

The Pathalyzer: a Tool for Analysis of Signal Transduction Pathways

David L. Dill¹, Merrill A. Knapp², Pamela Gage³, Carolyn Talcott², Keith Laderoute²,
and Patrick Lincoln²

¹ Stanford University

² SRI International

³ Highwire Press

Abstract. The Pathalyzer is a program for analyzing large-scale signal transduction networks. Reactions and their substrates and products are represented as transitions and places in a safe Petri net. The user can interactively specify goal states, such as activation of a particular protein in a particular cell site, and the system will automatically find and display a pathway that results in the goal state – if possible. The user can also require that the pathway be generated without using certain proteins. The system can also find all individual places and all pairs of places which, if knocked out, would prevent the goals from being achieved. The tool is intended to be used by biologists with no significant understanding of Petri nets or any of the other concepts used in the implementation.

1 Introduction

Signal transduction pathways in eukaryotic cells relay information received in the form of physical or chemical stimuli from both the external and internal environment to various intracellular targets. The most common targets are found in the active genome, where subsequent changes in specific gene expression occur in response to the original signal(s). These pathways consist of sequences of biochemical processes that transduce a signal usually through the modification and translocation of proteins and other molecules. Signaling pathways are embedded in large networks having multiple interactions, leading to phenomena such as redundancy and cross talk.

To someone with a computer design background, signaling pathways appear to have some properties in common with hardware and software control mechanisms. For example, pathways can show conditional activation, allowing different or variable responses to different combinations of signals. Currently, much work in signal transduction biology is focused on understanding molecular details of individual pathways in signaling networks. However, as genomic, proteomic, and other high order experimental information accumulates, signaling and other biological pathways will be increasingly represented within complex biological networks. Therefore, it will be essential to acquire at least a qualitative understanding of the global properties of these networks using computational tools. For example, it would be of value to be able to compute an answer to the general question “How are signaling pathways perturbed when environmental conditions change, or when a biochemical process is disrupted?”

This paper describes a prototype software tool, called the Pathalyzer, for querying large-scale qualitative information about signaling pathways. This tool, which uses Petri nets to represent signaling pathways, is unique in several ways. It has a high-level Boolean approach to modeling, based on the Pathway Logic system [3, 19], that enables the analysis of large biological networks. The analysis can answer questions, of types commonly asked by bench biologists, that require searching *all possible pathways* consisting of specific signaling molecules and their associated reactions. Thus, *the model has no built-in concept of pathways*; instead, pathways are generated dynamically from a collection of rules describing individual biochemical steps in signal transduction in response to queries from a user. This capability allows the Pathalyzer to generate a very large number of alternative pathways under the control of a user, who can display differences between the generated pathways. The tool uses sophisticated algorithms developed within the Petri net community for solving the computationally difficult *reachability problem* – in real time. The user needs to understand very little about Petri nets or the various algorithms used in the tool, because the tool provides an easy-to-use interface for entering queries and displaying the results.

A model of a mammalian signaling network has been developed consisting of hundreds of signaling and other molecules involved in the epidermal growth factor receptor (EGFR) system. Cancer biologists are strongly interested in EGFR system, so it has been extensively studied experimentally as a paradigm for growth factor effects on the proliferation and survival of diverse mammalian cells. All of the information incorporated within the model was curated from the appropriate scientific literature. For this model, under development as part of the Pathway Logic project [3, 4, 18], the analysis algorithms of the Pathalyzer run in real time and produce results consistent with the published literature.

Briefly, in Pathway Logic biochemical processes are represented as rules written in Maude [1, 2], a system based on rewriting logic [13, 14]. The Petri nets used by the Pathalyzer are extracted automatically from these Maude rules, and displayed for analysis as described below. The models used by the Pathalyzer are not specific to Maude. It would be possible to generate them from SBML [7] models or to enter the models via a web form interface. Using an existing model requires no knowledge of the model data entry process.

An interactive demonstration of the pathway logic viewer is available at <http://mcs.une.edu.au/~iop/Bionet/>.

2 Modeling Pathways as Petrinets

2.1 Petri Nets

A Petri net is a graphical representation of possible system behavior that generalizes the idea of a state machine to make notions of concurrency and conflict explicit in the structure of the graph. Also, Petri nets have precise, mathematically defined behavior, and sophisticated analysis techniques have been developed over several decades.

A Petri net is a directed bipartite graph with two types of vertices, called *places* and *transitions* (see Fig. 1). Places are drawn as circles, and transitions are drawn as

boxes or thick bars. Each transition has a set of *input places* and a set of *output places*; similarly, each place has input and output transitions. There are directed edges (often called “arcs”), drawn as arrows, from the input places to the transition and from the transition to its output places.

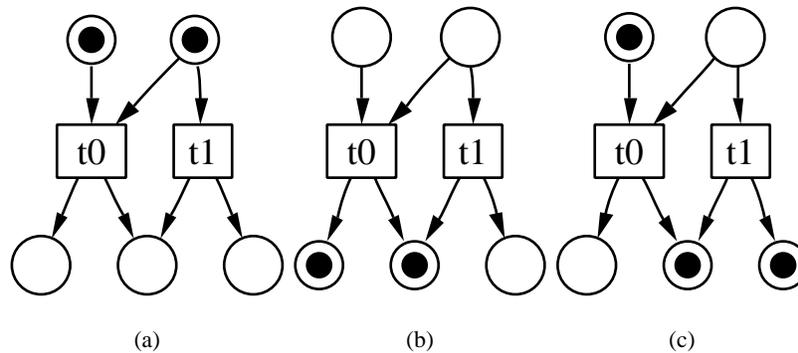


Fig. 1. A basic Petri net example.

The behavior of a system is captured by moving markers called *tokens* around on the net according to certain rules. In Fig. 1, the tokens are shown in the conventional way as smaller black circles inside the places. Mathematically, a placement of tokens on a Petri net is a *marking*, which is an assignment of a non-negative number of tokens (possibly 0) to each place. A place is said to be *marked* if it has one or more tokens in a given marking.

Starting with some initial marking, the behavior of the Petri net evolves in a series of steps that alter the marking. The rule for changing the marking is simple: if all of the input places to a transition are marked, the transition is said to be *enabled*. An enabled transition can *fire*, which removes a token from each input place and adds a token to each output place. A marking of a very small Petri net is shown in Fig. 1(a). Figures 1(b) and 1(c) show the results of firing transitions t_0 and t_1 , respectively.

Any tokens not removed or added by firing a transition remain unchanged. Transitions fire one-at-a-time, so firing one transition may disable other transitions by removing tokens from their inputs, as well as enabling other transitions by adding tokens to their inputs. In Fig. 1, firing either transition disables the other.

Given a Petri net and an initial marking, a *firing sequence* is a list of transitions which can be fired in sequential order. There are usually many possible firing sequences, because whenever several transitions are enabled in the same marking, there is a choice of which transition to fire, and each such choice results in a different firing sequence. A Petri net marking m is said to be *reachable* from an initial marking if, from that initial marking, there is some firing sequence that results in m when the last transition fires.

A marking m that is the result of some firing sequence is said to be *reachable*. The set of reachable markings is generally much larger than the Petri net. For the 1-safe

Petri nets used by the Pathalyzer, there can be as many as 2^n markings for n places. The problem of discovering whether a particular marking is reachable, or whether there is a reachable marking meeting a certain property, is computationally difficult – even the best known algorithms must search an exponential number of markings in the worst case.

There are many different types of Petri nets, and various applications often use Petri nets that are restricted in various ways. The Petri nets in the Pathalyzer are very simple and highly restricted. For example, unlike some Petri nets, they do not model quantitative time, the tokens are not labeled with data, and the transitions are not labeled with firing probabilities. These Petri nets are further limited since each place can only be marked with zero or one token. Petri nets with this property are called *safe* or *1-bounded* Petri nets. These restrictions enable analysis methods that would otherwise be difficult or impossible to implement efficiently.

2.2 Petri Nets for Biological Modeling

Petri nets are very general. The same system can be represented in many different ways using Petri nets, and different aspects of a system can be represented. Furthermore, the basic idea of a Petri net has been augmented in countless ways: adding time, probability, logical conditions on transitions, labeling tokens with data values, etc. There is a voluminous literature on Petri Nets [21].

There are many variants of the Petri net formalism and a variety of languages and tools for specification and analysis of systems using Petri nets. Petri nets have a graphical representation that corresponds naturally to conventional representations of biochemical networks. They have been used to model metabolic pathways and simple genetic networks [10, 17, 9, 11, 12, 5, 8, 6]. These studies have been largely concerned with dynamic or kinetic models of biochemistry, so the models are qualitatively different from those used by the Pathalyzer. The questions answered by these previous efforts, and the types of analysis applied to the net, are completely different from those of the Pathalyzer.

In other work, a more abstract and qualitative view has also been taken, mapping biochemical concepts such as stoichiometry, flux modes, and conservation relations to well-known Petri net theory concepts [23]. Generalized Petri nets have also been used to model higher-level processes, including Malaria parasites invading host erythrocytes [16].

The Pathalyzer is unique in that its analysis depends on solving the *reachability problem* for Petri nets. The Pathalyzer can reliably answer questions about all possible pathways that can evolve from a defined starting state of a model and reach a cell state satisfying specified conditions. Pathways are discovered by solving the *Petri net reachability problem*. Since this problem is computationally difficult and the EGFR network has hundreds of reactions, it would not be unreasonable to guess that the problem could not be even solved in practice. Surprisingly, the pathalyzer is able to answer such queries in a few seconds, so a pathway can be displayed interactively (if such a pathway exists). This effect is achieved through the use of sophisticated heuristics developed in the Petri net community over several decades, which happen to work surprisingly well for the Pathalyzer's models.

2.3 Modeling Pathways

As mentioned above, pathways are not modeled explicitly in Pathway Logic or using the Pathalyzer. Instead, a collection of related reactions or processes is modeled: The user queries the system as to whether something of interest can occur in response to a specific stimulus and, if the answer is affirmative, the Pathalyzer will display a pathway that achieves the result. This perspective has several advantages. First, it saves the redundant work entering individual pathways as data, including the problems of resolving inconsistencies between the pathways and reactions that comprise them. Second, because a large system of reactions/processes has a potentially huge number of similar pathways, representing them individually as data would be very inefficient. The Pathalyzer can dynamically generate pathways in response to user requests instead of storing them.

Here, the reactions/processes in a signaling network are transitions in a Petri net. Each place represents a combination of three factors: a molecule (e.g., the signaling protein *Gab1*), a possible modification (e.g., 'activated'), and a particular location within the cell (e.g., at the cell membrane). The transitions form a network because the places connect the reactions. For example, the product of one reaction is usually the substrate of another reaction.

In the Pathalyzer, the diagrams of Petri nets are drawn slightly differently from the examples of Fig. 1. The places are drawn as ellipses and labeled with the combination of molecule, location, and modification that they represent. The transitions are labeled with the reaction/process they represent. Any place that is both an input and output to a transition is depicted as an input to the transition with a dashed line; finishing the transition will not unmark such a place (the conventional way to draw this situation would be to have arrows in both directions or a double-headed arrow, which tends to clutter the diagrams). For the analysis, some combinations of molecules and modifications in certain locations are assumed to be present initially. This situation determines the initial marking of the Petri net. Places that are initially marked are colored gray in the graphs generated by the tool. Figure 2 represents the reaction *353.Egfr.act.Gab1*, which is enabled when EGFR is activated in the cell membrane (*EGFRact(CM)*), *Gab1* is present in the cytoplasm (*Gab1(cyto)*), and *Grb2* is present in the cell membrane (*Grb2(CM)*). When the reaction finishes, it activates *Gab1* and recruits it to the cell membrane (*Gab1act(CM)*). In the process, *Gab1(cyto)* is used up (there is a solid arrow from *Gab1(cyto)* to the reaction), but *EGFRact(CM)* and *Grb2(CM)* are not (they have dashed arrows). *Gab1(cyto)* is initially present (so it is gray), but *EGFRact(CM)* and *Grb2(CM)* are not.

Hundreds of reactions/processes have been curated, making up pathways involved in signaling from the activated EGFR. The tool displays all of these reactions as a graph.

3 Analysis methods

A primary advantage of Petri nets is that they can be analyzed efficiently to answer non-trivial questions about the behavior of multiple interacting signaling pathways. Several kinds of analysis have been implemented in the Pathalyzer.

The tool is predicated on the assumption that the user is interested in discovering whether certain cell states can be reached, and, if so, how they can be reached. Such

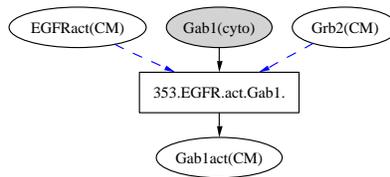


Fig. 2. A single reaction in a model, with associated substances. The inputs of the reaction are activated EGFR in the cell membrane, Gab1 in the cytoplasm, and Grb2 at the cell membrane. The reaction represents the recruitment of Gab1 to the cell membrane by Grb2 and the phosphorylation of Gab1 by EGFR. In the reaction, Gab1 is used up but the other two inputs are not.

queries can reveal the possible interactions between pathways and show the effects of perturbations on pathways. In practice, the analysis methods have also been very helpful for checking the accuracy of the rules entered into the system.

All of the analysis methods are based on the user specifying one or more *goals*, which are places that are to be marked simultaneously in some reachable marking of the network, and *avoid places*, which are places that are not to be marked in the process of reaching a goal (*avoid places* will be treated as being unmarked initially, even if the Petri net includes them in the initial marking). For example, the user may be interested in whether it is possible for the transcription factors *cJun* and *cFos* to be activated in the nucleus at the same time.

3.1 Relevant Subnet

There are several queries that can be automatically answered given a particular goal. The simplest is: “What part of the net is relevant to achieving this goal?” The relevant subset of the net is often much smaller and easier to comprehend than the full net, which is usually quite large.

The determination of what is relevant is a two-stage process. First, places are classified as *potentially markable*. This set is approximate: every place that can be marked starting from the initial marking by some sequence of transition firings is guaranteed to be classified as potentially markable, but there may be potentially markable places that cannot actually be marked in a firing sequence. In practice, this approximation has turned out to be very accurate.

The computation of the potentially markable places is based on iterating two simple rules: a place is potentially markable if it is in the initial marking or if some input transition is potentially fireable. A transition is potentially fireable if *all* of its input places are potentially markable. The algorithm uses these rules to enlarge the set of potentially markable places and potentially fireable transitions until no more places or transitions can be added. If one of the goal places is not potentially markable, the goal cannot be achieved, and the relevant subnet is empty. This fact is reported to the user.

If all of the goal places are potentially markable, the second stage of the analysis is executed. This stage searches backwards from the goal places for places and transitions

that are relevant, again using simple rules: all goal places are relevant and all input places of a relevant transition are relevant, and all input transitions in to a relevant place are relevant. Once again, the rules are applied repeatedly until no more places and transitions can be added to the relevant set.

3.2 Finding a Pathway

Another useful kind of analysis is to search for a particular pathway that leads to specified goals. This analysis first finds the relevant subset of the Petri net as described above, and then searches for a firing sequence starting from the initial marking that results in the goal places being marked.

Finding a pathway can be computationally difficult. The problem of whether a particular marking can be reached from a given initial marking is NP-complete even for safe acyclic Petri nets (and the nets in the Pathalyzer have cycles, which makes them harder to analyze). The obvious algorithm would be to search the graph of all reachable markings, saving the markings in a table to avoid redundant computation. Unfortunately, this method is slow and often does not complete because relatively small Petri nets can have large sets of reachable markings.

Fortunately, efficient algorithms have been devised over the years for solving reachability problems in Petri nets. The approach we use, with great success, is called *stubborn set reduction*. It searches the set of markings selectively, avoiding redundancy stemming from non-interacting transitions. Stubborn set reduction is subtle, so it is not described in more detail here; the interested reader can learn the details elsewhere [15]. In this application, stubborn set reduction works so well that there is little noticeable delay in finding a pathway to a particular goal, if it exists.

The firing sequence that reaches a specified goal would not be easy to interpret if it were just printed as a list. Instead, the tool supports visualization of the pathway by displaying the subset of the original graph consisting of those reactions appearing in the firing sequence and their associated input and output places. Unlike the firing sequence, which has many reactions/processes that occur in arbitrary order, the graphical display shows the dependencies between them, and shows when they are independent. This display is intended to depict a signaling pathway in a way familiar to most biological researchers. Fig. 3 shows a pathway in our EGFR model that leads to *cFos* being activated in the nucleus.

3.3 Avoid Places

One problem is that there are often many different pathways to the same goal. Users would like to have more control over which pathways are displayed by the system. In addition, users may wish to experiment *in silico* by examining the results of perturbations on the net. To provide this control, the Pathalyzer allows the user to specify additional places to *avoid*, by checking a box in the same menu as the goals. A place marked as “avoid” is not allowed to appear in the relevant subset of the net or in a pathway. If the goal cannot be achieved without using an avoid place, it is reported to be unreachable.

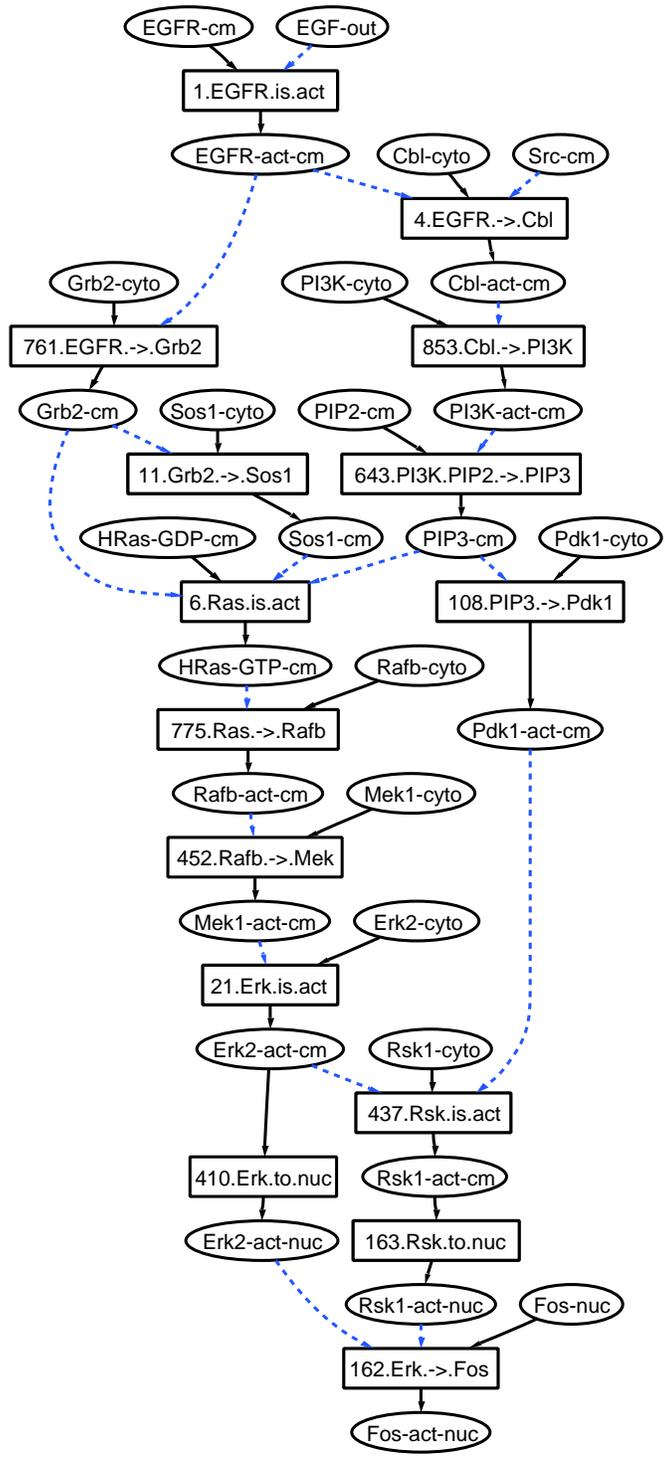


Fig. 3. A pathway as displayed by the tool that shows how the presence of EGF outside the cell can cause *cFos* to be activated in the cell nucleus.

Avoid places are simple to implement in the relevant subset computation. The rules are modified so that avoid places are never added to the potentially markable set, and their input transitions are never added to the potentially fireable set (if any of these transitions were to fire, it would mark an avoid place). The effect of these changes is a list of additional places and transitions to be omitted from both phases of the relevant subnet computation.

No additional effort is required to exclude avoid places in the search for a pathway, since the search is performed on the relevant subset of the net, which has had the avoid places and their associated reactions removed before the pathway search. Hence, no avoid place will appear in a pathway displayed by this command.

Avoid places are useful to reduce clutter in the relevant net, search for a particular pathway, or explore redundancy in the pathways. The typical use of avoid places is to start with goals only and find the relevant subnet or a pathway. Unwanted places are identified in the result and designated as avoid places, then the command is re-run. Typically, it will display a smaller relevant net or a different pathway. Additional avoid places can be selected and the process repeated until the desired result is displayed, or the goals are found to be unreachable.

3.4 Knockout search

Finally, there is one kind of analysis (so far) that is more automatic and global than searching for a pathway. The user may be interested in whether removing one or more signaling molecules renders a particular goal unreachable. For example, this may be interesting to predict the behavior of gene knockouts or identify drug targets. The Pathalyzer has a command to find all “single and double knockouts” relative to a particular specification of goal and avoid places.

The basic method for finding them is simply to automate the process by which it would be done manually. There is a loop that systematically selects one or two additional avoid places and attempts to find a pathway to the goal, logging all failures. The results are then listed for the user. This computation is greatly accelerated by observing that a single knockout *must* appear in every pathway to the goal, so the Pathalyzer searches for a pathway first, then only considers as knockouts places that appear in that pathway.

4 Implementation

The tool is implemented primarily in Java, although it uses two other programs. The structure of the Petri net and its initial marking is stored in an XML file, which is read by the tool. All graphs are laid out by the **dot** program from AT&T Bell Laboratories (it is available free as part of the GraphViz suite of tools from AT&T Bell Labs [22]). The Java interface provides an interactive display for the graphs using the Java-2D graphics library.

Because the graph of an entire network can be too large to fit conveniently on a display screen, the Pathalyzer makes it possible to zoom in and out, and to select a node that is to remain centered at all times. There is also a menu to search for a node by name;

if the node is found, it is highlighted in the network. The nodes immediately connected to a particular node can be displayed, and then the user can interactively enlarge the fragment of the graph that is displayed.

The user selects the goal and avoid places from a menu that lists all of the places, and provides buttons to selection them as “goal,” “avoid,” or “neither.” Once goals and avoids are selected, they persist until they are changed.

The user can cause the relevant subnet to be computed relative to the currently selected goal and avoid places, and then displayed in a separate window. The user can also request that a pathway be found that reaches the specified goals while not marking any of the avoid places. If the goals are in the relevant subnet, the tool invokes LoLA, a Petri net analysis tool that does stubborn set reduction. LoLA is available under GNU General Public License license [20]. If LoLA is able to find a firing sequence to the goals, it stores it in a temporary file which is read by the Java interface. The Java program then finds the subnet of the full Petri net corresponding to the firing sequence, prunes any unnecessary transitions by re-running the relevant subnet algorithm on it, and displays it in a new window using the same code as the display of the original network. For convenience, new analysis commands can be processed from the window displaying the pathway, working from the goals and avoids that resulted in that pathway.

Users often want to compare the pathways that are found by the tool with different choices of avoid places. The tool allows the two most recently computed pathways to be compared. The comparison display shows the nodes from both graphs, and color-codes those that appear only in the first graph, only in the second graph, or in both graphs.

The user can also request the single and double knockouts that would prevent the currently selected goals from being achieved given the currently selected avoid places. The program pops up a window with a simple textual list of the individual places and pairs of places that prevent the goal from occurring.

5 Discussion

At this point, the Pathalyzer is a demonstration prototype. The tool could be used in a variety of ways. At the very least, it could be valuable an interactive “desk reference,” summarizing the vast literature on signaling pathways. If so, an important feature would be to link the reactions to the relevant source literature.

More ambitiously, the Pathalyzer could be used by researchers to interpret the results of experiments, and generate hypotheses for new experiments. For example, various reactions could be knocked out, and the results predicted by the program could be compared with results in the laboratory to test the Pathalyzer’s model. Inconsistencies with experiments indicate a need to refine the model (or the experiments).

Answering the above question requires developing more comprehensive models, which will require expanding the community of curators. There needs to be a system for entering new models that is accessible to biologists who are not experts in Maude. Such a system would also allow researchers to develop pathway models focusing on their particular areas of interest. Work is underway on this question.

It is easy to imagine extensions for modeling reaction rates, quantitative time, and so on. However, any such additions makes analysis more difficult. The longer-run research

challenge is to strike the correct balance between modeling power and analytical power, which can only be found with more use.

Acknowledgments

Some of the early ideas about using Petri nets for this application emerged in discussions with Chris Myers in 2002. Much of the research and programming was done while the first author was on sabbatical at SRI, International in 2003.

Work by the first author on this publication was partially supported by NIH Grant Number NIH 5 U56 CA 112973 from the ICBP Program. Its contents are solely the responsibility of the authors and do not necessarily represent the official views of NIH.

References

1. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. Maude 2.0 Manual, 2003. <http://maude.cs.uiuc.edu>.
2. M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott. The Maude 2.0 system. In *Rewriting Techniques and Applications (RTA'03)*, Lecture Notes in Computer Science. Springer-Verlag, 2003.
3. Steven Eker, Merrill Knapp, Keith Laderoute, Patrick Lincoln, José Meseguer, and Kemal Sonmez. Pathway Logic: Symbolic analysis of biological signaling. In *Proceedings of the Pacific Symposium on Biocomputing*, pages 400–412, January 2002.
4. Steven Eker, Merrill Knapp, Keith Laderoute, Patrick Lincoln, and Carolyn Talcott. Pathway Logic: Executable models of biological networks. In *Fourth International Workshop on Rewriting Logic and Its Applications (WRLA'2002)*, Pisa, Italy, September 19 — 21, 2002, volume 71 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2002. <http://www.elsevier.nl/locate/entcs/volume71.html>.
5. J. S. Oliveira et al. An algebraic-combinatorial model for the identification and mapping of biochemical pathways. *Bull. Math. Biol.*, 63:1163–1196, 2001.
6. J. S. Oliveira et al. A computational model for the identification of biochemical pathways in the Krebs cycle. *J. Computational Biology*, 10:57–82, 2003.
7. M. Hucka et al. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *bioinformatics*, 19(4):524–531, 2003.
8. H. Genrich, R. Küffner, and K. Voss. Executable Petri net models for the analysis of metabolic pathways. *Int. J. STTT*, 3, 2001.
9. P. J. Goss and J. Peccoud. Quantitative modeling of stochastic systems in molecular biology using stochastic Petri nets. *Proc. Natl. Acad. Sci. U. S. A.*, 95:6750–6755, 1998.
10. R Hofestädt. A Petri net application to model metabolic processes. *Syst. Anal. Mod. Simul.*, 16:113–122, 1994.
11. R. Küffner, R. Zimmer, and T. Lengauer. Pathway analysis in metabolic databases via differential metabolic display (DMD). *Bioinformatics*, 16:825–836, 2000.
12. H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano. Hybrid Petri net representation of gene regulatory network. In *Pacific Symposium on Biocomputing*, volume 5, pages 341–352, 2000.
13. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.
14. José Meseguer. Rewriting logic and Maude: A wide-spectrum semantic framework for object-based distributed systems. In S. Smith and C.L. Talcott, editors, *Formal Methods for Open Object-based Distributed Systems, FMOODS 2000*, pages 89–117. Kluwer, 2000.

15. Mogens Nielsen and Dan Simpson, editors. *LoLA: A Low Level Analyser*, volume 1825 of *Lecture Notes in Computer Science*, 2000.
16. Mor Peleg, Iwei Yeh, and Russ B. Altman. Modelling biological processes using workflow and Petri Net models. *Bioinformatics*, 18(6):852–837, 2002.
17. V. N. Reddy, M. N. Liebmann, and M. L. Mavrouniotis. Qualitative analysis of biochemical reaction systems. *Comput. Biol. Med.*, 26:9–24, 1996.
18. C. Talcott, S. Eker, M. Knapp, P. Lincoln, and K. Laderoute. Pathway logic modeling of protein functional domains in signal transduction. In *Proceedings of the Pacific Symposium on Biocomputing*, January 2004. to appear.
19. <http://www.csl.sri.com/users/clt/PLweb/pl.html>. Pathway logic.
20. <http://www.informatik.hu-berlin.de/top/lola/lola.html>. Lola system.
21. <http://www.informatik.uni-hamburg.de/TGI/pnbib/>. The Petri Nets Bibliography.
22. <http://www.research.att.com/sw/tools/graphviz/>. The GraphViz Homepage.
23. I. Zevedei-Oancea and S. Schuster. Topological analysis of metabolic networks based on Petri net theory. *In Silico Biology*, 3(0029), 2003.