Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

SYMBOLIC MODELING OF SIGNAL TRANSDUCTION IN PATHWAY LOGIC

Carolyn Talcott

SRI International
Menlo Park, CA 94025, U.S.A.

ABSTRACT

Pathway Logic is a step towards a vision of *symbolic* systems biology. It is an approach to modeling cellular processes based on formal methods. In particular, formal executable models of processes such as signal transduction, metabolic pathways, and immune system cell-cell signaling are developed using the rewriting logic language Maude and a variety of formal tools are used to query these models. An important objective of Pathway Logic is to reflect the ways that biologists think about problems using informal models, and to provide bench biologists with tools for computing with and analyzing these models that are natural. In this paper we describe the Pathway Logic approach to the modeling and analysis of signal transduction, and the use of the Pathway Logic Assistant tool to browse and query these models. The Rac1 signaling pathway is used to illustrate the concepts.

1 SYMBOLIC SYSTEMS BIOLOGY

Biological processes are complex. They exhibit dynamics with a huge range of timescales—microseconds to years. The spatial scales cover 12 orders of magnitude—metabolite to single protein to cell to organ to whole organism. Oceans of experimental biological data are being generated. How can we use this data to develop better models? Important intuitions are captured in mental models that biologists build of biological processes and the cartoons they draw. The trouble is that these models are not amenable to computational analysis.

Symbolic systems biology is the *qualitative and quantitative* study of biological processes as integrated systems rather than as isolated parts. Our initial goals for symbolic systems biology include: modeling causal networks of biomolecular interactions in a logical framework at multiple scales; developing formal models that are as close as possible to domain experts (biologists) mental models; being able to compute with and analyze these complex networks. The latter includes techniques for abstracting and refining the logical models; using simulation and deduction to compute or check postulated properties; and make testable predictions about possible outcomes, using experimental results to update the models.

There are many challenges in developing symbolic systems models. One challenge is choosing the right abstractions. Biological networks (metabolic, protein, or regulatory, for example) are large and diverse. It is important to balance computational complexity against model fidelity and to be able to move between models of different levels of detail, using different formalisms in meaningful ways. Biological networks combine to produce high levels of physiological organization, for example, circadian clock subnetworks are integrated with metabolic, survival, and growth subnetworks. A second challenge is to be able to compose different views or models of different components into integrated system models.

Pathway Logic (Eker et al. 2002, Eker et al. 2002, Talcott et al. 2004) is one piece of a symbolic systems approach to modeling biological processes. It is an approach to the modeling and analysis of molecular and cellular processes based on rewriting logic. Pathway Logic (PL) models reflect the ways that biologists think about problems using informal models. They are curated from the literature, and written and analyzed using Maude (`<http://maude.cs.uiuc.edu>`), a rewriting-logic-based formalism. A Pathway Logic knowledge base includes data types representing cellular components such as proteins, small molecules, complexes, compartments/locations protein state, and post-translational modifications. Modifications can be abstract, just specifying being activated, bound, or phosphorylated, or more specific, for example, phosphorylation at a particular site. Collections of entities, treated as 'liquid' mixtures, are represented as multisets (unordered collections). Rewrite rules describe the behavior of proteins and other components depending on modification state and biological context. Each rule represents a step in a biological process such as metabolism or intra/inter-cellular signaling. A specific model is assembled by specifying an initial state (called a dish): the cells, their components, and entities such as ligands in the supernatant.

The Pathway Logic Assistant (Talcott and Dill 2005) provides an interactive visual representation of PL models. Using the Pathway Logic Assistant one can

- display the network of signaling reactions for a specified model;
- formulate and submit queries to find pathways, for example, activating one protein without activating a second protein, or exhibiting a phenotype signature such as apoptosis;
- compare two pathways;
- find single or double knockouts—individual or pairs of proteins whose omission prevents reaching a specified state;
- compute and display subnets for which given proteins are critical; and
- map gene expression data onto signaling networks.

The remainder of the paper is organized as follows. §2 provides a brief overview of other work in formally based symbolic approaches to modeling cellular processes. An introduction to rewriting logic and Maude is given in §3. The basic ideas of Pathway Logic are presented in §4, and illustrated with fragments from a model of Rac1 activation. Use of the Pathway Logic Assistant tool to browse and query models is discussed in §5. The paper concludes with a discussion of future directions in §6.

2 SYMBOLIC MODELING OF CELLULAR PROCESSES

Symbolic/logical models allow one to represent partial information and to model and analyze systems at multiple levels of detail, depending on information available and questions to be studied. Such models are based on formalisms that provide language for representing system states and mechanisms of change such as reactions, and tools for analysis based on computational rules or logical inference. Symbolic models can be used for simulation of system behavior. In addition properties of processes can be stated in associated logical languages and checked using tools for formal analysis.

A variety of formalisms have been used to develop symbolic models of biological systems, including Petri nets (Peterson 1981, Goss and Peccoud 1998, Oliveira et al. 2003), the pi-calculus (Milner 1989, Regev et al. 2001), stochastic variants (Priami et al. 2001); stochastic logics and associated model checkers (Calder et al. 2005, Hinton et al. 2006); membrane calculi (Regev et al. 2003, Nielson et al. 2003, L.Cardelli 2004); statecharts (Harel 1987, Efroni et al. 2003), life sequence charts (Kam et al. 2003); and rule-based systems including P-systems (Prez-Jimnez and Romero-Campero 2005); BioCham (Fages et al. 2004, Chabrier-Rivier et al. 2004); and Pathway Logic (Eker et al. 2002, Eker et al. 2002, Talcott et al. 2004). Each of these formalisms was initially developed to model and analyze computer systems with multiple processes executing concurrently.

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 (e.g., see Hofestädt 1994, Reddy et al. 1996, Goss and Peccoud 1998, Küffner et al. 2000, Matsuno et al. 2000, Oliveira et al. 2001, Genrich et al. 2001, Oliveira et al. 2003). These studies have been largely concerned with dynamic or kinetic models of biochemistry. In Zevedei-Oancea and Schuster (2003) a more abstract and qualitative view is taken, mapping biochemical concepts such as stoichiometry, flux modes, and conservation relations to well-known Petri net theory concepts.

A pi-calculus model for the receptor tyrosine kinase/mitogen-activated protein kinase (RTK/-MAPK) signal transduction pathway is presented in Regev et al. (2001). BioSPI, a tool implementing a stochastic variant of the pi-calculus, has been used to simulate both the time and probability of biochemical reactions (Priami et al. 2001).

In Calder et al. (2005) a continuous stochastic logic and the probabilistic symbolic model checker, PRISM, is used to express and check a variety of temporal queries for both transient behaviors and steady state behaviors. Proteins are modeled as synchronous concurrent processes, and concentrations are modeled by discrete, abstract quantities.

BioAmbients (Regev et al. 2003), an adaptation of the Ambients formalism for mobile computations has been developed to model dynamics of biological compartments. BioAmbient type models can be simulated using an extension of the BioSPI tool. A technique for analysis of control and information flow in programs has been applied to analysis of BioAmbient models (Nielson et al. 2003). This can be used, for example, to show that according to the model a given protein could never appear in a given compartment, or a given complex could never form.

Statecharts naturally express compartmentalization and hierarchical processes as well as flow of control among subprocesses. They have been used to model T-cell activation (Kam et al. 2001, Efroni et al. 2003). Although Statecharts is a mature technology with a number of associated analysis and verification tools, it does not appear that these have been applied to the T-cell model. Life Sequence Charts (Damm and Harel 2001) are an extension of the Message Sequence Charts modeling notation for system design. This approach has been used to model the process of cell fate acquisition during C.elegans vulval development (Kam et al. 2003).

P-systems is a multiset rewriting formalism that provides a built in notion of location. A continuous variant of P-systems is used in Prez-Jimnez and Romero-Campero (2005) to model intra-cellular signaling. Locations are used to represent compartmental structure of a cell. Abstract objects represent proteins and small molecules, with different objects used to represent different modifications / states of the same protein. The underlying relation between a protein and its modifications is not made explicit. A system state specifies the quantity of each object in each location. A rate function associates to each rule a function from system states to real numbers, representing the rate of the reaction in that state. This determines how a system state evolves over time. Such models can be used to predict concentration of objects, for example phosphorylated Erk, over time by a discrete step approximation method.

A simple formalism for representing interaction networks using an algebraic rule-based approach very similar to the Pathway Logic approach is presented in (Fages et al. 2004, Chabrier-Rivier et al. 2004). The language has three interpretations: a qualitative binary interpretation much like the Pathway Logic models; a quantitative interpretation in which concentrations and reaction rates are used; and a stochastic interpretation. Queries are expressed in a formal logic called Computation Tree Logic (CTL) and its extensions to model time and quantities. CTL queries can express reachability (find pathways having desired properties), stability, and periodicity. Techniques for learning new rules to achieve a desired system specification are described in (Calzone et al. 2005).

BioSigNet (BSN) (Baral et al. 2004) is a knowledge-based system for representing and reasoning about signaling networks. A BSN knowledge base encodes knowledge about a signal network, including logical statements based on symbols termed fluents and actions. Fluents represent the various properties of the cell and its components while actions denote biological processes (e.g., biochemical reactions, protein interactions) or external interventions. The logical statements describe the impact of these actions on the fluents, how actions can be triggered or inhibited inside the cell. A BSN knowledge base is queried using a temporal logic language over propositions expressing presence or absence of particular fluents. Three classes of queries are identified: prediction (can a state be reached); explanation (find initial conditions that lead to a specified condition); and planning (determining when an action should occur in order to achieve a desired result). In Shankland et al. (2005) BSN is used to model the Erk signaling network.

3 REWRITING LOGIC AND MAUDE

Rewriting logic (Meseguer 1992) is a logical formalism that is based on two simple ideas: states of a system are represented as elements of an algebraic data type; and the behavior of a system is given by local transitions between states described by *rewrite rules*. A rewrite rule has the form $t \Rightarrow t'$ *if* c where t and t' are patterns (terms possibily containing place holder variables) and c is a condition (a boolean term). Such a rule applies to a system in state s if t can be matched to a part of s by supplying the right values for the place holders, and if the condition c holds when supplied with those values. In this case the rule can be applied by replacing the part of s matching t by t' using the matching values for the place holders in t'. The process of application of rewrite rules generates computations (also thought of as deductions). In the case of biological processes these computations correspond to pathways.

Maude (`<http://maude.cs.uiuc.edu>`) is a language and tool based on rewriting logic. Maude provides a high performance rewriting engine featuring matching modulo associativity, commutativity, and identity axioms; and search and model-checking capabilities. Thus, given a specification S of a concurrent systems, one can execute S to find one possible behavior; use search to see if a state meeting a given condition can be reached; or model-check S to see if a temporal property is satisfied, and if not to see a computation that is a counter example.

To introduce Maude notation and give some intuition about how concurrent systems are specified in Maude we consder a specification of a simple Vending Machine. The specification is given in a module named `VENDING-MACHINE`.

```
mod VENDING-MACHINE is
  sorts Coin Item  Marking .
  subsorts Coin Item < Marking .
  op null : -> Marking .  *** empty marking
  op _ _ : Marking Marking -> Marking
        [assoc comm id: null] .
  ops $ q : -> Coin .  *** dollar, quarter
  ops a c : -> Item .  *** apple, cake
  rl[buy-c]: $ => c .
  rl[buy-a]: $ => a q .
  rl[change]: q q q q => $ .
endm
```

First several sorts (think sets or data types) are declared. The basic sorts are `Coin` and `Item`. They represent what you put in and get out of the machine. The sort `Marking` consists of multisets of items and coins. This is specified by the subsort (subset) declarations saying that coins and items are (singleton) markings; and the declaration of the union operator (_ _). The blanks indicate operator argument positions, and union of two markings is represented by placing them side by side, just as one represents a string of characters. The operator attributes `assoc`, `comm`, and `id:null` declare it to be associative and commutative with identity `null`, the empty marking. (Text following `***` is a comment.) After defining the data types to be used, some specific constants are declared: `$` (dollar) and `q` (quarter) of sort `Coin`; and `a` (apple) and `c` (cake) of sort `Item` (the keyword `ops` is used to when declaring multiple constants of the same sort). Finally there are three rewrite rules specifying the vending machine behavior. The rule labeled `buy-c` says that if you have a dollar you can by a cake. More formally, any marking containing an occurrence of `$` can be rewritten to one in which the `$` is replaced by a `c`. Similarly the rule labeled `buy-a` says that with a dollar you can also get an apple and a quarter change. The rule labeled `change` says that when four quarters have accumulated they can be changed into a dollar. Note that if a dollar is present in a marking, there are two ways that the marking could be rewritten, each with a different outcome. If four quarters are also present, the change rule could be applied before or after one of the buy rules without affecting the eventual outcome.

To find one way to use three dollars, ask Maude to rewrite, and a quarter, an apple, and two cakes are the result.

```
Maude> rew $ $ $ .
result Marking: q a c c
```

Although there are several ways to rewrite three dollars, the Maude rewrite command uses a specific strategy for choosing rules to apply, and in this case chose to apply `buy-c` twice and `buy-a` once.

To discover more possibilites Maude can be asked to search for all ways of rewriting three dollars, such that the final state matches some pattern. For example, we can find all ways of getting at least two apples using the pattern

```
a a M:Marking
```

that is matched by any state that has at least two `as`.

```
Maude> search $ $ $ =>! a a M:Marking .
Solution 1 (state 8)
M:Marking --> q q c
Solution 2 (state 9)
M:Marking --> q q q a
```

There are two ways this can be done. In one solution the remainder of the state consists of a cake and two quarters, (indicated by `M:Marking -> q q c` in Solution 1). In the other solution, there is a third apple and three quarters.

We can ask Maude to show us a path (list of rules fired) corresponding to one of these solutions using the function `findPath` and the property `nApples(2)` which holds just if the state matches the pattern `a a M:Marking`.

```
Maude> red findPath(vm($ $ $),nApples(2)) .
result SimplePath:
   spath('buy-c 'buy-a 'buy-a, vm(q q a a c))
```

The function `findPath` calls the Maude model-checker with the assertion that starting in a state with three dollars there is no way to apply to rules to reach a state with two apples. The model-checker looks at all possible sequences of rewrites (paths) and for each one checks the given assertion. If it finds a path for which the assertion fails it returns that path as a counter example to the given assertion. In the example above the path found applies `buy-c` then applies `buy-a` twice. If no counterexample is found, the model-checker returns the boolean `true`.

4 PATHWAY LOGIC BASICS

Pathway Logic models are structured in four layers: (1) sorts and operations, (2) components, (3) rules, and (4) dishes and queries. The *sorts and operations* layer declares the main sorts and subsort relations, the logical analog to ontology. The sorts of entities include Chemical, Protein, Complex, Location (cellular compartments), and Cell. These are all subsorts of the sort, Soup, that represents 'liquid' mixtures, as multisets (unordered collections) of entities. The sort Modification is used to represent post-translational protein modifications. They can be abstract, to specify that a protein is activated, bound, or phosphorylated, or more specific, for example, phosphorylation at a particular site. Modifications are applied using the operator `[ - ]`. For example the term `[EgfR - act]` represents the epidermal growth factor (Egf) receptor in an activated state and `[Rac1 - GTP]` represents Rac1 in its "on" state (loaded with GTP).

A cell state is represented by a term of the form

```
[cellType | locs]
```

where `cellType` specifies the type of cell, for example Macrophage, and `locs` represents the contents of a cell organized by cellular location. Each location is represented by a term of the form `{ locName | components }` where `locName` identifies the location, for example `CLm` for cell membrane, and `components` stands for the mixture of proteins and other compounds in that location.

The *components* layer specifies particular entities (proteins, genes, chemicals) and introduces additional sorts for grouping proteins in families. For example ErbB1L is a subsort of Protein. This is the sort of ErbB1 ligands whose elements include the epidermal growth factor Egf. The *rules* layer contains rewrite rules specifying individual reaction steps. In the case of signal transduction, rules represent processes such as activation, phosphorylation, complex formation, or translocation. The sorts and operations, components, and rules layers make up a Pathway Logic knowledge base. The *dishes and queries* layer specifies initial states, relative to which queries can be answered, and properties of states to be used in formulating queries. Initial states are in silico Petri dishes containing a cell, with its components, and ligands of interest in the supernatant.

Below we give a brief overview of the representation in Maude of signal transduction processes, illustrated using a model of Rac1 activation. This model and several others are available as part of the Pathway Logic Demo available from the Pathway Logic web site `<http://pl.csl.sri.com/>` along with papers, tutorial material and download of the Pathway Logic Assistant tool.

4.1 Modeling Activation of Rac1 in Pathway Logic

Rac1 is a small signaling protein of the Ras superfamily. It functions as a protein switch that is "on" when it binds the nucleotide triphosphate GTP, and "off" when it binds the hydrolysis product GDP. The Pathway Logic model of Rac1 activation was curated using Schmidt and Hall (2002) and many other references (cited as metadata associated with individual rules). In the following we show an initial state for study of Rac1 activation and two example rules. We then briefly sketch some of the ways one can compute with the model. The initial state (called `rac1demo`) is a dish `PD( ... )` with a single cell, of type Fibroblast, and two ligands in the supernatant, Egf and FN, represented by the following term.

```
rac1demo = PD(Egf FN [Fibroblast |
  {CLo | empty}
  {CLm | EgfR Ia5Ib1 PIP2}
  {CLi | [Hras - GDP] [Rac1 - GDP]
       [Cdc42 - GDP] Src}
  {CLc | [Actin - poly] [Ksr1 - phos]
       1433x1 1433x2 C3g Cas Cbl Crk
       Dock Abi1 Elmo Eps8 Erk Fak
       Gab1 Grb2 Mek Pak Pax Pdk1 Pi3k
       PP2a Raf1 Rsk Shp2 Sos1 Vav2 }]) .
```

The location just outside of the cell membrane (tag `CLo`) is empty. The cell membrane (tag `CLm`) has an Egf receptor (EgfR), an integrin, Ia5Ib1, that binds to FN, and the chemical PIP2. The inside of the cell membrane (tag `CLi`) contains Rac1, Hras and Cdc42 loaded with GDP (`[Rac1 - GDP]`, `[Hras - GDP]`,`[Cdc42 - GDP]`) and Src. Thus initiallyl, Rac1 in its "off" state.

The cell cytoplasm (location tag `CLc`) contains additional proteins used in one or more of the activating pathways.

One way to activate Rac1 begins with the activation of the Egf receptor due to the presence of the Egf ligand. The following rule represents this signaling step.

```
rl[1.EgfR.on]:
  ?ErbB1L:ErbB1L
  [CellType:CellType | ct
  {CLo | clo}
  {CLm | clm EgfR} ]
  =>
  [CellType:CellType | ct
  {CLo | clo [?ErbB1L:ErbB1L - bound]}
  {CLm | clm [EgfR - act]} ] .
```

The symbol `?ErbB1L:ErbB1L` is a variable ranging over the sort `ErbB1L`. The symbol `ct` is a variable standing for the remaining locations of the cell, and the symbols `clo` and `clm` are variables standing for the remaining components in their respective locations. The rule matches the `rac1demo` dish by binding the variable `?ErbB1L:ErbB1L` to `Egf`, `ct` to the locations tagged `CLi` and `CLc`, `clo` to `empty`, and `clm` to `Ia5Ib1 PIP2`. Applying the rule replaces `EgfR` by `[EgfR - act]` resulting in the dish

```
 PD(Egf FN [Fibroblast |
  {CLo | [Egf - bound]}
  {CLm | [EgfR - act] Ia5Ib1 PIP2}
  {CLi | [Hras - GDP] [Rac1 - GDP]
       [Cdc42 - GDP] Src}
  {CLc | [Actin - poly] [Ksr1 - phos]
       1433x1 1433x2 C3g Cas Cbl Crk
       Dock Abi1 Elmo Eps8 Erk Fak
       Gab1 Grb2 Mek Pak Pax Pdk1 Pi3k
       PP2a Raf1 Rsk Shp2 Sos1 Vav2 }]) .
```

The following is one of three rules characterizing conditions for the Rac1 switch to be turned on.

```
rl[256.Rac1.on-3]:
  {CLi | cli [Cas - act] [Crk - reloc]
       [Dock - act] [Elmo - reloc]
       [Rac1 - GDP]}
  =>
  {CLi | cli [Cas - act] [Crk - reloc]
       [Dock - act] [Elmo - reloc]
       [Rac1 - GTP]} .
```

This rule describes activation resulting from assembly of Elmo and Crk (modifier `reloc`), with activated Cas, and Dock at the cell membrane. Executing the rule replaces `[Rac1 - GDP]` by `[Rac1 - GTP]`, turning Rac1 on, and leaves the remaining components unchanged.

As explained in §3, Maude provides several ways to compute with or query a model. One can rewrite an initial state, such as `rac1Demo` above, to see a possible final state, or search for all states satisfying some predicate. The main form of query used in PL is a *goals-avoids* query. Goals and avoids are specified in terms of occurrence properties satisfied when a particular protein occurs in a given modification state and location. A pathway satisfies a goals-avoids property if the final state satisfies each of the goals and no state along the pathway satisfies any of the avoids.

As an example, to find a path activating Rac1 that is stimulated by FN alone, we use a query consisting of one goal and one avoid. The goal is the property that Rac1 loaded with GTP is present in the location `CLi`, and the property to be avoided is that Egf is present on the outside of the cell, thus forcing the FN stimulus to be selected. This query could be answered by directly asking Maude, however the textual representation of cell states and pathways quickly becomes difficult to use as the size of a model grows, and an intuitive graphical representation becomes increasingly important. In addition, it becomes important to take advantage of the simple structure of PL models when searching for paths and carrying out other analyses. In the next section we show how the Pathway Logic Assistant can be used to visualize a model as a network of reaction rules, to browse the network, and to specify and execute queries.

5 THE PATHWAY LOGIC ASSISTANT

The Pathway Logic Assistant (PLA) provides an interactive graphical view of a PL knowledge base. Given a dish, PLA generates a Petri net model. Petri nets have a natural graphical representation, and additionally, there are very efficient tools for analyzing the Petri net models generated by PLA. Our Petri net models are a special case of Place-Transition Nets given by a set of occurrences (places in Petri net terminology) and a set of transitions (Stehr 2000). Occurrences can be thought of as atomic propositions asserting that a protein (in a given state) or other component occurs in a given compartment. For example Efg outside a cell is represented by the occurrence `<Egf, out>` and Egfr activated in the cell membrane is represented by `<[EgfR - act], CLm>`. A system state is represented as a set of occurrences (called a marking in Petri net terminology), giving the propositions that are true. A transition is a pair of sets of occurrences. A transition can fire if the state contains the first set of occurrences. In which case the first set of occurrences is replaced by the second set. PL goal properties translate to Petri net properties expressed as occurrences that must be present (places to be marked) and avoids properties translate to occurrences that must not appear (places not to be marked) in a computation. Paths leading from an initial state to a state satisfying a set of goals can be represented compactly as a Petri net consisting of the transitions fired in the path, thus giving query results a natural graphical representation. Execution of the path net starting with the initial state, leads to a state satisfying the goals, and the net representation makes explicit the dependency relations between transitions: some can fire concurrently (order doesn't matter), and some require the output of other transitions to be enabled.

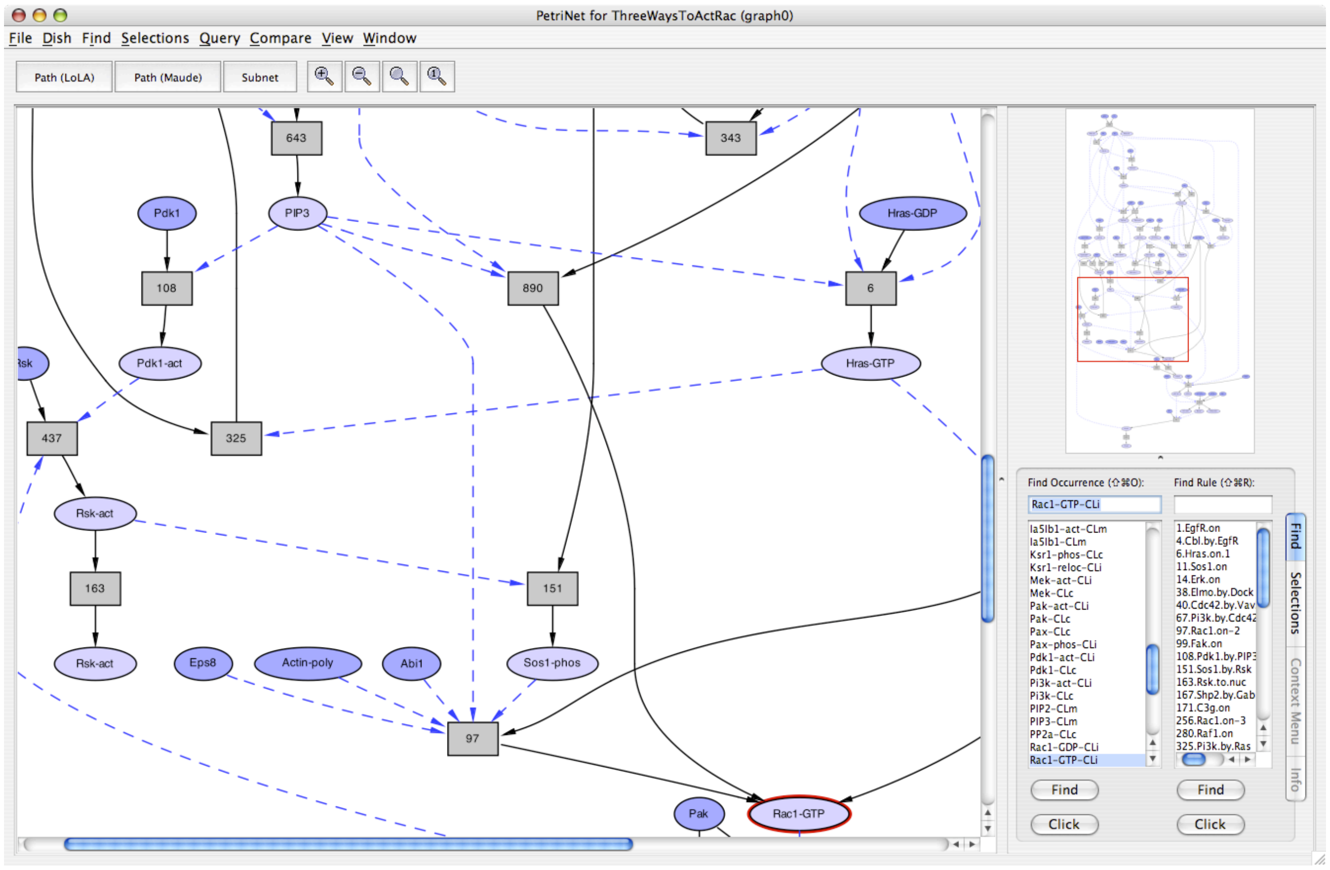
To generate a Petri net model given a PL knowledge base and a dish, the first step is to find all rule instances that could possibly be enabled by repeated rewriting starting from that dish. (Actually, a super set is computed for efficiency.) Then each rule instance is converted into a Petri net transition which consists of the rule label and two sets of occurrences, one corresponding to the lefthand side of the rule and one corresponding to the righthand side. In each case the occurrences are the location components paired with their location name. For example the rule labeled `[1.EgfR.on]` shown in §4 becomes the transition

```
pnTrans[1.EgfR.on]:
 <Egf, out> <Egfr, CLm>
  =>
 <Egf - bound, CLo> <[Egfr - act], CLm> .
```

There is only one instance of this rule in the model generated from the `rac1Demo` dish as there is only one ErbB1L ligand in the dish.

Figure 1 shows a screen shot of the Petri net model generated by PLA from the `rac1Demo` dish. Ovals are occurrences, with initial occurrences darker. Rectangles are transitions. Dashed arrows indicate an occurrence that is both input and output. A magnified view of the portion in the red rectangle is shown in the main view. The thumbnail sketch in the upper right shows the full network. The main frame shows a magnified version of the portion of the network in the red rectangle. The view in the main frame can be changed by dragging the red rectangle around in the thumbnail frame. It can also be changed using the scroll bars. The Finder in the lower right allows one to locate occurrences and rules by name, and center the view on the selected node. To make a query, goals and avoids can be specified either by clicking on the occurrence and selecting goal or avoid in the selection window that appears, or by using the selection window directly. Once goals and avoids have been specified the user can ask to see the relevant subnet or to find a path. The relevant subnet contains all of rules needed for any (minimal) pathway satisfying the query, while the path is just the first path found by the analysis tool. Figure 2 shows the path found in response to the query in which the goal is activation of Rac1 (`<[Rac1 - GTP], CLi>`) and there are no avoids. This path uses downstream elements stimulated by both the Egf and the FN ligands.

In addition to generating subnets and pathways, two subnets and/or pathways can be compared. For this, the two networks are merged into one. Figure 3 shows the result of comparing two other pathways found in the rac1Demo model: one stimulated by Egf only (purple/darker color), and one stimulated by FN only (blue-green/lighter color). The common parts (Src and Rac1) are white. PLA can also be asked to list the knockouts for a given goal (components



Rac1-GTP-CLi

Ia5Ib1-act-CLm
Ia5Ib1-CLm
Ksr1-phos-CLc
Ksr1-reloc-CLi
Mek-act-CLi
Mek-CLc
Pak-act-CLi
Pak-CLc
Pax-CLc
Pax-phos-CLi
Pdk1-act-CLi
Pdk1-CLc
Pi3k-act-CLi
Pi3k-CLc
PIP2-CLm
PIP3-CLm
PP2a-CLc
Rac1-GDP-CLi
Rac1-GTP-CLi

1.EgfR.on
4.Cbl.by.EgfR
6.Hras.on.1
11.Sos1.on
14.Erk.on
38.Elmo.by.Dock
40.Cdc42.by.Vav
67.Pi3k.by.Cdc42
97.Rac1.on-2
99.Fak.on
108.Pdk1.by.PIP3
151.Sos1.by.Rsk
163.Rsk.to.nuc
167.Shp2.by.Gab
171.C3g.on
256.Rac1.on-3
280.Raf1.on
325.Pi3k.by.Ras

Figure 1: Rac1 Activation Model as a Petri Net.

which if deleted prevent the goal from being reached). In the case of Rac1, Src is the only single knockout.

The logic underlying the goals/avoids queries is a temporal logic and in principle is it possible to formulate more complex queries, for example expressing that a particular element is a check-point, or that a particular activation state is always eventually reachable. In Apolzan (2005) a study was carried out in which Pathway Logic models were exported to the SAL language (Shankar 2000) and comparison of the effectiveness of several model-checkers in answering temporal logic queries was made. For the large models that we are interested in querying, bounded model checking was able to find counter-examples and thus to generate pathways, but the special purpose Petri net analysis seems to scale much better, and the goals/avoids queries are easier for the biologists to understand.

6 CONCLUSIONS

We have described the Pathway Logic approach to modeling and analysis of signal transduction networks. The current Pathway Logic knowledge base has more that fifteen hundred components and more than nine hundred curated reaction rules. The current state of Pathway Logic is one step towards the grander vision of symbolic systems biology.

The Pathway Logic Assistant uses several representations of reaction networks. Petri nets have been used for both visualization and analysis. Special purpose formal notations exist for representing molecular interactions and pathways. For example the notation developed by Kitano for representing pathways (Kitano 2003a, Kitano 2003b), and the notation developed by Kohn for representing (Kohn 1999, Kohn 2001). The Kitano notation has existing tool support (Cell Designer) that connects the graphical representation to an underlying computational semantics. This could be used as a graphical interface for curation in Pathway Logic. An important feature of PLA is the ability to generate pathways as query results. In this case the ability to automate the drawing of pathways is crucial. Automatic drawing of computed pathways using the Kitano or Kohn notations is a difficult problem, without good solutions currently.

Pathway Logic does not currently support representing quantitative information such as reaction rates. The focus has been on understanding static and dynamic structure of

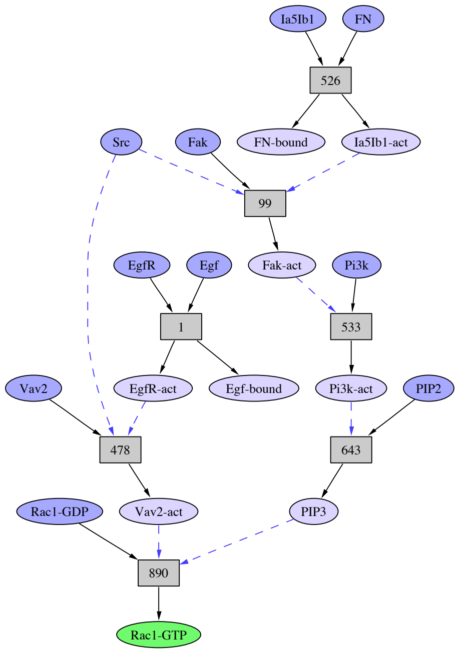


Figure 2: A Pathway Activating Rac1.

reaction networks as this is what the underlying formalism is best suited for. In ongoing projects we are exploring ways to combine other modeling techniques including stochastic simulation and information theoretic analyses.

There are several future directions for development. One is to scale to much bigger models. To make larger models manageable, it will be necessary to develop algorithms to collapse flat networks into hierarchical networks where nodes in the hierarchy correspond to meaningful signaling modules. In addition, developing property preserving abstractions will be important to be able answer queries against the more complex networks. Another direction is to apply the basic approach to different types of systems, such as metabolic networks, gene-regulation networks, or multi-cellular systems, and to integrate models of different types of systems to develop a systems level view.

ACKNOWLEDGMENTS

The author wishes to thank the anonymous referees for helpful criticisms. This work was supported in part by NIH/NIGMS grant GM68146 and by a Public Health Service Grant CA112970 from the National Cancer Institute.

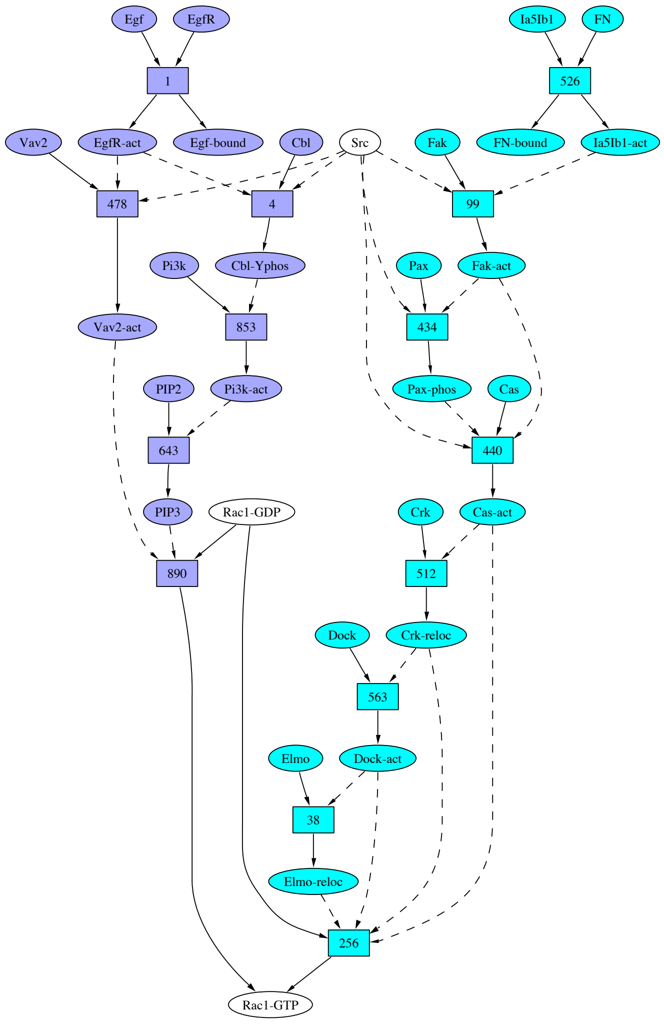


Figure 3: Comparison of Two Pathways Activating Rac1.

REFERENCES

Apolzan, R. 2005. Rapid prototyping applications of formal reasoning tools to biological cellular signalling networks. <http://mcs.une.edu.au/~iop/Data/Papers/>.

Baral, C., K. Chancellor, N. Tran, N. Tran, A. Joy, and M. Berens. 2004. A knowledge based approach for representing and reasoning about signaling networks. *Bioinformatics* 20:i15–i22.

Calder, M., V. Vyshemirsky, D. Gilbert, and R. Orton. 2005. Analysis of signalling pathways using the PRISM model checker. In *Proceedings of the Third International Conference on Computational Methods in System Biology*, ed. G. Plotkin.

Calzone, L., N. Chabrier-Rivier, F. Fages, L. Gentils, and S. Soliman. 2005. Machine learning bio-molecular interactions from temporal logic properties. In *Proceedings of the Third International Conference on Computational Methods in System Biology*, ed. G. Plotkin.

Chabrier-Rivier, N., M. Chiaverini, V. Danos, F. Fages, and V. Schächter. 2004. Modeling and querying biomolecular interaction networks. *Theoretical Computer Science* 351 (1): 24–44.

Damm, W., and D. Harel. 2001. Breathing life into message sequence charts. *Formal Methods in System Design* 19 (1).

Efroni, S., D. Harel, and I. Cohen. 2003. Towards rigorous comprehension of biological complexity: Modeling, execution and visualization of thymic t-cell maturation. *Genome Research*. Special issue on Systems Biology, in press.

Eker, S., M. Knapp, K. Laderoute, P. Lincoln, J. Meseguer, and K. Sonmez. 2002, January. Pathway Logic: Symbolic analysis of biological signaling. In *Proceedings of the Pacific Symposium on Biocomputing*, 400–412.

Eker, S., M. Knapp, K. Laderoute, P. Lincoln, and C. Talcott. 2002. Pathway Logic: Executable models of biological networks. In *Fourth International Workshop on Rewriting Logic and Its Applications*, Volume 71 of *Electronic Notes in Theoretical Computer Science*: Elsevier.

Fages, F., S. Soliman, and N. Chabrier-Rivier. 2004. Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of Biological Physics and Chemistry* 4 (2): 64–73.

Genrich, H., R. Küffner, and K. Voss. 2001. Executable Petri net models for the analysis of metabolic pathways. *Software Tools for Technology Transfer* 3.

Goss, P. J., and J. Peccoud. 1998. Quantitative modeling of stochastic systems in molecular biology using stochastic Petri nets. *Proceedings of the National Academy of Science* 95:6750–6755.

Harel, D. 1987. Statecharts: A visual formalism for complex systems. *Science of Computer Programming* 8:231–274.

Hinton, A., M. Kwiatkowska, G. Norman, and D. Parker. 2006. PRISM: A tool for automatic verification of probabilistic systems. In *12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06)*, ed. H. Hermanns and J. Palsberg, Volume 3920 of *Lecture Notes in Computer Science*, 441–444: Springer.

Hofestädt, R. 1994. A Petri net application to model metabolic processes. *Systems Analysis Modelling Simulation* 16:113–122.

Kam, N., I. Cohen, and D. Harel. 2001. The immune system as a reactive system: Modeling t cell activation with statecharts. In *Visual Languages and Formal Methods (VLFM'01)*, 15–22.

Kam, N., D. Harel, H. Kugler, R. Marelly, A. Pnueli, J. Hubbard, and M. Stern. 2003. Formal modeling of C.elegans development: A scenario-based approach. In *First International Workshop on Computational Methods in Systems Biology*, Volume 2602 of *Lecture Notes in Computer Science*, 4–20: Springer.

Kitano, H. 2003a. A graphical notation for biochemical networks. *Biosilico* 1:169–176.

Kitano, H. 2003b. The standard graphical notation for biological networks. In *The Sixth Workshop on Software Platforms for Systems Biology*.

Kohn, K. W. 1999. Molecular interaction map of the mammalian cell cycle control and dna repair systems. *Molecular Biology of the Cell* 10 (8): 2703–2734.

Kohn, K. W. 2001. Molecular interaction maps as information organizers and simulation guides. *Chaos* 11 (1).

Küffner, R., R. Zimmer, and T. Lengauer. 2000. Pathway analysis in metabolic databases via differential metabolic display (DMD). *Bioinformatics* 16:825–836.

L.Cardelli 2004. Brane calculi interactions of biological membranes. In *Computational Methods in Systems Biology*: Springer.

Matsuno, H., A. Doi, M. Nagasaki, and S. Miyano. 2000. Hybrid Petri net representation of gene regulatory network. In *Pacific Symposium on Biocomputing*, Volume 5, 341–352.

Meseguer, J. 1992. Conditional Rewriting Logic as a unified model of concurrency. *Theoretical Computer Science* 96 (1): 73–155.

Milner, R. 1989. *Communication and concurrency*. Prentice Hall.

Nielson, F., H. R. Nielson, C. Priami, and D. Rosa. 2003. Control flow analysis for bioambients. In *BioConcur*.

Oliveira, J. S., C. G. Bailey, J. B. Jones-Oliveira, D. A. Dixon, D. W. Gull, and M. L. Chandler. 2001. An algebraic-combinatorial model for the identification and mapping of biochemical pathways. *Bulletin of Mathematical Biology* 63:1163–1196.

Oliveira, J. S., C. G. Bailey, J. B. Jones-Oliveira, D. A. Dixon, D. W. Gull, and M. L. Chandler. 2003. A computational model for the identification of biochemical pathways in the Krebs cycle. *Computational Biology* 10:57–82.

Peterson, J. L. 1981. *Petri nets: Properties, analysis, and applications*. Prentice-Hall.

Priami, C., A. Regev, E. Shapiro, and W. Silverman. 2001. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*. In press.

Prez-Jimnez, M., and F. Romero-Campero. 2005. Modelling EGFR signalling cascade using continuous membrane systems. In *Proceedings of the Third International Conference on Computational Methods in System Biology*, ed. G. Plotkin.

Reddy, V. N., M. N. Liebmann, and M. L. Mavrovouniotis. 1996. Qualitative analysis of biochemical reaction systems. *Computational Biological Medicine* 26:9–24.

Regev, A., E. Panina, W. Silverman, L. Cardelli, and E. Shaprio. 2003. Bioambients: An abstraction for biological compartments.

Regev, A., W. Silverman, and E. Shapiro. 2001. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Pacific Symposium on Biocomputing*, Volume 6, 459–470: World Scientific Press.

Schmidt, A., and A. Hall. 2002. Guanine nucleotide exchange factors for rho gtpases: turning on the switch. *Genes and Development* 16:1587–15609.

Shankar, N. 2000. Symbolic analysis of transition systems. In *Proceedings of the International Workshop on Abstract State Machines, Theory and Applications*, 287–302. Springer.

Shankland, C., N. Tran, C. Baral, and W. Kolch. 2005. Reasoning about the ERK signal transduction pathway using BioSigNet-RR. In *Proceedings of the Third International Conference on Computational Methods in System Biology*, ed. G. Plotkin.

Stehr, M.-O. 2000. A rewriting semantics for algebraic nets. In *Petri Nets for System Engineering – A Guide to Modelling, Verification, and Applications*, ed. C. Girault and R. Valk. Springer-Verlag.

Talcott, C., and D. L. Dill. 2005. The pathway logic assistant. In *Third International Workshop on Computational Methods in Systems Biology*, ed. G. Plotkin, 228–239.

Talcott, C., S. Eker, M. Knapp, P. Lincoln, and K. Laderoute. 2004, January. Pathway logic modeling of protein functional domains in signal transduction. In *Proceedings of the Pacific Symposium on Biocomputing*.

Zevedei-Oancea, I., and S. Schuster. 2003. Topological analysis of metabolic networks based on Petri net theory. *In Silico Biology* 3 (0029).

AUTHOR BIOGRAPHY

CAROLYN TALCOTT is a program manager in the Computer Science Laboratory of SRI International, Menlo Park CA, where she leads the Symbolic Systems Technology and Pathway Logic groups. Her web page is `<http:/www-formal.stanford.edu/clt/>`.