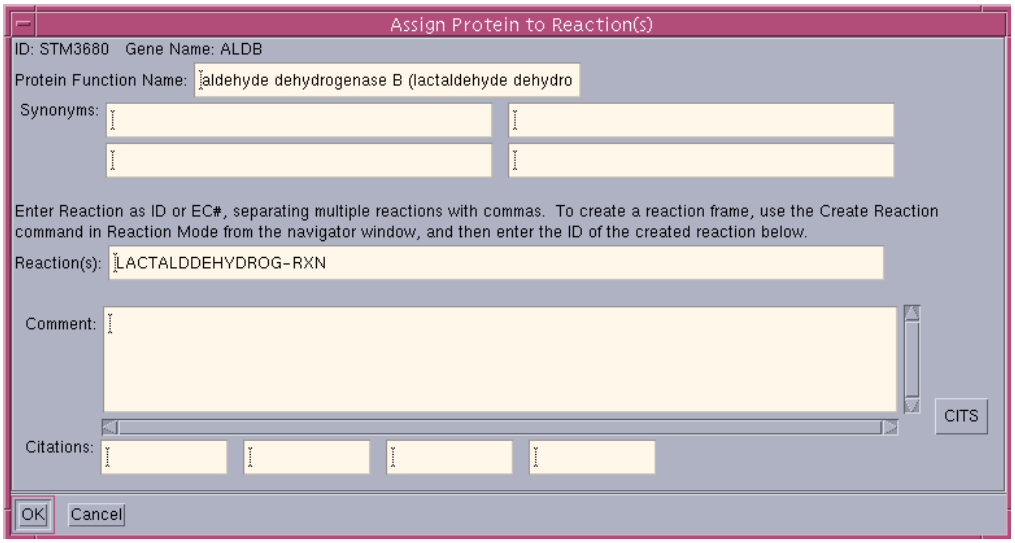


Figure 1-6 Dialog for assigning a protein to one or more reactions

1.4.2 Re-Run Name Matcher

If new names or synonyms have been added to MetaCyc or to one of the name files described in "Matching of Enzymes to Reactants" in section 1.1.2.1, then the user may wish to re-run the name matcher after the PGDB has been built, to match those additional names. This operation does not alter existing reaction assignments, but may make new assignments.

1.4.3 Rescore pathways

The pathway scoring algorithm runs as part of the automated build procedure. However, additional manual steps, such as assigning ambiguous proteins, re-running the name matcher, or manually adding, deleting or changing protein assignments using the editing tools may change the set of pathways that would be inferred to be present in the organism. Rescoring pathways

imports any pathways for which there is new evidence, re-imports any pathways which may have changed in MetaCyc, and deletes pathways (and their reactions and compounds where appropriate) which were previously inferred but are now determined not to be present. This command should be invoked after the kinds of changes described above. However, if the user has manually deleted or imported any pathways, those changes will be lost (e.g. deleted pathways may be re-imported), so the user should attempt to complete all editing of protein assignments and invoke this command before manually importing or removing any pathways.

1.4.4 Create Protein Complexes

A functional enzyme is often composed of several polypeptides (subunits) that aggregate to form a protein complex. To represent this biological situation faithfully, we must create a new PGDB object for the protein complex, and link that object to the PGDB object that represents the reaction that the enzyme catalyzes. However, the PGDB currently links the object for each polypeptide to the reaction it catalyzes, implying that each polypeptide is a functional isozyme. The PathoLogic protein-complex building tool (see Figure 1-7), invoked by **Refine -> Create Protein Complexes,** allows the user to specify new complexes that should be built from monomer subunits, and automatically unlinks the monomers from reactions, and links the new protein complexes to appropriate reactions.

The protein-complex building tool iteratively considers every reaction in the PGDB that has more than one enzyme connected to it via an enzymatic-reaction frame, meaning that more than one protein had a name or an EC number that corresponds to that reaction. These proteins must either be isozymes (separate proteins that catalyze the same activity), or subunits of an enzyme complex, or subunits of multiple isozymes for this reaction. For each such reaction, the tool groups together all polypeptides linked to that reaction and presents them to the user. The user scans the groupings to identify those polypeptides that, based on their names, appear to represent subunits of enzyme complexes. To aid the user in deciding which subunits might belong within one complex, the Navigator window (which may be hidden behind other windows) displays the MetaCyc version of that reaction, allowing the user to easily inspect the corresponding enzymes in MetaCyc.

The user can avoid creating a complex for a group of monomers by clicking the **Skip** button. Or, the user can create one or more complexes. The user indicates which monomers should be assigned to new protein complexes by selecting monomers with the mouse from one of the lists presented. Additionally, the **Add current history item to Complex** button allows the user to add one or more other monomers in the PGDB to any complex, if the user is aware of additional such monomers. The user is asked to choose from among the items on the Navigator history list, which contains all of the PGDB objects the user has viewed in the recent past.

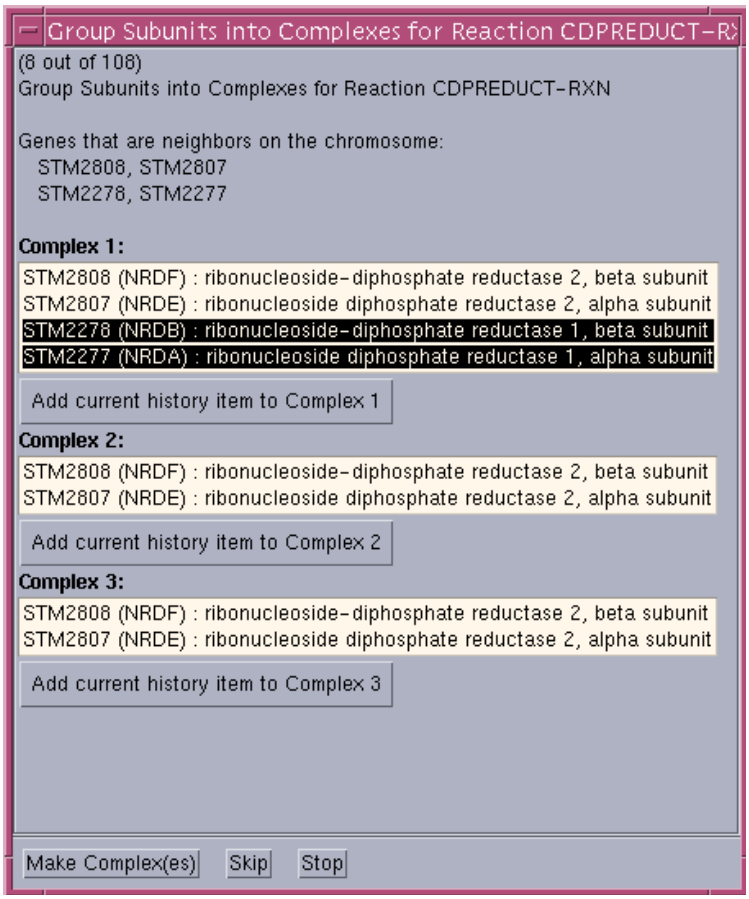


Figure 1-7 Creating Protein Complex(es). The user will create Complex 1 containing the alpha and beta subunits of ribonucleoside-diphosphate reductase 1. In this example, the user should also create Complex 2 by selecting the alpha and beta subunits of ribonucleo-diphosphate reductase 2 in the Complex 2 menu.

From the names of the monomers, the user must make a judgment as to which subunits make up a complex. Typically, the name of the enzyme would indicate both its function, and identify which subunit it is within a complex (e.g., “ribonucleoside-phosphate reductase 1, alpha subunit” and “ribonucleoside-phosphate reductase 1, beta subunit”).

To create one complex, use the mouse to select the one or more monomers that belong to the first protein complex in the menu under **Complex 1**. To create a second complex, select the one or more monomers that belong to the second complex in the menu under **Complex 2**.

Clicking on the **Make Complex(es)** button pops up a window (see Figure 1-8) for the user to specify coefficients of the subunit within the complex, in the (somewhat unlikely) case that these coefficients are known.

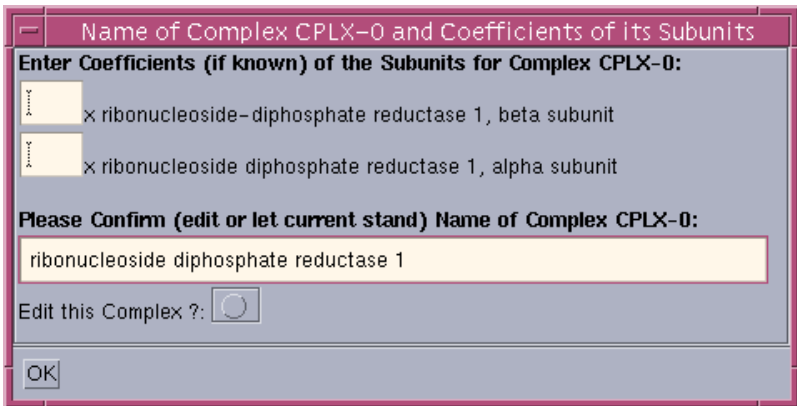


Figure 1-8 Specifying Complex Subunit Stoichiometries

In addition, the user may edit the name automatically chosen for the complex. Clicking on the **OK** button now automatically creates a PGDB object for the protein complex, and appropriately links it to other objects in the PGDB.

As shown in Figure 1-7, the user has the option to go on to the next potential protein complex (thus creating no new protein complex from the current set of monomers) by clicking the **Skip** button. Alternatively, if the user wants to stop working on complexes (and return to them later), they should click the **Stop** button.

1.4.5 Assign Modified Proteins

Some metabolic reactions use proteins as substrates, that is, proteins become modified by an enzyme-catalyzed reaction (such as phosphorylation or acetylation). The alternative chemically modified forms of a protein are represented in a PGDB as a *base form* and their *modified forms*. The base forms are the products of those genes that encode them. The modified proteins contain a slot, *Unmodified Form*, which points to (holds the ID of) the base forms. The *Modified Form* slot of the base forms gets the ID of the modified species.

When PathoLogic imports a new reaction into the PGDB that has a protein substrate (such as the Acyl Carrier Protein, or ACP), it does not know which gene product within the PGDB corresponds to ACP. The **Refine -> Assign Modified Proteins** command creates a dialog that allows the user to indicate which protein substrates within metabolic reactions correspond to which gene products within the PGDB. PathoLogic then creates a frame for the modified form of the protein and fills the appropriate slots for it. The user is presented with a pane (see Figure 1-9) containing a list of unmodified proteins that are referenced in metabolic reactions that PathoLogic has inferred to be present in the PGDB.

For each protein substrate, PathoLogic presents a list of its best guesses as to which PGDB gene product might correspond to that protein substrate, in a selector button. There is also a empty box where the user may enter a gene ID if none of the gene names in the list is deemed to be correct. Pressing the **Show** button instructs the Pathway/Genome Navigator to display information about the protein substrate that is currently selected using the selector button.

If the user is unsure about what gene ID to choose from the selection list and cannot find the appropriate ID by manual investigation, they should select *leave unconnected* from the list.

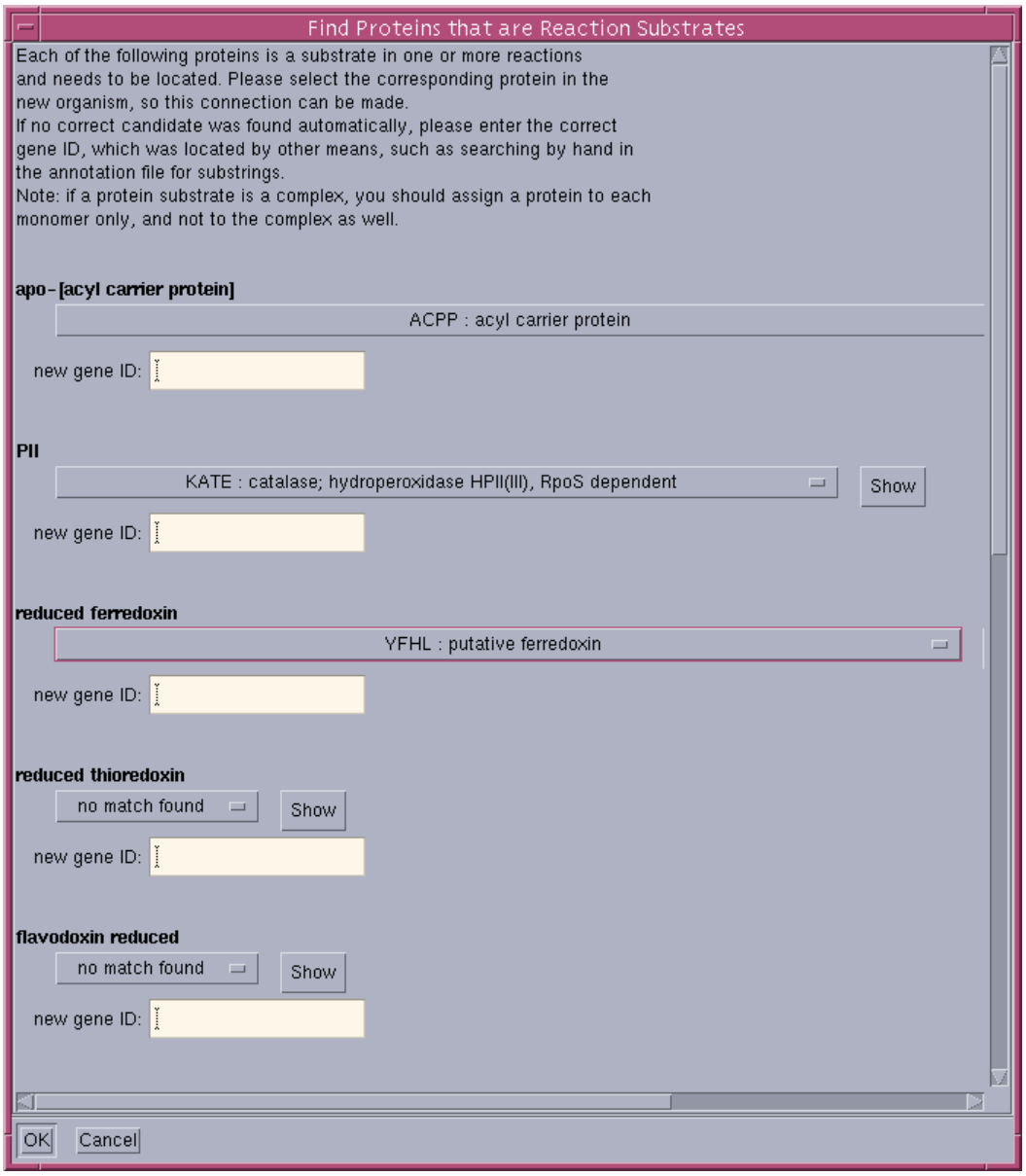


Figure 1-9 Find Proteins that are Reaction Substrates

1.4.6 Run Consistency Checker

Once all automated and manual changes have been made to the PGDB, it is advisable to run the consistency checker. This procedure will find violations of internal consistency constraints. In some cases, the violations may be fixed automatically, but in most cases, the violation will be reported in the PathoLogic window, and the user will have to decide whether or not to fix the

problem manually using the editing tools. Not all warnings will need to be fixed -- some are provided merely for informational purposes and left up to the user's discretion. The user should carefully examine the output from the consistency checker to see what if any changes need to be made.

1.4.7 Predict Transcription Units

PathoLogic can predict transcription units (TUs) on an entire genome or on genetic elements selected by the user. In this context, we also call them operons when comprising more than one gene. The predictor produces a set of predicted TUs that encompass all genes in the selected genetic element(s), and generates the corresponding transcription unit frames in the PGDB. Notice that the predicted TU frames will only include the genes that comprise the TU, as the predictor does not generate regulatory sites (transcription start sites, transcription factor binding sites, terminators).

The **Refine -> Predict transcription units** command creates a dialog (Figure 1-10) that allows the user to select one or more genetic elements from those in the current PGDB. You can click on **Select All** to predict TUs on the entire genome. Once genetic elements have been selected, click on **Predict TUs** to initiate the prediction process. Click **Done** when finished.

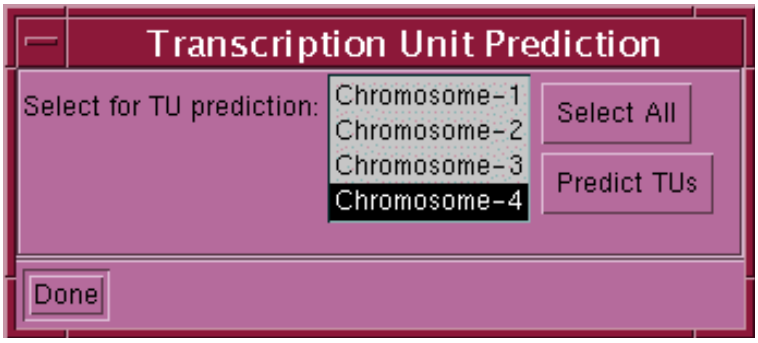


Figure 1-10 The Transcription Unit Prediction dialog

The TU predictor works by partitioning the genetic element into regions of contiguous genes that are transcribed in the same direction, called directons. A directon starts where two contiguous genes are transcribed in opposite directions (i.e., a directon boundary), and includes all contiguous genes transcribed in the same direction up to the next directon boundary (see Figure 1-11).

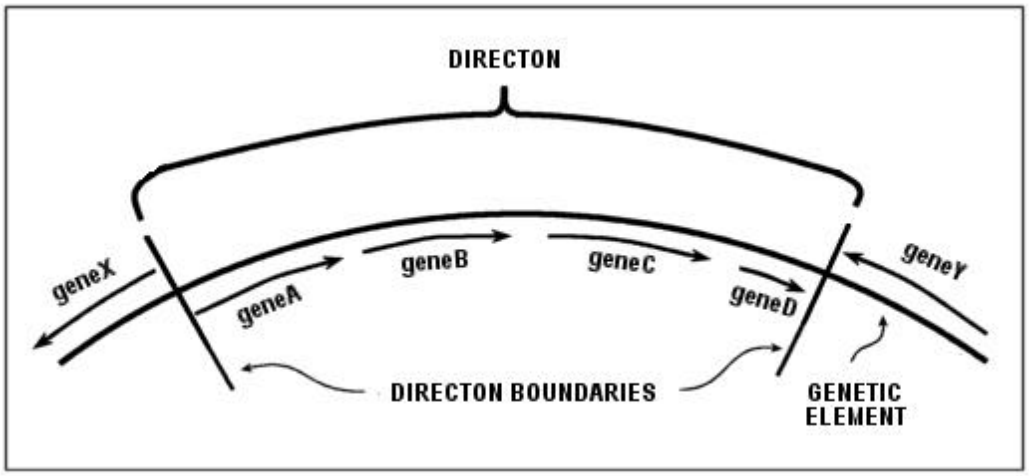


Figure 1-11 Directon. Notice the directon boundaries at the positions where transcription direction changes.

A special case is a single-gene directon, that is, a gene that is transcribed in one direction, whereas both its neighboring genes (up- and downstream) are transcribed in the opposite direction. Such a single-gene directon is in fact a single-gene TU. Thus, the TU predictor labels all single-gene directons as single gene TUs.

Longer directons are partitioned into contiguous gene pairs. The TU predictor then proceeds to classify each gene pair into two classes: Either the gene pair is contained within an operon (i.e., a WO, or "Within Operon" pair), or it corresponds to a boundary between adjacent TUs (i.e., a TUB or "Transcription Unit Boundary" pair). The predictor classifies gene pairs as WO or TUB based on six parameters:

1. The base-pair distance between the genes in a pair.
2. The genes' functional classes (if known).
3. Whether or not both genes' products belong to the same protein complex.
4. Whether or not both genes' products catalyze reactions in the same pathway.
5. Whether or not the product of one gene catalyzes a reaction in a pathway and the other gene's product is a transporter for a substrate in the same pathway.
6. Whether or not a gene up- or downstream from the studied gene pair (within the same directon) is related to one of the pair's genes as in 3, 4 or 5 above.

Once prediction is carried out on all gene pairs within a directon, the corresponding TUs are assembled out of WO and TUB pairs. Finally, TU frames are added to the PGDB for all predicted TUs, both single-gene directons and TUs predicted within longer directons.

All predicted TU frames will have appropriate Pathway Tools evidence codes identifying them as artificially inferred (EV-AINF) without manual oversight. If the genes from a predicted TU

correspond exactly to those of an existing, non-artificially-inferred TU, the software will just add the appropriate AINF evidence code to the evidence already in the evidence slot of the existing TU frame. A new TU frame is not generated in this last case.

The TU predictor deletes all previously predicted TUs for a genetic element, if any, before performing a new prediction. To determine whether a TU frame was predicted and should be deleted, the predictor checks the evidence slot for the EV-AINF evidence codes. In case the EV-AINF evidence code for a particular TU is accompanied by other evidence codes, the TU frame is not deleted: only the AINF evidence code is removed from the TU frame evidence slot. This prevents curated TU frames from being deleted in this phase, even if they have also been predicted by the PathoLogic TU predictor.

The predictor sends output to the main PathoLogic window indicating the prediction process' status, and the name of the file where the TU prediction report is going to be saved. The report includes some global statistics on the prediction, like number of TUs and operons predicted, and prints all directons on which a prediction was carried out, including the predicted and known or curated (if any) TUs. For each predicted TU, the name of the gene products of the component genes is also printed. When curated TUs exist in the PGDB, the report also includes some statistics on how well these known TUs where predicted by PathoLogic.

1.4.8 Transporter Identification Parser

The PathoLogic command **Refine -> Transport Identification Parser** (TIP) identifies proteins in the current PGDB that are likely to catalyze transport reactions. Such proteins are termed probable transporters. If sufficient evidence is present, TIP infers the transport reaction for the probable transporter, and creates a corresponding reaction frame in the current PGDB. In some cases TIP will also identify multi-subunit transporters, and create protein-complex frames for those transporters in the PGDB. All probable transporters are presented for review in the Transport Identification Parser interface, which allows acceptance, rejection, or editing of inferred transporters by the user.

To identify certain transporters correctly, operons/transcription units must first have been inferred for the PGDB (bacteria only). If not, those transporters that consist of systems of several proteins will be defined incorrectly, with a transport reaction being created for each component monomer of the transport system rather than one reaction for the entire system.

TIP operates by first inferring which proteins are transporters, and then creating reactions and enzymatic reaction frames for the highly probable transporters. The TIP GUI is then presented, allowing en masse acceptance (or rejection) of these transporters, or review and editing of each individual transporter.

1.4.8.1 Transport Inference

The major computational steps performed by TIP in identifying probable transporters and constructing their reactions, enzymatic reactions, and protein complexes are as follows:

1. Find candidate transporter proteins.
2. Filter candidates.
3. Identify substrate(s).
4. Assign an energy coupling to transporter; identify any cosubstrate.
5. Identify compartment of each substrate.
6. Identify transporters that consist of subunits; group into complexes.
7. Construct full reaction including substrates, compartment assignments, and coupling.
8. Construct enzymatic reaction linking each reaction with transport protein.

Each of the steps is described below.

1.4.8.1.1 Find candidate transporter proteins

TIP operates through a syntactic analysis of the common name (also referred to as annotation) field of each protein. It does not perform a sequence analysis, but attempts to understand the transport activity already assigned to proteins.

To identify candidate transporters, the common name is examined for certain words indicative of transport activity, such as “transporter”, “permease”, or “channel”. Overly long annotations (currently those longer than 12 words) are excluded.

1.4.8.1.2 Filter candidates

Candidate transporters are excluded from further consideration if their annotation exactly matches one of a number of generic annotations that contain no substrate information, such as XXX.

A candidate is also excluded if it contains a counterindicator word such as “regulator”. Such counterindicators increase the likelihood that the protein is related to a transporter rather than actually being a transporter.

1.4.8.1.3 Identify substrate(s)

The annotation of each candidate transporter is searched for the names of transported substrates. This is done by looking up each word pair and individual word, as well as variants thereof, in the MetaCyc KB. When a match is found, that compound is added to the list of possible substrates for that transporter. Processing details include:

- Excluding non-substrates that look like compounds such as "as" "be" "c" "i“.
- Name canonicalization, such as stripping plurals.
- Handling substrates with affixes such as "-transporting" and “-specific".

- Handling special ionic forms such as "cuprous", "ferric“, and "hydrogen".
- Detecting two-word substrates.
- Detecting multiple substrates separated by a delimiter such as “sodium/proton”.

In general, a protein can have multiple substrates. There are two cases of multiple substrates. The first is when two substrates have been identified, and one substrate is a cosubstrate facilitating transport of the other, primary, substrate. In this case, the transporter will have only one primary substrate, and one reaction.

In the second case, more than one substrate has been identified, but none are cosubstrates. In this case, the transporter has N substrates and N different reactions.

Note that certain multivalent elemental substrates such as iron and cobalt may be incorrectly identified as their elemental rather than their ionic form.

1.4.8.1.4 Assign an energy coupling to transporter; identify any cosubstrate

The energy coupling mechanism used by the transport protein is represented by a controlled vocabulary that indicates the subclass of Transport-Reactions in which the reaction associated with a transporter will be assigned. The terms in that controlled vocabulary are:

- UNKNOWN: Transporters of Unknown Mechanism
- MECHANICAL: Mechanically driven Transporters
- LIGHT : Light-driven Transporters
- ELECTRON: Electron-flow-driven Transporters
- DECARB: Decarboxylation-driven Transporters
- PTS: PEP-dependent Transporters
- ATP: ATP-driven Transporters
- SECONDARY: Secondary Transporters
- CHANNEL: Channel-type Facilitators

An energy coupling is assigned to each transporter as follows:

1. For a small number of substrates (exs: enterobactin, glycerol) a particular coupling mechanism is inferred.
2. Sometimes an annotation contains an indicator word for a coupling mechanism (exs: “atp” “channel” “permease”).
3. If neither of the above is successful, the coupling mechanism is UNKNOWN.

If the energy coupling is SECONDARY, an attempt is made to determine the cosubstrate that enables the transport. The cosubstrate is transported in the same direction (for symporters) or in the opposite direction (for antiporters) as the primary substrate(s). The cosubstrate is assigned as follows:

1. If H^+ (proton) has been identified as a substrate, use it as the cosubstrate; it is removed from the (primary) substrates of the transporter.
2. If Na^+ has been identified as a substrate, use it as the cosubstrate; it is removed from the (primary) substrates of the transporter.
3. If there are exactly two substrates, choose one arbitrarily as the cosubstrate; it is removed from the (primary) substrates of the transporter. The reaction that will be created is independent of this decision. An incorrect guess of the primary may, however, lead to some confusion when results are reviewed.
4. If only one substrate has been identified, H^+ is assumed to be the cosubstrate.

1.4.8.1.5 Identify the compartment of each substrate

Currently, it is assumed that primary substrates are transported either into or out of the cytosol. It is further assumed that they cross the membrane closest to the cytosol. Simple inferencing is done to determine this compartment. For gram-negative bacteria, this compartment is the periplasmic space. For gram-positive bacteria, it is the extracellular space.

Representation of cellular compartments in PGDBs is currently evolving, and may change in the near future.

The default assumption is that the primary substrate(s) are transported into the cytosol. If the indicator words “export”, “uptake”, or “efflux” are present in the annotation, transport in the opposite direction is inferred.

If the energy coupling of the transporter is SECONDARY, both the primary substrate and the cosubstrate change compartments. If the indicator word “symport” or “symporter” is present in the annotation, it is inferred that they move in the same direction, that is, they start in the same compartment. If the indicator word “antiport” or “antiporter” is present, it is inferred that they move in opposite directions.

1.4.8.1.6 Identify transporters that consist of subunits; group into complexes

Certain transporters consist of systems of proteins, rather than individual proteins. For such transporters, an attempt is made to identify the component proteins of the transporter and to group them into a complex. This complex is added to the KB, and any reactions and enzymatic reactions for the transporter are associated with the complex, rather than with its components.

Proteins are grouped into a complex if the following conditions all hold:

- The energy coupling is ATP or PTS.
- The same substrate has been identified for each.
- The genes of all such proteins share the same operon/transcription unit.

Note that if operons/transcription units have not been predicted for the PGDB, many monomer proteins are likely to be erroneously identified as individual transporters.

1.4.8.1.7 Construct full reaction

For each primary substrate of this transporter, either import a reaction (from EcoCyc) that conforms to its substrates and compartments thereof or create a new one.

The full transport reaction consists of the primary substrate plus any auxiliary substrates indicated by the coupling, as well as their compartments. For example, ATP transporters have ATP and water on the left side and ADP and phosphate on the right.

If a single transport reaction from EcoCyc matches the full reaction, it is imported. If multiple reactions match, they are retained for possible import later on but not yet imported. If no imports are found, a new reaction is created.

The semantics of matching require that each substrate be present in the reaction, and that its inferred compartment is identical. Since different organisms have different cellular compartments, reactions that are analogous but that do not match exactly with respect to compartments will not be imported.

Note that EcoCyc is used as the reference PGDB for importation. Organisms with different cell compartments such as gram-positive bacteria will import no reactions; all inferred reactions will be newly created.

1.4.8.1.8 Construct enzymatic reaction linking reaction with transport protein

Each reaction constructed is associated with its protein by constructing an enzymatic reaction frame that links them together. A history string and evidence code are added to it to track how and why it was created.

1.4.8.2 Reviewing and modifying the results

When TIP inference completes, all transporters that have been identified are displayed in a table, as shown in Figure 1-12. This display includes the review status of the transporter, the gene and annotation of the protein, and the inferred substrate, coupling, and reaction if present.

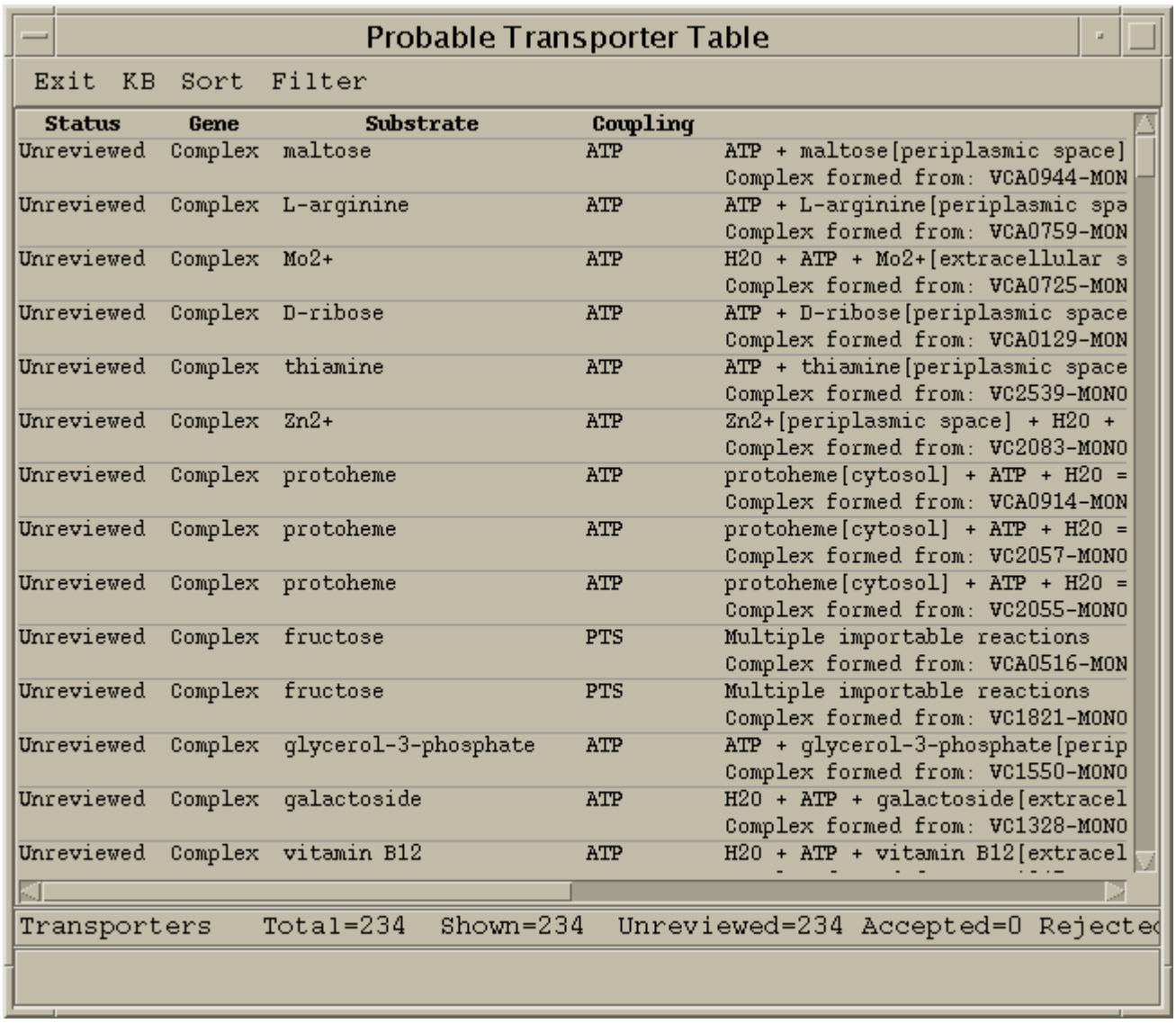


Figure 1-12: Results of transport inference, in tabular form.

To view additional information about a transporter, pass the mouse over its row in the table. The protein frame ID is shown, along with the number of substrates of multisubstrate transporters, the protein frame ID, and brief explanations of certain inferences.

A single transport protein may in general have several substrates, and several different transport reactions. In this case, each substrate is shown on a separate row of the table. These multisubstrate transporters have the same gene name and annotation. Also, when the mouse is passed over its row, the explanatory text at the bottom of the table displays the number of substrates.

The initial table display shows both high- and low-probability transporters. Low-probability transporters have no reaction frames associated with them. Low-probability transporters include

those for which no substrate was identified, and hence no reaction was created. If desired, these may be filtered out using the **Filter** command. Filtering by status is also supported.

1.4.8.2.1 Main display

The Status column indicates whether that transporter has been marked as reviewed by the user, as well as indicating whether the transporter has been accepted, rejected, or whether the decision to accept or reject has been deferred.

1.4.8.2.2 Individual transporter dialog

To review an individual transporter in detail, left-click on its row in the table display. A dialog box similar to the one shown in Figure 1-13 appears.

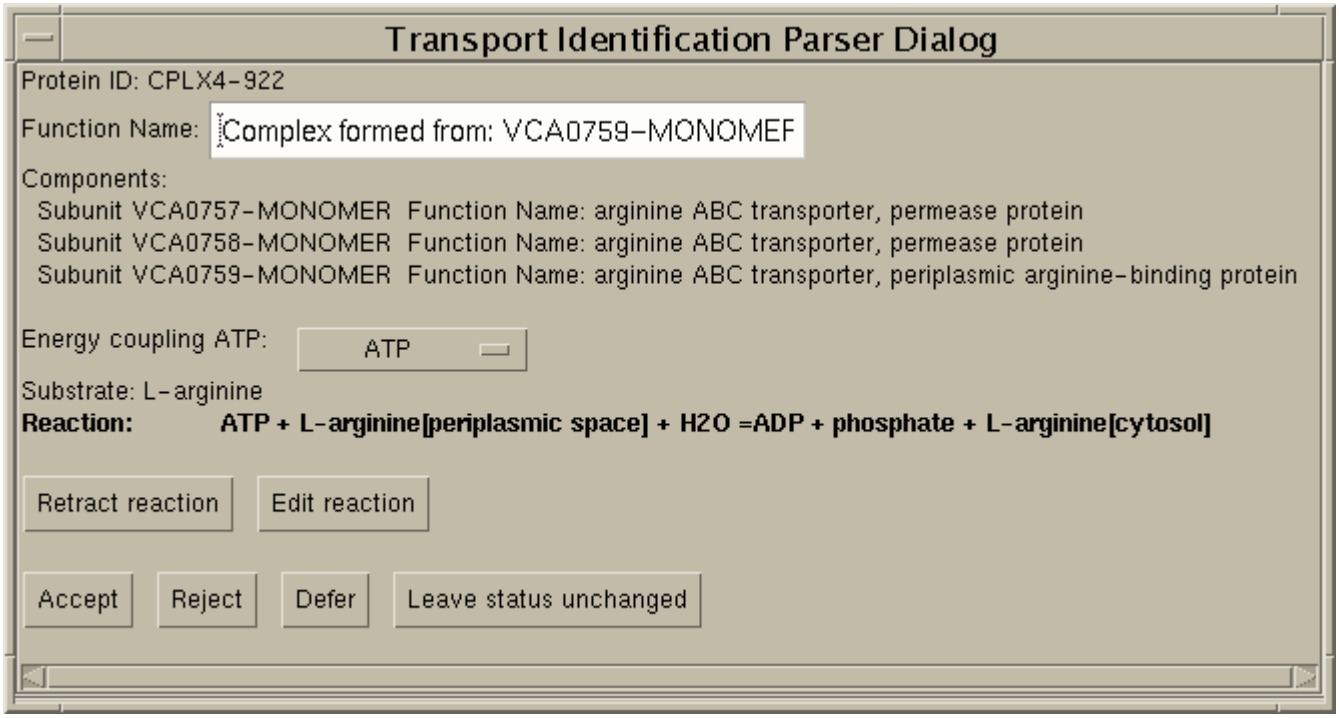


Figure 1-13: Dialog for review of an individual transporter.

Up to four exit boxes are shown at the bottom of the dialog. If a reaction has been created for the transporter, the Accept and Reject commands are present. Clicking **Accept** causes the inferred reaction and enzymatic reaction to remain in the PGDB; if the PGDB is saved, it will be incorporated into the PGDB. Clicking **Reject** causes the reaction and enzymatic reaction to be retracted from the PGDB; regardless of whether the PGDB is saved, these will not be incorporated into the PGDB. Clicking **Defer** simply changes the status indicator in the table display; it is useful in conjunction with the **Sort** and **Filter** commands for grouping transporters. Clicking **Leave status unchanged** does simply that.

The annotation of the protein is displayed, and may be edited. The inferred energy coupling is also shown, and may be changed. The assigned coupling determines the subclass of the transport reaction as noted above. The primary substrate is displayed and may not be edited. However, the **Edit reaction** button allows arbitrary changes to the reaction, including changing the primary substrate.

If a single reaction is shown, transport inference was able to determine an unambiguous reaction for the transporter, either by creating it from scratch or by importing it from EcoCyc.

If multiple imports are shown, two or more EcoCyc reactions conform to the substrates and energy coupling that have been identified. In this case, one of the reactions may be selected. When an import is selected, the **Import reaction** button appears; clicking it causes the reaction to be imported, and an enzymatic reaction associating it with the transport protein to be created.

If no reaction is shown, then either no substrate was identified, or evidence was sufficiently ambiguous to preclude inferring a reaction.

Clicking the **Edit reaction** button allows modification of the inferred reaction, or creation of a reaction in none is present. This is shown in Figure 1-14.

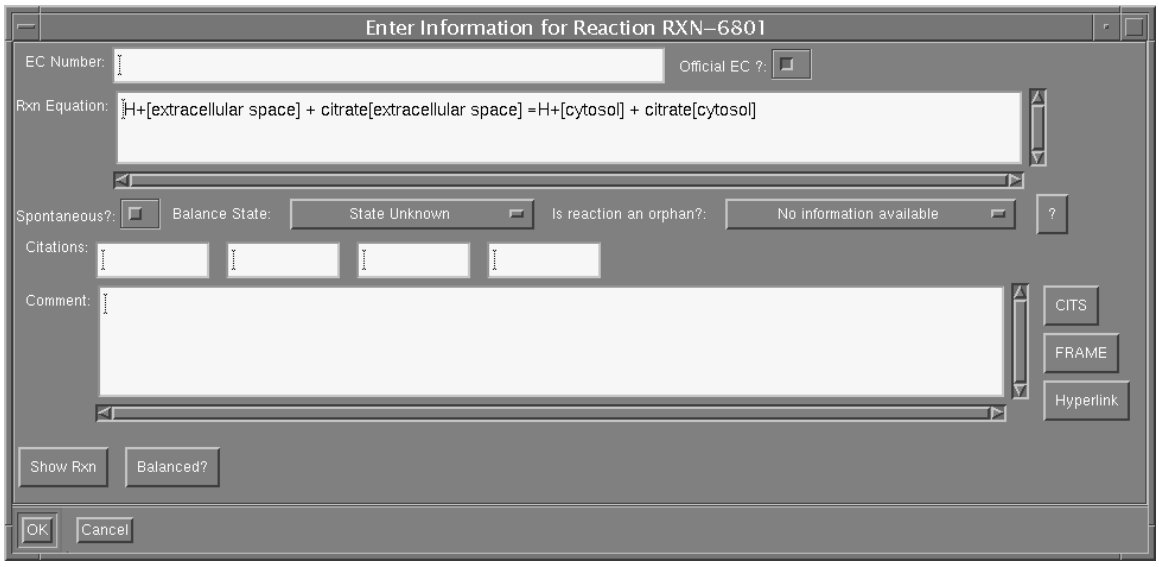


Figure 1-14: Reaction Editor dialog

1.4.8.2.3 Saving and exiting

There are four commands that exit TIP:

- Exit -> Exit, rejecting all unresolved predictions. All frames that were created for unresolved transporters (those with status of DEFERRED or UNREVIEWED) are retracted from the PGDB. TIP is then exited without saving the PGDB. To preserve all

previously ACCEPTED changes, the PGDB must be saved at a later time.
- Exit -> Exit, accepting all unresolved predictions. All frames that were created for unresolved transporters (those with status of DEFERRED or UNREVIEWED) are accepted into the PGDB as is. TIP is then exited without saving the PGDB. To preserve all ACCEPTED changes, the PGDB must be saved at a later time.
- KB -> Save KB & Exit, rejecting all unresolved predictions. All frames that were created for unresolved transporters (those with status of DEFERRED or UNREVIEWED) are retracted from the PGDB. The PGDB is then saved; this preserves all previously ACCEPTED changes.
- KB -> Save KB & Exit, accepting all unresolved predictions. All frames that were created for unresolved transporters (those with status of DEFERRED or UNREVIEWED) are accepted into the PGDB as is. The PGDB is saved; this preserves all ACCEPTED changes.

Each of these commands shows a summary of PGDB changes that have been made and the number of transporters that have been neither accepted nor rejected. Note that if the PGDB is not saved, either here or subsequently, no transport information will be saved to the PGDB, regardless of what has been accepted or rejected. In other words, **accepting a transporter alone is not sufficient** to update the PGDB.

1.4.8.3 Repeated invocations of TIP

TIP inference excludes from consideration any proteins that already have at least one transport reaction associated with it. Therefore, if TIP is run repeatedly, any transporters that are accepted and saved will not be re-inferred.

Note that as of this time, rejecting a transporter does not reject it permanently; subsequent invocations of TIP will re-infer it.

1.4.9 Update Overview

The command **Refine -> Update Overview** constructs a new Cellular Overview diagram for the PGDB, taking into account any changes to reactions or pathways since the overview was last generated. The new overview diagram is saved automatically.

1.4.10 Pathway Hole Filler

1.4.10.1 Overview of the Pathway Hole Filler Algorithm

A pathway hole occurs when, based on PathoLogic's predictions, the organism's genome appears to lack one or more enzymes required to complete a pathway. The activity of each pathway hole is known from the pathway inferred by PathoLogic. We use a set of isozyme sequences encoding the required activity in other genomes to search for candidate enzymes in the genome of interest.

After identifying candidate sequences, our program uses a Bayes classifier to evaluate each candidate. Rather than evaluating the candidates based solely on their similarity to the set of search sequences, we determine the probability that the candidate has the desired function. Our classifier considers evidence from the homology search (e.g., E-values, alignment lengths, and the rank of the candidate in the BLAST output) from the pathway context of the missing reaction (e.g., is the candidate gene adjacent to a gene catalyzing an adjacent reaction in the pathway?) and operon-based data (i.e., is the candidate gene likely to appear in an operon with another gene in the pathway?).

A more detailed description of the Pathway Hole Filler is available at http://www.biomedcentral.com/1471-2105/5/76.

1.4.10.2 Pathway Hole Filler Operation

1.4.10.2.1 Installation

The instructions provided here for running the Pathway Hole Filler assume that a local installation of the BLAST program capable of XML output (newer than BLAST version 2.1.2) is available and correctly configured. The NCBI BLAST executables and instructions for installation can be aquired at ftp://ftp.ncbi.nlm.nih.gov/blast/executables/LATEST and ftp://ftp.ncbi.nlm.nih.gov/blast/documents/blast.txt, respectively.

1.4.10.2.2 Overview

The Pathway Hole Filler runs in two phases - training and prediction.

Training: In the training phase, the known reactions in the PGDB are used to train the Bayes classifier to be able to predict which candidates have the desired function and which do not. The training phase gathers evidence about the known reactions in the PGDB (e.g., given that gene A catalyzes the reaction adjacent to the reaction catalyzed by gene B in a pathway, what is the probability that genes A and B are neighboring genes in the genome? what is the distribution of the rank in the BLAST output of genes assigned to reactions in the PGDB?). The classifier can then compare the evidence for a particular candidate sequence to the prior evidence gathered about sequences known to have the correct function, and predict the probability that that candidate has the desired function.

The Pathway Hole Filler requires a list of known reactions in the PGDB from which to gather the data needed for training. Using a larger dataset for training should produce more robust results in the prediction phase; however, gathering a larger dataset will require more time to run the Pathway Hole Filler. Given a random selection of reactions, the specific reactions selected should not impact the accuracy of the prediction phase; hence, which reactions the user chooses is likely less important than the number of reactions chosen. Gathering training data for 100 reactions takes about 2 hours (depending on the number of isozyme sequences available for the reactions selected) and should scale approximately linearly with the number of reactions.

Prediction: After the training data has been gathered, the program will make predictions for holes in pathways in the PGDB. The default setup groups the missing reactions by pathway and makes

predictions for all pathways with holes. Since making predictions for all pathways with holes may take several hours, the user may wish to run the Pathway Hole Filler in stages. To do this, the user may select any subset of pathways with holes from the list provided by the interface. The program may also be configured to search for a list of reactions selected by the user, rather than searching for holes in a list of pathways.

1.4.10.2.3 Operation of the Pathway Hole Filler

To identify missing enzymes in the genome predicted to fill these pathway holes, use the command **Refine** -> **Pathway Hole Filler** to access the Pathway Hole Filler.

The Pathway Hole Filler can be operated in one of three modes:

- Fully-Automatic - In this mode, the entire hole-filling procedure is carried out with no interaction required from the user.
- Wizard - In this mode, the user is guided through each step of the hole-filling procedure before the program is run.
- Power User - In this mode, the user must initiate each step of the hole-filling process by selecting the appropriate command from the Pathway Hole Filler menu.

In each of these modes, the program runs in three distinct steps as listed in the Pathway Hole Filler menu - Step I: Prepare Training Data, Step II: Identify and Evaluate Candidates, and Step III: Choose Holes to Fill in DB. The operation of each step described below applies to both the Wizard and the user-driven modes. In Wizard mode, the interface guides the user through the necessary selections for Step I and Step II, these two steps are executed, then the Wizard prompts the user to make the required selections for Step III. In the user-driven mode, each step is preceded by dialog boxes where the user must make selections. For the Fully-Automatic mode, the program uses the default options when user selections are indicated.

In the first step, "Prepare Training Data", data is extracted from the known reactions (reactions to which an enzyme has been assigned) in a PGDB. The user must select from which PGDB to extract the training data (the default is the currently selected PGDB). If training data for that PGDB have been generated previously, the user must decide whether to use the existing data or generate new data. Likewise, if training data have been generated for another organism, the user may select that PGDB from the list and use that data to train the Bayes classifier for any other organism they wish to predict. When generating new training data, the user must also decide which known reactions in the PGDB to extract data from (the default is to use all known reactions in the PGDB). The flowchart in Figure 1-15 also describes the process for preparing training data.

The second step, "Identify and Evaluate Candidates", requires the user to select the set of pathway holes that the program will attempt to fill (the default is to search for holes in all pathways in the PGDB). The program can search for holes in all pathways with holes, holes in a list of pathways selected by the user, or a list of holes (reactions) selected by the user.

In the last step of filling pathway holes, "Choose Holes to Fill in the DB", the user can specify which pathway holes to fill with which candidate identified in the second step of the process. The first window that opens after selecting Step 3 from the Pathway Hole Filler submenu

displays a list of the top-scoring candidates for pathway holes where the probability computed for the candidate is above the user-specified cutoff (Figure 1-16). The user may enter any value between 0 and 1 in the box next to the button labeled "Update display for new cutoff". After changing the value and clicking this button, only candidates with P greater than or equal to the specified cutoff will be displayed.

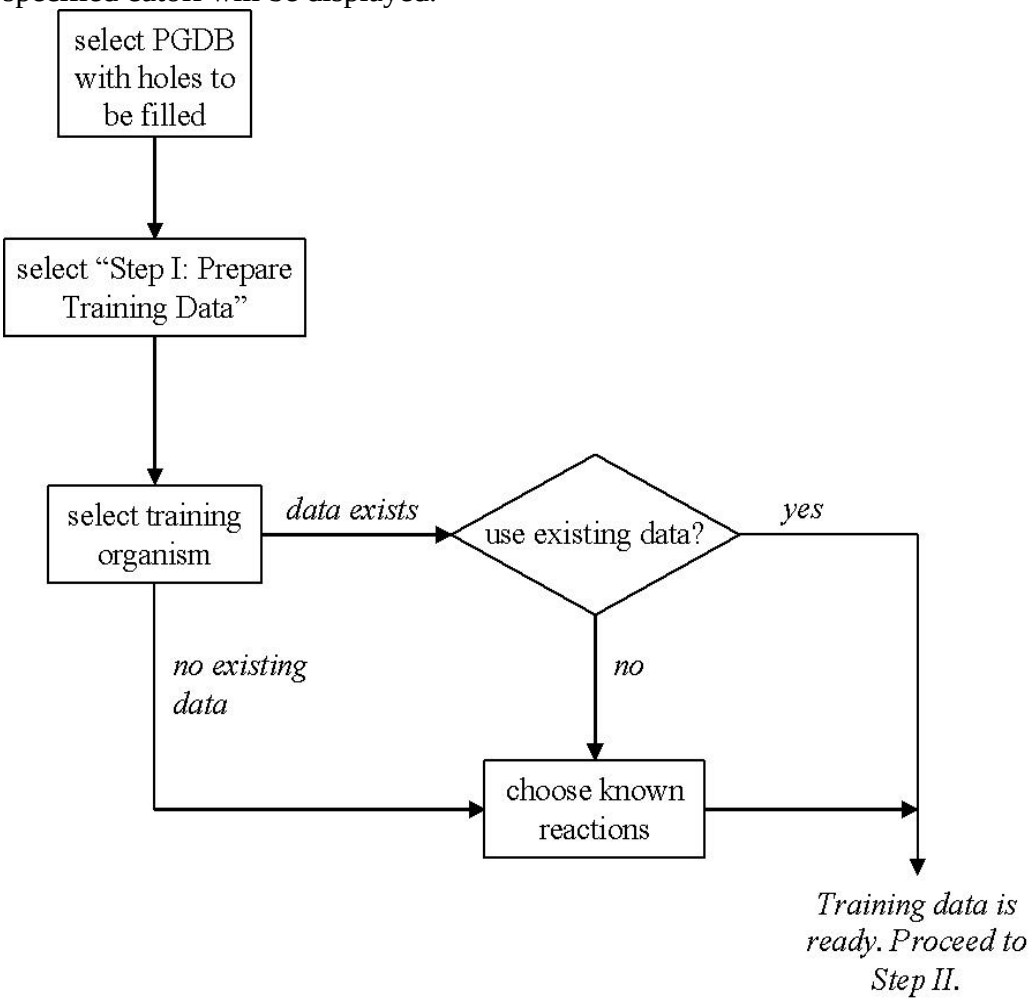


Figure 1-15 Flowchart for Step I: Prepare Training Data

The candidates above the cutoff appear in a table with three columns - "Holes/Reactions", "Top candidate", and "Fill hole with top candidate?". If Step 2 was executed using a list of pathways with holes, all candidates for a pathway will be grouped together in the list.
For each pathway hole, the actual missing reaction and its EC number (if any) are displayed in the first column. The top-scoring candidate, its computed probability, and any existing functional annotation for the candidate appear in the second column. The last column holds a radio-button list where the user may specify the fate of the top candidate for each pathway hole. If you wish to fill a pathway hole with the top-scoring candidate, make the appropriate selection in column 3. There are three options:

- "No" - Do not fill the pathway hole with this candidate.
- "Yes, by adding function" - Fill the pathway hole with this candidate. Assign the function of the pathway hole/missing reaction to the candidate in addition to any currently known

function (i.e., you believe the candidate may be a multifunctional protein).

- "Yes, by replacing function" - Fill the pathway hole with this candidate. Assign the function of the pathway hole/missing reaction to the candidate, replacing any currently known function (i.e., you believe the originally assigned function is incorrect).

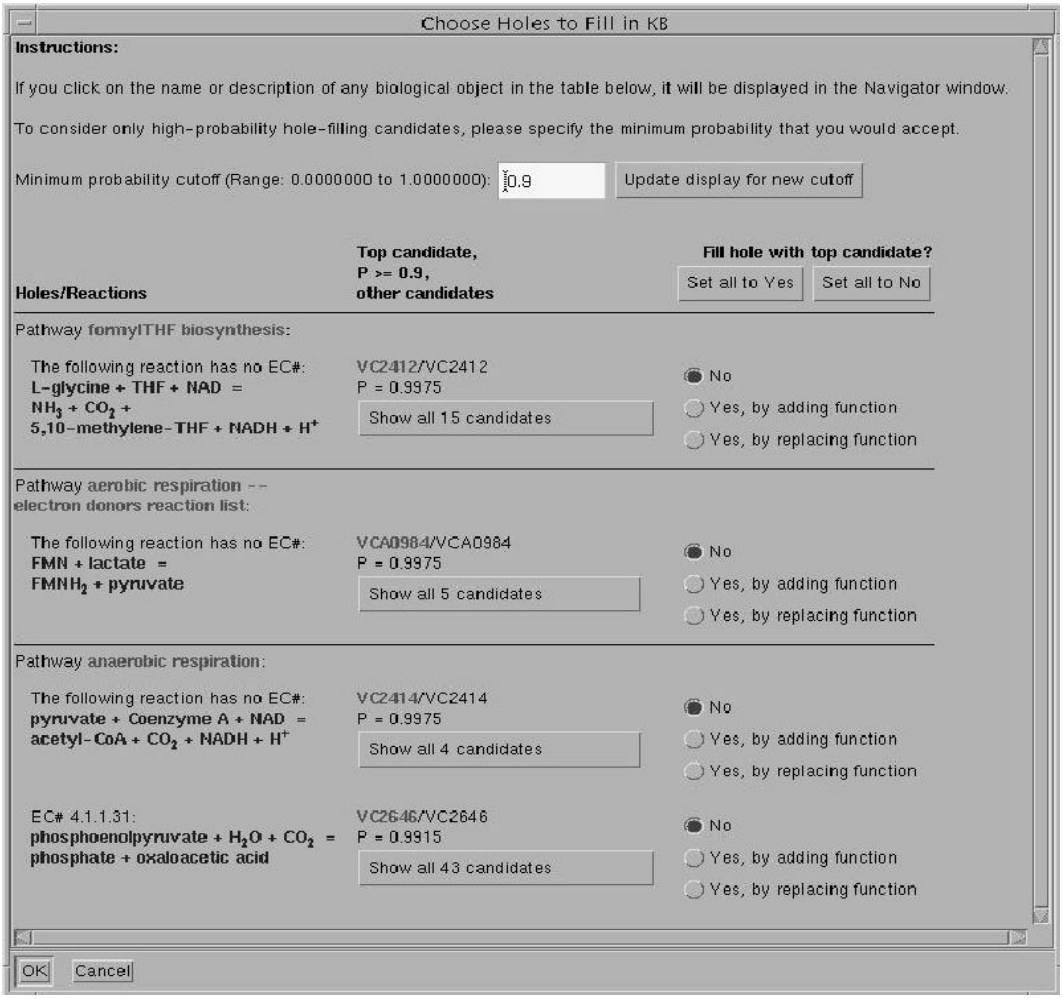


Figure 1-16 "Choose Holes to Fill in DB" page in the Pathway Hole Filler. In this case, the user selected a list of pathways for hole-filling during Step 2 of the process. If the user had selected a list of reactions for hole-filling, the pathway subheadings in the first column would be eliminated.

The second column includes a button labeled "Show all X candidates", where X is the number of candidates identified in Step 2. Clicking this button will open a second window titled "Candidates to fill pathway hole: …". This window displays detailed evidence for each candidate for filling the pathway hole, as well as additional information describing the pathway hole and the candidate itself. Each column contains data for one candidate. Refer to Figure 1-17 and Table 1-1 for a description of each data item in the page. To fill a pathway hole with a particular candidate enzyme, select the appropriate radio button in the "Fill hole?" row. The data displayed in this page can be very helpful in choosing which pathway holes to fill.

Like the previous window, each candidate may be used to fill a pathway hole. The same radio button list is included in row E, "Fill hole?" of this window. The difference now is that any number of candidates may be selected to fill the pathway hole. This feature may prove to be useful, for example, for cases where multiple isozymes appear to catalyze the same reaction, or when a reaction might be catalyzed by an enzyme complex. If the pathway hole filler identifies multiple subunits of the complex, the user can assign each subunit to the reaction.

After making the appropriate selections in the "Candidates to fill pathway hole: …" window, click the "OK" button in the lower left-hand corner of the window to return to the "Choose Holes to Fill in DB" window. Any selections that were made in the previous window will be reflected in the column labeled "Top-scoring candidate". A list of genes encoding the candidates selected by the user will be shown under the heading "Other candidates already selected". After clicking the "OK" button in the "Choose Holes to Fill in DB" window, the selected changes will be made in the DB.

Table 1-1 Description of fields in the "Candidates to fill pathway hole:…" window.

Key	**Description**
A	The missing reaction that may be catalyzed by the candidates shown in the table.
B	A list of all pathways in which this missing reaction appears.
C	Toggles the definitions for the table rows on and off.
D	Candidate hole filler is the putative enzyme identified by the Pathway Hole Filler to potentially catalyze the reaction shown at A. This row in the table shows the protein frame name, the currently assigned function of the protein (if any), and a button to move the candidate to the last column in the table.
E	The "Fill hole?" row holds a radio button list where the user may specify the fate of the candidate with respect to the pathway hole. There are three choices: 1. "No" - Do not fill the pathway hole with this candidate. 2. "Yes, by adding function" - Fill the pathway hole with this candidate. Assign the function of the missing reaction (A) to the candidate in addition to any currently known function (i.e., you believe the candidate may be a multifunctional protein). 3. "Yes, by replacing function" - Fill the pathway hole with this candidate. Assign the function of the missing reaction (A) to the candidate, replacing any currently known function (i.e., you believe the originally assigned function is incorrect).
F	The "Gene" row includes the name of the gene which encodes the candidate enzyme and the common name of the gene, if any.
G	The "Probability" is the probability that this candidate enzyme catalyzes the missing reaction (A). This value is computed by the Pathway Hole Filler.
H	"Candidate's Gene-Reaction Schematic" displays the gene-reaction schematic for the candidate gene in the organism where pathway holes are being filled (i.e., the current DB). The diagram will display any functions that will be replaced if you choose to fill the pathway hole "by replacing

	function".
I	"MetaCyc Gene-Reaction Schematic" displays the schematic for the missing reaction/pathway hole as it appears in MetaCyc. This schematic may be useful for evaluating candidates for filling pathway holes.
J	"Average rank" is the average the average rank of the candidate enzyme sequence in the BLAST output lists (e.g., if a candidate is the best hit in each search, the average rank for the candidate is 1).
K	"Best E-value" is the negative log of the E-value for the best alignment of the candidate with a query sequence.
L	"Shotgun score" is the number of query sequences whose BLAST output included the candidate sequence.
M	"Average fraction aligned" is the average of each BLAST alignment length normalized by the length of the query sequence.
N	"Adjacent reactions?" shows a list of reactions adjacent to the pathway hole for which the gene encoding the product that catalyzes the reaction is adjacent to the candidate in the genome.
O	Is the candidate gene in a "potential operon" as another gene in the same pathway; we define a "potential operon" as a contiguous series of genes transcribed in the same direction.
P	The user may enter a history note to associate with the enzymatic reaction created by filling the hole.

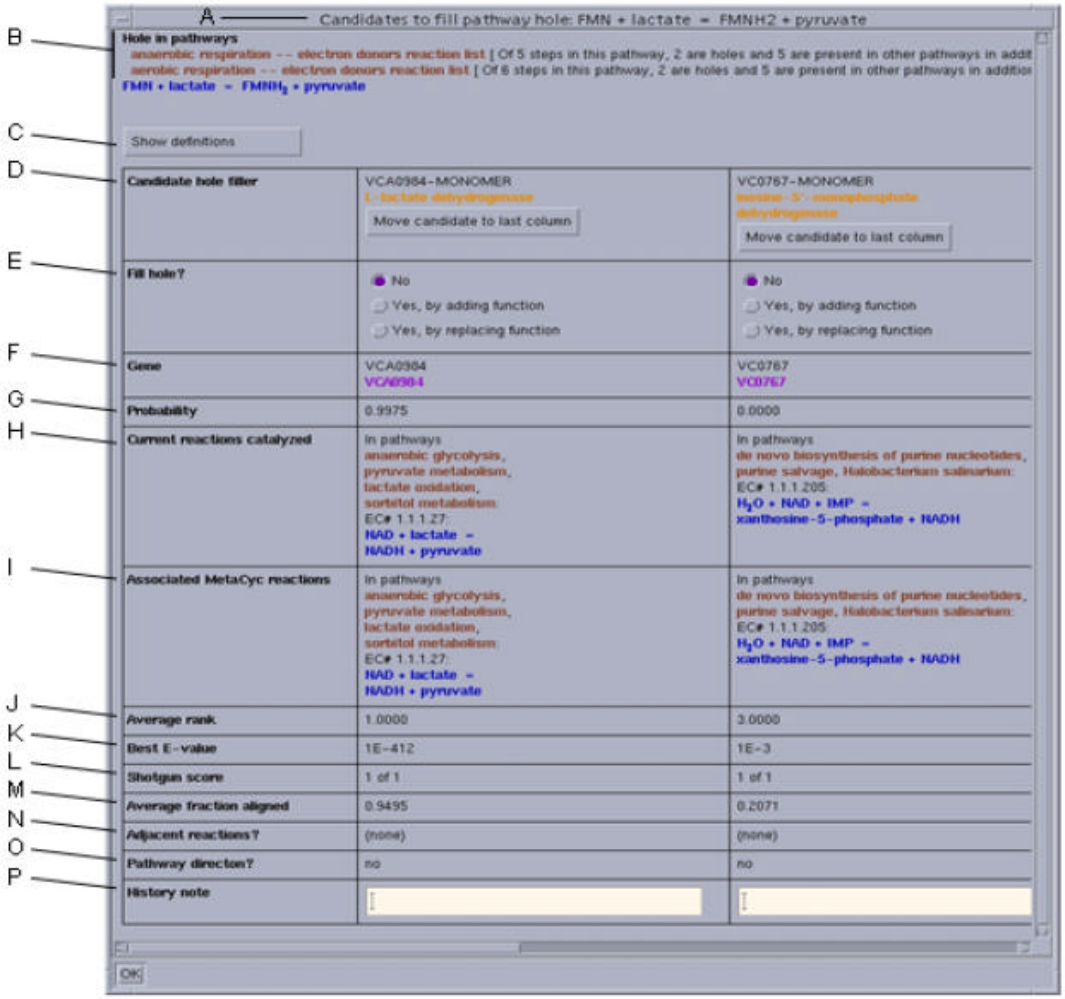


Figure 1-17 Candidates to Fill Pathway Holes page from the Pathway Hole Filler.

1.5 OUTPUT FROM THE PATHOLOGIC PATHWAY PREDICTOR

1.5.1 Pathway/Genome Database

The principal output as a result of running the Predictor is the subject organism pathway/genome database.

1.5.2 PathoLogic Pathway Predictor Summary Pages

Following construction of a pathway/genome database for a subject organism (O), the user can view a series of reports, which can also be output to HTML files that may be viewed via a Web browser. To view the reports, select the **Summary of Organisms** command in the Navigator

menu. Select the newly created PGDB, click on the button **Summary of Pathway Evidence**, and select the desired report. To generate an HTML file for one of these reports, right-click on the report name in the index page, and select **Save Page as HTML**. These reports provide a convenient summary of the output created by the Predictor and are grouped into two categories:

1. Pages summarizing evidence for the presence of pathways in O.
2. A page that lists reactions missing from O that would complete its partial pathways.

The pathway evidence pages are grouped into 5 distinct sets based upon pathway function and/or the *X*|*Y*|*Z* score (see "Assigning Evidence Scores to Predicted Pathways" in section 1.1.2.2, for definition of this score as well as a description of how evidence is assigned) assigned by the Predictor to each pathway. One grouping is of pathways by functional category. Within each category pathways are ranked on the basis of assigned scores (i.e., ranked on the basis of the numerical value, Y/X, from largest to smallest). An example of this page is shown in Figure 1-15. A second grouping is of all pathways ranked by Y/X values (largest to smallest). The remaining three groupings represent a partitioning of all pathways based upon pathway score, *X*|*Y*|*Z*. The first group is of pathways probably present in O, where probably means there is evidence for 50% or more of the steps of each pathway in this group, i.e., $Y/X > 0.5$. The second group is of pathways each of which have evidence for at least one step but less than 50% of the reactions in the pathway, i.e., $0 < Y/X > 0.5$. These are pathways deemed as being possibly present in O. The last grouping is of pathways probably not present in O. All pathways in this group have assigned Y values of 0.

A "pathway glyph" is included for each pathway entry on these pages, which summarizes graphically the evidence for each step in the pathway. The glyph shows the sequence of steps in the pathway, with reaction steps drawn in different colors depending on whether or not there is evidence for catalysis of the reaction in O, and whether or not the reaction step is unique to the pathway. The meaning of the colors is explained at the end of the **Contents** section of the page.

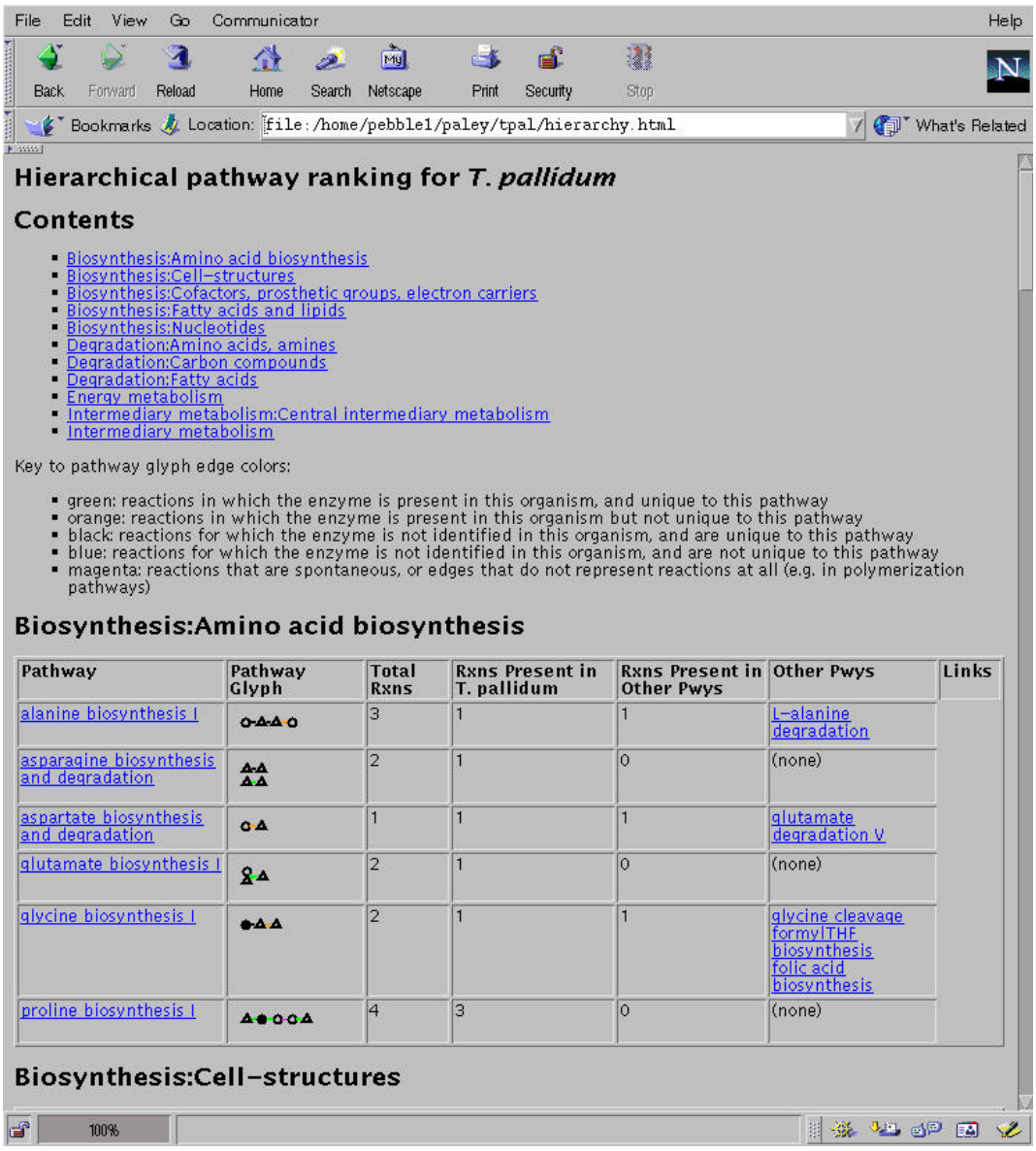


Figure 1-18 Pathway Predictor Summary Page organized by pathway functional category

1.5.3 Interpretation of the PathoLogic Pathway Predictor Summary Pages

In this section, we provide guidelines for interpretation of the PathoLogic Pathway Predictor Summary Pages. First, we present a set of guidelines for interpretation of the pathway listings and associated pathway evidence scores. They provide a framework for outlining issues relevant to interpretation of pathway evidence scores. We then discuss the summary page that lists reactions not predicted to be carried out in O that would complete its partial pathways.

It is important that the user appreciate that the pathway predictions presented in the newly created pathway/genome database are based upon the genomic sequence of a specific strain of the listed microbial species and that there may be significant metabolic pathway variation between different strains of a given species.

When the results of pathway analysis suggest that O has genes coding for enzymes that can catalyze all of the steps of a pathway in the reference database, two interpretations are possible:

The pathway is in fact found in O and is active.

The pathway is not active in O for several possible reasons, e.g., the genes that encode these enzymes might never be expressed, or the genes might contain mutations that render one or more enzymes within the pathway inactive, or the assignment of enzyme function through sequence analysis was incorrect.

When pathway analysis finds partial evidence for the presence of a pathway P in O, several interpretations are possible:

1. The pathway is active in O, but genes encoding enzymes that catalyze the remaining steps in the pathway remain unidentified in the genome; the enzymes used by O may have no detectable homology to previously known enzymes that perform those functions; or there may be no known sequences for enzymes that perform those functions.
2. A variant of pathway P is active in O; that variant might substitute slightly different reaction steps, catalyzed by different enzymes, for reactions in P.
3. Only a segment of the full pathway P is active in O, and that segment connects with O's metabolic network in a different way than P connects with the reference metabolic network. For example, some pathways contain two or more alternative routes for accomplishing a given transformation; if only one of these routes is present in O, the transformation can still be accomplished, but the score for the pathway will be artificially low (examples of pathways containing such alternative transformations are methionine biosynthesis and purine biosynthesis).
4. The pathway is not active because the genome does not in fact code for the enzymes that catalyze the remaining "missing" steps (pathway holes).

The pathway is not active in O because most or all of the putatively identified enzymes for the pathway were in fact wrongly identified by sequence analysis.

When pathway analysis finds no evidence for the presence of a pathway P in O, several interpretations are possible:

1. The pathway is in fact not active in O.
2. The pathway is in fact active in O because all of the enzymes that catalyze the steps within the pathway remain unidentified in the genome.
3. A related pathway that accomplishes a similar biological function (such as catabolism of a sugar), but using a different set of reaction steps, is in fact active in O. That

pathway would not be detected in O by the Predictor if that pathway was not present in the reference database.

One of the PathoLogic Pathway Predictor Summary pages lists reactions predicted not to be carried out in O that would complete its partial pathways, i.e., lists pathway "holes". This list may be used to guide ongoing sequence analysis efforts to find genes for enzymes that patch these "holes". This approach, pathway-driven gene finding, was successfully applied by Pangea scientists to make several functional assignments to *H. pylori* ORFs listed in the original genome annotation.

1.6 BATCH MODE

Pathologic can be run in batch mode to create one or more PGDBs in an automated fashion, skipping the manual steps listed in "Refining the PGDB". Use the -patho command-line argument[1] in either of the following forms:

- -patho param-dir
- Creates a single PGDB based on parameters specified in the following files in the directory you specify by param-dir:
    - organism-params.dat (see "File organisms-params.dat" in section 1.6.1)
    - genetic-elements.dat (see "File genetic-elements.dat" in section 1.2.1)
    - and for each replicon (chromosome or plasmid) listed in genetic-elements.dat:
        - gene sequence FASTA file, [repliconid].fna or [repliconid].fsa
        - gene annotation file: one of the following:
            - [repliconid].pf (see "The PathoLogic File Format" in section 1.2.2)
            - [repliconid].gbk (see "GenBank File Format" in section 1.2.3)
- -patho -f file
- Creates a set of PGDBs; you need to specify the path to a file containing a list of param-dirs, one per line.

Note: The directory or file you specify on the command line should not be the standard PGDB directory location (see "The PathoLogic File Format" in section 1.2.2) because Batch Mode needs to create the standard directories and files for you. So, you need to create the directory or file somewhere else.

1.6.1 File organism-params.dat

Each line of organism-params.dat should be of the form:

ATTRIBUTE value

[1] Command-line arguments are discussed in volume I of the User's Guide, section 2.2.1.

The separator character is the Tab character. Case is not important, except for literal strings. Valid attributes are:

- ID (unique id -- 2-10 alphanumeric characters, no intervening spaces, must start with a letter)
- STORAGE (either File, Oracle, or MySQL; defaults to File)
- NAME (full species name)
- ABBREV-NAME (abbreviated species name)
- STRAIN
- PRIVATE? (either T or NIL, defaults to NIL, specifies whether database can be published to the general public over the web)
- DOWNLOAD-TIMESTAMP
- ORG-COUNTER (a number that uniquely identifies this database)
- DOMAIN (one of Bacteria, Eukaryota, Archaea or Viruses)
- CODON-TABLE (a number between 1 and 15 -- if not supplied, the standard codon table will be used)
- AUTHOR (project author -- either name only, or name:institution)
- CITATION (medline ID)
- HOMEPAGE (URL for project home-page)
- EMAIL (primary contact email address for the project)
- DBNAME (The "pretty name" for the project database)
- COPYRIGHT (a copyright string, in html format, to appear on web pages)
- FOOTER-CITATION (a citation string, in html format, to appear on web pages, that users of the data should cite)
- NCBI-TAXON-ID (the ID from the NCBI organism taxonomy database)

Of these, only ID and NAME are required. Multiple values may be supplied for AUTHOR and CITATION, in which case, each should appear on its own line. If multiple values are supplied for any other field, all but the first will be ignored. Once the file and the project are generated, the ID and STORAGE fields should *not* be edited. The other attributes may be edited so long as the correct format is maintained. Case is important only for species and strain names . Attributes other than these will be ignored.

1.7 ERROR SOURCES IN PATHOLOGIC PREDICTIONS

A pathway/genome database constructed for a subject organism may contain three classes of errors:

- Errors in the input sources that would be propagated to the subject organism database.
- Errors introduced by bugs in the Predictor that we have not yet discovered.
- Errors introduced by the user during the manual steps of database construction.

In this section we briefly outline possible sources of some such errors.
Any errors in the inputs (in particular, the annotated genome and the reference database) will be carried through to the subject organism pathway/genome database. Error rates in annotated genomes are difficult to quantitate since the accuracy of functional assignment by homology remains to be precisely defined.

However, in general, it is expected that some functional assignments in a genome will be incorrect. In part, this reflects the limitations of methods for prediction of function based on sequence data. Another contributing factor is that functional assignments in one genome may be based on sequence homology to another gene in another genome whose function was also assigned by sequence analysis rather than based on definitive experimental data. As a functional assignment becomes further removed from its original source, i.e., solid experimental data, it becomes increasingly error prone.

Software-introduced errors may result as result of the name matching step. This could potentially lead to wrong connections between enzymes and reactions being made. Also tRNA parsing could result in unpredictable results since the syntax used for specifying tRNAs is highly variable.

Some errors may be introduced into the pathway/genome database during editing of pathways or entry of new pathways/compounds. To prevent some such errors, consistency checks are performed by the Predictor.

Computationally predicted metabolic pathways will necessarily be incomplete since the genomes upon which the predictions are based are incompletely annotated. A requirement for recognizing that a particular enzymatic step is carried out in the subject organism is that at least one enzyme that catalyzes that step must have been sequenced in the past. However, of the 3600 EC numbers defined in the ENZYME database [1], version 34 of SwissProt contains sequences for only 37% of these EC numbers. Therefore, many enzymes cannot be detected by sequence-similarity searches, and thus cannot be counted during pathway analysis. These will constitute genes for which no functional assignment can be made, i.e., ORFs. As additional functional assignments are made to ORFs based either on sequence analysis and/or new experimental data, annotations of genomes will become increasingly complete. This completeness will help make computational pathway prediction increasingly robust and less susceptible to error.

1.8 ONGOING PATHWAY CURATION

Each pathway for which there is evidence needs to be checked to make sure it was correctly represented in the newly created PGDB. “Holes” in predicted pathways are then used to identify any enzyme names incorrectly/not matched to reactions. Additional enzyme-reaction matches are sometimes made at this point. This information may be added in a number of ways:

- By editing the annotation file and rebuilding the database. If a lot of manual work has been done, this is not the recommended way since the Pathway Tools have no way of keeping manual changes after a PGDB has been reinitialized.
- By editing the local file that maps synonyms to reactions. Again, if much manual work has been done, this is not the recommended approach.
- By using the editing software that is part of the Pathway Tools to manually alter protein-reaction relationships and pathway definitions. This is the recommended approach if a lot of manual work has been done up to this point.

For some organisms, literature searches should be conducted to identify:

- Predicted pathways for which there exists direct experimental evidence
- Experimentally isolated enzymatic activities for which no gene was identified in the

genome. This information is then added to the relevant entries in the database. Any information about an enzyme that is not linked to a gene represents an experimentally confirmed enzymatic activity for which no gene was identified in the genome. Gene finding efforts and/or ongoing ORF annotation should focus on these enzymes as genes for these enzymes almost certainly reside in the genome.

MetaCyc does not contain all microbial pathways of small molecule metabolism. The process of PGDB construction is such that any pathways not in MetaCyc but present in the subject organism are not incorporated into the newly constructed PGDB.

2 EDITING PATHWAY/GENOME DATABASES

Pathway/Genome Editors, hereafter referred to as simply the Editors, are a collection of tools for interactively modifying the objects within a Pathway/Genome Database (PGDB). Each editing tool is oriented toward modifying a particular class of objects, such as genes, reactions, and metabolic pathways.
Some of the topics addressed in this chapter include:

- Section 2.1 tells you how to begin editing.
- Section 2.2 describes the object data model used by the Ocelot DBMS, which underlies all PGDBs. We recommend that you also consult later sections of this chapter that describe the other Editors in more detail.
- Section 2.3 provides examples of editing operations.
- Section 2.4 describes detailed aspects of editing, such as how to create URL links from a PGDB to other Web-accessible databases, and restrictions on editing of PGDBs that are designed to protect your work.

For information about strategies and policies for curation of PGDBs, refer to the Pathway Tools Curator's Guide, available at
`http://bioinformatics.ai.sri.com/ptools/curatorsguide.pdf`.

2.1 OVERVIEW OF THE EDITORS

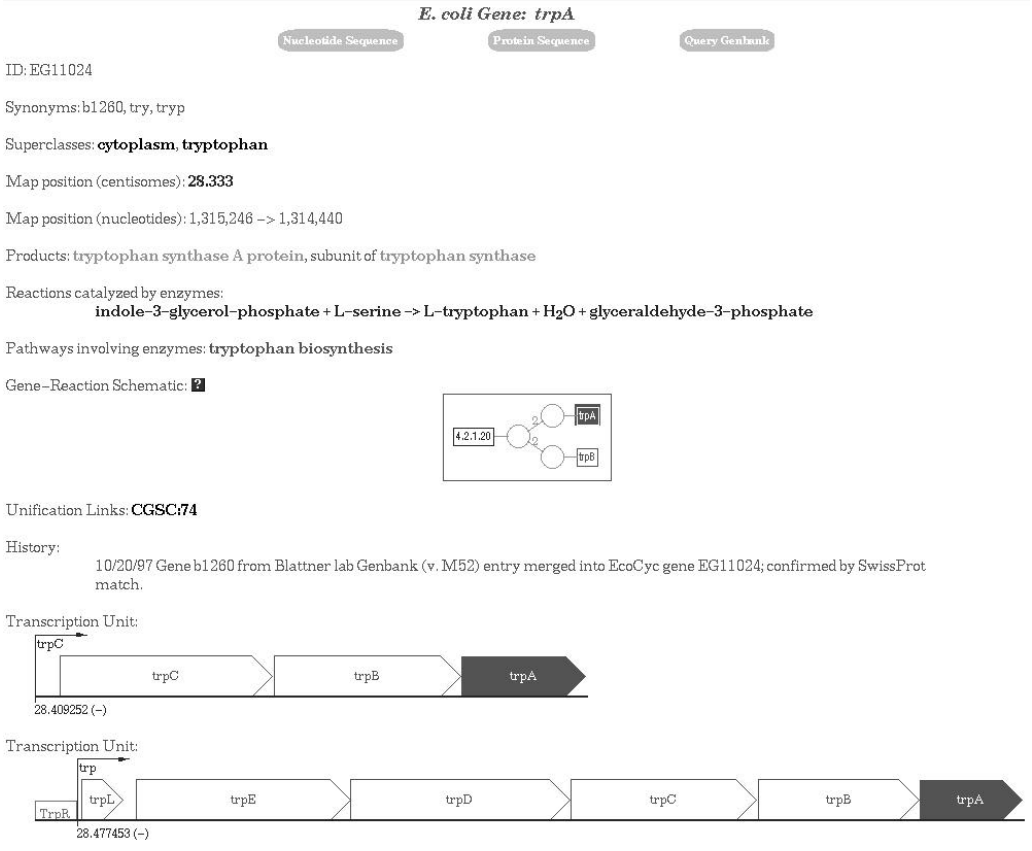
This section provides a listing of the available Editors, a description of how to invoke each Editor, and a brief discussion of how to save and undo changes to a PGDB. Later sections describe each Editor in more detail, and provide a more elaborate description of the Ocelot database system used by Pathway Tools.

2.1.1 Right-Click on Object Handles to Invoke Editors

The Editors are used both to modify existing entities in PGDBs, and to create new entities. All Editors are launched from the Navigator window. Whenever you see an object within the Navigator, your can launch an Editor by right-clicking on the *handle* for an object, and selecting an Editor from a command menu that appears after right-clicking. Navigator displays have two types of object handles. The first type is simply the name of an object that appears in the title region at the top of the Navigator display window for that object. For example, if we are viewing the display for the *E. coli trpA* gene within the Navigator display (see Figure 2-1), the handle for *trpA* is the text "*trpA*" at the top of that window. You can edit the *trpA* gene by right-clicking on the title handle for *trpA*. The second type consists of the names of other objects in a Navigator display. For example, the display for *trpA* contains the following handles, and by right-clicking on any of them, you can edit the respective objects:

- "tryptophan synthase A protein" is the handle for the polypeptide product of trpA .
- "tryptophan synthase" is the handle for protein complex in which that polypeptide is a subunit.
- "indole-3-glycerol-phosphate + L-serine - -> H2O + glyceraldehyde-3-phosphate + L-tryptophan" is the handle for the reaction catalyzed by that protein complex.

- "L-serine" is the handle for a substrate in that reaction.
- "tryptophan biosynthesis" is the handle for the pathway in which that protein complex is an enzyme.



?

Figure 2-1 Typical gene display

2.1.2 List of Editors

Use the right mouse button menu commands to launch the Pathway/Genome Editors and modify existing objects, and use the other commands described here to create new objects with each Editor. Many object types can also be created from the **Edit -> Create** right-click menu.

- Gene Editor (Section 2.3.1)
  - For modification, right-click on the gene handle and choose **Edit -> Gene Editor**
  - For creation of new genes, **Gene -> New Gene**
- Transcription Unit Editor (Section 2.3.2)
  - For modification, right-click on the transcription unit handle and choose **Edit -> Transcription Unit Editor**
  - For creation of new operons, **Gene -> New Operon**
- Intron Editor (Eukaryotes only) (Section 2.3.3)
- To create a new splice form for a gene, or to edit the set of introns associated with a gene or particular splice form of a gene, right-click on the gene and select Edit -> Intron Editor.

- Protein Editor (Section 2.3.4)
    - For modification, right-click on the protein handle and choose **Edit -> Protein Editor**
    - For creation of new proteins, **Protein -> New**
    - For creation of a new enzyme for a given biochemical reaction, right-click on the reaction and select **Edit -> Create/Add Enzyme**
- Pathway Editor (Section 2.3.5)
    - For modification, right-click on the pathway handle and choose **Edit -> Pathway Editor**
    - For creation of new pathways, **Pathway -> New**
- Reaction Editor (Section 2.3.6)
    - For modification, right-click on the reaction handle and choose **Edit -> Reaction Editor**
    - For creation of new reactions, **Reaction -> New**
    - For creation of a new enzyme for a given biochemical reaction, right-click on the reaction and select **Edit -> Create/Add Enzyme**
- Chemical Compound Editor (Section 2.3.7) (edits common-name, synonyms, citations, and so on)
    - For modification, right-click on the compound handle and choose **Edit -> Compound Editor**
    - For creation of new compounds, **Compound -> New**
- JME Compound Structure Editor (see Section 2.3.8)
- Compound Structure Editor (see Section 2.3.11)
- Publication Editor. (Section 2.3.12)
- PGDBs contain two types of literature references: references to publications that are included in the PubMed database, and references to publications that are not in PubMed. You enter the first type of reference in the form of a bracketed PubMed ID number, such as [78345682], whereas the ID's for references to non-PubMed articles are based on the first author's last name, followed by the publication year, such as [Smith95].
    - To modify a reference entry, right-click on the citation in the References section at the bottom of any display page.
    - If you enter a reference that is not included in PubMed in a citation field of any editor, such as the enzyme editor, you will automatically enter a dialog window for creating the new publication object in the PGDB (or for searching for an existing publication in case it already exists), in which you would enter the full citation information. This dialog window would not appear if the reference contains only digits, as it is assumed to be PubMed ID. The full information for PubMed citations can be periodically downloaded in bulk from the NCBI website with the command **File -> Import -> Citations from PubMed**.

2.1.3 Saving Edits

Modifications that you make to a PGDB using the Editors are **not saved** until you explicitly save them using the **Save DB** button in the Navigator toolbar (this command is also accessible from the **File** menu as **File -> Save Current DB**, and with the associated keyboard shortcut Ctrl-S). Until **Save DB** is executed, edits will be lost if a power failure or computer crash occurs.

You can use the menu command **File -> Revert Current DB...** to discard all PGDB modifications (creation of new objects and changes to existing objects) that you have made since the last **Save DB** command was executed, if you decide not to retain them.

2.2 THE FRAME DATA MODEL

To understand PGDBs, you must understand the frame data model that is used to structure information in all *Ocelot DBs* (see the Glossary for definitions of these and other terms). The frame data model is very similar to the object data model used in object-oriented databases. The frame data model comprises the following components.

2.2.1 Frames

The representation of an individual entity in the database is referred to as an *instance.* For example, the amino acid lysine and the *E. coli* gene *trpA* are instances. Similar instances often belong to one or more *classes,* which are entities that represent collections of semantically related instances. To continue with the same example, a class called Amino Acids includes the instance lysine, while a class called Genes includes the *trpA* gene instance.

The representation of both instances and classes in the database is in the form of *frames*. A frame is an object with which data are associated. Examples for instance frames include the individual compounds, genes, reactions, proteins, pathways etc, while examples for class frames include Amino acids, Genes, tRNAs, etc. Table 2-1 covers the most important classes in a PGDB.

Table 2-1 Important classes in a PGDBa

Class	Description
Reactions	All reactions, including enzyme and transport reactions
Pathways	All pathways, including superpathways and signal-transduction pathways
Genes	All genes
Polypeptides	All monomers
Protein Complexes	All protein complexes
Proteins	The parent of Polypeptides and Protein-Complexes

a. Note the class names are case sensitive.

A Database, or DB, is a collection of frames and their associated slots, values, facets, and annotations. DBs are saved permanently in Oracle or MySQL database management systems. A *knowledge base*, or KB, is a different name for a database.

Each frame has a frame name (also called a frame ID, or a key) that uniquely identifies that frame within the DB.

2.2.1.1 Slots

Slots encode the attributes or properties of a frame, and also represent relationships between frames. A slot is a mapping from a frame and a slot name to a collection of zero or more values. Each value can be any Lisp object (such as a symbol, list, number, or string). Ocelot supports three collection types for slot values: sets, multisets, and lists. For example, the slot called **`components`** in the AROH-CPLX frame in Figure 2-2, on the next page, lists the frames that represent the polypeptide subunits of the enzymatic complex. The figure shows a printed representation of two frames, AROH-MONOMER and AROH-CPLX. The former is an instance of the class **`Polypeptides`** and the latter is an instance of the class **`Protein-Complexes`**. For each frame we list the name of a slot within the frame, followed by a colon, followed by the zero or more values of that slot. An annotation records the coefficient of the AROH-MONOMER component within the ARZOH-CPLX.

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

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


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


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

Figure 2-2 Printed representation of frames for AROH-MONOMER and AROH-CPLX.

2.2.1.2 Facets

Facets encode information about slots. Facets are uniquely identified by a facet name, a slot name, and a frame. A facet has as its value a set of Lisp objects. Facets allow comments or citations to be associated with an entire slot.

2.2.1.3 Annotations

To store information about individual slot values, we use annotations. An annotation associates a labeled set of data objects with a particular slot value. Annotations are used to attach comments and citations to slot values. In addition, as shown in Figure 2-2, an annotation stores the coefficient of each subunit within a protein complex. The slot value AROH-MONOMER has an annotation whose label is **`coefficient`** and whose value is 2.

2.2.2 Relationship of Frames to Pathway Tools Displays

An important thing to keep in mind when modifying a PGDB is that the display windows generated by the Pathway/Genome Navigator and Pathway/Genome Editors do not literally display the exact contents of the PGDB. Every Navigator window is computed by a program that takes the information in a Pathway Tools frame (such as a pathway frame), plus information from related frames (such as the reactions within the pathway), and computationally maps the information in these frames into a drawing. These programs display some slots more or less verbatim, such as the **comment** slot although even that slot is processed to convert citations into bracketed, clickable links. Other slots are not displayed at all. For example, when displaying a compound, the slot that stores the chemical bonds within the compound is not displayed literally, although the information it contains is used to produce a drawing of the compound structure.

Sometimes fairly complicated transformations are applied to create these drawings. For example, when displaying an enzyme, Pathway Tools displays some information from the enzyme frame itself, but it also makes use of information from the enzymatic-reaction frame(s) connected to the enzyme frame, the reaction frame(s) connected to the enzymatic reaction(s), and the compound (substrate) frames listed in the reaction. When Pathway Tools displays those related objects, the name it prints for a given frame is usually the value of the **common-name** slot for that frame, or else the frame name if that slot has no value.

Therefore, it may not always be obvious which frame you must edit to effect a given change in a display window. For example, to change the name shown for a chemical compound in the display of a pathway, you should edit the chemical compound frame, not the pathway frame. As you gain experience with the system, you will learn what editor to use to alter a given part of a Navigator display.

2.3 THE EDITORS

Each editing tool is designed for entering information about a particular data type, such as genes, pathways, or proteins. In some cases you will need to use more than one editor to implement what may seem like a “single” change in biological knowledge. For example, if an enzymatic function is assigned to a gene whose previous function was unknown, you would use the gene editor to name the gene and describe its function, and also use the protein editor to connect the enzyme to a reaction (perhaps within a pathway), and to define properties of the enzyme such as its cofactor and inhibitors, when known.

2.3.1 The Gene Editor

By a *gene* we mean a region of DNA that encodes a protein, tRNA, or other type of gene product. The gene editor is shown in Figure 2-3. Fields within the gene editor include the following:

- Class: Assigns the gene to a functional classification scheme developed by Monica Riley.
- Common-Name, Synonyms: The primary name for the gene, and alternative names for the gene.

- Product: The name of the gene product is printed here. However, the name of the gene product is specified in the Common-Name slot of a frame that represents the gene product. If you want to edit that frame, click the Edit button.
- Product-Types: Specifies the one (or more) broad categories of function of the gene product, when known.
- Evidence: Specifies the type of evidence used to infer the function of the gene product.
- Links to Databases: Defines Web-based links to information about this gene in other databases. Select a database to link to, and enter the unique identifier assigned to this gene in that database.
- Gene Location: The position of the gene on the chromosome is specified by giving its left-end position and right-end position (the former must always be the smaller number, unless the gene spans the origin of replication), and by specifying the direction of transcription for the gene. The sequence indicated by that position and direction is displayed by the editor.
- Add History Note: A history note is a text comment that is marked with the name of the user who created it, and the date. We recommend that when information about a gene is revised (such as its function), you should create a history note explaining the rationale for the change. Click this button to create a new history note, which will be displayed in the window for this gene by the Navigator.

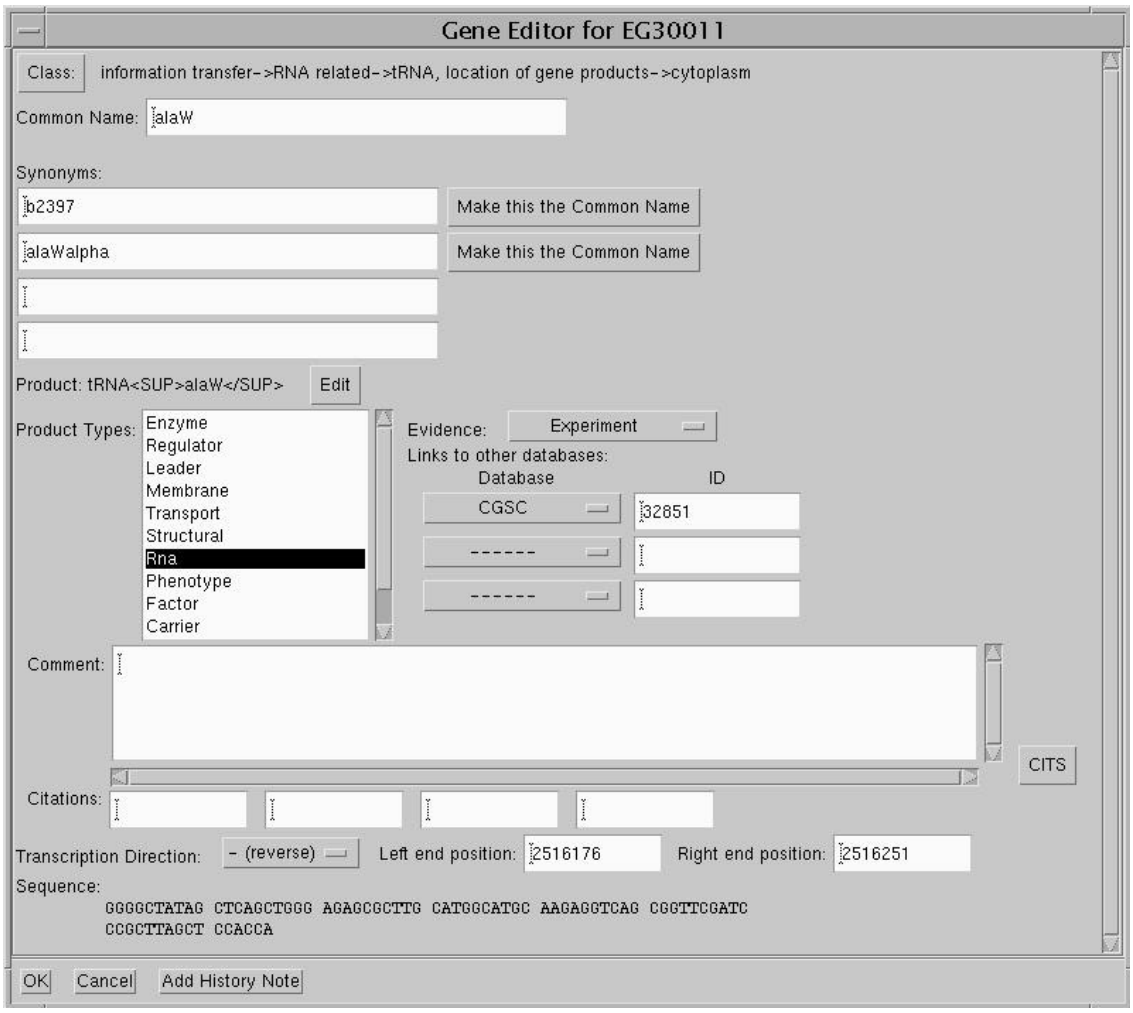


Figure 2-3 Gene Editor

2.3.2 The Transcription Unit Editor

Definition: A *transcription unit* is a region of DNA in a prokaryotic organism that includes a single transcription start site (TSS), the transcription-factor binding site(s) that modulate the rate of transcription initiation at that transcription start site, and the gene(s) and transcription terminator(s) that are downstream of that transcription start site. There is a one-to-one relationship between transcription units and transcription start sites. A transcription unit differs from an operon because by definition an operon must contain more than one gene, whereas a transcription unit may contain one or more genes; an operon may also include more than one transcription start site, whereas a transcription unit must contain a single transcription start site (when known), by our definition. Therefore, our approach defines a different transcription unit for each transcription start site in an operon containing multiple transcription start sites.

The transcription-unit editor is a set of dialog windows that, given a preexisting set of genes and transcription factors, allows you to define frames that describe the complex set of molecules and interactions that yield the genetic network of an organism, including:

- Transcription units
- Transcription-factor binding sites
- Transcription terminators
- Molecular interactions between
    - Small-molecule ligands and transcription factors that bind to those ligands
    - Transcription factors and the DNA binding sites they recognize
    - RNA polymerases and promoter regions (including modulation of those interactions by transcription factors and their associated binding sites)

The stages in defining a genetic-network fragment centered around a single transcription unit are as follows:

2.3.2.1 Step 1: Create Transcription Unit

Create a new transcription unit with the command **Gene New Operon**. A dialog box such as that shown in Figure 2-4 allows you to enter information such as a name and synonyms for the transcription unit, an ordered list of the genes it contains, and the type of evidence used to infer the existence of the transcription unit.

With this dialog, you can also enter a name and synonyms for the transcription start site associated with the transcription unit, and you can define its position as either an absolute nucleotide position on the chromosome, or as a position relative to the start codon of the first gene in the transcription unit. The dialog window displays the sequence corresponding to that region of the chromosome. You can also specify the type of evidence for the existence of the transcription start site, the sigma factor required for RNA polymerase to recognize this promoter, and certain numerical constants for the promoter. The dialog creates a frame describing the interaction between RNA polymerase and this promoter.

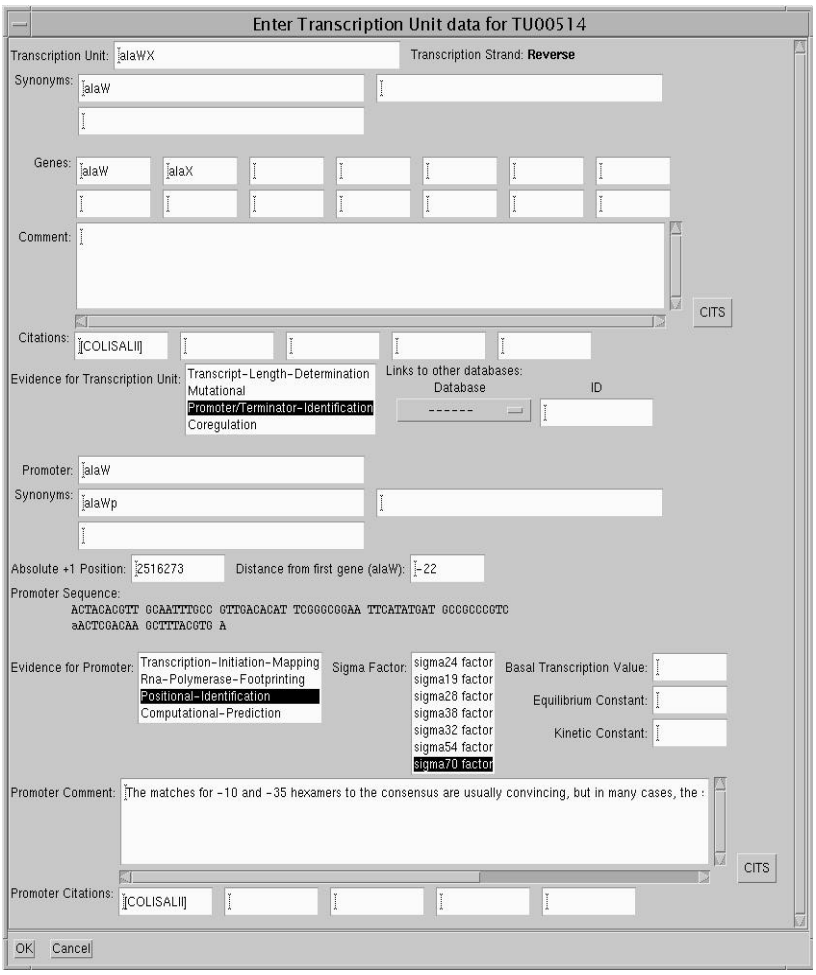


Figure 2-4 Transcription Unit Editor

Once you click OK and exit, the transcription unit is created. If one or more transcription terminators are known, you can add them to this transcription unit by right-clicking the Transcription Unit and selecting Edit Terminators, which brings up the dialog shown in Figure 2-5. This dialog creates frames for the one or more terminators that you specify, and associates them with the transcription factor.

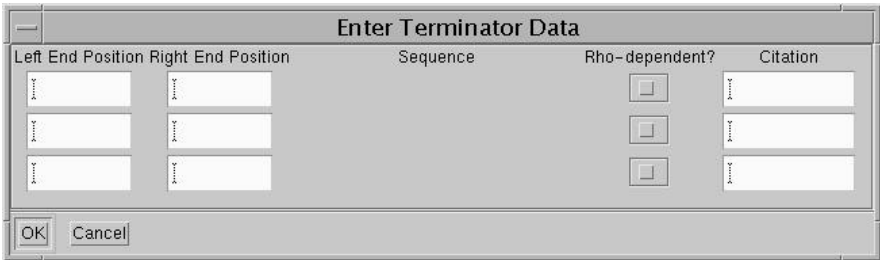


Figure 2-5 Terminator Editor

To create a new transcription-factor binding site associated with this operon, right-click the Transcription Unit, and select Edit/Create Binding Interaction. The Regulatory Interaction Editor (Figure 2-6) opens. There you can enter the name of the binding site and its synonyms. To

specify the protein that binds, you can either select a protein that is already in the database, or create one by clicking the Create button, which takes you to the Protein Editor (Section 2.3.4).

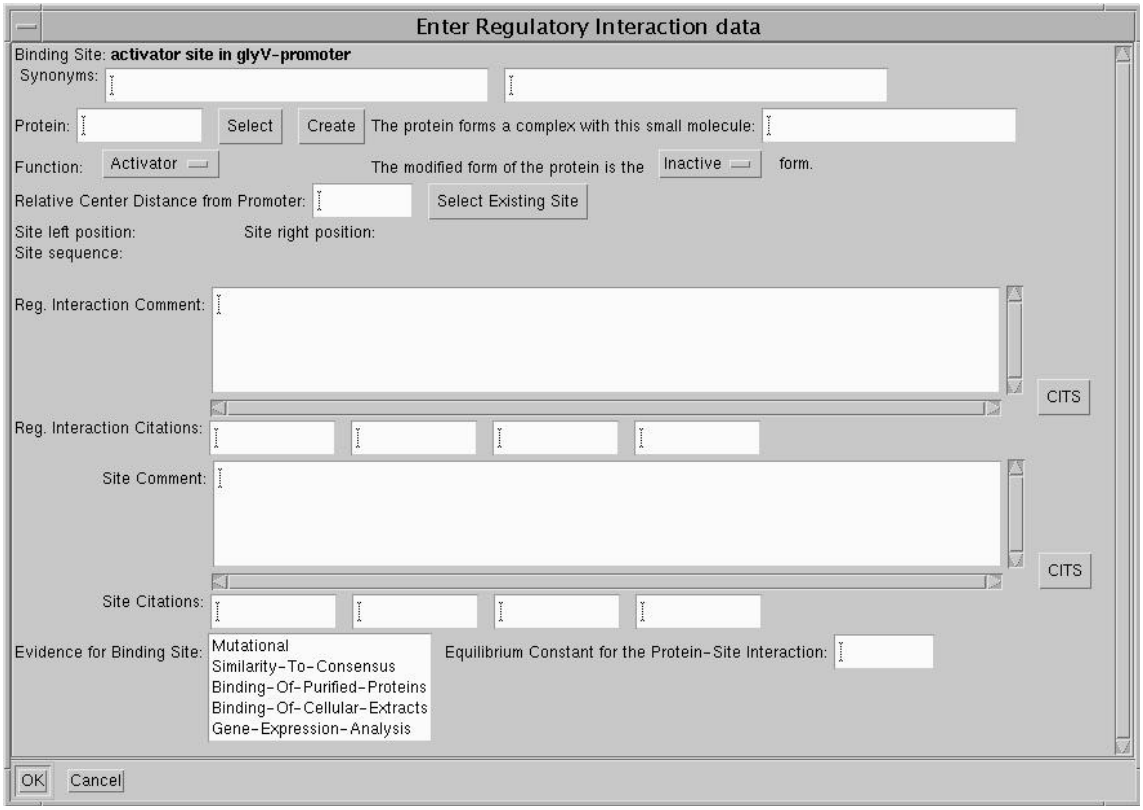


Figure 2-6 Regulatory Interaction Editor

2.3.3 The Intron Editor

The Intron Editor, shown in Figure 2-7, is available only in PGDBs for eukaryotic organisms. You can invoke it by right-clicking on a gene and selecting menu **Edit -> Intron -> Editor**. It contains the following three fields:

- Splice Form Gene Product: This is a menu of gene products associated with each splice form for a gene. To edit the introns for a particular splice form, select the corresponding gene product. To create a new splice form and specify its introns, select the Create New Splice Form option. You are then asked to enter a name for the product of the new splice form. A new polypeptide or RNA frame is created with that name, and is linked to the gene.
- Relative/Absolute Base Numbers: Indicate whether you will be entering base numbers that are relative to the start of the gene or that represent absolute positions in the chromosome.
- Intron positions: Enter start and end positions for each intron. Introns may be entered in any order. If an alternate splice form has a start position inside the gene, specify an intron with start position 1 (relative) and end position one base before the actual start of coding. Create an analogous intron if the splice form has an end position inside the gene. As intron positions are entered, some basic checks are run, and you are alerted to any

possible problems that are identified. These are warnings only, and can be ignored at your own risk.

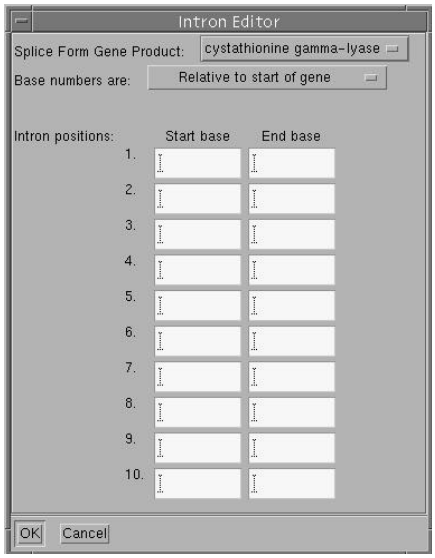


Figure 2-7 Intron Editor

2.3.4 The Protein Editor

Two different editors are used to define and edit a protein, the *Protein Subunit Structure Editor*, and the *Protein Editor*. You can select either of these editors individually from the right-click edit menu for a protein. When you create a protein, you are asked to fill in both forms, in turn.

The *Protein Subunit Structure Editor*, shown in Figure 2-8, is used to specify the genes and subunits (if any) that compose the protein. First, specify whether the protein you want to describe is active as a monomer, or as a protein complex (multimer). If it is a multimer, you should specify the number of distinct gene products that comprise the complex (not the number of copies of each gene product), and then enter the names or IDs of each gene product or subunit.

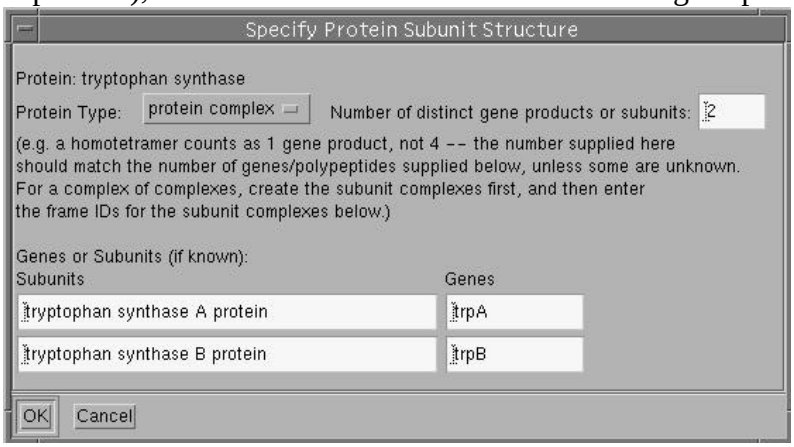


Figure 2-8 Protein Subunit Structure Editor

To create a multi-enzyme protein complex whose components are themselves protein complexes, first create the individual component enzymes and link them to the reactions they catalyze, if

any. Then, when creating the multi-enzyme complex, enter the names or frame IDs for the subunit complexes in the dialog under Subunits. Do not enter any genes.

The *Protein Editor*, shown in Figure 2-9, is used to specify additional information about the protein. This window is divided into several sections. The top section lets you define general information about the protein such as commentary, citations, database links, and one or more cellular locations known for the protein.

Subsequent sections allow you to specify one or more known enzymatic activities for the protein. Information you can define in each of these sections includes a name for this enzymatic activity (a multifunctional protein will have multiple names, one for each function), and

- The reversibility of this reaction with respect to this enzyme. For example, if the reaction is essentially irreversible in the left-to-right direction, specify it as such. If this enzyme typically catalyzes the reaction in the right-to-left direction, choose "Physiologically unidirectional, right-to-left." If the reaction is reversible, specify it as such. This entry will determine whether this step will appear in the pathway diagrams as a unidirectional or a bidirectional arrow.
- One or more modulators (activators or inhibitors), cofactors, prosthetic groups, or alternative substrates for the enzyme. For each molecule, specify the name of the molecule, its relationship to the enzyme (e.g., cofactor), and whether its interaction with the enzyme is physiologically relevant or not (for example, a molecule that has been demonstrated to inhibit the enzyme in vitro but is not known to occur in vivo should not be marked as physiologically relevant). You can also specify one or more citations in support of this information. If you run out of slots, close the dialog window and reopen it – more empty slots will show up.

Additional sections of this window (not shown) allow you to specify information specific to each subunit of the multimer, including its copy number (coefficient) within the complex, molecular weight, and Swiss-Prot ID.

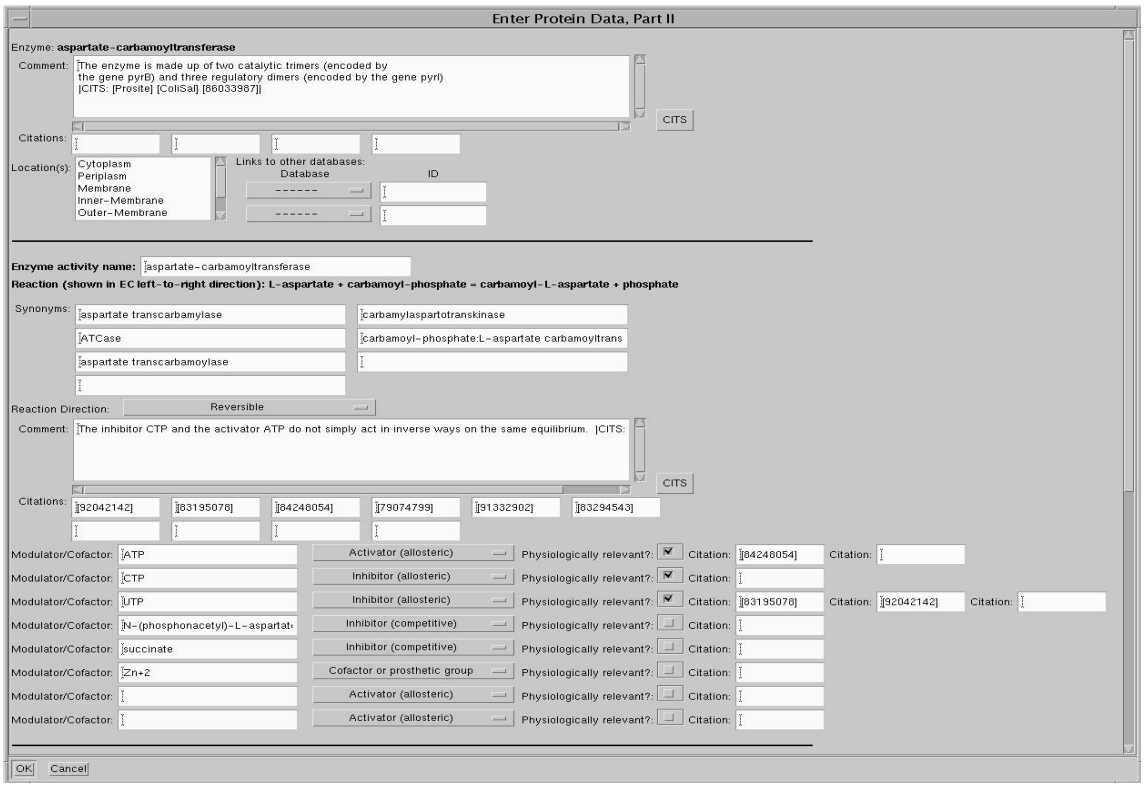


Figure 2-9 Protein Editor

2.3.4.1 Adding an Enzyme to a Reaction

There are two ways to link an enzyme to the reaction it catalyzes. If the reaction and the protein frame have both already been created, right-click on the protein name in a Navigator display, and select **Edit -> Add Reaction(s)**. You can then enter one or more reaction frame IDs or EC numbers. Once the reactions have been linked to the protein, the Protein Editor appears so that you can add information about directionality, activators/inhibitors/cofactors, citations, and so on. The second way to connect an enzyme to a reaction is by right-clicking on the reaction and selecting **Edit -> Create/Add Enzyme**. If the enzyme already exists, and you know the frame name or ID, you can enter it directly. Otherwise, select **Search by Genes or Create New Protein** to bring up the Protein Subunit Structure Editor. You can list the genes or polypeptides (or subunit complexes, for enzymes that are a complex of protein complexes) that make up the enzyme. If a protein already exists that matches what you have entered, it will be linked to the reaction. If a protein complex is specified and no such complex yet exists, it will be created. If you are not sure whether the enzyme is a polypeptide or a protein complex, you can leave the class unspecified while you search for an existing protein. However, if you wish to create a new protein, you must specify the class. Finally, the Protein Editor will pop up to allow entry of additional information about the enzyme.

2.3.4.2 Disconnecting an Enzyme from a Reaction

To disconnect an enzyme from a reaction, right click the enzyme, and in the menu that pops up, select **Edit > Remove Reaction**.

2.3.4.3 Creating Active/Inactive Forms of an Enzyme

Some proteins may be covalently modified in order to activate or inactivate them. Pathway Tools treats the activation/deactivation process as an ordinary chemical reaction, with the protein and its modified form as the reaction substrates. To create a protein activation/inactivation reaction, first read Section 2.4.11 on Modified Proteins and Section 2.3.6 to learn about the Reaction Editor. Enter **Reaction Mode** and select **Create Reaction**. Enter the reaction equation (e.g., XyzA + P_i = XyzA-P) and click **Show Rxn**. In general, if XyzA is recognized as a protein, and XyzA-P does not yet exist as a frame, you are asked if XyzA-P should be created as a modified form of XyzA. Once the reaction has been created, specify which form is active by linking it to a reaction, as described in the previous section. This same process can be used to generate active and inactive forms of a protein for transcriptional regulation, signal transduction pathways and so on.

2.3.4.4 Creating Protein Features

In the Protein Editor section for a polypeptide, there is a button named **Edit Protein Feature(s)** which allows for creating and editing of sequence features of various types. To create a new feature, click the button and select Create New Feature. You will be presented with a hierarchy of possible feature types, such as Phosphorylation-Modifications, Active Sites, Nucleotide-Phosphate-Binding-Regions, etc. If the exact feature type you need is not listed, you may use a nonspecific type. For example, choose Amino-Acid-Binding-Sites (which describe features in which a moiety binds to a single amino acid) if no sub-type exists for the type of attached moiety that you are trying to describe. After selecting the feature type, the Feature Editor, shown in Figure 2-10, will appear. You may use this editor to enter descriptive information about the feature: name, comment, citations, and the attached group if the feature type requires one. The protein sequence is provided, and the feature location can be specified by either typing in the residue number or by selecting it with the mouse (a sequence search box is provided to help in locating the feature within the sequence). You may also supply a sequence motif that may or may not be part of the actual binding or modification site but which is indicative of the feature.

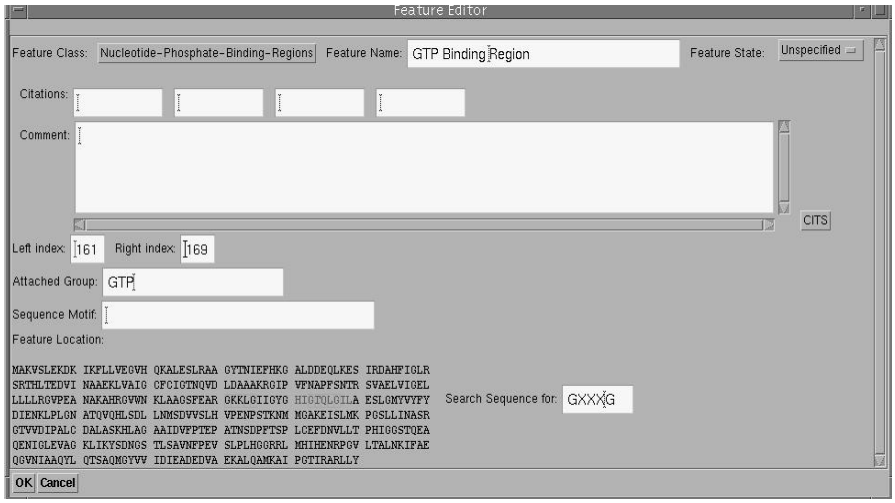


Figure 2-10 The Feature Editor

2.3.5 The Pathway Editors

The Pathway Editors allow you to graphically create new pathways, and to modify existing pathways. A pathway is a connected set of chemical reactions. You can manipulate a pathway by manipulating its component reactions: reactions can be added to or deleted from a pathway, and the connections among reactions can be altered. Reactions are connected by shared substrates. Two reactions share a substrate if a product of one reaction, *R*1, is a reactant of the next reaction in the pathway, *R*2. In this case we say that *R*1 is the *predecessor* reaction of *R*2.

The Pathway Editors consists of three tools: the *Pathway Info Editor* (which can be invoked separately from the other tools), the *Pathway Editor*, and the *Segment Editor*. With the Pathway Info Editor, you can enter non-topological information about a pathway, such as commentary, its list of synonyms, its classification in the taxonomy of pathways, and links to other databases. With the Segment Editor, you can enter a linear pathway segment — a linear sequence of reactions within a pathway. The Segment Editor is typically used to enter new pathways, or to add a new sequence of reactions to an existing pathway. With the Pathway Editor, you can add new reactions to a pathway, remove reactions from a pathway, and alter connections between individual reactions.

The Pathway Editor is always invoked before the Segment Editor; it displays the current form of an entire pathway. The Segment Editor can then be used to add new linear segments to a (possibly empty) existing pathway. A short linear pathway can be entered using one invocation of the Segment Editor. More complex pathways may take several operations:

- To enter a long linear pathway (more than seven reactions), use the Segment Editor to enter several segments and then connect the segments together with the Pathway Editor.
- To enter a branching pathway, use the Segment Editor to enter each linear segment, and then use the Pathway Editor to connect the segments together to form the branching structure.
- To enter a circular pathway, use the Segment Editor to construct a linear form of the pathway, and then connect the ends together within the Pathway Editor.

2.3.5.1 Invoking the Pathway Editor

You can invoke the Pathway Editor either to create a new pathway or to modify an existing pathway.

To create a new pathway, select **Pathway -> New**. The Pathway Info Editor dialog box appears; click **Class** to choose a pathway class in a cascading menu. Note that the general class "Pathways" is chosen by default – make sure to uncheck it when you choose a more appropriate class. When you click OK, a new instance of that pathway class is created, and the Pathway Editor will pop up, letting you specify the individual reactions of the new pathway.
To modify an existing pathway, right click on the pathway handle, and select either the **Edit -> Pathway Info Editor** or the **Edit -> Pathway Editor** command.

To create a super-pathway, create a new pathway as above, using the **Pathway -> New** command. In the Pathway Editor, instead of adding a linear sequence of reactions using the

Segment Editor, or individual reactions using the Reactions commands, you can add whole sub-pathways using the commands in the Pathways menu. Additional reactions or connections between sub-pathways can be added as needed in the usual fashion.

2.3.5.2 The Pathway Info Editor

With the Pathway Info Editor, you can specify the class of pathways to which the pathway belongs, specify the primary name and synonyms for the pathway, and give a comment and citations for the pathway. You can create links to this pathway in other pathway databases. You can also specify that one or more of the reactions in the pathway are considered hypothetical, generally because presence of the enzyme, reactants, or products has not been experimentally demonstrated.

2.3.5.3 Pathway Editor Operations

The Pathway Editor contains two main display panes (see Figure 2-11). The left pane shows reactions that have been added to the pathway but have not yet been connected to other reactions. Reactions are listed by frame ID, EC number, and reaction equation. The right pane draws the connected reactions of the pathway. With the Pathway Editor, you can add new reactions to the pathway, connect together reactions in the pathway, and delete reactions from the pathway.

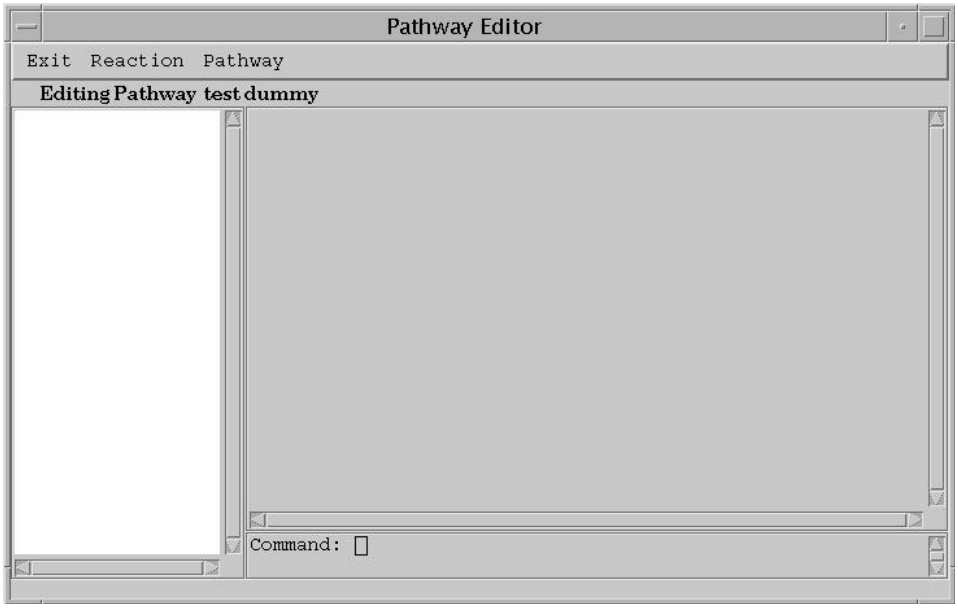


Figure 2-11 Connection Editor

The mechanism for connecting reactions is as follows. First select a *predecessor* reaction (in either pane) by clicking on it (or on its terminal compound in the right pane). That reaction is highlighted in red; all reactions that can potentially follow it, based on substrate overlap, are highlighted in green. Click on one of the green *successor* reactions to indicate that the green reaction should directly follow the red reaction, or on the background to cancel the operation. The panes are updated to reflect the change. Until a reaction is actually linked to another reaction in the pathway, its direction is unspecified, so you may see more reactions highlighted in green

than you would have expected (this can also be true if the reaction has commonly occurring substrates such as ATP).

Unconnected reactions from sub-pathways are displayed in the right pane. They can be manually connected to reactions in the super-pathway or any sub-pathway(s). To keep the integrity of sub-pathways, reactions that originally came from sub-pathways are not allowed to be disconnected or deleted.

The Pathway Editor contains additional commands.

- Exit
    - **Keep Changes:** Exit the Pathway Editor, transferring all changes made here back to the PGDB.
    - **Abort Changes:** Exit the Pathway Editor without transferring any of the changes made to the pathway back to the PGDB. However, if any new reaction frames were created or edited during the session, those changes will *be maintained*.
- Reaction Any of the following commands that require you to first select a reaction are also accessible by right-clicking on the reaction.
    - **Add Reaction:** Provides you with several alternative ways of specifying a reaction to add to the pathway. In a dialog window, you specify the reaction by EC number, by frame ID, according to the name of the enzyme that catalyzes the reaction, or according to the substrates of the reaction. In the latter case, the program finds the one or more reactions containing the specified substrates. If no reactions contain the specified substrates, the Reaction Editor is invoked to create a new reaction (see "The Reaction Editor" in section 2.3.6). The specified reaction is then added to the left pane of the Pathway Editor.
    - **Add Reaction(s) from History:** Pops up a list of all reactions on the Pathway Tools history list. Those selected are added to the pathway and appear in the left pane of the Connections Editor. This offers a convenient way to add reactions to a pathway: find and display all desired reactions through the Navigator *before* invoking the Pathway Editor, then the desired reactions are at the front of the history list and are easy to select.
    - **Create New Reaction Frame:** Invokes the Reaction Editor (see section 2.3.6) to create a new reaction, which is added to the pathway.
    - **Clone a Reaction Frame:** You are prompted for the reaction frame (frame ID or EC number) that is to be copied and a new frame ID for the resulting copy. The new reaction frame is created as a copy of the specified frame and added to the pathway.
    - **Add Connection:** Connects a reaction to its predecessor.
    - **Delete Predecessor/Successor Link:** Prompts for a reaction and pops up a list of connections between the reaction and its predecessors and successors. You can select any or all of these to delete. If any reaction becomes completely disconnected as a result of this operation, it is removed from the right pane and redrawn on the left pane.

- **Disconnect Reaction:** Prompts for a reaction and deletes all predecessor and successor links to or from the reaction. The reaction and any others that become disconnected reappear in the left pane.
- **Delete Reaction from Pathway:** Prompts for a reaction and removes it from the pathway. This does *not* delete the reaction frame from the DB.
- **Choose Main Compounds for Reaction:** The Pathway Editor automatically infers the main and side compounds for each reaction in the pathway based on substrate overlap between reactions and other heuristics. Use this command if you want to specify different or additional main compounds for a reaction. Because reactant and product mains are specified separately, this command also provides a mechanism for specifying the direction of a reaction if the direction is ambiguous. Thus, this dialog also pops up on occasion when a reaction direction cannot be inferred directly from its predecessor/successor links.
- **Edit Reaction Frame:** Prompts for a reaction and pops up the Frame Editor window for the reaction.

• Pathways

- **Enter a Linear Pathway Segment:** Invokes the Segment Editor to enter a linear pathway (see "Pathway Segment Editor" in section 2.3.5.5).
- **Guess Pathway Predecessor List:** This command attempts to guess how the reactions within the pathway should be connected based on some simple substrate overlap heuristics. Any previously specified predecessor links are ignored. While this command can save time when constructing simple pathways, it may produce incorrect results for more complicated branching pathways. Be sure to check the resulting pathway for accuracy.
- **Disconnect All Reactions:** Removes all predecessor links between any reactions in the pathway. All reactions are moved back to the left pane, and the right pane is blank. This command is useful when a pathway is misconnected and it is easier to start rebuilding it from scratch rather than attempting to fix the links.
- **Invoke Relationships Editor:** Brings up a window containing the Relationships Editor view of this pathway and its relationship to other frames in the DB. This editor is a very useful way to visualize the web of relationships between all frames involved in one pathway.
- **Add Subpathway by Name:** A dialog prompts you for a frame ID of a pathway. A menu shows the pathway with that frame ID that you can choose to add.
- **Add Subpathway by Substring:** A dialog prompts you for a substring of the common name or synonym of the pathway. A menu shows all the relevant pathways, from which you can choose one or more to add.
- **Add Subpathway by Class:** A cascading menu allows you to choose one or more metabolic pathways by first selecting one class from a menu of pathway classes, and then one or more pathways from a menu of all pathways in that class.
- **Delete Subpathway:** A menu shows all the subpathways in the current superpathway. You can choose one or more subpathways to delete.

In addition, right-clicking on a compound name brings up a menu containing whichever of the following commands are applicable:

- Add Link from/to Pathway: A pathway link indicates that a particular substrate in a pathway either comes from or feeds into some other pathway. It is displayed using an arrow connecting the pathway substrate to the name of the linked pathway. Links are one-way; because a link is created in the display of Pathway A from Compound X to Pathway B, it does not mean that a link should also appear in the display of Pathway B connecting Compound X to Pathway A. Use this command to create pathway links to or from the selected compound in the display for the current pathway. A cascading menu allows you to choose the pathway to which the link is to be created. The software itself determines the directionality of the link. To delete a pathway link created in this fashion, right-click on the link arrow and select Delete Link.
- Place this Compound at Cycle Top: This command is available for compounds within a cycle. Selecting this command causes the pathway to be redrawn with the indicated compound at the top of the circle.
- Create Polymerization Link: This command is available when the software detects the possibility for a polymerization cycle. Invoking this command joins the selected compound to its other instantiation within the pathway by a polymerization link.
- Delete Polymerization Link: Deletes a polymerization link created by the previous command.
- Assign Polymerization Name: When two different instantiations of a compound exist in a pathway, connected by a polymerization link, this command can be used to assign the display name for each instantiation by selecting from a menu of possibilities. The list of possibilities is derived from the values of the slots COMMON-NAME, N-NAME, N+1-NAME, and N-1-NAME for the compound.

2.3.5.4 Pathway Editor Limitations

In general, the order in which you specify links to generate a pathway is unimportant. However, in some complicated situations, specifying one link before another may introduce ambiguity in the partially constructed pathway. Depending on the situation specifics, one of several things might happen: the link may be ignored, a dialog box may pop up asking you to disambiguate, or the pathway may be drawn in a bizarre arrangement. If any of these occur, try removing the offending link and adding the links in a different order. In general, links that introduce the least ambiguity should be specified first. For example, if our pathway consists of three reactions, R1: A = B, R2: B + C = D, and R3: D + C = E, and the first link we specify is from R2 to R3, then the direction of R2 remains ambiguous and we may see unpredictable results. If, however, the first link we specify is from R1 to R2, then we can infer that R2 must proceed from left to right and can safely add the link R2 to R3 later.

2.3.5.5 Pathway Segment Editor

With the Segment Editor (see Figure 2-12), you can enter a linear sequence of connected reactions in a faster manner than by adding reactions one-by-one through the Pathway Editor. Each reaction is specified using either its EC number — which when known provides a fast

shorthand for the reaction — or using the reaction substrates. One segment can contain as many as seven reactions.

Each reaction is depicted in the Segment Editor by a straight arrow that connects the main substrates of the reaction, plus a curved arrow that connects the side substrates of the reaction. To specify each reaction in the sequence, either enter its EC number in the box to the left of the arrow, or enter the side and main substrates in the appropriate boxes above, below, and to the right of the arrow. Only one box is provided to enter each of the side reactants and the side products, respectively. Multiple side reactants (or products) can be entered by separating them with the "/" character.

Several types of correctness checking are performed by the Segment Editor. As soon as you enter an EC number or a substrate name, the Editor verifies whether they are defined in the PGDB or in MetaCyc. If they are unknown, pop-up windows indicate that condition. Incorrect EC numbers or substrate names can be altered by retyping them within their box.

In general, it is highly preferable to construct new pathways from existing reaction and compound frames in the PGDB, to minimize data redundancy and data entry. Therefore, every effort should be made to find existing objects before creating new ones. If a chemical compound is not found using the first name you try, try again using other names for the compound, or using substring searches for a root of the compound name. If you still cannot find the compound within the PGDB, you should create it. Repeated searches and compound creation can be performed within the Compound Resolution Tool (see "Compound Resolution Tool" in section 2.3.6.3). You can invoke that tool by clicking the **Search/Create** button in the dialog window that informs you a compound name is unknown.

Once you have successfully entered EC numbers and substrate names, you must perform a second phase of checking by clicking the **Check** button at the bottom of the Segment Editor. The checks performed in this phase include

- Checking consistency of EC number, reaction substrates, and reaction directions
- Matching substrates for each specified reaction against reactions stored in the PGDB. If a specified reaction is not found, the Reaction Editor (Section 2.3.6) is invoked to create it.

Reactions for which a problem is found are drawn in red; correct reactions are drawn in green. Modify red reactions to correct the indicated problem.

When all reactions are green, click the **Done** button to exit the Segment Editor. You cannot exit until you have used the **Check** button to perform checking. You can click **Cancel** at any time to abort the session, which will abort construction of the pathway segment.

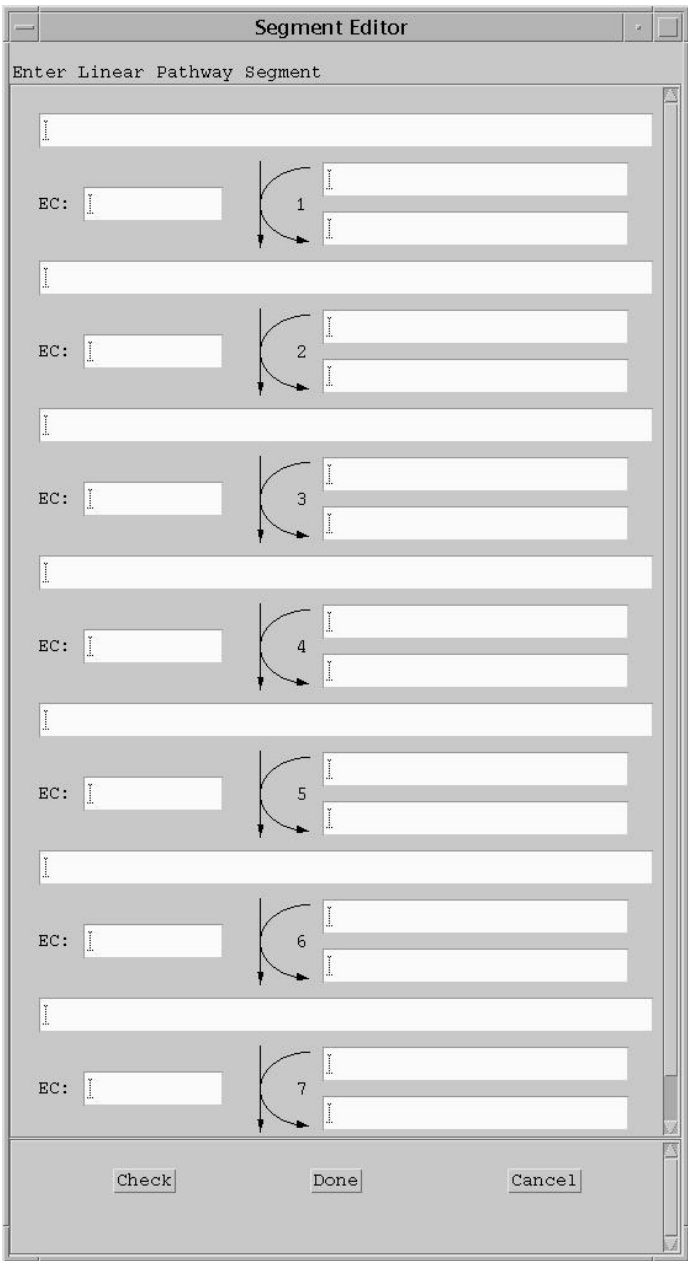


Figure 2-12 Segment Editor

2.3.6 The Reaction Editor

The Reaction Editor facilitates the creation of a new reaction frame in a PGDB. Reaction frames contain links to frames for the chemical compounds that are the building blocks of reactions.

The literature often refers to the same chemical compound using many different chemical names. It is desirable that the PGDB not contain multiple frames that describe the same chemical compound using different names. MetaCyc contains extensive lists of compound synonyms that will help you identify an already known compound, but in some cases, chemical names that you enter will not be known. The Reaction Editor contains a tool called the Compound Resolver for

helping to find existing compound frames in a newly entered reaction equation, while avoiding retyping of synonyms if a new compound frame has to be created, after all attempts to find an existing one have failed. As reactions are usually created in conjunction with adding more information about the enzyme involved in the transformation, the Reaction Editor can also automatically create the enzymatic-reaction and protein frames and crosslink them by the appropriate relations.

2.3.6.1 Invoking the Reaction Editor

To create a new reaction, select **Reaction -> New** from the Reaction menu.

Reaction editing occurs in two phases: first, entering basic information such as the equation, EC number and frame names; second, identifying those chemical compounds that are already defined within the PGDB, and those for which new frames must be created.

2.3.6.2 Entering Reaction Equations

The Reaction Editor dialog requests several pieces of information:

- EC number: If the EC number for the reaction is known, enter it. A valid entry consists of four consecutive numbers (or dashes, standing for currently unspecified categories), which are separated by one dot each (e.g. 6.3.2.24).
- Official EC? Mark the checkbox only if the reaction is 100% identical to the official EC reaction. You can specify multiple reactions with the same EC number, but only one reaction in the DB can be the official EC reaction. This is useful when the EC reaction is a general one, and you need to specify a reaction which constitutes a special case of this general reaction.
- Reaction Equation: Enter the reaction equation in the form A + B = C + D. The "+" signs separate individual compound names, and the "=' sign separates the left and right sides. The operators "+" and "=" must be surrounded by spaces. Specify transport reactions that involve translocation between periplasm and cytoplasm by suffixing the name of the compound that is located within the periplasm with [periplasm]; e.g., A[periplasm] + B = C + D. When displayed, such reactions show the translocation.
- Spontaneous? Mark the checkbox only if the reaction is spontaneous.
- Balance State: Select the appropriate state regarding what is known about the reaction equation in terms of balancing all atoms on the left with those on the right side. This is important primarily if the reaction can not be balanced, for example because the full equation is unknown at the time.
- Citations: Enter as many as four citations, which can be any combination of PubMed IDs, citation IDs that are already in the current database, or citation IDs for citations that you are ready to put into the database.
- Show Rxn: Clicking this button starts the compound resolution phase (see "Compound Resolution Tool" in section 2.3.6.3), which pops up the second panel if there are still compound names that have not been automatically resolved. To use Compound Resolution, at the very minimum the Reaction Frame ID and the Reaction Equation will need to have been entered.
- Balanced? To test whether the reaction is mass balanced, click on this button. If all the reaction substrates contain a chemical formula, and the reaction is unbalanced, a pop-up

window shows the difference in atoms between the reactants and products. A reaction may be unbalanced because of an error in the reaction equation or an error in a compound structure.

- OK: After all editing is completed, click the OK button to create and cross-link all the remaining frames (unless they already existed), and to exit the Reaction Editor.
- Cancel: Clicking this button does not create the specified frames, and deletes the new reaction frame, and any newly created compound frames as well.

2.3.6.3 Compound Resolution Tool

This tool is entered from the Reaction Editor or Pathway Segment Editor when a reaction contains compound names that could not be identified in the PGDB. A new dialog appears, presenting a choice of the names that were still unresolved or that were completely ambiguous, meaning that more than one compound contained the same string as its common-name.

"Resolving a compound name" means matching it unambiguously to a compound frame in the PGDB by using successive searches to find the compound under different names or name fragments. For example, you might encounter the name "N2-succinyl-L-arginine" in an article, but a search under that name does not yield a match. You might then try searching using the substrings "succinyl" or "arginine," and find that the compound is indeed already in the PGDB under the name "N2-succinyl-arginine," and then add the synonym to the existing compound. To resolve a name, click on it, and enter an alternative synonym or substring. Click on **Exact Match** to search for a compound with exactly that name; click on **Substring Match** to search for a compound whose name contains the specified substring.

If no match was found, the string that was entered in the text line is transferred into the box that collects all the synonyms for this compound, ensuring that all the entered synonyms are retained, to save retyping them later.

If matching frames were found, they are mentioned in the Messages section of the panel. If multiple frames were found, a list pane allows selection of any one of the hits. Simultaneously, the selected compound is displayed in the lower half of the main Pathway Tools window. This should help you see whether the compound is the desired one. Two buttons exist for accepting the compound. Either the collected synonyms are transferred to the selected compound frame, or not. It can be useful to transfer the synonyms to allow future retrieval of this compound by these new names, which have previously failed, even though they ought to be valid names under which the compound should be findable.

At all times, it is possible to create a new compound frame from scratch, if you conclude that this compound truly does not exist in the PGDB. Compound creation transfers all the synonyms collected so far, to avoid having to retype them. The first of these synonyms, which was the name originally entered as part of the reaction equation, becomes the common-name of the new compound frame. A pop-up window allows entering of a frame name, and the first synonym is again suggested as a suitable default.

2.3.6.4 Limitations of the Compound Resolver

Currently, compound frames that were modified by adding synonyms during compound resolution are not reverted to their unmodified state, after you press the **Abort** button. Only frames that were created from scratch, both compound frames and the new reaction frame, are deleted in an abort.

The "Protein Complex?" button is not yet properly connected, so that its change does not percolate through automatically to the protein frame ID, to switch between the **MONOMER** and **CPLX** suffixes. You must type in the correct suffix. However, this button does control linking of the new frame into the right class at frame creation time (when **Done** is pressed).

2.3.6.5 Transport Reactions

Transport reactions can be specified and displayed in a special way. See section 2.3.6.2, "Entering Reaction Equations".

2.3.7 Chemical Compound Editor

Chemical compounds are the small-molecule substrates for metabolic reactions, as well as activators, inhibitors, or cofactors for enzymes. To edit an existing compound, right-click on a compound handle and choose **Edit -> Compound Editor**; to create a new compound, click **Compound -> New**.

With the compound editor (see Figure 2-13) you can specify the class of chemicals to which the compound belongs, specify the primary name and synonyms for the compound, and add a comment and citations for the compound.

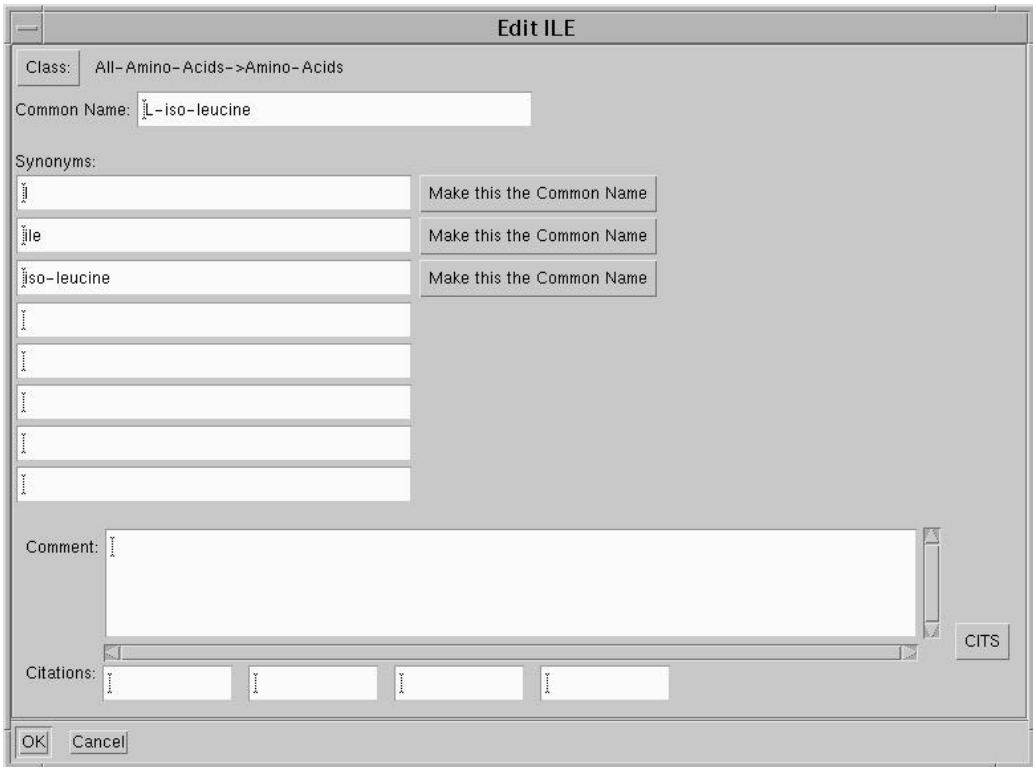


Figure 2-13 Chemical Compound Editor

2.3.8 Marvin Compound Structure Editor

An additional editor for compound structures is now supported starting with Pathway Tools version 9.5, which is more feature rich and easier to install than the JME editor described in the next section. The new editor consists of an interface to Marvin, a popular Java applet. Its key advantages are:

- support for stereochemistry with wedged bonds
- smarter in handling the way the Navigator represents aromatic bonds, so it will not need the OpenBabel software, simplifying the installation, compared to the JME editor
- atom aliases allow broader flexibility in naming atoms: They can be called R, R1, R2. Even frame IDs for proteins, such asACP, can be used to represent modified protein structures.
- easier to obtain

2.3.8.1 Installation

Before this editor can be used, one piece of external software needs to be obtained and installed properly. Marvin is not included in the Pathway Tools distribution.

Marvin MSketch (version 3.5.6 is known to work): Please download this from http://www.chemaxon.com/marvin/ . Obtaining and using MSketch is free of charge for most classes of users, though it is a commercial product. The distribution file will look like

marvin-bin-3.5.6.tar.gz or **marvin-bin-3.5.6.zip** (the version number will probably be higher). It should be unzipped into a newly created directory in the local file system. On UNIX platforms, the command to execute in a shell would look like
`unzip marvin-bin-3.5.6.zip` or `tar zxvf marvin-bin-3.5.6.tar.gz`

Afterwards, the additional required installation step is to set the shell environment variable MARVIN_PATH to the full directory path in which the MARVIN distribution was unpacked. It is best to do this in one of the user's shell init scripts such as **.cshrc** or **.login**. For example, in csh syntax, it would look similar to
`setenv MARVIN_PATH /example/of/full/marvin/path.`

2.3.8.2 Structure Editing

When the Marvin editor is invoked by right-clicking on a compound handle and selecting Edit -> Marvin Compound Structure Editor, a WWW browser window pops up, in which the Marvin applet will run. The Pathway Tools program starts a local WWW server (currently on port 1557) that is used for transferring data to and from Marvin. The structure of the compound (if any) will show inside the Marvin panel. In order to add to an existing structure or to start drawing from scratch, it is necessary to first select the appropriate tool by clicking on one of the square boxes lining the drawing area at the top. This can be one of several bond types (single, double, triple, and wedged bonds), one of several prefabricated ring systems, or an element type. By moving the cursor in the drawing area and over existing parts of the compound structure, an attachment location can be selected. Marvin will highlight the selected location by temporarily drawing a small circle around it. If an atom is highlighted, a new bond or ring system will be attached to this atom by a left-click on the mouse. Usually, the default suggested geometry for the new bond is acceptable. If it needs to be changed, it is possible to rotate into a different geometry by not releasing the left mouse button until a satisfactory geometry has been reached. An element can be converted to another by clicking on the desired element in the tool box and then selecting it. The tool box labelled "More" allows bringing up a large element panel with an entry box at the bottom, where atom aliases can be entered, for example for indicating an R-group with "R". Charges on an atom can be changed with the buttons labeled + and - . By clicking several times on the atom, all the charges can be iterated through. A ring system can be fused alongside the highlighted bond when a prefabricated ring system was selected and is dragged over a bond.

Stereochemistry is supported by using the two wedged bond types for indicating whether a bond on a stereocenter is pointing "up" (a solid wedge) or "down" (a hashed wedge), relative to the drawing plane. If neither wedge is chosen, the assumption is that the bond is oriented flat in the drawing plane.

If a drawing mistake was made, it can be undone by clicking the "Undo" button. It is always possible to close the WWW browser window entirely to start over, before submitting. Individual atoms and bonds can be deleted by clicking on the "Erase" button, and then on the parts of the compound structure that need to be deleted.

After the structure is fully edited, the "Submit" button below the Marvin panel needs to be

clicked, to cause the new structural information and the coordinates to be sent back to the Pathway Tools. The Navigator window should now show the submitted and updated compound, and the PGDB will have been modified accordingly. If the compound looks fine, the PGDB should be saved, so the changes will not get lost.

It is possible to submit the structure information to a different compound frame than the one used for invoking Marvin, by editing the frame ID in the text entry box next to the "Submit" button before actually submitting. This is useful when a relatively large structure needs to be added to a compound frame, which differs only slightly from some other structure. This can be accomplished by invoking Marvin on the structure which is to be copied, make the modifications, and change the frame ID next to the submit button to the ID of the destination frame before submitting the structure back to Pathway Tools.

2.3.8.3 Limitations

While the interaction with Marvin is generally smooth and more intuitive than with the old structure editor, there currently are a few known problems:

- Lost aromatic heteroatom hydrogens: Differences in the way Marvin and Pathway Tools recognize aromaticity may cause implied hydrogens on aromatic heteroatoms to get lost. In some cases, Pathway Tools appears to not infer hydrogens correctly where it should. An example of such a compound is indole, which needs a hydrogen atom on its nitrogen atom. The work around for this problem is to explicitly draw a single bond from the nitrogen, and to then change the element type at the other end of the bond to an H. Once the structure is submitted to Pathway Tools, the explicit bond to the explicit hydrogen will then be shown correctly and taken into account for computing the empirical formula and molecular weight.
- No coordination bond type: A few compounds in BioCyc contain a coordination type bond to a metal atom. Marvin does not have a corresponding bond type, and when such a compound is edited, the coordination bonds will be converted to single bonds. To change the bond type back to a coordination bond, the user needs to use the old Compound Structure Editor.

2.3.9 JME Compound Structure Editor

A new editor for compound structures is now supported starting with Pathway Tools version 8.5, which is recommended over the old editor described in the next section. The new editor consists of an interface to JME, a popular Java applet. Its key advantages are:

- support for stereochemistry with wedged bonds
- greater ease of use

2.3.9.1 Installation

Before this editor can be used, two pieces of external software need to be obtained and installed properly. They are not included in the Pathway Tools distribution.

- JME applet distribution, version 2004.01 or later: Please request this by email from Peter Ertl at Novartis (peter.ertl@pharma.novartis.com). Obtaining and using JME is free of charge. More information about JME, including the license and instructions for using it, can be found at http://www.molinspiration.com/jme/ . The distribution will arrive as an email attachment in the form of a ZIP archive. It should be unzipped into a newly created directory in the filesystem. On UNIX platforms, the command to execute in a shell would look like unzip jme-0401.zip. Afterwards, the additional required installation step is to set the shell environment variable JME_PATH to the full directory path in which the JME distribution was unzipped. It is best to do this in one of the user's shell init scripts such as .cshrc or .login. For example, in csh syntax, it would look similar to

  ```
  setenv JME_PATH /example/of/full/jme/path.
  ```

- OpenBabel version 1.100.2 or later: This is a format converter between many 2D molecular structure file formats. It is currently needed if a Pathway Tools compound that contains aromatic rings needs to be edited. JME does not currently understand the aromatic bond type directly, and babel is used for converting the aromatic bonds to the alternating single/double bond patterns that JME understands. OpenBabel is completely open source and the source code can be downloaded from http://openbabel.sourceforge.net/. The program needs to be built by following its installation instructions. This should result in an executable program called babel. This program needs to be installed in a directory location that is mentioned in the PATH shell environment variable. If babel is not installed or not found, aromatic bonds will show up in JME labeled with a question mark (?), and it will be necessary for the curator to manually change the bond types of the aromatic substructures to an appropriate pattern of alternating single/double bonds.

2.3.9.2 Structure Editing

When the JME editor is invoked by right-clicking on a compound handle and selecting **Edit -> JME Compound Structure Editor**, a WWW browser window pops up, in which the JME applet will run. The Pathway Tools program starts a local WWW server (currently on port 1557) that is used for transferring data to and from JME. The structure of the compound (if any) will show inside the JME panel. In order to add to an existing structure or to start drawing from scratch, it is necessary to first select the appropriate tool by clicking on one of the square boxes lining the drawing area to the left and top. This can be one of several bond types (single, double, triple, and wedged bonds), one of several prefabricated ring systems, or an element type. By moving the cursor in the drawing area and over existing parts of the compound structure, an attachment location can be selected. JME will highlight the selected location by temporarily drawing a small box around it.

If an atom is highlighted, a new bond or ring system will be attached to this atom by a left-click on the mouse. Usually, the default suggested geometry for the new bond is acceptable. If it needs to be changed, it is possible to rotate into a different geometry by not releasing the left mouse button until a satisfactory geometry has been reached. An element can be converted to another by selecting it and then clicking on the desired element in the tool box. Charges on an atom can be changed with the button labeled +/-. By clicking several times on the atom, all the plausible

charges can be iterated through.
If the cursor highlights a bond, its bond type can be switched upon a left mouse click, or a ring system can be fused alongside the highlighted bond.

Stereochemistry is supported by using the two wedged bond types for indicating whether a bond on a stereocenter is pointing “up” (a solid wedge) or “down” (a hashed wedge), relative to the drawing plane. If neither wedge is chosen, the assumption is that the bond is oriented flat in the drawing plane.

If a drawing mistake was made, it can be undone by clicking the UDO button. However, this appears to be only a one-step undo. It is always possible to close the WWW browser window entirely to start over. Also, the CLR button erases the entire structure from the drawing area. Individual atoms and bonds can be deleted by clicking on the DEL button, and then on the parts of the compound structure that need to be deleted.

After the structure is fully edited, the submit button below the JME panel needs to be clicked, to cause the new structural information and the coordinates to be sent back to the Pathway Tools. The Navigator window should now show the submitted and updated compound, and the PGDB will have been modified accordingly. If the compound looks fine, the PGDB should be saved, so the changes will not get lost.

It is possible to submit the structure information to a different compound frame than the one used for invoking JME, by editing the frame ID in the text entry box next to the submit button before actually submitting. This is useful when a relatively large structure needs to be added to a compound frame, which differs only slightly from some other structure. This can be accomplished by invoking JME on the structure which is to be copied, make the modifications, and change the frame ID next to the submit button to the ID of the destination frame before submitting the structure back to Pathway Tools.

2.3.9.3 Limitations

While the interaction with JME is generally smooth and more intuitive than with the old structure editor, there currently are a few known problems:

- Lost aromatic heteroatom hydrogens: Both JME and Pathway Tools recognize aromaticity and display implied hydrogens on heteroatoms. However, the information about the implied hydrogens is not sent by JME back to Pathway Tools. In some cases Pathway Tools appears to not infer hydrogens correctly where it should. An example of such a compound is indole, which needs a hydrogen atom on its nitrogen atom. The work around for this problem is to explicitly draw a single bond from the nitrogen, and to then change the element type at the other end of the bond to an H. The latter has to be done by selecting the X element type button in JME, which will pop up a little text entry box that would allow changing the element type to other unusual elements. By default, it is already set to H. Once the structure is submitted to Pathway Tools, the explicit bond to the explicit hydrogen will then be shown correctly and taken into account for computing the empirical formula and molecular weight.
- No coordination bond type: A few compounds in BioCyc contain a coordination type bond to a metal atom. JME does not have a corresponding bond type, and when such a

compound is edited, the coordination bonds will be converted to single bonds. To change the bond type back to a coordination bond, the user needs to use the Compound Structure Editor.

- Bond lengths are not adjustable: Newly drawn bonds in JME only come in one standard length. Their angles can be hand-tuned, but not their length.
- Element X lock-up: It appears that after clicking on the X element type button and editing the element name from H to something else, it is no longer possible to change it to a different element. It is necessary to exit the WWW browser completely (not just close the current window), before this behavior is reset.

2.3.10 MDL Molfile Import/Export

The 2D structure of a compound can be exported and imported to/from the widely used MDL Molfile file format, by right-clicking the compound handle in the Pathway/Genome Navigator window and selecting menu items **Edit -> Export compound structure to molfile...** and **Edit -> Import compound structure from molfile...**.

2.3.11 Compound Structure Editor

The Compound Structure Editor is an older editor, and to a large extent has been replaced by the JME editor described in the previous section. One key reason is that this editor does not show nor allow editing of stereochemistry. However, it is still included for the benefit of users who wish to use it.

The Compound Structure Editor contains graphical tools with which you can draw the structure of a chemical compound and store the structure within a DB. You can also use these tools to modify an existing chemical structure. To edit an existing compound's structure, right-click on the compound handle, and select **Edit -> Compound Structure Editor**.

Read this note about modes: The Compound Structure Editor may be confusing until you understand its use of modes. When you select a modal command (indicated by "[MODAL]" where the command is described below), the Compound Structure Editor remains in that command's mode until you right-click anywhere in the window to exit that mode. **Simply clicking on a different command will have no effect.** For example, if you click the command **Add Atom**, then the Compound Structure Editor remains in **Create Atom** Mode until you right-click somewhere. Modes allow you to quickly edit several atoms or bonds, but it is important to remember that you must exit the current mode before you will be allowed to select another command.

The **File** menu contains two commands for exiting the Compound Structure Editor.

- Save and Exit: Exit the Compound Structure Editor, displaying the new chemical structure in the compound display page, and keeping the modified structure (it will not be permanently saved until you run the Save DB command).
- Exit without Saving: Exit without keeping any of the structure modifications made in this session of the structure editor.

The **Structure** menu contains commands that manipulate the structure as a whole.

- Replace: Replace the current drawing with a structure you select by Frame ID from the database.
- Insert: [MODAL] Append another structure, selected by Frame ID, to your drawing; finely position the newly added structure with the mouse.
- Align: [MODAL] Click on a first atom to use as the basis for horizontal and vertical alignment. Then click other atoms to align them to the first atom, using the vertical alignment axis or the horizontal alignment axis, whichever is closer. Right-click to deselect the atom being used as the basis for alignment. You can click another atom to use as the basis for alignment. An additional right-click is required to exit Align Mode.
- Rescale: Enter numerical values to adjust the size of the structure in the X and Y dimensions.
- Rotate/Flip: Select from a list of simple rotations or flips.
- Adjust Bond Length: Automatically adjust the lengths of all unconstrained bonds (i.e., those that can be adjusted without forcing the adjustment of adjoining bonds) to the structure's average bond length. This adjustment can make structures look more regular.
- Adjust Atom Pos: Because it is difficult to manually make a bond perfectly vertical or horizontal, this command can be invoked to help out. When you select this command, bonds that are nearly vertical or horizontal are made perfectly vertical or horizontal, respectively.
- Aromaticity: This command provides the only way to specify aromatic bonds. Prior to running this command, specify alternating single and double bonds in rings that should be aromatic. When you select this command, rings of alternating single and double bonds are converted to aromatic rings.

The **Atom/Bond** menu contains commands for creating and manipulating atoms and bonds:

- Add Atom: [MODAL] Click to specify the position of a new atom, type an element symbol, and press Enter.
- Add Bond: [MODAL] Click the pair of atoms to connect; the bond type is defined as a single bond but can be changed later by using the Change Bond command.
- Delete Atom/Bond: [MODAL] Click atoms or bonds in any order to delete them.
- Change Atom: [MODAL] Click an atom, type a new element symbol to replace the current one, and press Enter.
- Change Bond: [MODAL] Click bonds and choose the type of bond from a list. Allowable values are single, double, triple, and coordination. Aromatic bonds are handled separately by the command Aromaticity.
- Move Atoms: [MODAL] Click on each of the atoms to move. Then click to specify the new location of the last atom you had clicked. All the clicked atoms will move in unison.

2.3.12 Publication Editor

When a PGDB frame cites a publication that does not have a PubMed ID, the reference for that publication must be defined as a publication frame within the PGDB. Every such publication frame must have a unique identifier (ID), such as SMITH95. That unique ID is used to refer to the publication frame in other editors, for example, to cite the article by Smith you can enter the text “[SMITH95]” in the Citations field within the protein editor. The publication editor can be

used to create new publication frames, and to find the unique ID of publication frames when you do not know the unique ID.

2.3.12.1 Creating New Publication Frames

You can create new publication frames in two ways.

- From an editor such as the protein editor, type the new unique ID for the publication frame you want to create in a citation field (e.g. JONES98) and then click on a different field. A dialog window will show up and give you two choices: (1) search for existing publications or create a new publication, or (2) return to the protein editor. If you selection option (1), the Publication Editor (Figure 2-14) appears to accept the definition of the new publication.
- From the right-click menu of most objects, select Edit -> Create frame -> Publication. The Publication Editor (Figure 2-14) pops up to accept the definition of the new publication.

Within the publication editor, enter the identifier that you will use to cite the publication, for example, **SMITH95** (use all uppercase, with no spaces or punctuation). Then enter other information such as author, title, and source (journal, volume, and pages) in the appropriate fields, and click **OK** when done.

Until the editor is changed, you may need to use the frame editor of the publication frame to enter the year of publication.

2.3.12.2 Finding Existing Publication Frames

If you are inside an editor such as the protein editor and you want to find the unique ID for an existing publication frame, simply enter a random citation name such as “[XXX99]”. A dialog window gives you two choices: (1) search for existing publications or create a new publication, or (2) return to the protein editor. If you selection option (1), the Publication Editor (Figure 2-14) appears. Type in the title or the surname of an author, and click **Search for Existing Publications Frame**. A menu of matching publication frames appears.

When you close the dialog, either by clicking **OK** or by selecting an existing publication from the provided menu, the ID of the publication frame is printed both to the listener pane and to the terminal window, so that you can use it in your citations.

Figure 2-14 Publication Editor

2.3.13 Editing Examples

Examples of editing tasks illustrate the procedures involved.

2.3.13.1 Changing Annotation for a Gene

Imagine that you have discovered a new function for a gene that is currently annotated as an ORF in the PGDB, and you want to alter the gene frame to reflect the new function you have assigned. Perform the following steps.

- Search for the gene in the Navigator (Gene -> Search by Substring) to display the gene (we will use gene yaaA in B. subtilis as an example).
- Right-click on yaaA in the title bar, and select Edit -> Gene Editor to invoke the Gene Editor.
- Imagine that the gene product is a cell-division protein that forms cytoplasmic filaments due to its similarity to the E. coli gene cafA. First, we change the gene name to cafA by erasing yaaA in the Common Name field and typing "cafA" in its place.
- We want to keep the name yaaA as a synonym for this gene, so type "yaaA" in the first Synonyms field.
- Change the classification for the gene by clicking the Class button. A window pops up showing the gene classification hierarchy. Expand the "cell processes" class and select "cell division". Other terms may be selected also. When done, click OK.
- Next we want to insert a history note on this gene to record the rationale for our change. Click on the Add History Note button at the bottom of the dialog box. Enter the text of the note in the pop-up window (e.g., "Function as cytoplasmic filament protein inferred due to very significant (P=10E-50) Blast hit to E. coli cafA") and click OK.
- Click OK in the Gene Editor dialog to exit. The gene is redisplayed with its new name.
- Next we want to alter the name assigned to the gene product. Right-click on "YaaA" in the line showing the gene product, and select Edit -> Synonym Editor to invoke the

Synonym Editor. (We could have used the Synonym Editor to change the gene name, too, but could not have added the history note that way.)
- Change "YaaA" in the Common Name field to "cytoplasmic filaments", and click OK.

2.3.13.2 Changing Annotation for an Enzyme Gene

The preceding example was relatively simple because the gene in question coded for a structural protein rather than for an enzyme. Enzymes are more complex to encode in a PGDB because we use a combination of frames to encode the function of an enzyme.

Imagine that we want to change the annotation of the *B. subtilis* gene *yhfR* from the product phosphoglycerate mutase to the product phosphoglycerate kinase. To do so we would perform the following steps.
- Search for the gene in the Navigator (Gene -> Search by Substring) to display the gene yhfR.
- Use the preceding procedure to change the gene name (if desired) and to change the protein name.
- First, remove the reaction that this gene product is currently connected to. In the gene window, right-click on the reaction equation "3-phosphoglycerate = 2-phosphoglycerate" and then select Edit -> Detach Enzyme(s).
- A list of enzymes for the reaction appears. Select the appropriate one (currently named "putative phosphoglycerate mutase (glycolysis)", but if you have changed the name, the new name appears in the menu instead) and click Use These Values. The reaction (now with only one enzyme) is displayed in the Navigator window.
- Now you should connect the product of yhfR to the reaction for phosphoglycerate kinase, EC# 2.7.2.3. Display this reaction in the Navigator by selecting Reaction -> Search by EC#, and typing in 2.7.2.3.
- Right-click on the EC number in the title, and select Edit -> Create/Add Enzyme.
- Type in the name of the product of yhfR (since YhfR is a synonym for the protein, you can type that instead of a longer name this will also serve to disambiguate between proteins if, for example, you changed the name of the gene product to "phosphoglycerate kinase", since there was already a protein by that name in the PGDB). Click OK.
- The Protein Editor appears. If you have any additional information to add, such as information about activators or inhibitors, or a citation or comment, you can enter it here. In this example, you should add an evidence code for this enzymatic activity. The appropriate code is EV-COMP-HINF-FN-FROM-SEQ. When you are done, click OK. The reaction is redisplayed, showing the link to the new enzyme.

2.3.13.3 Entering a New Pathway

Imagine that we want to enter the pathway for phenylalanine biosynthesis, as shown in Figure 2-15, into a PGDB.

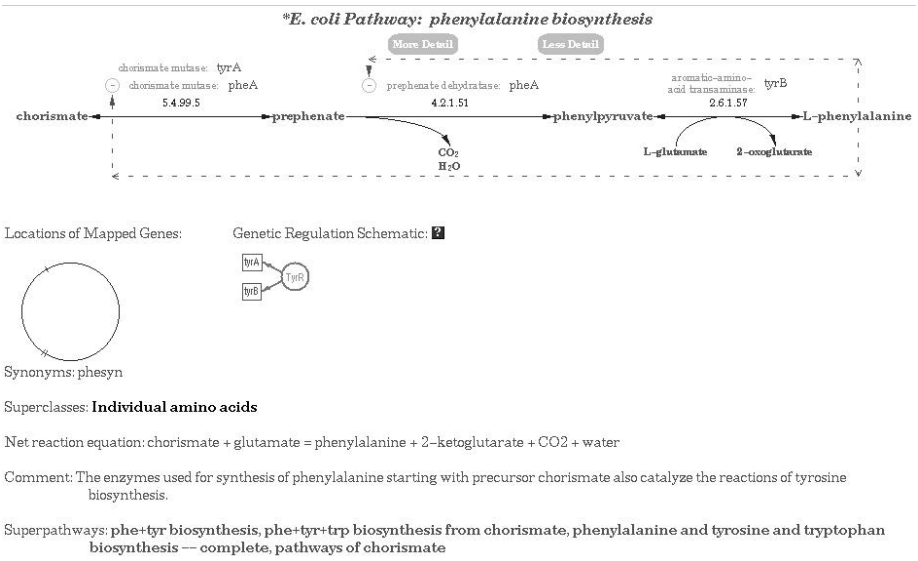


Figure 2-15 The EcoCyc pathway for phenylalanine biosynthesis

Entering the pathway for phenylalanine biosynthesis into a PGDB is accomplished using two steps. The first step is to build the pathway out of its component reactions. Once the pathway has been constructed, the enzymes that catalyze each reaction can be linked to each reaction using the **Edit -> Create/Add Enzyme** command.

Create the new pathway frame by selecting **Pathway -> New**. The Pathway Info Editor appears so that you can classify the pathway, and enter a common-name, a comment, and perhaps literature citations for the pathway. Don't forget to add an evidence code with an appropriate citation. When you exit from the Pathway Info Editor, the pathway display window for this frame is displayed, although no reactions have been added to the pathway yet, and the Pathway Editor appears.

- From the Pathway Editor, select Pathway -> Enter a linear pathway segment to invoke the Segment Editor. Since each reaction in this pathway has an EC number, you can simply enter the three EC numbers for these reactions in the "EC number" boxes to the left of the first three reaction arrows in the Segment Editor. Enter them in the same order that the reactions should appear in the pathway. For any reactions that did not have EC numbers, specify the reactants and products of the reaction. Click Check to check the pathway, and then click Done when the checking phase is complete.
- You are now back in the Pathway Editor. The pathway is complete, so select Exit -> Keep changes to exit from this editor. The pathway is redrawn by Pathway Tools.
- Now right-click on a reaction arrow in the display and select Edit ->Create/Add Enzyme. If you know the name or frame ID for the enzyme, and you know it already exists in the PGDB, you can enter it directly. If the enzyme is a protein complex that does not yet exist in the PGDB, you must first create it. We use a complex in the example that follows.
- Let us create an enzyme that is a heterodimer. In the Choose Protein dialog box, select Create New Protein. In the window that pops up, the type protein complex should already be selected. Specify that the number of distinct gene products is 2, and type in two gene names, for example tyrA and pheA (assuming those genes exist in the PGDB — note that

this selection is for demonstration purposes only, as it is unlikely that those two gene products form a complex in any real organism). The Choose Protein dialog box should reappear, with the text field filled in with a generated frame ID. Click OK.

- The Protein Editor dialog window appears appear, and you can enter information about the protein complex and its catalysis of the reaction. Note that the name for the enzyme is derived from its enzyme activity name(s) (because, for example, a bi-functional enzyme has two different names for its two activities), so filling in the enzyme activity name field also updates the enzyme name. If you know the number of copies of each subunit within the complex, you can enter a coefficient in the section for each subunit. Click OK when you have finished. The updated reaction page is displayed in the Navigator window, but you can use the Back command to return to the pathway display.

2.4 ADVANCED EDITING TOPICS

This section provides more comprehensive information about editing PGDBs, such as

- How the Ocelot DBMS handles simultaneous updates to a PGDB by multiple users
- Restrictions on editing PGDBs designed to protect your work
- The full list of commands accessible through right-clicking on an object handle
- Editing commands in the Tools menu
- Encoding literature citations in a PGDB
- Rules for encoding special formatting of text, such as Greek letters
- Conventions used to name PGDB frames
- Creating URL links from PGDBs to other Web databases
- When Pathway Tools recommends a change

2.4.1 Ocelot Concurrency Control

Multiple users can update a PGDB simultaneously. The Ocelot DBMS uses an *optimistic concurrency control mechanism* to coordinate these updates, ensuring that they occur in an orderly fashion. The mechanism is optimistic in the sense that it assumes that conflicts among updates will be infrequent. Therefore, an appropriate way to control these conflicts is to allow users to make updates at will, and to check for conflicts between updates at the time the updates are *saved*.

The Ocelot update model is based on the notion that at any given point in time, there is a *public version* of a PGDB, and that in addition, developers may have their own private versions of the PGDB in *private workspaces*. Whenever users open the PGDB stored in the Oracle or MySQL server, they are creating a new private workspace. They can make as many updates as they like to that private workspace. Those updates exist on their workstations only until they execute a **Save DB** operation (that operation is in both the Pathway Tools main menu and in the GKB Editor **DB** menu as **Save DB**). The **Save DB** operation proceeds in two phases. In the first phase, Ocelot checks whether any changes made by the user conflict with changes that may have been saved recently by other users. If no conflicts are found, Ocelot proceeds to the second phase, in which the changes are saved to the Oracle or MySQL server, and any recent changes made by other users are loaded into the user's workspace.

PGDB updates are not saved until the **Save DB** operation has completed successfully. The **Save DB** operation saves changes for the current organism only. If you have updated more than one organism, you must select each organism in turn and perform a **Save DB**. An asterisk ("*") next to the name of an organism in the organism-selector menu and in the Organism Summary screen indicates that unsaved changes exist for that organism.

Figure 2-16 illustrates a scenario in which several different users are updating the DB at overlapping times.

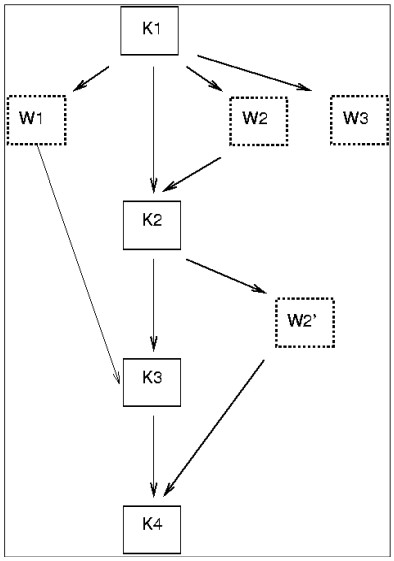


Figure 2-16 Sample set of relationships between public DBs and private workspaces

In the scenario represented in Figure 2-16, three different users open an Oracle or MySQL PGDB at roughly the same time, and thus are all working from the same public version of the DB, which we call *K1*. Each user is working at a different workstation, and has his or her own private workspace, which we call *W1*, *W2*, and *W3*, respectively. Fred works in *W2* for 15 minutes or so, makes several modifications to enzymes in the glycolysis pathway, and then does a **Save DB**. Since no other user has saved any changes during Fred's session, no conflicts can occur. Therefore, Fred's changes are saved into the DB to produce a new version of the DB that we call *K2*.

Mary works in *W1* on the arginine biosynthesis pathway for about an hour, and then does a **Save DB**. Since she performs the save after Fred saved *W2*, Ocelot compares her changes with the changes made in *W2*. However, since Mary and Fred are working on completely different pathways, no conflicts are detected, and the changes made in *W1* are also saved successfully, resulting in a new version of the public DB called *K3*. In addition, the changes to the glycolysis enzymes made in *W2* are loaded into *W1*, and are now available to Mary.

Fred continues his work, and modifies the comments in several reactions of the glycolysis pathway. He now saves his workspace, which we call *W2*'. Since he saves after Mary saved *W1*, his changes are compared against Mary's changes for conflicts. Again, no conflict is found, so Fred's changes are saved successfully as a new version of the DB that we call *K4*. The changes

made by Mary are loaded into Fred's workspace as well. Fred might continue his work by making some changes to the glycolysis pathway itself.

However, he might decide that his changes are not suitable, and that he wants to undo them. To do so, he can execute the **File -> Revert Current DB** operation, which will undo all changes he has made since his last **Save DB** operation. In general, **Revert Current DB** restores the PGDB to its state as of the last **Save DB** operation, or the last **Revert Current DB,** or since a user started a session, whichever occurred most recently. **Revert Current DB** then loads in any changes that were saved by other users (in Fred's case, there were none).

Imagine that Fred continues his work for the next day in the same Pathway Tools session (i.e., without exiting out of the Pathway Tools and back to Unix). Once a night, at 2am, Ocelot performs a **Refresh All Open DBs** operation, *if the user does not have any unsaved updates in their workspace.* The **Refresh All Open DBs** operation loads in all recent changes that have been saved by other users to decrease the likelihood of conflicts occurring. Users can invoke the **Refresh All Open DBs** operation at any time through the **File** menu.

If Fred wants to save his changes, but the **Save DB** command is either undesirable (because he does not yet want his changes to be visible to other users) or unavailable (because the connection to Oracle or MySQL is temporarily unavailable), he can save them to a local file using the command **File -> Checkpoint Current DB Updates to File**. He can later restore these changes using the command **File -> Restore Updates from Checkpoint File**. This latter command loads in and replays all changes stored in the checkpoint file, on top of any updates that have been made since the last checkpoint, save, or revert. To save those changes permanently to the PGDB, you must still then perform a **Save DB** operation.

The full list of database commands available in the **File** menu are as follows:

- **Summarize Databases**: Provides a summary table for all the PGDBs in the system.
- **List Unsaved Changes in Current DB**: Describes the modifications you have made to the current PGDB since your last Save DB operation.
- **Revert Current DB**: Takes the DB back to its state since you last saved it, or since the start of your session, whichever was most recent. That is, all changes you have made since that time are discarded.
- **Refresh All Open DBs**: Brings your version of the DBs up-to-date with respect to any transactions committed by other users. If there have been any such transactions, a window appears containing information about the transactions and the frames that have been modified. You should not invoke this command if you have modified the DB but not yet saved your changes, or else some of your changes may be overwritten. The DB is automatically refreshed when you save it, and nightly (at approximately 2:00 a.m.) if you leave the Pathway Tools running overnight and you do not have any unsaved changes. You need only invoke this command yourself to refresh at other times.
- **Checkpoint Current DB Updates to File**: Saves to a file all changes to the current DB made since the last Save DB or Revert DB operation. Those changes are not saved permanently in the PGDB. This operation is provided as a last-resort means of capturing changes, such as when there is a network outage or the database server is unavailable. The Checkpoint DB command is also a fast way of capturing changes if Save DB

operations work very slowly in your environment. You could run Checkpoint DB operations frequently so you will not lose work if your computer crashes, and run slower Save DB operations less frequently.

- **Restore Updates from Checkpoint File**: Retrieves changes that were saved using the Checkpoint Current DB Updates to File operation. To save those changes permanently to the PGDB, you must next perform a Save DB operation.
- **Delete a DB**: Deletes all traces of a PGDB from permanent storage, including the disk-based directory tree for the DB, and (for PGDBs stored in Oracle or MySQL) deletes the Oracle or MySQL data for the PGDB. Do not use this command unless you really want to remove the PGDB from Pathway Tools, such as if you intend to use PathoLogic to generate a newer version of the DB from scratch.
- **Save Current DB**: Saves all changes.
- **Attempt to Reconnect to Database Server**: If you are editing a PGDB that resides in an Oracle or MySQL DB, this command opens a connection to the Oracle or MySQL server computer in which your DB resides. An existing connection to Oracle or MySQL can become lost for various reasons (such as loss of Internet connectivity), which will prevent you from being able to save your changes to the DB. If you successfully reopen the connection, you will be able to save. If the connection does not reopen successfully, try again later, or create a checkpoint. (Note: there is no harm in reopening a live connection.)
- **Print**: Sends the current display to the printer.
- **Import**: See the section “The Import/Export Facility" in volume I of the user guide for discussion of the import/export facility.
- **Export:** See the section “The Import/Export Facility" in volume I of the user guide for discussion of the import/export facility.

2.4.2 Editing Restrictions

SRI’s support policy for Pathway Tools requires that you follow these restrictions on updating PGDBs:

- Do not alter PGDB schemas, such as by adding or removing classes or slots.
- Do not modify any SRI-supplied PGDBs, such as EcoCyc or MetaCyc because any modifications you make will be lost when you upgrade Pathway Tools.

2.4.3 Right-Button Menu

The following additional commands are available via the mouse right button by clicking on an object handle. A number of these commands refer to a set of editing tools called GKB Editor, which is not described in detail in the Pathway Tools User’s Guide. GKB Editor is an older editor system than the Pathway/Genome Editors. See URL **`http://www.ai.sri.com/~gkb/user-man.html`** for more information on GKB Editor.

The individual commands are arranged into three submenus as follows:

- **Edit**

- **Frame Editor:** Invoke the Frame Editor on this frame. With this command and the other GKB Editor editing commands below, you can continue using the Pathway Tools while editing the frame. Exiting the Frame Editor causes the Pathway Tools to redisplay the edited frame to illustrate the effects of any changes.
- **Synonyms Editor:** Invoke the Synonyms Editor which provides a quick access to the list of synonyms for an object. You can also choose the Common Name of the object from this editor.
- **Relationships Editor:** Invoke the Relationships Editor on this frame, to view links between this frame and those to which it is related.
- **Ontology Editor:** Invoke the hierarchical Ontology Editor, beginning by browsing from the current frame.
- **Object-Type Editor:** Depending on what type of PGDB object you are viewing within the Navigator, the name of one or more appropriate editors appears here (such as Gene Editor).
- **Copy frame name to Clipboard:** Saves the name of the selected frame into the clipboard so that you can later paste it somewhere else during editing with the GKB Editor.
- **Create frame:** Upon right-clicking on the Create Frame command you need to select the type of object to be created. This will open the appropriate editor (e.g. protein editor).
- **Delete frame:** Deletes the selected frame from the DB.

- **Notes**
  - **Add to history:** Add a note to the history of this frame. This feature allows users to attach multiple comments to an object that are stamped with the current date and user.
- **Show**
  - **Show frame:** Display a printed representation of the data in this frame in the Unix window in which you originally started Pathway Tools.
  - **Show frame name:** Write the frame ID for the current frame both in the Lisp window and in the listener pane.
  - **Show compound/reaction/pathway in overview:** Display the Overview graph, highlighting this compound, reaction, or pathway.
  - **Show frame in other species:** Display this same biological object for another species whose database is available, for example, the same chemical compound or a gene with the same name (if one exists). You select the species of interest in a cascading menu.
  - **Print frame:** Print the data in this frame to a printer.
  - **Print pathway or reaction frames:** Print the specified data to a printer.
  - **Print pathway and its enzymes:** Print the specified data to a printer.

- **Show changes:** Pop up a window listing the modification history of this frame. The modification history is stored as a sequential log of Generic Frame Protocol (GFP) operations that have been applied to the frame.
- **Refresh object display:** Refreshes the display on the screen.

2.4.4 The Tools Menu

Please see section 3.9 of Volume I.

2.4.5 Object Names Visible to PGDB Users

Every biological entity in a PGDB is encoded as a frame (an object) in the DB. When you see an object display in the Pathway/Genome Navigator, you typically see the biological names that are stored in the slots called Common-Name and Synonyms that are defined within most frames. That is, whenever a Pathway Tools display window refers to a given object, the name that the Pathway Tools shows is computed by first attempting to retrieve the value of the Common-Name slot; if that slot is empty, then the frame name is displayed.

Conventions used by SRI in the EcoCyc and MetaCyc DBs are

- Object names are generally all lowercase.
- Bacterial gene names have the standard bacterial convention of “abcD”.
- Bacterial protein names are generally lowercase (such as “tryptophan synthetase”), except for protein names derived directly from the gene name, such as “TrpA”.

Developers of other PGDBs can follow other conventions if they so choose — the most important thing is for all developers of a PGDB to follow a consistent set of conventions.

There are two exceptions to the previous rule that displayed object names are derived from the Common-Name slot: reactions and enzymes. Since reactions typically do not have any meaningful name, they are referred to by either the reaction equation, or the EC number. Enzymes are complicated because for a multifunctional enzyme we want to use different names to refer to the enzyme in different contexts. An enzyme that catalyzes two reactions has a different name associated with each reaction. When referring to the enzyme in the context of one of those reactions, or in the context of a pathway containing one of those reactions, we use the name associated with that reaction; that name is taken from the Common-Name slot of the enzymatic reaction frame that links the enzyme to that reaction. However, when referring to the enzyme outside the context of any particular reaction, we string together all of the names associated with all the reactions. One final complication is that any names for the enzyme that are independent of the reaction(s) that it catalyzes are stored in the protein frame (e.g., names that refer to physical properties of the protein rather than the function of the protein).

2.4.6 Frame Naming Conventions

Each frame has a frame name (also called an ID, or a key) that uniquely identifies that frame within the DB. Frame names must begin with a letter, and can contain numbers and hyphens (used to separate multiple words). The length of frame names is restricted to 40 characters.

Instance names are written in uppercase, singular (as opposed to plural). Example instance frame names: are TRPA-MONOMER, G00010, RXN0-152. Most class names are written as capitalized plurals. Example class names are Compounds, Genes, Polypeptides, Amino-Acids.

Generally, frame names are assigned in a systematic way so that names convey some information about the biological objects that they describe, and so that PGDB authors can remember (or at least recognize) names for some DB objects. The remainder of this section describes the systems used to name different objects within a PGDB. We advise following these guidelines.

Frame names sometimes become out of date because of changes in biological knowledge. For example, if the name of a gene is changed, the name of the frame describing the product of that gene then becomes obsolete (e.g., the name XYZA-MONOMER becomes obsolete if gene xyzA is renamed to qrtA). However, we strongly recommend that frame names not be changed if they have been in use for more than an hour or so. The reason is that frame names are used as "pointers" to allow one frame to refer to another frame. When a frame is renamed, Ocelot attempts to change all pointers that used the old name so that they use the new name; however, in some cases it is not easy to find all pointers that use a given frame name. Therefore, frame names should only be changed shortly after they are created, and some frame names may not serve as accurate mnemonics for the biological objects they identify. However, you can easily alter the biological name for an object by changing the value of the Common-Name slot, thus changing the name that users will see.

By default, all new frame names you enter will be coerced to uppercase. This makes it easier to enter names of instance frames. If you are entering a class name and do not want it coerced to uppercase, surround the frame name by vertical bars when you first type it. Example: |Compounds|.

You can set a preference in the GKB Editor (under the **Preferences -> Miscellaneous** menu) so that this coercion to uppercase does not occur. If this preference is set, you do not need to type the vertical bars, but you must make sure that you type instance names in all uppercase.

Different conventions are used to name instances in different classes in a PGDB. The conventions include the systematic use of a suffix or prefix, of mnemonic names, and of names generated from some numeric sequence that therefore have no obvious meaning. (Names from a numeric sequence are less meaningful, but they have the advantage that a name with no meaning cannot become obsolete — unless the corresponding biological object disappears completely or is merged with another object.) The rules for naming instances in different classes are as follows.

2.4.6.1 Naming Compounds

Some compounds are written using mnemonics, while others are written using a "C" prefix and a number. Examples: TRP, ARG, C0002.

2.4.6.2 Naming Gene Frames

Different conventions for naming gene frames are used for different organisms. Typically, we try to use the same ID for the frame as the unique ID that has been assigned to a gene by a genome project.

2.4.6.3 Naming Polypeptide Frames

Polypeptide frame names are written using a mnemonic name suffixed by -MONOMER. The mnemonic name is sometimes derived from the name of the gene. Examples: TRANSKETOI-MONOMER, XYLISOM-MONOMER, CYDA-MONOMER.

2.4.6.4 Naming Protein Complex Frames

Protein complex names are written using a mnemonic name suffixed by -CPLX. Examples: THIOESTERII-CPLX, SUPEROX-DISMUTMN-CPLX.

2.4.6.5 Naming Enzymatic Reaction Frames

Enzymatic-reaction names are written using a mnemonic name suffixed by -ENZRXN. Examples: SAICARSYN-ENZRXN, THIOESTERI-ENZRXN.

2.4.6.6 Naming Reaction Frames

Reaction frame names are written using a mnemonic name suffixed by -RXN, or as an EC number suffixed by -RXN. Examples: BUTANAL-DEHYDROGENASE-RXN, 1.2.1.9-RXN.

Reaction classes defined in the enzyme nomenclature system are written as “EC-1.2”.

2.4.6.7 Naming Pathway Frames

Pathway names are written using a mnemonic name suffixed by -PWY. Examples: LEUSYN-PWY, HISTSYN-PWY.

2.4.6.8 Naming Slots

All slot names are stored internally in uppercase. Typically, slots that take a single value have singular names, and slots that take multiple values have plural names. Examples: CATALYZES, COMPONENTS, LEFT-END-POSITION.

If you are using the Frame Editor to enter a value for a slot, and the value should be a frame name, but no frame exists with the name that you entered, one of two things will happen, depending on the slot. If the value must be a frame (based on the definition of the slot), a dialog asks you if you want to create the frame. If you create the frame, the name of the frame is the name that you originally entered, and the class under which the frame is created is determined by the slot definition. Alternatively, for some slots, such as Left, for which the value can be either a frame or a string, if the value you type does not match an existing frame it is left as a string

(there may be other cases where a name is left as a symbol). Even if you later create the frame to which you were referring, the value of the slot remains a string until the Pathway Tools consistency checking code is run, at which point the correct frame name is substituted. You can tell whether a value of a slot is a frame, a string, or a symbol by looking at how it is displayed in the Frame Editor. Strings have double-quotes surrounding them, symbols are displayed in a fixed-width font and are usually all uppercase, and frames are displayed in a variable-width font without quotes.

2.4.7 Special Formatting of Text

Pathway Tools provides special formatting conventions allowing you to include citations and special characters (italic, bold,) in text that is stored in the Comment slot. Most of these formatting conventions are loosely modeled after HTML.

- Formatting of special characters is described at **`http://ecocyc.org/ecocyc/ec-greek.html`**.

2.4.8 Citations

You can include citations within the text of a comment, as illustrated in the following example. The first line shows the syntax of the text that is to be entered into a DB within the comment field of an editor such as the gene editor; the second line shows what that text will look like when displayed by Pathway Tools (after new PubMed references have been imported from the PubMed database using the **Import -> References from PubMed** command).

Pyruvate modulates the activity of the enzyme over a 10-fold range
|CITS:[84183611],[SMITH95]|.

Pyruvate modulates the activity of the enzyme over a 10-fold
range [Band84,Smith95].
When a user clicks on a citation displayed at the bottom of the page, the corresponding reference will be displayed by Pathway Tools or by your Web browser.

Each citation is identified by a unique identifier enclosed by square brackets. For references indexed in PubMed (the simpler case), use the PubMed ID number (e.g. 84183611) as the identifier to allow direct linking to PubMed. For references not in PubMed, you should assign a unique ID of as many as 20 uppercase alphanumeric characters (ideally composed of the first author's last name followed by the year of publication, such as "SMITH95". If this ID is already in the database, use the format SMITH95a), and create a frame for each citation, as described in the section 2.3.12 – Publication Editor.

2.4.9 Creating Publication Frames

See "Publication Editor" in section 2.3.12.

2.4.10 Creating Links between a PGDB and External Databases

Pathway Tools allows you to create URL-based links from objects in a PGDB to objects in other databases outside the Pathway Tools environment that are accessible via Web queries. In addition, if you want to create links from internal databases at your organization to a PGDB, Pathway Tools can generate files called *linking tables* that specify the unique identifiers for PGDB objects, to allow you to link to them. For example, you could create links from an EcoCyc gene to homologs within a database at your site, and you could create links from those homologs to EcoCyc genes.

If your PGDB exists in conjunction with a previously existing database of genes for an organism, you may wish to have the Pathway Tools web server send users directly to the gene pages generated by the other database, rather than to the Pathway Tools-generated gene pages. To accomplish this, you must first use one of the procedures described below to create links from the PGDB to the other database. See the description of the **-gene-link-db** command line argument to Pathway Tools in Volume I of the User Guide for more details.

2.4.10.1 Object Correspondence

To create links in either direction, you must first identify correspondences between PGDB objects and objects that you want to link to. To facilitate this process, Pathway Tools can generate a set of files listing the unique identifiers of PGDB objects, plus other information about those objects such as their names or EC numbers. For example, the EcoCyc genes file lists the EcoCyc ID, the common name, and the synonyms, for each EcoCyc gene. It also lists correspondences to other databases such as SwissProt. You could do a name-based search to identify correspondences between genes of interest to you, and EcoCyc genes. You could also use the SwissProt IDs to pull out sequences for each EcoCyc gene, and to compare those sequences to gene sequences of interest to you.

The linking files, and the columns provided in each file are as follows. The ORGID is a mnemonic for the name of the organism.

ORGID-gene-links.dat ID EG# b# SP-ID CGSC-ID Name Synonyms...

ORGID-protein-links.dat ID Gene-ID Name Synonyms...

ORGID-reaction-links.dat ID EC#

ORGID-pathway-links.dat ID Name Synonyms...

ORGID-compound-links.dat ID SMILES Name Synonyms...

Where:

ID = PGDB ID

EG# = ID from ecogene database

b# = ID from Blattner GenBank entry

```
SP-ID     =  SwissProt ID
CGSC-ID   =  Coli Genetic Stock Center ID
EC# =    Enzyme Commission number
```

2.4.10.2 Creating Links to a PGDB

We assume that you have determined correspondences between objects in your local databases and PGDB objects, and that you can add links to Web-accessible objects to your local databases. The document **`http://biocyc.org/linking.shtml`** provides instructions on what URLs to use to link to PGDB objects.

Links are accessible in both the X-windows PGDB and the Web server version.

2.4.10.3 Creating Links from a PGDB

To create links from a PGDB to another database you must define the database itself to the PGDB, and you must define the links to the PGDB. These definitions are created manually using the GKB Editor, and they can be loaded in bulk by using two different files: a database-definition file and a link-definition file.

For each database that you link to, you will define a unique identifier for the database, a name for the database, and a URL that can be used to query an object from that database given a unique identifier for that object.

2.4.10.4 Manual Creation of Links

Each of the editors for the different object types contain fields for entering links to other databases. Simply select the desired external database, and enter the ID for the object in that database. This is possible however only if information about the external database is already available in the PGDB. Many databases, such as SWISSPROT, are already defined in any PGDB that you create. If you want to define a new database, however, such as one that is local to your organization, you can do so as follows:

- Select Tools -> Ontology Browser
- Select View -> Browse from new root(s) and enter "Databases" in response to the question in the listener window
- Select the frame called Databases.
- Select Frame -> Create -> Instance.
- Enter the frame name, which is the identifier you will use to refer to the database in link specifications (e.g., "GENEDB").
- Select Frame -> Edit to invoke the Frame Editor.
- You can enter a value in the Common-Name slot if you want the name of the database to be displayed differently than its frame name when links are printed.
- Enter a new value for the slot Static-Search-URL. Whenever a user clicks on a link, the

Pathway Tools will call Netscape on a URL that is computed by substituting the ID of a given object for the string "~A" in the slot value that you enter here. An example URL is: **http://gene.pharma.com/dbquery?~A**

- If you want the database to be one of the possible selections in a particular editor, you must supply one or more values for the slot Search-Object-Class. Each value should name a class in the Pathway Tools Schema. For example, if you want your database to be one of the options in the Gene Editor, you would add the value Genes to this slot.
- Exit from the Frame Editor and save your changes.

2.4.10.5 Bulk Loading of Links

A database-definition file defines each database that you will link PGDB objects to. An example database-definition file is as follows. Each line in the file describes one database; each column in the file is separated by a TAB.

```
# Sample database file.

DB1     ChemicalDB  http://chem.pharma.com/dbquery?~A

DB2     SequenceDBhttp://seq.pharma.com/dbquery?~A
```

Imagine that DB2 is a DNA-sequence database. We want to link the EcoCyc gene whose unique identifier is EG10004 to the sequence for a homologous gene in DB2. Let us assume that the unique identifier for that homologous gene in DB2 is S9993. The preceding entry for DB2 specifies that to query object S9993, we would query **http://seq.pharma.com/dbquery?S9993** (the object identifier is substituted for the string "~A").

The link itself can be created using a link-definition file, whose format is as follows (each column in the file is separated by a TAB).

```
# Sample linktable file.

$ORGANISM ECOLI

TRP     DB1     C000111

EG10004 DB2     S9993    Homolog
```

The first non-comment line in the file must be a **$ORGANISM** specification, which indicates the PGDB in which these links are defined. You can determine organism identifier by right-clicking on the organism's name in the Organism Summary page; the organism ID appears in the resulting menu. Subsequent lines in the file define links from one PGDB object to one object in another, Web-accessible database (one link per line).

The line beginning with **TRP** creates a unification link (a link between two pieces of the same biological object that reside in different databases). The link is from an EcoCyc object whose ID is **TRP** to an object in DB1 whose ID is **C000111**. The line beginning with **EG10004** creates a relationship link (a link between two different biological objects based on some relationship between those objects) from **EG10004** to **S9993** in DB2; the relationship is specified as "Homolog".

Unification links and relationship links are displayed slightly differently by Pathway Tools (the word "Homolog" is printed in the second link).

To load the database descriptions and links defined in these files into Pathway Tools, use the command line arguments **-dbdef** *<database-file>* and **-linkdef** *<linktable-file>.* If these links are being loaded into a PGDB that you can edit, then you need only load them one time like this, and then save the relevant PGDB(s). If they are being loaded into a PGDB which you cannot edit (such as the EcoCyc or MetaCyc PGDBs included in your software distribution), then you will have to load the links in this fashion every time.

2.4.11 Modified Proteins

Modified proteins are polypeptides that have one or more chemical groups attached. They can therefore be considered a kind of hybrid between a protein and a compound. We consider them instances of Polypeptides, but allow them to have values for many of the same slots as Compounds (e.g., chemical structure information). The unmodified protein is grouped together with all the modified versions under a single subclass of Polypeptides. For example, the class **Gln-B** has two instance: the unmodified protein **protein-pii** and the modified **uridylyl-pii**. The class **All-ACPs** contains the unmodified ACP-MONOMER along with instances for all the various ACP derivatives.

The unmodified protein should carry all the usual information and comments as a regular protein. It will be the only polypeptide in the class that has a link to the gene (because it is the direct gene product).

The modified protein(s) need not carry much information about the protein, but can carry additional, separate comments that apply specifically to the modified form. Each modified protein needs to link back to the unmodified form through the Unmodified-Form slot. The unmodified protein then points to all its modified versions through the Modified-Form slot. These two slots are maintained as inverse links, so the information needs to be added in one direction only.

Any protein or modified protein can be a substrate in a reaction, just like a small molecule.

2.4.12 When Pathway Tools Makes Recommendations

When Pathway Tools detects that you have made an allowable but not recommended edit, a window appears stating a recommended change and asking you whether you want Pathway

Tools to make the recommended change for you. For example, if you specify the COMMON-NAME of an acid to be Artelinic acid, then Pathway Tools asks you to approve automatically changing the COMMON-NAME to artelinate.

If you need to disable this behavior during a Pathway Tools session, type the following to the Lisp Listener:

```
(setq *auto-recommend?* nil)
```

To later re-enable this behavior, type the following to the Lisp Listener:

```
(setq *auto-recommend?* t)
```

BIBLIOGRAPHY

1. Bairoch The enzyme data bank in 1995. *Nuc. Acids R*es., 24:221--222, 1996.
2. F.R. Blattner, G. Plunkett III, C.A. Bloch, N.T. Perna, V. Burland, M. Riley, J. Collado-Vides, J.D. Glasner, C.K. Rode, G.F. Mayhew, J. Gregor, N.W. Davis, H.A. Kirkpatrick, M.A. Goeden, D.J. Rose, B. Mau, and Y. Shao. The complete genome sequence of *Escherichia coli* k--12. *Science*, 277:1453-1462, 1997.
3. NCBI DDBJ, EMBL. *The DDBJ/EMBL/GenBank Feature Table Definition*, version 2.0 edition, December 1997. http://www.ncbi.nlm.nih.gov/collab/FT/index.html.
4. P. Karp. Database links are a foundation for interoperability. *Trends in Biotechnology*, 14:273-279, August 1996.
5. P. Karp. The EcoCyc user's guide. Unpublished; see WWW URL http://ecocyc.PangeaSystems.com/ecocyc/doc/ecocyc-uguide/paper.html, 1996.
6. P. Karp and T. Gruber. The generic frame protocol. Available via World Wide Web URL http://www.ai.sri.com/~gfp/doc/paper.html, 1995.
7. P. Karp and M. Mavrovouniotis. Representing, analyzing, and synthesizing biochemical pathways. *IEEE Expert*, 9(2):11--21, 1994.
8. P. Karp and S. Paley. Representations of metabolic knowledge: Pathways. In R. Altman, D. Brutlag, P. Karp, R. Lathrop, and D. Searls, editors, *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 203-211, Menlo Park, CA, 1994. AAAI Press.
9. P. Karp and S. Paley. Automated drawing of metabolic pathways. In H. Lim, C. Cantor, and R. Robbins, editors, *Proceedings of the Third International Conference on Bioinformatics and Genome Research*, pages 225-238. World Scientific Publishing Co., 1995. See also WWW URL ftp://ftp.ai.sri.com/pub/papers/karp-bigr94.ps.Z.
10. P. Karp and S. Paley. Integrated access to metabolic and genomic data. *Journal of Computational Biology*, 3(1):191-212, 1996.
11. P. Karp and M. Riley. Representations of metabolic knowledge. In L. Hunter, D. Searls, and J. Shavlik, editors, *Proceedings of the First International Conference on Intelligent Systems for Molecular Biology*, pages 207-215, Menlo Park, CA, 1993. AAAI Press.
12. P. Karp, M. Riley, S. Paley, A. Pellegrini-Toole, and M. Krummenacker. EcoCyc: Electronic encyclopedia of E. coli genes and metabolism. *Nuc. Acids Res.*, 26(1):50--53, 1998.
13. P.D. Karp. A knowledge base of the chemical compounds of intermediary metabolism. *Computer Applications in the Biosciences*, 8(4):347--357, 1992.
14. M. Riley. Functions of the gene products of Escherichia coli. *Microbiological Reviews*, 57:862--952, 1993.
15. J.-F. et. al. Tomb. The complete genome sequence of the gastric pathogen helicobacter pylori. *Nature*, 388:539--547, 1997.
16. R.A. VanBogelen, P. Sankar, R.L. Clark, J.A. Bogan, and F.C. Neidhardt. The gene-protein database of *Escherichia coli:* Edition 5. *Electrophoresis*, 13:1014–1054, 1992.
17. Edwin C. Webb. *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, 1992.

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

INDEX

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U