Electronic Notes in Theoretical Computer Science 71 (2002)
URL: http://www.elsevier.nl/locate/entcs/volume71.html 18 pages

Pathway Logic:

Executable Models of Biological Networks

Steven Eker, Merrill Knapp, Keith Laderoute, Patrick Lincoln and Carolyn Talcott

Computer Science Laboratory, SRI International, Menlo Park, CA 94025, USA
Email: <firstname>.<lastname>@sri.com

Abstract

In this paper we describe the use of the rewriting logic based Maude tool to model and analyze mammalian signaling pathways. We discuss the representation of the underlying biological concepts and events and describe the use of the new search and model checking capabilities of Maude 2.0 to analyze the modeled network. We also discuss the use of Maude's reflective capability for meta modeling and analyzing the models themselves. The idea of symbolic biological experiments opens up an exciting new world of challenging applications for formal methods in general and for rewriting logic based formalisms in particular.

1 Introduction

In this paper we describe the use of the rewriting logic based Maude tool [7,6] to model and analyze mammalian signaling pathways. A preliminary report on this work appeared in [16]. Here we present the Maude model in more detail, and describe the use of the new search and model checking capabilities of Maude 2.0 [6] to analyze the modeled network. We also discuss the use of Maude's reflective capability [4] for metamodeling and analyzing the models themselves. This turns out to be quite important for managing complex models. The naturalness and relative ease (for biologists) of building and using the Maude models of signaling pathways is very encouraging. The idea of symbolic biological experiments opens up an exciting new world of challenging applications for formal methods in general and for rewriting logic based formalisms in particular.

Biological Signaling Pathways.

The tremendous growth of genomic sequence information combined with technological advances in the analysis of global gene expression has revolutionized research in biology and biomedicine [43]. However, the vast amounts of experimental data and associated analyses now being produced have created an urgent need for new ways of integrating this information into theoretical models of cellular processes for guiding hypothesis creation and testing. Investigation of mammalian signaling processes, the molecular pathways by which cells detect, convert, and internally transmit information from their environment to intracellular targets such as the genome, would greatly benefit from the availability of such predictive models. Although signaling pathways are complex, fundamental concepts have emerged from contemporary research indicating that they are amenable to analysis by computational methods. For example, most signaling pathways involve the hierarchical assembly in space and time of multi-protein complexes or modules that regulate the flow of information according to logical rules [22,28]. Moreover, these pathways are embedded in networks having stimulatory, inhibitory, cooperative, and other connections to ensure that a signal will be interpreted appropriately in a particular cell or tissue [18,36].

Modeling Cellular Signaling Networks.

Various models for the computational analysis of cellular signaling networks have been proposed involving approaches that incorporate rate and/or concentration information to simulate responses to specific stimuli [27,44,2]. Alternatively, networks consisting of signaling pathway targets (genes) are modeled using differential equations to represent changes in the concentrations of both the targets and their pathway components (reviewed in [42]). The objectives here include prediction of transcription signal sequencing, gene expression and understanding of regulation mechanisms, including feedback and multistable switches. Models are used to test hypotheses about elements that are not known experimentally.

C. Tomlin's group at Stanford uses techniques from control theory to analyze the dynamics of reactions that control signalling processes and determine possible equilibrium states [19]. They are able to simulate moderate size arrays of cells using fixed parameters for switching thresholds and to determine analytic relations between possible equilibrium states and threshold values for one and two cell systems. An alternative approach is to apply abstraction techniques to hybrid systems models of metabolic reactions [30]. This gives finite state models that are then subjected to model checking. In [34] biological decision circuits (circuits with feedback mechanisms) are modeled using hybrid logical/kinetic models of circuit elements.

The HELIX project at INRIA has developed the GNA (Genetic Network Analyzer) tool for modeling and simulation of genetic regulatory networks [10]. Models are specified by giving upper and lower bounds for protein concentrations and reaction rates. The tool computes a qualitative state transition graph from which possible qualitative behaviors can be read.

Work by B. Palsson (UCSD) and colleagues uses constraint based Flux balancing techniques to model genome scale metabolic pathways [12,37,9]. A long term objective is to specify metabolic phenotypes that will be expressed by an organism under different environmental conditions. The basic idea is to start with a metabolic network—a graph whose nodes are the metabolites of the modelled system, and whose arrows represent fluxes between these metabolites (arrows may start or end externally). The graph is turned into a system of symbolic differential equations using variables where rate data is unknown. Constraints on the variables are added for known or predicted relations. (Currently regulation is not modeled.) Linear programming techniques (flux based analysis) are used to determine the space of solutions states, and to find optimal solutions relative to objectives such as growth or by-product secretion. The results say what the cell can not do, and give possibilities for what it might do—hypotheses to be tested in laboratory experiments.

A number of tools designed for hardware or sofware analysis have been applied to biological problems [31]. It is interesting to contrast objectives: in the case of software/hardware systems the objective is typically validation or verification of designs, algorithms, implementations; while in the case of biological systems, tools are used for hypothesis generation and testing in models of systems that are usually only partially known.

The pathFinder tool [32], developed for static timing analysis of digital designs has been used to determine families of proteins that will disconnect a target from a source, by enumerating all paths between them. The tool does not work on networks with feedback (cycles). Baysian nets have been used to find statistical dependencies from massive gene expression array data [41]. An analog of the SPICE hardware simulator, BioSpice, is being developed for numerical simulation of biological networks [1]. The π-calculus has been used to represent and forward-simulate a small signaling pathway [38].

Computational predictions of hundreds of metabolic pathways have been made in sequenced genomes using tools developed by Karp's group at SRI [25,24,26], and by the KEGG [23] and WIT [35] groups. The PathoLogic program [26] makes predictions by anology to experimentally known pathways using AI techniques. This tool set also includes a tool for animating biological pathways in the EcoCyc database [39].

Simulations using in silico models founded on kinetic measurements of signaling pathways or networks are important for achieving a detailed understanding of the biochemistry of signal transduction. However, the development of such models is impeded by the great difficulty in obtaining experimental data. The stochastic nature of cellular scale populations of signaling molecules gives additional complications [31,33]. Moreover, new computational methods will be needed to overcome of complexity of simulating and analyzing such detailed models [17].

Our approach focuses on developing highly abstract qualitative models of metabolic and signaling processes that can be used as the basis for analysis by powerful tools, such as those developed in the formal methods community, to study a wide range of questions.

2 Executable Models in Symbolic Systems Biology

The Maude case study presented here together with the EcoCyc work [25] and other initial case studies [30,14] applying formal modeling techniques to cellular pathways is a first step toward developing a new science of *Symbolic Systems Biology*. This new science could provide new capabilities for biologists through power tools that enable the generation of novel inferences about relations among biological entities. These tools would facilitate our understanding of complex biological systems and accelerate research to test hypotheses concerning their operation *in vivo*.

In our approach, rewrite rules describe local change that could occur when an instance of the left-hand side of the rule exists in a modeled system. Rewriting logic allows reasoning about possible complex changes given the basic changes specified by the model. These complex changes can be concurrent; that is, different parts of a compartment can change simultaneously and independently. Furthermore, the changes can be multi-step, allowing reasoning about possible future states of the system. Thus, under very reasonable assumptions rewrite theories can be executed in Maude to describe a biological signaling process over time according to a symbolic model, and can be formally analyzed to reason about properties of the states reachable from an initial state.

Given an executable model such as that described above, the path graph of a given initial state is a graph whose nodes are the reachable states and whose edges are the rules connecting them. Paths through the graph then correspond to possible ways a system can evolve. An execution strategy picks out a particular path among those possible. In such a model, there are many kinds of computation that can be carried out, including: static analysis, forward simulation, forward search, backward search, explicit state model checking, and meta analysis.

Static analysis allows one to examine the structure of the model and to understand how the elements are related and organized (the sort structure). Static analysis also provides a means to check for inconsistencies or ill-formed declarations and to look for missing information.

Forward simulation runs the model from a given initial state using a built-in strategy either for a fixed number of steps, or until no more rewrites apply. This is extremely fast, and very useful for initial exploration.

Forward search is a breadth-first search of all paths through the transition graph for a given initial state. It will find ALL possible outcomes from a given initial state. Search can also be constrained to find a possibly limited number of states satisfying a given property.

Backward search runs the model backwards. For models satisfying certain constraints, backwards search can answer the question: “From what initial states can we get to this state?”. For example it can be used to find all possible precursors to a particular checkpoint.

Explicit state model checking expands the collection of properties that can be investigated. Search concerns only properties of individual states. Model checking also considers properties of paths. For example, we can ask: “If we reach a state that satisfies P then do we always later reach a state satisfying Q?"

Meta analysis allows us to reason about the models themselves. Essential features of models can be abstracted to form families of related models, allowing us to work with uncertainty about reactions. Starting with a base set of known reactions, different instantiations of sets of reactions can be explored. For example, we can search for models where a given path property is true in a given initial state. In addition, rules themselves can be abstracted into families of rules, each family corresponding, for example, to a particular type of reaction, such as activation, inhibition, or transduction.

3 The Signaling Path Model BP

In this section we describe a Maude model, BP (BioPathways), of a major receptor-mediated pathway in mammalian cells [20], focusing on the part centered around the epidermal growth factor receptor (EGFR). BP formalizes reaction data carefully curated from the literature by expert biologists. Currently BP includes more than 650 proteins and more than 500 rules describing possible reaction steps.

In subsection 3.1 we describe how the relevant biological concepts are represented as a (membership) equational theory. In subsection 3.2 we describe the representation of the different types of signaling events using rewrite rules. In subsection 3.3 we give some example analyses focusing on two different initial states. First we will consider a small example corresponding to part of a signaling network concerned with regulation of novel protein kinase C ε (PKCe) by phosphorylation. Here the number of reachable states is small and one can examine the path graph in some detail. The second example involves the larger EGF signaling network. Here the reachable states is large (39992) and one explores the model by selective search and by model checking different initial configurations. In subsection 3.4 we discus the use of reflection and metamodeling.

3.1 Biological sorts and elements

Two basic sorts of interest are `Protein` and `Chemical` (non-biological entities such as calcium ion `Ca++`). We introduce a sort `Thing` to collect together the various basics sorts from which more complex components are formed. These sorts together with examples of their specific members, are declared by the following:

```
sorts Protein Chemical Thing .
subsorts Protein Chemical < Thing .
ops EGFR EGF PIP3 Pdk1 PKCe : -> Protein .
ops  Ca++ : -> Chemical .
```

The keywords `op` and `ops` are used to declare operators from a list of domain sorts into a range sort. In both of the examples above, the list of domain sorts is empty, indicating that the declared operators are constants. Thus `EGFR`, `EGF`, `PIP3`, `Pdk1`, and `PKCe` are declared to be a constants of the sort `Protein`.

Several important concepts from cellular biology are needed to model the signaling networks. They include *protein modification*; *protein association*; and *cellular compartmentalization*.

Protein Modification.

BP includes a comprehensive algebra of protein modifications. Here we show only a small part that is relevant for the later examples.

```
sorts Modification ModSet .
subsort Modification < ModSet .
ops GDP GTP act deact : -> Modification .
op none : -> ModSet .
op _ _ : ModSet ModSet -> ModSet [assoc comm id: none] .
op [_-_] : Protein ModSet -> Protein [right id: none] .
```

There are constants `GDP`, `GTP`, `act`, and `deact` of sort `Modification` which are abstractions of classes of modification with similar effect as regards signaling. `ModSet` is the sort of multisets of modifications, formed from singletons (expressed by the subsort declaration) by multiset union (represented by juxtaposition). `none` is the empty modification set. Sets of modifications are applied to proteins using the operator `[_-_]`. For example `[EGFR - act]` represents the activated form of `EGFR`.

Protein Association.

Signaling proteins commonly associate to form functional complexes [29]. This important phenomenon is algebraically represented by the following declarations:

```
sort Complex .
subsort Complex < Thing .
op _:_ : Thing Thing -> Complex [comm] .
```

Thus each protein or other thing is a singleton complex and two such complexes can be associated by the ":" operator to obtain a multicomponent complex. Complexes are also considered to be `Things`. An example is the inhibitory complex:

```
(IqGap1 : (Ecadherin : bCatenin))
```

that describes `IqGap1` sequestering `Ecadherin` and `bCatenin` so they cannot participate in other reactions.

Protein Compartmentalization.

In eukaryotic cells (cells with a nucleus) proteins and other molecules exist in complex mixtures that are compartmentalized [22]. These compartmentalized mixtures (here termed `Soup`) are algebraically represented by the following declarations:

```
sorts Soup Enclosure MemType .
subsort Thing < Soup .
op empty : -> Soup .
op _ _ : Soup Soup -> Soup [assoc comm id: empty] .
ops CM NM : -> MemType .
op {_|_{_}} : MemType Soup Soup -> Enclosure .
```

An `Enclosure` has a membrane part and an interior, each with its own constituent soup. A `MemType` is used to specify a particular membrane type such as the cell membrane (`CM`) or the nuclear membrane (`NM`). Formally, a soup is multiset of things. The multiset union operation models the presumed fluid or dynamic nature of some subcellular compartments where order in which molecules exist in the soups does not matter. The term

```
{CM| cm:Soup PIP3 [Pdk1 - act] {cyto:Soup PKCe}}
```

represents a cell containing the chemical `PIP3` and the activated protein `[Pdk1 - act]` in the cell membrane and the protein `PKCe` in the cytoplasm (cell interior). The notation `cm:Soup` is a variable declared on the fly, with name `cm` and sort `Soup`. The variables `cm:Soup` and `cyto:Soup` stand for the remaining cell contents. Finally, we declare a sort `Dish` that serves as a top-level soup, and a container for carrying out *in silico* experiments.

```
sort Dish .
op PD : Soup -> Dish .
```

3.2 Biochemical events

To model biochemical events such as signaling processes, we use the dynamic part of a rewrite theory, rewrite rules, to express biochemical processes or reactions involving single or multiple subcellular compartments. For example, the following text describes the first step in the activation of the EGFR signaling pathway, the binding of EGF to the EGFR [40] “Activated `Erk1` is rapidly translocated to the nucleus where it is functionally sequestered and can regulate the activity of nuclear proteins including transcription factors.” In Maude syntax, this signaling process is described by the following rewrite rules:

```
rl[410.Erk1/2.to.nuc]:
  {CM | cm:Soup {cyto:Soup [Erk1 - act] {NM | nm:Soup {nuc:Soup}}}}
   =>
  {CM | cm:Soup {cyto:Soup {NM | nm:Soup {nuc:Soup [Erk1 -
act]}}}} .
rl[438.Erk.act.Elk]:
  [?Erk1/2 - act] Elk1 => [?Erk1/2 - act] [Elk1 - act] .
```

Rule `410` describes the translocation of activated `Erk1` from the cytoplasm to the nucleus. Rule `438` describes the activation of `Elk1` (a transcription factor) by activated `Erk1`. Note that rule labels in BP consist of a number followed by an abbreviated description, which allows a meaningful description of reaction paths using just rule labels. We typically refer to the rule using only the number. We also adopt the convention that symbols beginning with `?` are variables with sort named by omitting the `?` (declared somewhere in the module). Thus `?Erk1/2` is a variable of sort `Erk1/2` (a subsort of `Protein` containing proteins `Erk1` and `Erk2`).

As another example, the following rule represents the experimental result [3] that, in the presence of `PIP3`, activated `Pdk1` recruits `PKCe` from the cytoplasm to the cell membrane and activates it.

```
rl[757.PIP3.Pdk1.act.PKCe]
  {CM| cm:Soup PIP3 [Pdk1 - act] {cyto:Soup PKCe}}
   =>
  {CM| cm:Soup PIP3 [Pdk1 - act][PKCe - act] {cyto:Soup}}
  [metadata "cite = 21961415"] .
```

A rule declaration may also contain additional information, including a metadata attribute. The metadata attribute allows rules to be annotated with arbitrary information that is ignored by the core rewriting engine, but available for use by metalevel operations. In the above rule, the metadata `"cite = 21961415"` gives the Unique Identifier for the MedLine database citation as justification for the rule. Other metadata for biological network rules include reaction rate or concentration data, other forms of literature citations, database links, and confidence ratings.

Figure 1. shows a representation of part of the PKC network that we have modeled, using one of the biological network notations, together with some of the corresponding Maude rules. Arrows connect before and after states of reactants. Rule enabling elements that are not consumed are indicated by lines perpendicular to the arrow. From the picture we see there is an equilibrium between `PIP2` and `PIP3` (indicated by the pair of arrows between them). This equilibrium is expressed by the pair of rules (`643`,`107`). Also, we see (from the pair of vertical arrows) that there are (at least) two ways for `PKCe` to be activated (rules `757` and `758`). In rule `758` the variable `?nPKC:nPKC` ranges over a protein subsort that contains `PKCe` and a small number of related proteins.

3.3 Computing with BP

Now we give some examples showing the use of BP.

3.3.1 PKC analyses

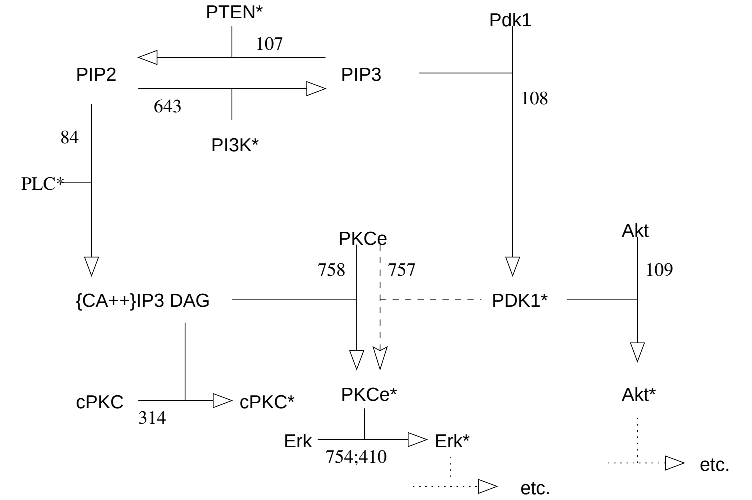
We first consider the PKC network, and carry out some analyses on the initial state `q14` which is defined as follows.

```
op q14 : -> Dish .
eq q14 =
      PD(Ca++ {CM | PIP2 [PI3Ka - act] [PLCb1 - act] [Pten - act]
                          { Erk1 Pdk1 PKCa  PKCe }}) .
```

First we simply rewrite according to two different builtin strategies, invoked by the commands `rewrite` and `frewrite` respectively. The command `rewrite` invokes the Maude default strategy, which is a top down rewriting strategy. Using this strategy we have:

```
rewrite q14 .
result Dish:
    PD({CM | Ca++ DAG IP3 [PI3Ka - act] [PLCb1 - act] [Pten - act]
              {Erk1 Pdk1 [PKCa - act] [PKCe - act]
                 {NM | empty {empty }}}})
```

The command `frewrite` invokes Maude's position fair rewrite strategy. Us-




```
rl [643.PI3KI.mets.PIP2.to.PIP3]:
  PIP2 [?PI3KI:PI3KI - act]  => PIP3 [?PI3KI:PI3KI - act] .

rl [107.Pten.pp.PIP3.to.PIP2]:
  PIP3 [Pten - act] => PIP2 [Pten - act] .

rl [84.PLC.act.PIP2.Ca]:
  Ca++ {CM| PIP2 cm:Soup [?PLC:PLC - act]{cyto:Soup}}
  =>
  {CM|cm:Soup (Ca++ DAG IP3) [?PLC:PLC - act]{cyto:Soup}} .

rl [758.DAG.act.nPKC]:
  {CM | DAG cm:Soup {cyto:Soup ?nPKC:nPKC}}
   =>
  {CM | DAG cm:Soup {cyto:Soup [?nPKC:nPKC - act]}} .
```

Fig. 1. PKC regulation network. A "*" following a protein indicates that it is activated.

ing this strategy we have:

```
frewrite q14 .
result Dish:
    PD({CM | Ca++ DAG IP3  [PI3Ka - act] [PLCb1 - act]
                  [Pten - act] [Pdk1 - act]
                  {Erk1 [PKCa - act] [PKCe - act]
                  {NM | empty {empty }}}})
```

Notice that using the position fair rewrite strategy, `Pdk1` is activated and recruited to the cell membrane, while using the top down strategy it remains unactivated in the cytoplasm. These two executions already show that there are multiple possible outcomes starting with `q14`. We can find all the possible outcomes using the search command.

```
search q14 =>! D:Dish .
```

This constructs the entire path graph for the PKC regulation network starting with initial state `q14` and reports three final states, numbered 6, 20, and 23. State 6 is

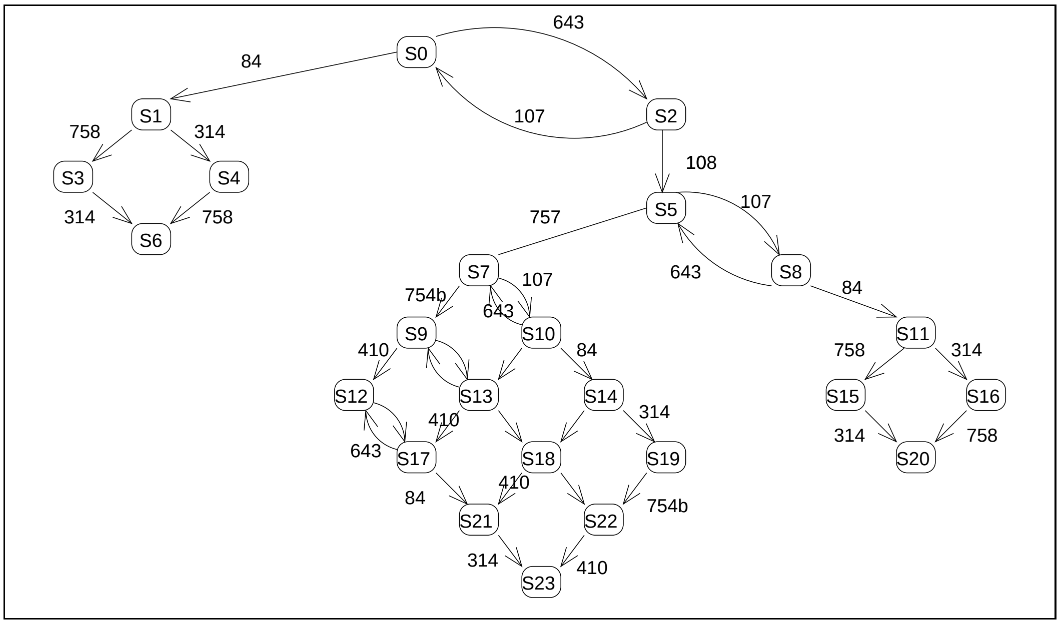


Fig. 2. PKCe path graph

the result obtained using rewrite, and state 20 is the result obtained using frewrite. In state 23 (shown below) `Erk1` has been activated and translocated from the cytoplasm to the nucleus.

```
Solution 3 (state 23)
D:Dish -->
  PD({CM | Ca++ DAG IP3 [PI3Ka - act] [PLCb1 - act] [Pten - act]
                [Pdk1 - act][PKCe - act]
                {[PKCa - act]{NM | empty {[Erk1 - act]}}}})
```

Figure 2. shows the structure of the full path graph for `q14` (S0).
Some of the rule labels in the repeated 'commuting diamonds' are omitted to reduce clutter. Note that the graph essentially splits into three parts, one for each solution. However equilibrium expressed by the pair of rules (`107,643`) adds some complication from a graph analysis point of view.

3.3.2 The EGF-EGFR network

Our analysis of the EGF-EGFR network begins with the state `q1` defined by

```
op q1 : -> Dish .
eq q1 = PD(EGF {CM | EGFR Pak1 PIP2 nWasp [H-Ras - GDP]
                         {Akt1  Gab1 Grb2 Gsk3 Eps8 Erk1
                           Mek1 Mekk1 Mekk4 Mkk4 Mkk3 Mlk2
                           Jnk1 p38a p70s6k Pdk1 PI3Ka PKCb1
                           Raf1 Rsk1 Shc Sos [Cdc42 - GDP]
                           {NM | empty {cJun cFos }}}}) .
```

Suppose we want to find out if a state with `cJun` and `cFos` activated is reachable. And further suppose we want to know what happens if `PI3Ka` is removed from `q1` (call it `q1x`). The Maude model-checker [13,15] can be used to find a counter-example to the assertion that such a state is never reachable. The interface to the Maude model checker is embodied in the MODEL-CHECKER module. This module defines syntax for LTL (Linear Temporal Logic) formulas built over a sort `Prop`. It also introduces a sort `State` and a satisfaction relation $\models$ on states and LTL formulas.

To use the model checker, the user defines particular states and propositions and axiomatizes satisfaction on these states and propositions. LTL semantics lifts satisfaction from propositions to arbitrary LTL formulas. In our BP experiments states are dishes, thus `Dish` is declared to be a subsort of `State`. A proposition `prop1` that is satisfied by dishes with `cJun` and `cFos` activated is defined as follows.

```
subsort  Dish < State .
op prop1 : -> Prop .
eq PD(out:Soup
      {CM | cm:Soup
               {cyto:Soup
               {NM | nm:Soup
                        {nuc:Soup [cJun - act] [cFos - act] }}}})
   |= prop1 = true .
```

The formula `~ <> prop1` says that `prop1` never holds. Executing the command

```
red  q1  |= ~ <> prop1 .
```

results in a counter example – a reachable state satisfying `prop1` together with a path leading to that state. A more biologist friendly notation for such uses of the model-checker is obtained by defining

```
eq findPath(S:State,P:Prop) = getPath(P:Prop, S:State |= ~ <> P:Prop) .
```

where `getPath` abstracts the counter example returned by Maude to a list of rule labels followed by a state satisfying the given proposition. If we execute `findPath(q1,prop1)` and `findPath(q1x,prop1)` we discover that indeed, in both cases a state with `cJun` and `cFos` both activated is found. However, `Akt1` and `Eps8` are both activated when `PI3Ka` is present, but not when it is removed. (For details see Figure 4 in the appendix.)

3.4 Meta analysis

There is a variety of metadata associated with the executable model of a signaling network (and many other systems). This includes information justifying or qualifying a rule and rate information about a rule. There is also ordering information such as "activation rules should be applied before translocation rules". There are patterns that rules of a particular class of event should follow, and "biological type checking" can be used to ensure that these patterns and other constraints on the model structure are satisfied. Furthermore, as the model of a system grows, consistency checks become more important and it is useful to be able to answer simple questions such as "What are all the rule labels?" or "What constants of sort `Protein` have been declared?". Using the reflective capability of Maude it is easy to formalize such metadata and use it to further control experiments or to answer questions and carry out various meta-analyses. As an example, we outline how to answer the question concerning the declared proteins. The BP metamodel extends the Maude META-LEVEL module [5], which provides a metarepresentation of modules and rewriting. For questions about constants, an operator `constSetBySort` is defined which takes a (metarepresented) module and a sort and returns the set of quoted identifiers naming all declared constants having the given sort in the given module. For example, assuming `BP` names the metarepresentation of the BP model (this can be generated using Full Maude 2.0 [11]), reducing `constSetBySort(BP, 'Protein)` results in a listing of all the protein constants of the model. The operator `constSetBySort` selects the set of operation declarations from the module and, using auxiliary functions `constSetBySortX` and `constBySortX`, checks each of the declarations to see if the operation is a constant and, if so, whether the declared sort is less or equal to the given sort in the sort partial order of the module. The operator declarations and the equation defining the key operation `constBySortX` are as follows.

```
op constSetBySort : Module  Sort  -> QidSet .
op constSetBySortX : OpDeclSet  SortSet  -> QidSet .
op constBySortX : OpDecl  SortSet  -> QidSet .
...
eq constBySortX((op q:Qid :
                 tl:TypeList -> t:Type[atts:AttrSet].),
                 ss:SortSet)
    = if ((tl:TypeList == nil) and member(t:Type,ss:SortSet))
      then q:Qid
      else none
      fi .
```

Application of `constBySortX` destructs an operator declaration by pattern matching, extracting the operation name, `q:Qid`, the argument type list `tl:TypeList`, the result type `t:Type`, and any attributes. There are many more useful questions that can be answered using similar techniques, such as “find all rules that activate `Pdk1`” or “find all rules that move activated `Erk1` to the nucleus”.

In addition to being able to ask questions about the model structure and content our biologist would like to visualize rules and path segments in a form more intuitive than the Maude rule syntax. In the literature pictures called “cartoons” are used for this purpose. Although suggestive, these pictures leave much information implicit, they can be ambiguous, and there are many different ways of picturing a given reaction or event (shades of OO design notations). There have been some proposals for “standard” graphical elements [8,21] but they lack an underlying formal model and thus are still just icons to manipulate by hand. We have defined a cartoon data type in the BP metamodel to serve as a formal foundation for visualization. Formal cartoons contain all the information of a rule, given some assumptions about cell structure, they eliminate information such as place holders that are irrelevant for visualizing, and they make explicit other information such as biological

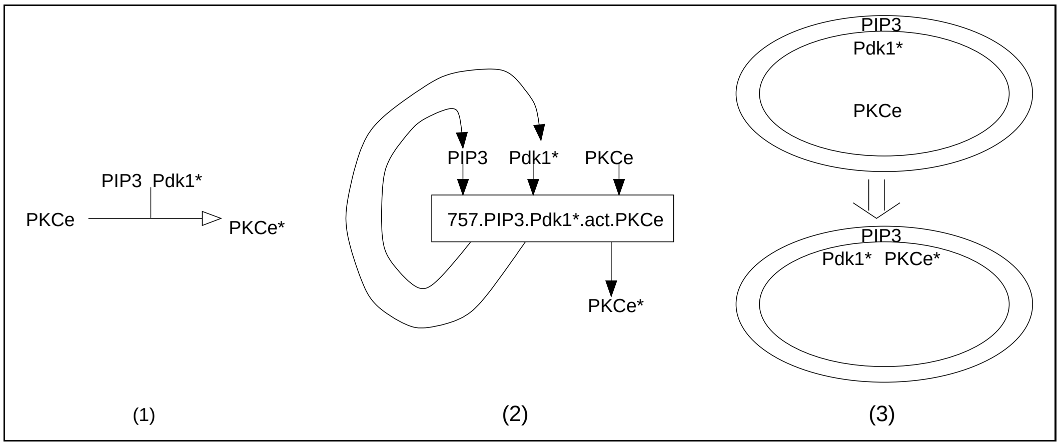


Fig. 3. Three ways to visualize the PKCe rule

sorts. The relevant elements of a rule are its label and the non-placeholder "items" in the left- and right-hand sides. Each item is represented in the cartoon by its print name, its biological sort, its location within the rule soup, and an abstraction of its modification state. Thus a cartoon in the BP metamodel is a term of the form:

```
ctoon(ruleid:Qid, lhs:ItemSet, rhs:ItemSet)
```

where an item is a term of the form

```
item(pname:String, bsort:BioSort, loc:Location, a:Activation)
```

A metaoperation `rs2cts` has been defined to map rules to cartoons. Applying `rs2cts` to the rule `757.PIP3.Pdk1.act.PKCe` (see subsection 3.2 produces the cartoon:

```
ctoon('757.PIP3.Pdk1.act.PKCe,
      ( item("PIP3", Protein, membrane('CM), noact)
        item("PKCe", Protein, interior('CM), noact)
        item("Pdk1", Protein, membrane('CM), act) ),
      ( item("PIP3", Protein, membrane('CM), noact)
        item("PKCe", Protein, membrane('CM), act)
        item("Pdk1", Protein, membrane('CM), act) ))
```

Figure 3.4 shows three possible visualizations that could be generated automatically from this formal cartoon. View (1) is the representation of the rule given in Figure 3.2. This is a common signalling network representation that shows what is changed and what else must be present. View (2) is a visualization inspired by the Petri Net formalism for networks of computer processes. It shows the rule label in a box, the required cell contents before the reaction are indicated by arrows leading to the box and resulting contents after the reaction by arrows leading away from the box. Thus the feedback loops show elements that are unchanged (catalysts). View (3) shows a representation of the cell state before and after, adding information about where in the cell the reaction elements are located. The double line represents the cell membrane, and the inside of the oval represents the cytoplasm. Location information could easily be added to the picture in (2).

The ability to generate different visualizations from the rules is important for a number of reasons, including: as an aid in understanding the signal flow or chemical processes in a path; and as an additional check that a rule correctly captures known data about the reaction. Although they have the look of "cartoons" these visualizations have an underlying formal representation that can be computed with and composed to form more complex rules representing path fragments whose internal structure can be hidden.

4 Conclusions and Future Work

We have described a representation of biological signaling networks in Maude and shown how such a model can be used for various in silico experiments and advanced forms of symbolic analysis. Executable models such as this put new kinds of computational capabilities into the hands of biologists. Their predictive power can be used to generate hypotheses to be tested by other means and to design experiments. They also serve as a starting point for a wide range of exciting applications of formal modeling.

Although the BP model is already quite advance there is more data to add, and much more to do to improve and extend the analysis and metamodeling capabilities. An important issue is development of methods to simplify and abstract signalling path models for particular classes of question, in order to be able to scale to analysis of much larger systems and answering more complex questions.

Acknowledgments. We would like to thank the anonymous referees for their helpful suggestions. We also thank other past and present members of the Pathway Logic group—José Meseguer, John Bashkin, Kemal Somnez, Harry Bratt, and Rasika Kumar—for the many fruitful discussions during the development of the model and visual representations.

References

[1] A. P. Arkin Labs. `http://genomics.lbl.gov/~aparkin/`.

[2] A. R. Asthagiri and D. A. Lauffenburger. A computational study of feedback effects on signal dynamics in a mitogen-activated protein kinase (mapk) pathway model. *Biotechnology Progress*, 17:227–239, 2001.

[3] V. Cenni, H. Döppler, E. Sonnenburg, N. Maraldi, A. C. Newton, and A. Toker. Regulation of novel protein kinase C ε by phosphorylation. *Biochem J.*, 363:537–545, 2002.

[4] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Marti-Oliet, and J. Meseguer. Metalevel Computation in Maude. In C. Kirchner and H. Kirchner, editors, *2nd International Workshop on Rewriting Logic and Its Applications, WRLA'98*, volume 15 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1998. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[5] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, and J. Quesada. Maude as a metalanguage. In C. Kirchner and H. Kirchner, editors, *2nd International Workshop on Rewriting Logic and Its Applications, WRLA'98*, volume 15 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1998. `http://www.elsevier.nl/locate/entcs/volume15.html`.

[6] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, and J. Quesada. Towards Maude 2.0. In *Third International Workshop on Rewriting Logic and Its Applications (WRLA'00), Kanazawa, Japan, September 18 — 20, 2000*, volume 36 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 2000. `http://www.elsevier.nl/locate/entcs/volume36.html`.

[7] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. Quesada. A tutorial on Maude. `http://maude.csl.sri.com`, March 2000.

[8] D. L. Cook, J. F. Farley, and S. J. Tapscott. A basis for a visual language for describing, archiving and analyzing functional models of complex biological systems. *Genome Biology*, 2(4), 2001. `http://genomebiology.com/2001/2/4/research/0012`.

[9] M. W. Covert and B. O. Palsson. Transcriptional regulation in constraints-based metabolic models of Escherichia coli. *Journal of Biological Chemistry*, 2002. in press.

[10] Hidde de Jong, Johannes Geiselmann, Celine Hernanedez, and Michel Page. Genetic network analyzer: A tool for qualitative simulation of genetic regulatory networks. Technical Report RR 4262, INRIA Rhone-Alpes, September 2001.

[11] P. Duran. Full Maude 2.0 alpha release notes, 2002.

[12] J. S. Edwards, M. Covert, and B. O. Palsson. Metabolic modeling of microbes: the flux balance approach. *Environmental Microbiology*, 4(3):133–140, 2002.

[13] S. Eker. Maude 2.0 alpha release notes, 2002.

[14] S. Eker, P. Karp, P. D. Lincoln, and P. Romero. From bugs to BDDs: Diet planning for E. coli. draft of work in progress.

[15] S. Eker, J. Meseguer, and A. Sridharanarayanan. The Maude LTL model checker. this volume, 2002.

[16] Steven Eker, Merrill Knapp, Keith Laderoute, Patrick Lincoln, José Meseguer, and Kemal Sonmez. Pathway logic: Symbolic analysis of biological signaling. In *Pacific Symposium on Biocomputing*, pages 400–412. World Scientific, January 2002.

[17] D. Endy and R. Brent. Modelling cellular behaviour. *Nature*, 409:391–395, 2001.

[18] D. Fambrough, K. McClure, A. Kazlauskas, and E. S. Lander. Diverse signaling pathways activated by growth factor receptors induce broadly overlapping, rather than independent, sets of genes. *Cell*, 97:727–741, 1999.

[19] R. Ghosh and C. J. Tomlin. Lateral inhibition through Delta-Notch signaling: A piecewise affine hybrid model. In M. D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control HSCC'01*, volume 2034 of *LNCS*, pages 232–246. Springer, 2001.

[20] A. Gschwind, E. Zwick, N. Prenzel, M. Leserer, and A. Ullrich. Cell communication networks: epidermal growth factor receptor transactivation as the paradigm for interreceptor signal transmission. *Oncogene*, 20:1594–1600, 2001.

[21] Michael Hucka, Andrew Finney, Herbert Sauro, and Hamid Bolouri. Systems biology markup language, 2001. `http://www.cds.caltech.edu/erato/sbml/docs/index.html`.

[22] J.D. Jordan, E. Landau, and R. Iyengar. Signaling networks: The origins of cellular multitasking. *Cell*, 103:193–200, 2000.

[23] M. Kanehisa, S. Goto, Shuichi Kawashima, and Akihiro Nakaya. The KEGG databases at genomenet. *Nucleic Acids Research*, 30:42–46, 2002.

[24] P. Karp. Biocyc web site, 2002. `biocyc.org`.

[25] P. D. Karp. Pathway databases: A case study in computational symbolic theories. *Science*, pages 2040–2042, 2001.

[26] P.D. Karp, S. Paley, and P. Romero. The pathway tools software, 2002. in press.

[27] K. W. Kohn. Functional capabilities of molecular network components controlling the mammalian G1/S cell cycle phase transition. *Oncogene*, 16:1065–1075, 1998.

[28] K. W. Kohn. Molecular interaction map of the mammalian cell cycle control and DNA repair systems. *Mol Biol Cell*, 10:2703–2734, 1999.

[29] W. Kolch. Meaningful relationships: The regulation of the Ras/Raf/MEK/ERK pathway by protein interactions. *Biochem J.*, 351:289–305, 2000.

[30] P. Lincoln and A. Tiwari. Automated techniques for stability analysis of delta-notch lateral inhibition mechanism, 2002. in preparation.

[31] L. Lok. Software for signaling networks, electronic and cellular. *Science STKE*, PE11, 2002.

[32] L. Lok, L.Eiden, and M. Dayhoff-Brannigan. Pathfinder, 2002. `http://intramural.nimh.nih.gov/lcmr/smn/special_projects.html`.

[33] H. H. McAdams and A. Arkin. It's a noisy business! genetic regulation at the nanomolar scale. *Trends Genet*, 15:65–69, 1999.

[34] H. H. McAdams and L. Shapiro. Circuit simulation of genetic networks. *Science*, 269:650–656, 1995.

[35] R. Overbeek, N. Larsen, G. D. Pusch, M. D'Souza, E. Selkov Jr., N. Kyrpides, M. Fonstein, N. Maltsev, and E. Selkov. WIT: integrated system for high-throughput genome sequence analysis and metabolic reconstruction. *Nucleic Acids Research*, 28:123–125, 2000.

[36] T. Pawson and T. M. Saxton. Signaling networks–do all roads lead to the same genes? *Cell*, 97:675–678, 1999.

[37] N. D. Price, J. A. Papin, and B. O. Palsson. Determination of redundancy and systems properties of the metabolic network of helicobacter pylori using genome-scale extreme pathway analysis. *Genome Research*, 12:760–769, 2002.

[38] A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the π-calculus process algebra. In R. B. Altman et. al., editor, *Pacific Symposium on Biocomputing*, pages 459–470. World Scientific, 2001.

[39] P. R. Romero and P. D. Karp. Nutrient-related analysis of pathway/genome databases. In R. B. Altman et. al., editor, *Pacific Symposium on Biocomputing*, pages 471–482. World Scientific, 2001.

[40] J. Schlessinger. Cell signaling by receptor tyrosine kinases. *Cell*, 103:211–225, 2000.

[41] B. Shipley. *Cause and Correlation in Biology: A User's Guide to Path Analysis, Structural Equations, and Causal Inference*. Cambridge University Press, Cambridge, 2000.

[42] P. Smolen, D. A. Baxter, and J. H. Byrne. Mathematical modeling of gene networks. *Neuron*, 26:567–580, 2000.

[43] O.G. Vukmirovic and S.M. Tilghman. Exploring genome space. *Nature*, 405:820–822, 2000.

[44] G. Weng, U. S. Bhalla, and R. Iyengar. Complexity in biological signaling systems. *Science*, 284:92–96, 1999.

Appendix: The EGFR counter examples

Figure 4 shows the results of using the model-checker to find paths leading to activated `cJun` and `cFos` with and without `PI3Ka` present initially. To make comparison of the two counter examples easier, `***`s tag lines of rule labels that appears in one but not the other.

```
red findPath(q1,prop1) .
result SimplePath:   spath('1.EGFLike.act.EGFR '353M.EGFR.act.Gab1
    '418B.EGFR.act.Cdc42 '349.Cdc42.act.Mekk1 '471.Cdc42.act.Mekk4
    '470.Cdc42.act.Mlk '3.EGFR.act.Shc
    '429.Gab1.act.PI3K '498.PI3K.act.Eps8  ***
    '411.Shc.act.Grb2 '324.Shc.Grb2.act.Sos '618.Cdc42.act.Wasp
    '6.Sos.act.Ras '280.Ras.act.Raf '274.Raf1.act.Mek1/2
    '643.PI3KI.mets.PIP2.to.PIP3 '108.PIP3.act.Pdk1 ***
    '109.PIP3.Pdk1.act.Akt '122.Akt.deact.Gsk3 '121.Akt.to.nuc ***
    '21.Mek1/2.act.Erk1/2 '437.Erk.act.Rsk '410.Erk1/2.to.nuc
    '163.Rsk.to.nuc '36.Mekk1.act.Mkk4/7 '457.Mekk4.act.Mkk3/6
    '61.Mkk3/6.act.p38 '450.p38.to.nuc. '43.Mkk4/7.act.Jnk
    '49.Jnk.to.nuc '50B.Jnk.Gsk3.act.cJun '162.Erk.Rsk.act.cFos,
  PD(EGF {CM | Grb2 Pak1 PIP3 [Cdc42 - GTP] [EGFR - act]
         [Eps8 - act] [Gab1 - act] [Pdk1 - act] [PI3Ka - act]
         [Raf1 - act] [H-Ras - GTP] [Shc - act] [Sos - act]
         [nWasp - act]
         {p70s6k PKCb1 [Gsk3 - deact] [Mek1 - act] [Mekk1 - act]
          [Mekk4 - act] [Mkk3 - act] [Mkk4 - act] [Mlk2 - act]
           {NM | empty {[Akt1 - act] [cFos - act] [cJun - act]
                        [Erk1 - act] [Jnk1 - act]
                        [p38a - act] [Rsk1 - act] }}}}))

red findPath(q1x,prop1) .  *** PI3Ka removed
result SimplePath:  spath('1.EGFLike.act.EGFR '353M.EGFR.act.Gab1
    '418B.EGFR.act.Cdc42 '349.Cdc42.act.Mekk1 '471.Cdc42.act.Mekk4
    '470.Cdc42.act.Mlk '3.EGFR.act.Shc
    '411.Shc.act.Grb2 '324.Shc.Grb2.act.Sos  '618.Cdc42.act.Wasp
    '6.Sos.act.Ras '280.Ras.act.Raf '274.Raf1.act.Mek1/2
    '21.Mek1/2.act.Erk1/2 '437.Erk.act.Rsk '410.Erk1/2.to.nuc
    '36.Mekk1.act.Mkk4/7 '457.Mekk4.act.Mkk3/6 '61.Mkk3/6.act.p38
    '450.p38.to.nuc. '43.Mkk4/7.act.Jnk '49.Jnk.to.nuc
    '149.Rsk.iphos.Gsk3  ***
    '50B.Jnk.Gsk3.act.cJun '163.Rsk.to.nuc '162.Erk.Rsk.act.cFos,
  PD(EGF {CM | Grb2 Pak1 PIP2 [Cdc42 - GTP] [EGFR - act]
         [Gab1 - act]
         [Raf1 - act] [H-Ras- GTP] [Shc - act] [Sos - act]
         [nWasp - act]
         {Akt1 Eps8 Pdk1
          p70s6k PKCb1 [Gsk3 - deact] [Mek1 - act] [Mekk1 - act]
          [Mekk4 - act] [Mkk3 - act] [Mkk4 - act] [Mlk2 - act]
          {NM | empty {[cFos - act] [cJun - act] [Erk1 - act]
                  [Jnk1 - act] [p38a - act] [Rsk1 - act] }}}}))
```

Fig. 4. Model checking EGF-EGFR pathway.