Automated Drawing of Metabolic Pathways*

Peter D. Karp
Suzanne Paley
Artificial Intelligence Center
SRI International
333 Ravenswood Ave., EJ229
Menlo Park, CA 94025
pkarp@ai.sri.com

October 10, 2000

1 Abstract

The EcoCyc system consists of a knowledge base that describes the genes and intermediary metabolism of *E. coli*, and a graphical user interface (GUI) for accessing that knowledge. This paper presents algorithms for drawing metabolic pathways by dynamically querying the underlying knowledge base. These algorithms provide a foundation for building graphical user interfaces to metabolic databases. Pathway drawing is a graph-layout problem. Our algorithms draw pathways of several different topologies, including linear, cyclic, and branching pathways, as well as larger groupings of such pathways. The algorithms provide several visual presentations of metabolic pathways, for example, compounds can be drawn as names and/or chemical structures, and enzyme names and side compounds can be drawn or omitted. The GUI also provides several facilities for navigating in the space of biochemical pathways, such as traversing connections between pathways, and exploding or collapsing a pathway to include or exclude neighboring pathways.

2 Introduction

The EcoCyc system consists of a knowledge base that describes the genes and intermediary metabolism of *E. coli*, and a graphical user interface for accessing that knowledge. Taken together, the knowledge base (KB) and the graphical user interface (GUI) constitute an electronic encyclopedia that allows scientists to visualize an integrated collection of genomic and biochemical information.

*Appears in Proceedings of the Third International Conference on Bioinformatics and Genome Research, H. Lim, C. Cantor and R.Bobbins eds., 1994.

The EcoCyc KB is stored in a frame knowledge representation system. It contains information about genes, the enzymes those genes encode, the reactions catalyzed by those enzymes, and the chemical compounds that participate in these reactions. In addition, the KB describes a number of metabolic pathways, each consisting of a collection of bioreactions. The EcoCyc GUI dynamically generates displays of each of these types of objects— such as a genetic map, a reaction, an enzyme, or a metabolic pathway—by querying the KB and applying drawing algorithms to the query results.

We present algorithms for drawing metabolic pathways that provide a foundation for building graphical user interfaces to metabolic databases. These algorithms have been tested on 25 pathways from the EcoCyc KB, such as the TCA cycle, glycolysis, and tryptophan biosynthesis. The algorithms provide several visual presentations of metabolic pathways, for example, compounds can be drawn as names and/or chemical structures, enzyme names can be drawn or omitted, and connections to other pathways can be shown. In addition, they can display either single pathways or groups of interconnected pathways.

3 The Pathway-Drawing Problem

Figure 1 shows a drawing of the biosynthetic pathway for threonine biosynthesis that was generated by EcoCyc, which closely mimics drawings found in biochemistry textbooks. The drawing shows a sequence of bioreactions that transform a number of reactant compounds into a number of product compounds.[1] Every transformation in the sequence is catalyzed by one or more enzymes, whose name(s) is drawn next to the reaction arrow (an exception is that a few bioreactions are known to occur spontaneously).

The nature of a pathway is that adjacent bioreactions operate on shared compounds: the *main substrates* that lie along the backbone of the pathway (such as homoserine). The unshared compounds are the *side substrates*; they are drawn off to the side of each bioreaction, with curved arrows showing whether particular side substrates are reactants or products of a reaction. We can represent a metabolic pathway as a graph whose nodes consist of enzymes, main substrates, and side substrates (identified as such); the edges of the graph are the straight lines that connect main substrates, and the curved lines that connect groups of side substrates. The pathway-drawing problem is: Given a pathway graph whose nodes and edges are typed and labeled with enzyme, main, and side information, compute appropriate display positions for each node and edge. This formulation considers pathway display to be a graph-layout problem. As discussed in [1], pathways within the EcoCyc KB are stored using the *predecessor-list* representation, not the graph representation that is the input to the pathway-drawing algorithms. Karp and Paley present an inference procedure for converting from the predecessor list to the graph representation that is the starting point for the algorithms described herein.

[1] We define the *reactants* as simply the compounds on the left side of the reaction; *products* are the compounds on the right side. *Substrates* are the union of products and reactants.

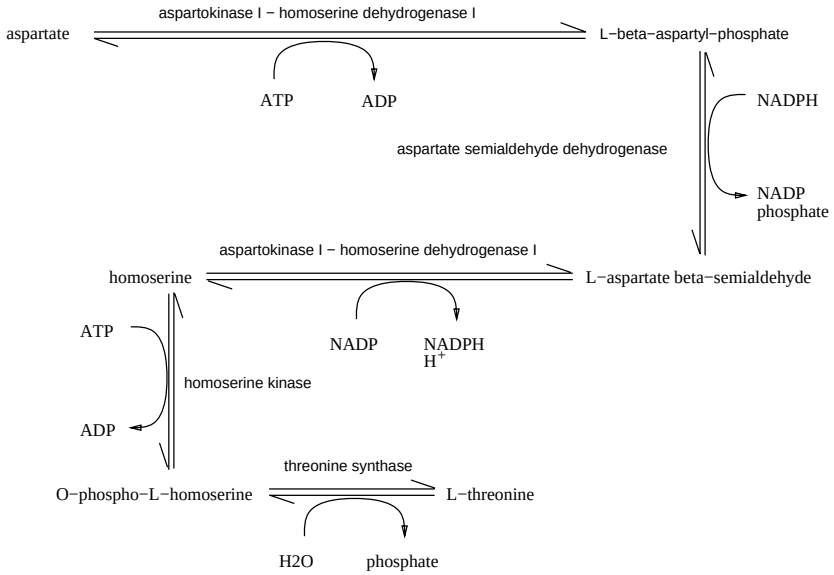

Figure 1: The *E. coli* pathway for threonine biosynthesis.

4 Pathway Drawing Algorithms

Our definition of the pathway-drawing problem invoked the notion of "appropriate" positions for graph nodes and edges. Although there is no simple definition of "appropriateness", a first approximation is to emulate the visual presentations found in biochemistry textbooks [8, 3]. Our approach is to first provide users with the type of drawings they are familiar with from textbooks, and then to build on that style of drawing to produce more powerful computer-generated displays that cannot practically be used in books.

Our conceptualization of pathway display, and its implementation, owe much to our present work at SRI on graph layout, display, and editing [2]. An important conclusion of that work is that we should not expect to find a single, all-purpose graph-layout algorithm. Instead, graphs of different topologies are best visualized using different layout algorithms. For that matter, different application domains sometimes lay out graphs of the same topology differently. In addition, relationships in a large, complex graph may be easiest to visualize if the graph layout is computed hierarchically: we decompose the graph into smaller subgraphs, then apply appropriate layout algorithms to each subgraph, and then arrange the individual drawings to produce a hierarchical layout. The Grasper-CL system developed at SRI provides a toolbox of graph-layout algorithms in the hope that some of these algorithms will be applicable to a new graph-layout application, but its authors recognize that new layout algorithms may be required for new applications.

This philosophy is appropriate for pathway display. Examination of biochemistry textbooks reveals that pathways of different topologies are indeed presented in different layouts; some of those layouts are familiar from other domains (e.g., circular layouts and tree layouts), whereas some layouts are more specific to biochemistry (such as by the "snake layout" for a linear pathway, as shown in Figure 1). Textbooks are usually quite limited as to the maximum-sized pathway they can show, because of constraints of the printed page. But as we move beyond those restrictions, we will wish to display both large interconnected collections of historically

defined pathways (e.g., glycolysis leading to the TCA cycle), and complex novel pathways that have been constructed by computer (such as the algorithms of Mavrovouniotis) [4]). In both cases, hierarchical displays of several individual layouts will be essential.

Here we present algorithms for drawing linear, circular, and tree-structured pathways. We divide the overall pathway-drawing problem into three steps:

1. Determine the topology of the input pathway, as linear, cyclic, or branched.
2. Apply the layout algorithm that is appropriate to the topology of the pathway to the main nodes of the pathway, thus assigning positions to the main nodes.
3. Assign positions to the side nodes and the enzyme nodes of the pathway. The same layout algorithm is applied in this step regardless of what layout algorithm was used in Step 2. In fact, this algorithm may be called by algorithms in Step 2 to determine the amount of spacing needed between adjacent main nodes to accommodate intervening side nodes.

Step 1 examines the connections among the main nodes of the pathway so that in Step 2 we know which layout algorithm to apply to the pathway. The actual topology tests employed are as follows:

- **Circular:** A standard cycle-detection algorithm is applied to the pathway graph. If a single cycle is found, and all main nodes are members of the cycle, then the pathway topology is circular.
- **Branched:** If the pathway contains no cycles, and any main node has more than one incoming edge or more than one outgoing edge, or the pathway graph is not fully connected, then the pathway topology is branched.
- **Linear:** If the pathway contains no cycles, and it is not branched, then the pathway topology is linear.
- **Complex:** If the above-mentioned cycle-detection algorithm finds more than one cycle, or one or more nodes are not members of the cycle, then the pathway topology is complex.

One limitation of this algorithm for topology detection is that although the branched category is meant to capture tree-structured pathways, a pathway with any complex set of connecting reactions will be assigned to this class as long as it contains no cycles.

4.1 Layout of Linear Pathways

We have developed layout algorithms for drawing linear pathways in three possible ways. The main nodes in the pathway can be drawn along a straight horizontal or vertical line; scroll bars are used when a large pathway does not fit on one screen. The algorithm for these styles of linear layout is trivial. Or, a linear pathway can be drawn using a *snake layout*, as found in many biochemistry textbooks, and as shown in Figure 1. This layout maximizes the usage of screen space, and is similar to word-filling algorithms used in word processors. We use a greedy algorithm that fits as many consecutive main nodes as possible into each horizontal line. When

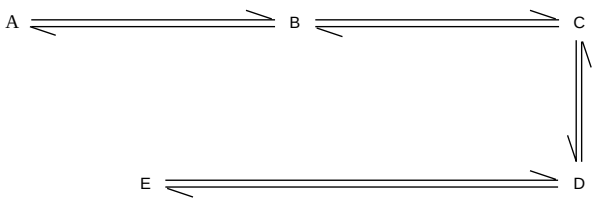

Figure 2: An illustration of the snake-layout algorithm.

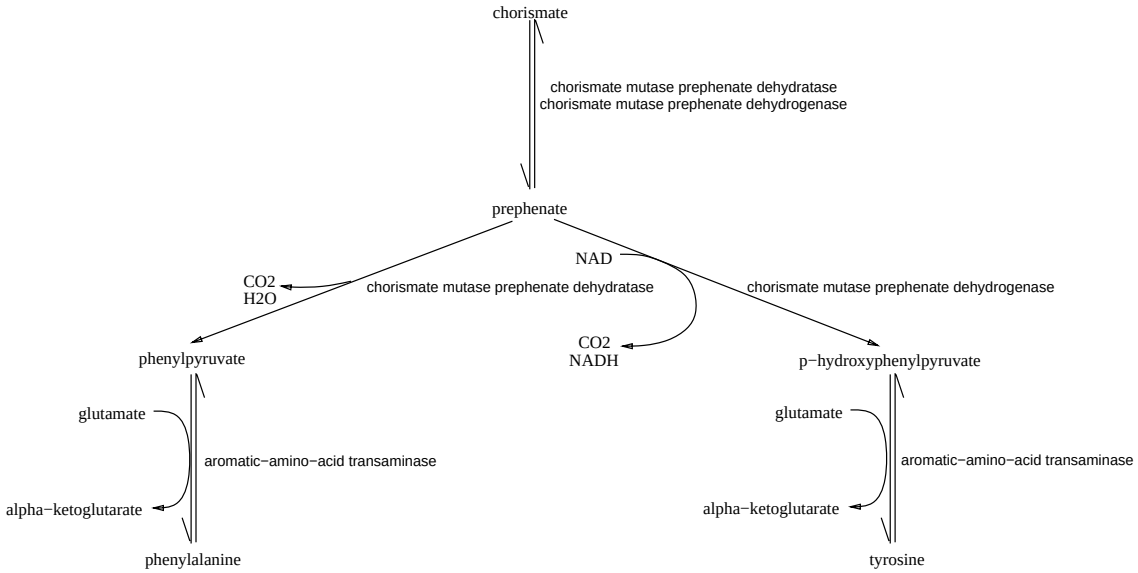

Figure 3: The *E. coli* pathways for biosynthesis of phenylalanine and tyrosine.

a line is full, a node is placed on a new line centered under the last node on the previous line, but with the direction reversed.

Our implementation separates pairs of main nodes on one line by the minimum space needed to hold the intervening side nodes and enzyme name(s). Since this approach may not completely fill a given line, the maximum length of the next line is decreased because the next line starts where the previous line ended. If in the example in Figure 2 there was not enough room to add E to the second line because of a long enzyme name for the $D \longrightarrow E$ transformation, the program would attempt to shift C to the right margin (this check is made only when attempting to add the second main node to a line). If there were still not enough room to fit E on the same line as D, E would be drawn below D. Another approach would be to space out all nodes that fit on a given line to fill the entire line, thus ensuring that the start of the next line is as close to the margin as possible.

4.2 Layout of Branched Pathways

We applied a tree layout algorithm from the Grasper-CL toolbox to the layout of branched pathways, with excellent results for pathways such as glycolysis and the biosynthesis of phenylalanine and tyrosine (see Figure 3).

The algorithm works by positioning the root node(s) of the tree at the top of the drawing.

The children of a given node are recursively positioned under their parents; then the parent is horizontally centered over its children. All children of a given parent are positioned at the same height. See [2] for more details.

The limitation of this algorithm is that although it produces acceptable drawings for pathways that are trees, it does not attempt to minimize edge crossings in graphs in which nodes have more than one parent. Therefore, an extremely tangled pathway may look more tangled than is necessary. But for the relatively simple historical pathways that are our initial focus, this is not a problem.

The initial implementation of this algorithm used a constant vertical offset between all parent nodes and their children, computed as the tallest set of side compound and enzyme names found in the pathway. Pathway drawings were more spread out vertically than they needed to be, wasting screen real-estate. We have since modified the implementation to use exactly the spacing required to accommodate each set of side compound and enzyme names, resulting in significantly better drawings.

4.3 Layout of Cyclic Pathways

We applied the circular layout algorithms from the Grasper-CL toolbox to the layout of cyclic pathways, testing it on the TCA cycle.

The algorithm produces an acceptable display for this pathway, except that the edges connecting the mains of the pathway are straight lines rather than the circular arcs typically seen in biochemistry books. Grasper-CL does not currently have the capability to draw graph edges as circular arcs, but we plan to add that capability soon.

The algorithm positions all of its argument nodes along the circumference of a circle. We tried one approach to computing node positions that often produced unacceptable results: positioning each node at a fixed angular offset along the circumference of the circle, that is, computing the angle θ_i for node i as $2\,\pi\,i/n$ radians. The results are unacceptable because nodes near the top and the bottom of the circle sometimes overlap.

Instead, our algorithm splits the nodes into two groups, called A and B, where each group of nodes is to lie along one side of the circle. The diameter of the circle is

$$D = max(\begin{array}{l} \sum_{i=1}^{n} height(node_i \mid node_i \in A) + vspace, \\ \sum_{i=1}^{n} height(node_i \mid node_i \in B) + vspace) \end{array}$$

Then, the y coordinate of each node in a group is computed to space the group evenly in the vertical dimension: $y_i = i\,D/2\,n$. The x coordinate is computed to place the node on the circumference of the circle: $x_i = \sqrt{r^2 - y_i^2}$ where r is the radius of the circle.

4.4 Layout of Complex Pathways

The layout for complex pathways is derived by partitioning the graph into subgraphs, each of which can be laid out using one of the above algorithms, and using Grasper-CL's hierarchical layout facility to place the subgraphs relative to each other. The nodes that make up the largest cycle form one such subgraph, C. The original graph minus C is then divided into connected

components. Any component connected to the nodes in C by two or more edges is a member of the subgraph I, to be drawn inside the circle; all others will be drawn outside the circle, forming the subgraphs $O_1 \ldots O_n$.

First, I is laid out, using whichever layout algorithm fits its topology (if I is itself a complex graph, this algorithm will be called recursively). Then, C is laid out so as to surround the layout of I. This is done using the above circular layout algorithm, but with the circle diameter expanded to accommodate the interior nodes. All the nodes in $C \cup I$ are then combined to form a single supernode. Each of the O_k is individually laid out using whichever layout algorithm is appropriate, with each forming its own supernode. Finally, the supernodes are laid out relative to each other using a tree layout algorithm. (Because the O_k are by definition disconnected from each other, all edges between supernodes must have one end at C. Thus, there can be no cycles, and the tree layout algorithm seems the most appropriate.)

The main limitation of this algorithm is that the subgraphs are laid out relative to each other as if they had no internal structure. Since any edge between subgraphs is actually an edge between a pair of nodes somewhere in the interior of the two subgraphs, ideally, we would like to place the supernodes so that nodes with edges between them are placed near each other (and perhaps do the individual component layout so as to make this possible). Grasper-CL's hierarchical layout facility does not support this, nor is it entirely clear how to best accomplish this even if we were to bypass the Grasper-CL layout mechanisms.

5 Layout of Side Nodes and Enzyme Nodes

A reaction might have side reactants, side products, both, or neither. We simplify the problem of positioning side nodes by first arranging the set of side products (and the set of side reactants) as two vertical groups. We then treat the two groups as two supernodes for layout purposes. Therefore, in a reaction that has both side products and side reactants, two supernodes must be positioned relative to the reaction arrow, leaving room for the curved side arrow in between.

In our experience the best positioning is produced when we consider two cases: reaction arrows that lie within 15° of horizontal, and arrows with greater slopes. Examples are shown in Figure 4 (a) and (b), respectively. In the horizontal case, we start with the supernodes horizontally centered above the reaction arrow, at the same height (a), and then adjust one supernode up and the other down such that the circled corners are the same distance from the reaction arrow. Of course, no adjustment is necessary if the reaction is completely horizontal. For a nonhorizontal reaction arrow, the supernodes start out aligned horizontally, and are adjusted analogously (b).

The side nodes are placed on different sides of the reaction arrow depending on the main-layout algorithm in use—for branched layout the side nodes are always placed on the left side, whereas for circular layout the side nodes should always be outside the circle; therefore, the position relative to the arrow varies. Similarly, the enzyme nodes(s) can be placed on the same side of the arrow as are the side nodes (used in circular layout), or opposite the side nodes (all other layouts).

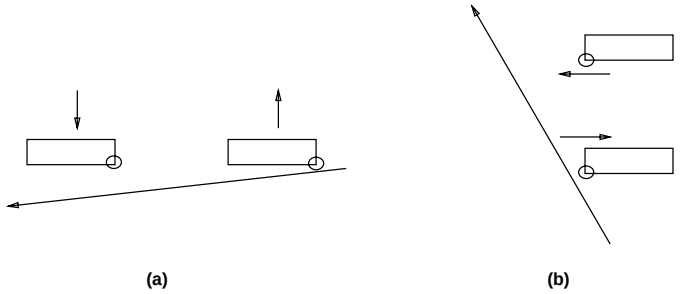

Figure 4: Considerations in layout of side compounds.

6 Complex Junctions

The preceding algorithms must handle additional complexity caused by intersections of more than two reactions in one pathway. For example, Figure 5 (a) shows two products of different reactions (A and B) that are shared reactants of a third reaction, and Figure 5 (b) shows two products of one reaction (B and C) that are shared reactants of a third reaction. Case (b) occurs in glycolysis. Case (b) must be distinguished visually from a situation in which two separate reactions convert A to B, and A to C, respectively, as shown in (c). An even more complex case would involve a single reaction in which multiple reactants and multiple products were all shared by other, separate reactions, as shown in (d). Note that the situations in (a – c) can also occur at the ends of a pathway when a given reaction has more than one terminal main node.

Another type of situation involves multiple interconversions among two consecutive main compounds in the same pathway. Figure4 5 (e) and (f) illustrates two examples, in which there are two and five reactions, respectively, that connect A to X. Case (e) occurs in methionine biosynthesis. Note that different side compounds, and perhaps enzymes, would be drawn next to each reaction arrow.

Figures 5 (g) and (h) show that the preceding types of situations can be combined to yield even more complexity. In case (h) we would want to order A, C, and B to minimize edge crossings. EcoCyc currently implements all these cases, but would not minimize edge crossings in (h).

7 Drawings are Parameterized

The EcoCyc user can view the same pathway in different perspectives by altering several parameters that the pathway drawing algorithms are sensitive to. These parameters control the following aspects of pathway drawing:

- Whether main compounds are drawn as chemical names, chemical structures, or both
- Whether side compounds are drawn as names, structures, or both, or are absent from the drawing
- Whether enzyme names are present or absent from the drawing

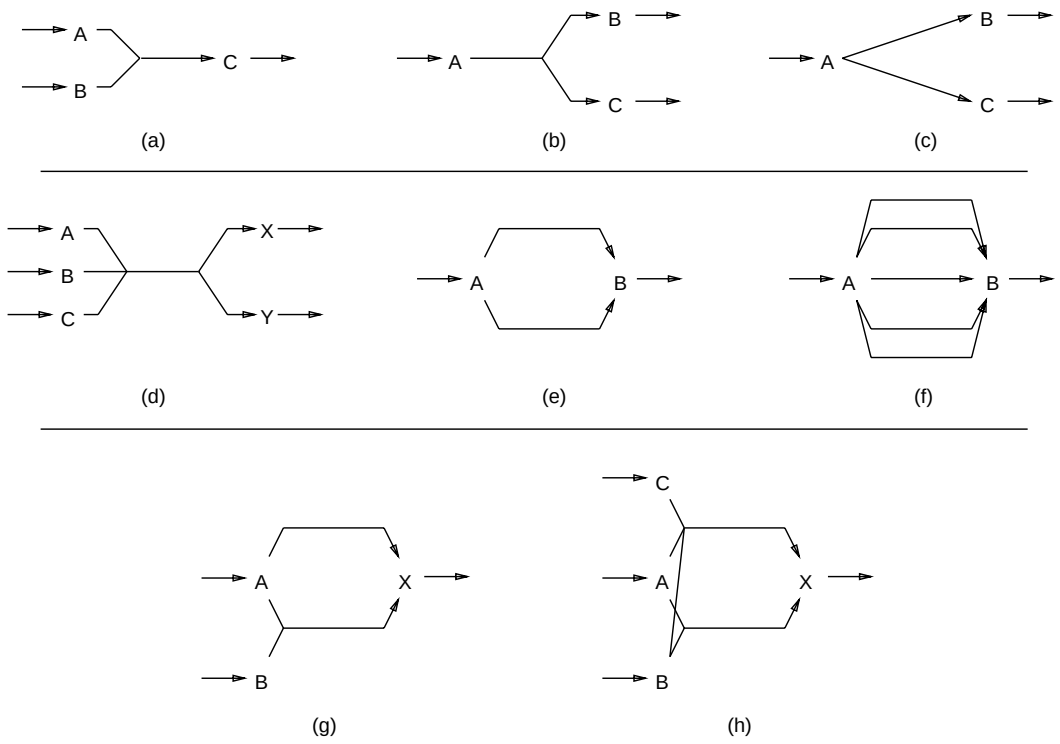

Figure 5: Complex junctions within pathways.

Compound structures are stored in the EcoCyc KB as atom-connectivity lists plus display coordinates. We modified Grasper-CL to support compound structures as a primitive node type; to display these nodes Grasper-CL calls special code that implements compound displays.

8 Pathway Navigation

Individual metabolic pathways are useful abstractions that allow us to focus on an isolated fragment of the metabolism. But all pathways are connected to other pathways, meaning that for every pathway P_1, there exists some compound in P_1 that is also a compound in some other pathway P_2. We call such compounds *connections* between pathways.

We are primarily concerned with the problems of how to allow a user to navigate among an interconnected set of metabolic pathways in order to view pathways within a larger context, and to examine all pathways that produce and consume a given compound. Note that our approach assumes that the user is interested in the relationships among existing, historically defined metabolic pathways, as opposed to the problem of finding new potential pathways in a large metabolic network defined by a database of biochemical reactions [4]. The latter problem is interesting and we intend to address it in future work, but here we focus on managing pathways that have already been defined.

8.1 Compounds that Interconnect Pathways

We use a simple visual cue to indicate compounds that connect pathways. When we draw a pathway, any compound that takes part in more than one pathway is drawn with a box around its name. Extremely common compounds that are found in a majority of pathways, such as water, ATP, and NAD, are never boxed.

A user middle-clicks on a boxed compound to see a list of the other pathways in which that compound appears. Choosing a pathway causes the pathway to be drawn, and the connecting compound is highlighted.

8.2 Superpathways

Individual pathways can be organized into related clusters. Consider aromatic amino acid biosynthesis — a group of related pathways that synthesize the amino acids tryptophan, tyrosine, and phenylalanine (Figure 6). Each of these three amino acids is synthesized from a common predecessor: chorismate. Chorismate, in turn, is synthesized from erythrose-4-P. We believe that users will want to visualize both the individual pathways and larger aggregations of the individual pathways involved in aromatic amino acid biosynthesis. Therefore, we have created pathway frames in EcoCyc that define

- Individual pathways
  - (a) Biosynthesis of tryptophan from chorismate
  - (b) Biosynthesis of tyrosine from chorismate
  - (c) Biosynthesis of phenylalanine from chorismate
  - (d) Biosynthesis of chorismate from erythrose-4-P
- Superpathways
  - (e) A branched pathway from chorismate to each of the three amino acids
  - (f) A branched pathway from erythrose-4-P to chorismate, to each of the three amino acids (equivalent to all of Figure 6)

While the user is looking at a display of one of these pathways, menu commands in the EcoCyc GUI allow navigation to either a superpathway or a subpathway. For example, while viewing (e) the user could click on `Expand` to see (f), or on `Collapse` to see any of (a–d) (which can be selected via a pop-up menu).

The user may also wish to perform another type of expansion and contraction, namely to remove intermediates from a long pathway that does not fit on the screen, in order to see a schematic view of the important branch points in the pathway. Such an operation would remove shikimate and shikimate-5-P and their neighboring intermediates from Figure 6. The inverse operation would add, to a pathway, intermediates that had previously been removed. These operations are fairly simple, but have not yet been implemented in EcoCyc.

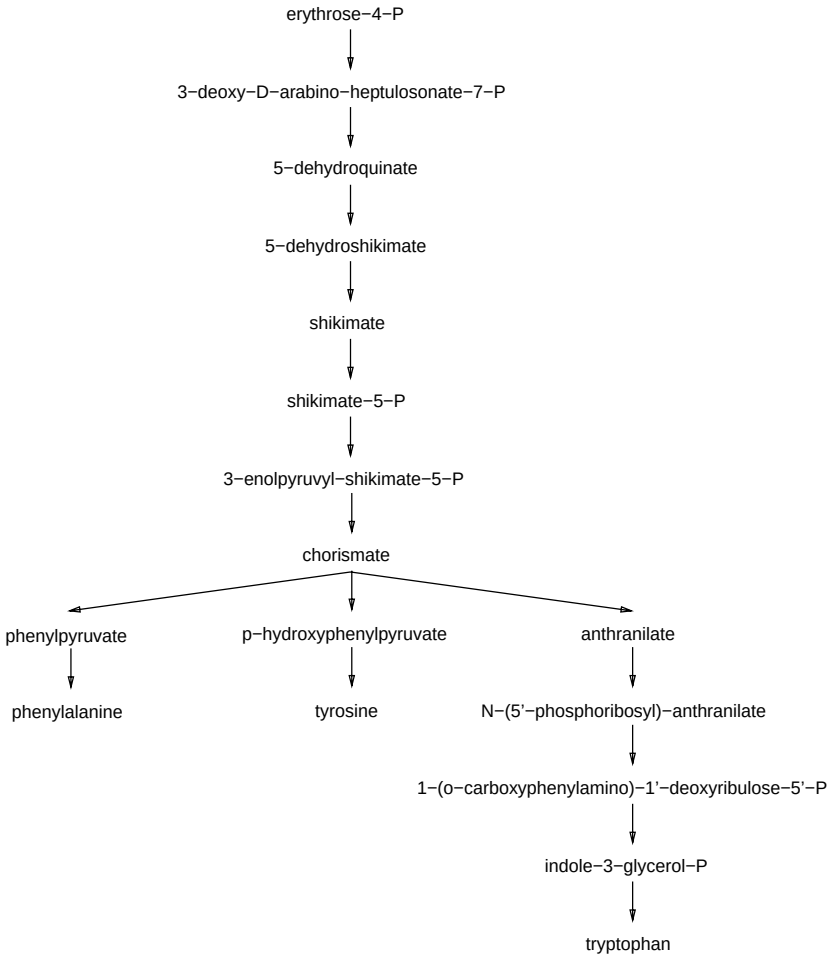

Figure 6: Biosynthesis of aromatic amino acids from erythrose-4-P (enzymes and side compounds are omitted from this drawing).

8.3 Representation of Superpathways

The representation for superpathways is a simple extension of the predecessor-list representation used for individual pathways. This representation simply lists, for each reaction R, the predecessor reaction of R in the pathway. For example, we would describe the pathway in Figure 7 (a) by the predecessor list (in prefix form): `(predecessor 1) (predecessor 2 1)`, meaning that reaction 1 has no predecessor, and the predecessor of reaction 2 is reaction 1. The predecessor list for pathway (b) is `(predecessor 1) (predecessor 3 1)`.

The predecessor list for a superpathway can be constructed as the union of the predecessor lists of its component pathways, modulo adjustments that are sometimes needed. For example, the pathway in Figure 7 (c) is a superpathway containing the pathways in Figure 7 (a) and (b), and its predecessor list is the union of the predecessor lists for (a) and (b). However, if we wish to define pathway (e) as a superpathway containing (a) and (d), we must add a single element to the predecessor list for (e), namely `(predecessor 5 2)`. This bit of "glue" is necessary to join the two component pathways together, but was not required for (c) because its component pathways have a common reaction. In some cases we wish to join pathways that do not even

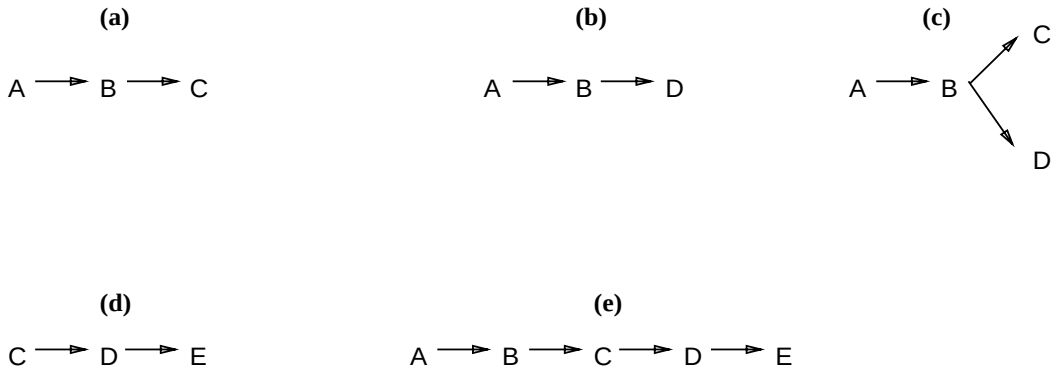

Figure 7: Pathways and superpathways.

Linear	23
Tree	5
Circular	4
Nontree	3
Complex Superpathways	1

Table 1: The number of pathways of different topologies in the EcoCyc KB. This list includes both individual pathways and superpathways. "Nontree" pathways are noncyclic pathways for which some nodes have multiple parents. "Complex superpathways" contain several subpathways of different topologies, such as a cyclic component and a linear component.

share a single compound. For example, it is natural to form a superpathway from glycolysis plus the TCA cycle, but neither of these pathways as commonly defined contains the reaction that converts pyruvate to acetyl-CoA. Therefore we must add two items to the predecessor list for this superpathway.

9 Experimental Results

EcoCyc is implemented in COMMON LISP using the CLIM (COMMON LISP Interface Manager) window system. COMMON LISP and CLIM are portable across a number of platforms; EcoCyc currently runs on the SUN workstation and Macintosh.

Table 1 summarizes the topologies of the 36 pathways currently in the EcoCyc KB. Our drawing algorithms produce excellent results on all of the linear, tree, and circular pathways. Results on the few nontree and complex superpathways are not particularly good (although understandable), because we must develop algorithms for these types of pathways, as we explain next.

10 Limitations and Future Work

The chief limitations of our work thus far concern the complexity of the superpathways EcoCyc can display, and its inability to present other types of information such as regulation. If we define superpathways with more than a few components — particularly if each component pathway has a different topology — our hierarchical layout algorithms do not produce pleasing results. We require more intelligent hierarchical layout algorithms that can properly compute relative positions for component pathways based on the shapes of those components and the locations of the connections between them.

The EcoCyc KB currently contains information about the activators and inhibitors that control each enzyme, and the subunit structures of each enzyme. We plan to add genetic regulatory data later. All of this information is potentially of interest to biologists and should be integrated into pathway visualizations when the user so desires.

We have mentioned the limitation that we cannot generate novel pathways between any user-specified compounds by searching the reaction database. Another limitation concerns pathways with topologies that are fairly close to being trees, but that are not trees. Using a tree layout algorithm on such pathways (such as the pentose phosphate pathway) can produce poor layouts. Good layout algorithms have been developed for such graphs (e.g., [5]); we plan to adapt these algorithms for use with EcoCyc.

11 Related Work

Rouxel et al. have developed a graphical system for displaying metabolic pathways called METALGEN [6]. Rather than computing pathway layouts automatically, they draw each pathway by hand. The advantage of this approach is that, currently, humans may be able to produce better layouts for extremely complex pathways than algorithms can. The disadvantages of this approach are that it is time consuming, that drawings cannot be automatically modified when the definition of a pathway changes (which may happen frequently as new enzymes are discovered), and that pathways with similar topologies may not be drawn consistently.

Letovsky has developed algorithms for pathway layout. Because these algorithms have not been published, a detailed comparison is difficult, but personal communications suggest that our approaches are quite similar. Letovsky's algorithms recognize subgraphs of different topologies, and apply appropriate layout algorithms to them. His layout algorithms include circular algorithms that can draw chordal segments, as well as an algorithm for drawing trees.

Selkov and his colleagues have also developed pathway-display algorithms (also unpublished) for use with the DBEMP database [7], which we have no information about.

12 Summary

The pathway-drawing problem is to assign positions to a pathway graph whose nodes are enzymes and chemical compounds (mains and sides) within the pathway, and whose edges are straight and curved reaction arrows within the pathway. We approach pathway drawing as a graph-layout problem and apply different graph-layout algorithms to biochemical pathways of

different topologies. The main nodes of circular, branched, and linear pathways are positioned by three corresponding layout algorithms. The layout for a pathway with a more complex topology is determined by partitioning the pathway into subgraphs with simpler topologies, applying a simple layout algorithm to each subgraph, and then treating each subgraph as a supernode that is positioned recursively by another layout algorithm. We identified several types of complex junctions between pathways that further complicate the positioning of main nodes and of the reaction arrows between them. Once positions for main nodes have been determined, another algorithm computes positions for side nodes and enzymes.

We described two mechanisms of navigation in the space of metabolic pathways. Our drawings indicate which compounds in a given pathway are also present in other metabolic pathways; users can click on one of these connecting compounds, and choose which connected pathway to draw from a menu of those pathways. In addition, the EcoCyc KB defines superpathways as collections of related subpathways. The user can explode the drawing of a subpathway to include a superpathway that contains it, and they can collapse the drawing of a superpathway to focus on one of its subpathways.

Acknowledgments

We are extremely grateful to Monica Riley for many stimulating discussions on these topics during our collaboration on EcoCyc. This work was supported by grant 1-R01-RR07861-01 from the National Center for Research Resources. The contents of this article are solely the responsibility of the authors and do not necessarily represent the official views of the National Institutes of Health.

References

[1] 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*, Menlo Park, CA, 1994. AAAI Press.

[2] P.D. Karp, J.D. Lowrance, T.M. Strat, and D.E. Wilkins. The Grasper-CL graph management system. *LISP and Symbolic Computation*. In press (see also SRI Artificial Intelligence Center Technical Report 521).

[3] C.K. Mathews and K.E. van Holde. *Biochemistry*. Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1990.

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

[5] E.B. Messinger. *Automatic Layout of Large Directed Graphs*. PhD thesis, University of Washington, 1988.

[6] T. Rouxel, A. Danchin, and A. Henaut. METALGEN.DB: Metabolism linked to the genome of *Escherichia coli*, graphics oriented database. *Computer Applications in the Biosciences*. In press.

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

[8] G. Zubay. *Biochemistry*. Macmillan Publishing Co., New York, NY, second edition, 1988.