The Pathway Logic Assistant

Carolyn Talcott
SRI International
clt@csl.sri.com

David L. Dill
Stanford University
dill@cs.stanford.edu

December 15, 2005

Abstract

This paper describes representations of biological processes based on Rewriting Logic and Petri net formalisms and mappings between these representations used in the Pathway Logic Assistant. The mappings are shown to preserve properties of interest. In addition a relevant subnet transformation is defined, that specializes a Petri net model to a specific query to reduce the number of transitions that must be considered when answering the query. The transformation is shown to preserve the query in the sense that no answers are lost.

Keywords: signal transduction, biological process, Pathway Logic, Rewriting Logic, Petri Net

1 Introduction

Pathway Logic [EKL+02a, EKL+02b, TEK+04] 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 [CDE+03a, CDE+03b] 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.

Using the reflective capabilities of Maude, several alternative representations are derived to support use of different tools for visualization and analysis. The Pathway Logic Assistant (PLA) manages these different representations and, using the IOP+IMaude framework [MT04], provides a user interface that supports visualization and interaction with the models, and access to tools such as the Pathalyzer for carrying out in silico experiments. In particular, PLA uses Petri nets which provide visual representations and algorithms for answering reachability queries interactively, displaying the results in a way that is natural for biologists.

Being able to use alternative representations with different expressive capabilities and tools for analysis is important both for scaling and being able to focus on different properties. In the

presence of multiple representations it is crucial to be able move between representations in semantically meaningful ways, preserving relevant properties. In this paper we describe a class of rewriting logic models of biological processes and a mapping of these models to Petri net models. We show that this mapping preserves computations and satisfaction of temporal formulae. Finally, we describe transformations that specialize Petri net models to specific queries by reducing the set of transitions that need to be considered, and show that transformations are safe in the sense that query results are not changed.

Plan. The remainder of this paper is organized as follows. This section concludes with a discussion of related work. To provide context and motivate the technical results, a brief overview of Pathway Logic and the ways that biologists can compute with and query Pathway Logic models is given in section 2. In section 3 the notion of occurrence-based rewrite theory is defined and mappings between such theories and Petri net models are defined and shown correct. The notion of subnet relevant for a particular query is introduced in section 4, and transformations for producing a safe approximation to the relevant subnet are defined. Section 5 concludes with a summary and discussion of future directions.

1.1 Related Work

Models of biological systems have been developed using a variety of computational formalisms and logics originally intended for modeling and analysis of computer systems. Much of the effort has been devoted to developing techniques to represent relevant biological concepts and to simulate their behavior Examples include Petri Nets [Hof94, RLM96, ZOS03]; variants of the Pi-calculus [RSS01, PRSS01]; membrane systems [PJRC05], Statecharts [EHC03]; and Live Sequence Charts [KHK+03]. Some modeling approaches based on computational or logical formalisms also use the associated logic to analyze and reason about the models. Examples include Pathway Logic [EKL+02b, TEK+04]; BioCham [FSCR04, CRFS04, CRCD+04, CCRF+05]; BioAmbients [RPS+03, NNPR03, PNN05]; BioSigNet-RR [STBK05]; and reasoning about continuous time Markov chains using continuous time stochastic logic [CVGO05].

The Pathway Logic Assistant extends the basic representation and execution capability with the ability to support multiple representations, to use different formal tools to simplify and analyze the models, and to visualize models and query results. Other efforts to integrate tools for manipulating models include the Systems Biology Workbench [HFS+02] the Biospice Dashboard [Bio04], IBM Discoverylink [IBM04], and geneticXchange, Inc [gen04].

2 Pathway Logic

As mentioned above, Pathway Logic models of biological processes are developed using the Maude system [CDE+03a, CDE+03b] a formal language and tool set based on rewriting logic. Rewriting logic [Mes92] 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*. Algebraic data types are specified by declaring sorts (names of the data types), subsort relations (one data type may

be a subset of another), and operations (naming functions and specifying argument and result types), and by giving equations that define the functions computed by the operations. Terms t that denote elements of the data types can be variables (denoting some unspecified element), constants, or the application of an operation to a tuple of argument terms, $f(t_1, \ldots, t_n)$. In its simplest form, a rewrite rule has the form $t \Rightarrow t'$ where t and t' represent a local part of the system state. This rule says that when the system has a subcomponent matching t, that subcomponent can evolve to t', possibly concurrently with changes described by rules matching other parts of the system state. The process of application of rewrite rules generates computations (also thought of as deductions). In the case of biological processes these correspond to paths. Using reflection, modules and computations are represented as terms of the Maude meta language. This makes it easy to compute with models and paths.

2.1 Pathway Logic Basics

Pathway Logic models are structured in four layers: (1) sorts and operations, (2) components, (3) rules, and (4) queries. The 'sorts and operations' layer defines the main sorts, subsort relations, and operations for representing cell states. The sorts of entities include `Chemical`, `Protein`, `DNA`, `Complex`, and `Enclosure` (cells and other compartments). These are all subsorts of the sort, `Soup`, that represents 'liquid' mixtures, as multisets. The sort `Dish` is introduced to encapsulate a soup as a state to be observed. Post-translational protein modification is represented by terms of the form `[P - mods]` where `P` is a protein and `mods` is a set of modifications. Modifications can be abstract, just specifying being activated, bound, or phosphorylated, or more specific, such as, phosphorylation at a particular site. For example, the term `[Cas - act]` represents the activation of the protein `Cas`. A cell state is represented by a term of the form `{CM | cm { cyto }}` where `cm` stands for a soup of entities in or at the cell membrane and `cyto` stands for a soup of entities in the cytoplasm.

The *components* layer specifies particular entities (proteins, chemicals, DNA) and introduces additional sorts for grouping proteins in families. For example `ErbB1L` is declared to be 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 signal transduction steps representing processes such as activation, phosphorylation, complex formation, or translocation. The *queries* layer specifies initial states and properties of interest.

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://www.csl.sri.com/~clt/PLweb` along with papers and tutorial material.

2.2 Example Pathway Logic Model: Activation of Rac1

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 [SH02] 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, and 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 and two stimuli in the supernatant, `EGF` and `FN`, represented by the following term.

```
rac1demo = PD(FN EGF
{CM | EGFR Ia5Ib1 Src PIP2 [Actin - poly][HRas - GDP][Rac1 - GDP]
        {Crk2 Erk2 Mek1  PI3K Shp2 bRaf C3g Dock Sos1
         Cas E3b1 Elmo Eps8 Fak Gab1 Grb2 Vav2 }} )
```

The cell membrane (shown on the line beginning `CM`) has an EGF receptor (`EGFR`) and an integrin (`Ia5Ib1`) that binds to `FN`. The term `[Rac1 - GDP]` represents the Rac1 protein in its 'off' state. The cell cytoplasm (shown on the last two lines) contains additional proteins that participate in the signaling process.

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

```
rl[1.EGFR.is.act]:
   ?ErbB1L:ErbB1L {CM | cm  EGFR        {cyto }} =>
   ?ErbB1L:ErbB1L {CM | cm [EGFR - act] {cyto }} .
   *** ErbB1Ls are AR EGF TGFa Btc Epr HB-EGF
```

The term `?ErbB1L:ErbB1L` is a variable ranging over the sort `ErbB1L`. The rule matches a part of the `rac1demo` dish contents by binding the variable `?ErbB1L:ErbB1L` to `EGF`, the variable `cm` to `Ia5Ib1 ... [Rac1 - GDP]` (every thing in the cell membrane except `EGFR`), and the variable `cyto` to the contents of the cytoplasm `{Crk2 ... Vav2}`. Applying the rule replaces `EGFR` by `[EGFR - act]` resulting in the dish

```
  PD(FN EGF
    {CM | [EGFR - act] Ia5Ib1 Src PIP2 [Actin - poly]
          [HRas - GDP][Rac1 - GDP]
        {Crk2 Erk2 Mek1  PI3K Shp2 bRaf C3g Dock Sos1
         Cas E3b1 Elmo Eps8 Fak Gab1 Grb2 Vav2}} )
```

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

```
rl[256.Rac1.is.act-3]:
 {CM | cm [Cas - act][Crk2 - act][Dock - act] Elmo [Rac1 - GDP]
      {cyto }} =>
 {CM | cm [Cas - act][Crk2 - act][Dock - act] Elmo [Rac1 - GTP]
      {cyto }} .
```

This rule describes activation resulting from assembly of `Elmo` with activated `Cas`, `Crk2`, and `Dock` at the cell membrane. The terms `cm` and `cyto` are variables standing for the remaining components in the membrane and cytoplasm, respectively. Executing the rule replaces

`[Rac1 - GDP]` by `[Rac1 - GTP]`, turning Rac1 on, and leaves the remaining components unchanged.

Maude provides several ways to compute with 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. To find a path satisfying some (temporal logic) property the Maude model-checker can be used. The properties of interest are expressed in Maude as patterns matching states with specific proteins, possibly with modifications, occurring in particular compartments (called goals), or requiring that particular proteins do not appear (avoids). For example, to find the path stimulated by `FN` alone, we define a property (called `racAct3`) that is satisfied when Rac1 is activated and the EGF stimulus is not used (`EGFR` is not activated), thus forcing the FN stimulus to be selected. The property `racAct3` is axiomatized by assertions stating which dishes satisfy the property (the relation `|=`) using patterns such as the following.

```
ceq PD (out:Soup
        {CM | cm:Soup  [Rac1 - GTP]  {cyto:Soup}}) |= racAct3 = true .
    if not(cm:Soup has [EGFR - act])
```

The model-checker is asked to check the assertion that there is no computation satisfying this property and a path can be extracted from a counterexample if one is found.

A Pathway Logic model, such as the Rac1 model, meeting certain simple conditions can be transformed into a Petri net model by specializing the rules to the model's initial state. The resulting Petri net model can then be analyzed using special purpose analysis tools.

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. Occurrences can be thought of as atomic propositions asserting that a protein (in a given state) or other component occurs in a given compartment. A system state is 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. Maude 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.
Figure 1 shows the result of the Petri net query corresponding to `racAct3` found by the LoLA [LoL04] Petri net analysis tool, as displayed by PLA using the Pathalyzer tool. For compactness, a computation/path is represented simply as a network of transitions, namely those used in the computation, with the occurrences in the initial state identified by giving them a darker color.

Lola uses "stubborn set reduction", which is a technique that exploits the ease of determining the independence of certain transitions in the Petri nets. For reachability queries on our nets, answering a reachability query that would have taken hours using a general purpose model-checking tool takes on the order of a second in LoLA—fast enough to permit interactive use.

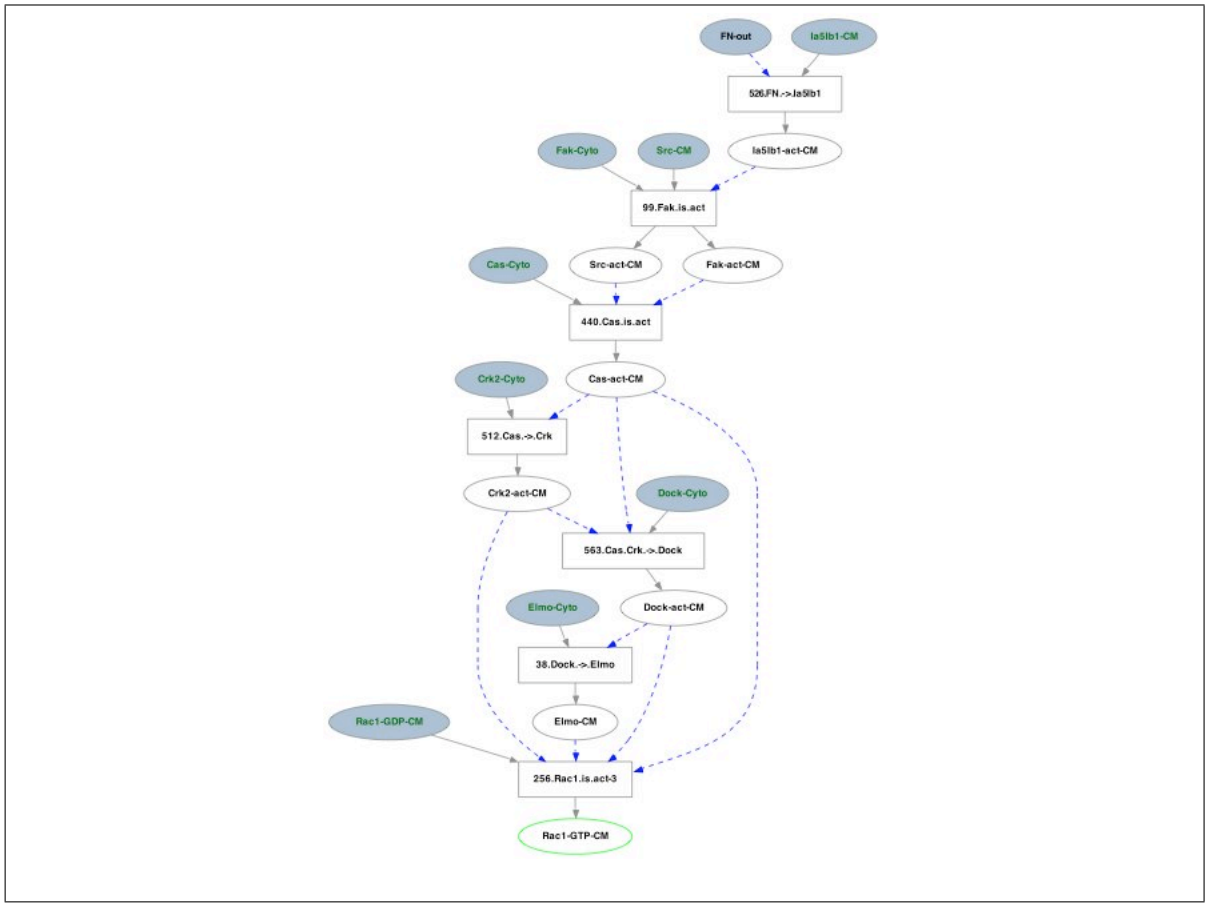

Figure 1: FN stimulation of Rac1 activation as a Petri net

2.3 The Pathway Logic Assistant

The Pathway Logic Assistant (PLA) manages the different model and computation representations and provides functions for moving from one representation to another, for answering user queries, displaying and browsing the results. The principle data structures are: PLMaude models, Petri net models, Petri subnets, PNMaude modules, computations (paths), and petrigraphs. Here we give an overview of the PLMaude and Petri net models and mappings between them. Details are given in the next section, where we define the notion of *occurrence-based rewrite theory* that abstracts the relevant features of PLMaude models, and specify the main properties required for mappings between and transformations of representations.

PLMaude models are Maude modules having the four layer structure described in the previous section, subject to the restriction that the rules preserve the property that component occurrences are not duplicated. As discussed above PLMaude states are represented as a mixture of cells and ligands where location of proteins and other chemicals is represented by the algebraic structure of a term. We define an alternative representation using multisets of occurrences, where an occurrence is a pair consisting of a protein, complex, or other chemical component and a location. The location uniquely identifies the position of the component

within the algebraic term (modulo multiset equality). For example, the dish

```
PD(EGF {CM | EGFR {Erk2}})
```

is represented by the occurrences

```
< EGF,out > < EGFR,cm > < Erk2,cyto >
```

Although soups and occurrences are formalized as multisets, initial states contain only sets (no duplication) and we have required that PLMaude rules preserve this property.

A *Petri net model* is a pair (T, I) consisting of a set of transitions T, and an initial state I (a set of occurrences). Each transition consists of a rule identifier, a pair of occurrence sets (the pre-occurrences and the post-occurrences). The mapping of a PLMaude model, with a specified initial dish D, to a Petri net model first determines an upper approximation to the set of components that might occur in each dish location by a collecting operation. This is done by starting with D, and repeating the collection cycle until nothing new is collected. In the collection cycle, for each rule that can be applied to the current dish, the current dish is merged with the result of applying the rule (by adding any new components to each compartment). For example, applying the rule `[1.EGFR.is.act]` the dish `rac1demo` in collecting mode would add `[EGFR -act]` to the membrane rather using it to replace `EGFR`. The set of transitions T is then the set of rule instances that apply to the collected dish, converted to occurrence pairs. For example the rule `[1.EGFR.is.act]` instantiated with `EGF` for `?ErbB1L:ErbB1L` is represented by the triple

```
(1.EGFR.is.act, < EGF, out > < EGFR, cm >,
                < EGF, out > < [EGFR - act], cm >)
```

The initial state I is the conversion of D to the occurrence representation. Because of the set preserving property of PLMaude rules mentioned above, the resulting Petri net model has the special form known as 1-safe, meaning that states are sets of occurrences, not more general multisets. This is important since efficient analysis algorithms have been developed for such Petri nets.

A *Petri subnet* is a tuple (T, I, G, A) consisting of a set of transitions, T, an initial marking, I a goal marking G, and an avoids set A. A Petri subnet specifies an analysis problem, namely finding a computation starting from the initial marking, and reaching a state with the goals marked, using the transitions in the given set, without ever marking an avoid. Petri subnets are generated by a 'relevant subnet' computation that simplifies the specified analysis problem. Although a Petri subnet is a Petri net, it is only equivalent to the original net for a goal set that is a subset of G or an avoid set that is a superset of A. For example, for a goal that is not in G there may be a path in the original net, but not in the subnet, since transitions needed for this goal may have been discarded as not being relevant.

Computations are data structures used to represent system executions. We model a computation as a sequence of steps, each step being a triple consisting of a source state, a rule instance or transition that applies to that state, and a target state, the state resulting from the rule application. The target state of the ith step of a computation must be equal to the source

state of the $i + 1$st step. A compact representation of a computation is the initial state together with the set of rule instances. We call the set of rule instances a *path*.

Petri graphs are used to represent Petri net models, subnets, and computations as data structures that have both a natural visual representation, and a clear connection to the computational structure. A petri graph has two kinds of node: occurrence and rule. Edges connect nodes representing occurrences of a rule premiss (lhs) to the rule node and the rule node to the nodes representing occurrences of the rule conclusion (rhs).

3 Relating PLMaude and Petri Nets

It is well known that Petri nets can be represented in rewriting logic [Ste00]. The various forms of PLMaude models have taken as the modeling ideas have matured have led us to identify a special class of rewrite theories, called *occurrence-based* rewrite theories, that, restricted to terms reachable from a given initial term, have a natural representation as Petri nets. The idea is to build on the equivalence of the dish and occurrences representations of states and to identify the features of PLMAude models needed to ensure that the translation to the Petri net formalism preserves computations and goals-avoids properties. Furthermore, the resulting Petri net models can be transformed back into rewriting logic, again preserving computations and goals-avoids properties. In this section we define the mappings between PLMaude and Petri net models and the subnet reduction, and sketch proofs of correctness. These mappings are implemented in Maude and used in PLA.

3.1 Some rewriting logic notation

We first introduce some notation for talking about rewrite theories. A rewrite theory, $\mathcal{R}$, is a triple $((\Sigma, E), R)$ where (Σ, E) is an equational theory (for example, in order sorted logic) with sorts and operations given by Σ and equations E, and R is a set of rules of the form $(t_0 \Rightarrow t_1 \text{ if } c)$ where t_0, t_1 are terms, the rule premise and conclusion respectively, and c is a boolean term, the rule condition. Viewing PLMaude as a rewrite theory, (Σ, E) is given by the first two layers (sorts and operations, components) and R is given by the rules layer.

A *context*, C, is a term with a single hole, denoted by $[\,]$, used to indicate the location of a rewrite application. $C[t]$ is the result of placing t in the hole of C.

A *substitution* σ is a finite mapping from variables to terms, preserving sort, and $\sigma(t)$ is the result of applying σ to the term t.

A *rule instance* is a triple $\rho = (r, C, \sigma)$ where r is a rule, C is context, and σ is a substitution. For a rule instance ρ as above we write $t \xrightarrow{\rho} t'$ if $t = C[\sigma(t_0)]$, $t' = C[\sigma(t_1)]$, and $\sigma(c)$ holds (rewrites to true). In this case we say that ρ is an application of r to t. We write $t \xrightarrow{r} t'$ if there is some $\rho = (r, C, \sigma)$ such that $t \xrightarrow{\rho} t'$. A computation over $\mathcal{R}$ is a sequence of rewrites of the form

$$\mathcal{R} \vdash s_0 \xrightarrow{\rho_1} s_1 \ldots \xrightarrow{\rho_k} s_k$$

with steps $s_{i-1} \xrightarrow{\rho_i} s_i$ for $1 \leq i \leq k$.

Note that rewriting is modulo E, that is the meaning of of the symbol '=' in the matching equations is defined by the equational theory E. The context makes explicit the location within a term where the rule applies. This is needed because when rewriting modulo equations the usual notion of path to a subterm of a syntax tree is not meaningful.

Rewriting Example. As an example, consider the following:

- S_0 = `EGF {CM | EGFR {Mek1 [Mekk3 - act]}}`
- S_1 = `EGF {CM | EGFR {[Mek1 - act] [Mekk3 - act]}}`
- r_{mek} = `[Mekk3 - act]  Mek1 => [Mekk3 - act] [Mek1 - act]`
- C = `EGF {CM | EGFR {[]}}`

Then $\rho = (r, C, \emptyset)$ is rule instance (with empty substitution, $\emptyset$) such that $S_0 \xrightarrow{\rho} S_1$. Note that S_0 can also be written `EGF {CM | {Mek1 [Mekk3 - act]} EGFR}`. Syntactically the subterm that matches the rule right hand side is at a different position in this case, but modulo associativity and commutativity the two ways of writing the term have the same meaning. The corresponding context `EGF {CM | {[]} EGFR}` is also equivalent to C, thus giving a representation of position that is independent to the representation of equivalence class.

3.2 Occurrence-based rewrite theories

There are five conditions to be met for a rewrite theory to be an occurrence-based rewrite theory, two conditions on the representation of state (**SC1** and **SC2**) and two conditions on rules (**RC1**, **RC2**) and one condition on the interaction of states and rules **SRC**).

In the following assume we are given a rewrite theory, $\mathcal{R}$, with distinguished sort $\mathbf{S}$ of elements representing system states to be analyzed.

SC1. The first condition is that $\mathbf{S}$ is generated from a base sort, by constructors such as the PLMaude enclosure constructors, in such a way that one only needs to know the 'location' of the base subterms to determine an element of $\mathbf{S}$. More precisely, we require that there be a base sort $\mathbf{B}$, a sort $\mathbf{L}$, of locations, and a sort $\mathbf{O}$ of occurrences, where elements of $\mathbf{O}$ have the form $<b, l>$ for base element b and location l, and two functions

$$s2o(\_) : \mathbf{S} \rightarrow \mathbf{P}_\omega[\mathbf{O}] \text{ and } o2s(\_) : \mathbf{P}_\omega[\mathbf{O}] \xrightarrow{\text{p}} S$$

such that $o2s(s2o(s)) = s$ where $\xrightarrow{\text{p}}$ denotes partial functions and $\mathbf{P}_\omega[\mathbf{O}]$ denotes finite sets from $\mathbf{O}$.

In PLMaude, the base sort is called `Thing`, which has subsorts `Protein` and `Chemical` amongst others. Each membrane enclosed compartment has two associated locations, the membrane and the interior. For example, a cell has locations `cm` and `cyto`, and things not inside a cell have location `out`.

SC2. We extend $s2o(\_)$ to contexts and terms with variables, by treating holes and variables as basic terms, and we require a function $cloc(C)$ that gives the location of the hole in a context.

We also relativize the map from states to occurrences so that $s2o(t, l)$ gives the occurrences for t in a context with hole location l. Thus

$$s2o(C[t]) = O \cup (s2o(t, l)) \text{ where } l = cloc(C), s2o(C) = O \cup <[\,], l>.$$

The **SC2** requirement is that if $s2o(s_0) = O \cup s2o(\sigma(t_0), l)$ then we can find C such that $cloc(C) = l$ and $s_0 = C[\sigma(t_0)]$.

Rewriting Example continued I. Using the notation of the example from section 3.1, we have

- $s2o(S_0)$ = `< EGF,out > < EGFR,cm > < Mek1,cyto > < [Mekk3 - act],cyto >`
- $cloc(C)$ = `cyto`
- $s2o(S_2, cloc(C))$ = `< Mek1,cyto > < [Mekk3 - act],cyto >`

where S_2 is the left-hand side of r_{mek}. From the discussion in the previous sections, it is easy to see that PLMaude modules satisfy conditions SC1 and SC2.

SRC. We require that there is an associative and commutative operation on states $merge : \mathbf{S} \to \mathbf{S}$ such that rewriting is preserved by merging. Specifically, if $s_0 \xrightarrow{(r,C,\sigma)} s_0'$, $s_1 = merge(s_0, s')$ and $s_1' = merge(s_0, s')$ then $s_1 \xrightarrow{(r,C',\sigma')} s_1'$ for some C', σ', such that $\sigma/B = \sigma'/B$ and $cloc(C) = cloc(C')$ where σ/B is the restriction of σ to basic variables. Using associativity and commutativity we extend the $merge$ operation to sets: $merge(s, S)$ is the result of merging elements of S into s in some order.

Rewriting Example continued II. Continuing the example we have

- $merge(S_0, S_1)$ = `EGF {CM | EGFR{Mek1 [Mek1 - act] [Mekk3 - act]}}`

RC1. We require that the variables appearing in rule terms either have basic sorts, or 'mixture' sorts (for example finite sets). This allows us to convert a rule application instance (r, C, σ) into a pair of occurrence sets that represent the actual change described by the rule. The mixture variables stand for the remaining basic terms and substructure at each location of interest that are not changed by the rule. Furthermore, we assume that the variables occurring in the rule condition have basic sorts.

Definition: Collection. Now we define a (partial) function that iteratively merges the reachable states into one state $\hat{s}$ in which each location contains all basic elements that could appear at that location in a reachable state. Given $s \in \mathbf{S}$ define $\hat{s}$ by

$$\hat{s} = s_k \text{ if } s_k = s_{k+1} \quad \text{where} \quad s_0 = s \quad \text{and} \quad s_{i+1} = merge(s_i, \{s' \mid (\exists \rho)(s_i \xrightarrow{\rho} s'\})$$

RC2. The final condition for $\mathcal{R}$ to be occurrence-based (relative to a choice of initial states) is that for any initial state s collection terminates, i.e. there is some k such that $s_k = s_{k+1}$.

3.3 Mapping occurrence-based rewrite theories to Petri nets

To define the mapping we need some Petri net notation. A transition τ over an occurrence set $\mathbf{O}$ is a pair $(O_i, O_o) \in \mathbf{P}_\omega[\mathbf{O}] \times \mathbf{P}_\omega[\mathbf{O}]$ (for simplicity we omit the transition labels). We define the pre- and post-occurrences of a transition as follows:

$$pre((O_i, O_o)) = O_i \qquad post((O_i, O_o)) = O_o.$$

The input and output occurrences are the pre- and post-occurrences with the shared occurrences removed.

$$in((O_i, O_o)) = O_i - O_o \qquad out((O_i, O_o)) = O_o - O_i.$$

Note that

$$in((O_i, O_o)) \cap out((O_i, O_o)) = \emptyset$$
$$(pre((O_i, O_o)) - in((O_i, O_o)))$$
$$= (post((O_i, O_o)) - out((O_i, O_o))) = (pre((O_i, O_o)) \cap post((O_i, O_o)))$$

A Petri net model over occurrences $\mathbf{O}$ is a pair (T, I) where T is a set of transitions and $I \in \mathbf{P}_\omega[\mathbf{O}]$ is the initial state/marking. A computation over T is a sequence

$$T \vdash O_0 \xrightarrow{\tau_1} O_1 \ldots \xrightarrow{\tau_k} O_k$$

such that $pre(\tau_{i+1}) \subseteq O_i$ and $O_{i+1} = (O_i - in(\tau_{i+1})) \cup out(\tau_{i+1})$.

Definition: Rule2Transition. We extend the occurrence mapping to map rule instances to transitions.

$$s2o(((t_0, t_1, c), C, \sigma)) = (s2o((t_0, C, \sigma)), s2o((t_1, C, \sigma)))$$

where

$$s2o((t_0, C, \sigma)) = s2o((\sigma/B)(t_0), cloc(C))^\dagger.$$

Where the † means to drop variable occurrences $<V, l>$ for mixture variables V. Note that if (r, C, σ) and (r, C', σ') are as in **RC2** then $s2o((r, C, \sigma)) = s2o((r, C', \sigma'))$. (See below for examples of $s2o(\_)$ applied to rules.)

Definition: OccB2Petri. Assume $\mathcal{R}$ is occurrence-based, with state sort $\mathbf{S}$, and s is an initial state. The Petri net model, $\mathcal{P}(\mathcal{R}, s)$, associated with $\mathcal{R}$ and s, has occurrences $s2o(\hat{s})$ (the result of collection), initial state $s2o(s)$, and a transition for each rule instance that applies to $\hat{s}$.

$$\mathcal{P}(\mathcal{R}, s) = (T_s, s2o(s)) \quad \text{where} \quad T_s = \{s2o((r, C, \sigma)) \mid (\exists s')(\hat{s} \xrightarrow{(r,C,\sigma)} s')\}$$

Theorem: Occ2Petri. For $\mathcal{R}$ an occurrence-based rewrite theory and s an initial state, the mapping to Petri nets preserves computations. Specifically, if $(T_s, s2o(s)) = \mathcal{P}(\mathcal{R}, s)$, then

$$\mathcal{R} \vdash s = s_0 \xrightarrow{\rho_1} s_1 \ldots \xrightarrow{\rho_k} s_k \;\Leftrightarrow\; T_s \vdash s2o(s_0) \xrightarrow{s2o(\rho_1)} s2o(s_1) \ldots \xrightarrow{s2o(\rho_k)} s2o(s_k).$$

Proof Sketch. By induction on the computation length k. If $s_i \xrightarrow{\rho_{i+1}} s_{i+1}$ then $s2o(\rho_{i+1}) \in T_s$ by RC2. Let $\rho_{i+1} = (r, C, \sigma)$ with $r = (t_0, t_1, c)$ and $l = cloc(C)$. Then $s_i = C[\sigma(t_0)]$,

$s_{i+1} = C[\sigma(t_1)]$, and for some occurrence set O $s2o(s_i) = s2o(\sigma(t_0), l) \cup O$ and $s2o(s_{i+1}) = s2o(\sigma(t_1), l) \cup O$. Thus $s2o(s_i) \xrightarrow{s2o(\rho_{i+1})} s2o(s_{i+1})$. Conversely, let $O_i \xrightarrow{\tau_{i+1}} O_{i+1}$, and by induction $O_i = s2o(s_i)$ for some reachable s_i. Also $\tau = (O_0, O_1) = s2o((r, C, \sigma))$ where (r, C, σ) applies to $\hat{s}$. We can find O' such that $O_i = O' \cup O_0$ and $O_{i+1} = O' \cup O_1$. By SC3 we can find C', σ' such that $s_i = C'[\sigma'(t_0)]$, and $s_i \xrightarrow{(r,C',\sigma')} s_{i+1}$ where $s2o(s_{i+1}) = O_{i+1}$.

Counterexample. To see that requirement (**RC2**) that merging preserves rewrites is needed, consider the following rule variants in the Pathway Logic language:

```
[r1]: {CM | Ras {cyto Rac}} => {CM | Ras [Rac - act]{cyto}}
[r2]: {CM | cm Ras {cyto Rac}} => {CM | cm Ras [Rac - act]{cyto}}
```

where `cyto` and `cm` are variables standing for any other components located in the cytoplasm or cell membrane respectively. Consider the state `{CM | Ras Grb2 {Src Rac}}` which can be obtained from `{CM | Ras {Src Rac}}` by a merge. The rule `r2` applies but `r1` does not, although `r1` applies to the 'before merge' state. Both rules transform to the same Petri net transition:

```
< Ras,CM > < Rac,Cyto > => < Ras,CM > < [Rac - act],CM >
```

which indeed applies to the corresponding occurrence state

```
< Ras,cm > < Grb2,cm ><  Rac,cyto > < Src,cyto >
```

Definition: Petri2RWL. The conversion of an occurrence Petri net to a rewrite theory is simple. If $(T_s, s2o(s)) = \mathcal{P}(\mathcal{R}, s)$, then $PS(\mathcal{R}, s)$ is the rewrite theory with the equational part of $\mathcal{R}$ extended with the definition of occurrences, and rules

$$\{O_i \Rightarrow O_o \mid (O_i, O_o) \in T_s\}$$

Theorem:Petri2RWL. The mapping PS preserves computations.

$$PS(\mathcal{R}, s) \vdash s2o(s) \xrightarrow{\tau_1} \ldots \xrightarrow{\tau_k} O_k \Leftrightarrow T_s \vdash s2o(s) \xrightarrow{\tau_1} \ldots \xrightarrow{\tau_k} O_k$$

4 Relating and transforming queries

4.1 Preservation of properties

The temporal logic used by the Maude model checker, LTL, is based on atomic propositions that can be defined by boolean functions in Maude. In the case of an occurrence-based rewrite theory, we restrict attention to to propositions that are positive (goals) and negative (avoids) occurrence tests – basic component b occurs (or does not occur) at location l. These propositions translate to simple membership tests $<b, l> \in s2o(s)$ in the corresponding Petri net model. For example, the property `racAct3` presented in section 2 contains one positive occurrence test (for the presence of `< [Rac1 - GTP], cm >`) and and one negative occurrence test (for the absence of `< [EGFR - act], cm >`).

Let LTLO be the Maude LTL language with propositional part restricted to occurrence propositions. Let ψ be an LTLO formula expressed in the PLMaude language and let $s2o(\psi)$ be the same property expressed in terms of occurrence membership, lifting $s2o(\_)$ homomorphically (on syntax) to LTLO formulas.

Theorem: LTLO. Given an occurrence-based rewrite theory $\mathcal{R}$ and initial state s, let π be a computation of $\mathcal{R}, s$, π' be the corresponding computation of $\mathcal{P}(\mathcal{R}, s)$, and π'' be the corresponding computation of $PS(\mathcal{R}, s)$. Then for any LTLO formula ψ

$$\pi \models \psi \;\Leftrightarrow\; \pi' \models s2o(\psi) \;\Leftrightarrow\; \pi'' \models s2o(\psi)$$

and thus

$$(\mathcal{R}, s) \models \psi \;\Leftrightarrow\; (T_s, s2o(s)) \models s2o(\psi) \;\Leftrightarrow\; (PS(\mathcal{R}, s), s) \models s2o(\psi)$$

This is a consequence of the isomorphism of computations and the preservation of satisfaction of occurrence properties by the occurrence translations.

Note that the **LTLO** theorem implies that counterexamples are also preserved. This is important, since queries asking for computations having certain properties are answered by asking a model-checker to find a counterexample to the assertion that no such computation exists.

4.2 Relevant Subnets for Goals-Avoids Queries

As indicated in section 2, we are especially interested in answering queries of the form "given initial state I, find a path that satisfies (G, A)" where (G, A) is a basic goals-avoids property with goals G and avoids A. We interpret this as meaning find a path (that is, a computation) starting with the initial state I, that reaches a state satisfying goals G, and such that no state in the path contains any occurrence of A. Without loss, we further require the path to be minimal. That is, if any transition is removed, the remaining transitions do not generate a path satisfying the goals.

The task of finding a minimal path satisfying a goals-avoids property can be simplified by considering only the set of transitions of a Petri net model that could possibly appear in any minimal path satisfying that property. We call these the *truly relevant* transitions. Finding just the truly relevant transitions means finding exactly the minimal paths satisfying a given property, the problem we are trying to simplify. Thus we will look for a safe approximation, that is a superset of truly relevant transitions set. Clearly, transitions that mention an occurrence to be avoided can be eliminated, as can transitions that do not contribute to reaching some goal, or transitions whose pre-set will not be a subset of a reachable state. In the following we define three transformations that formalize these intuitions. The first transformation removes transitions that mention an occurrence to be avoided. The second transformation is a backwards collection of transitions that contribute to reaching a goal, either because the post-occurrences contain a goal, or recursively contain a pre-occurrence of some contributing transition. The third transformation is a forward collection of transitions applicable to reachable states. We show that any minimal path meeting a goals-avoids property using transitions in T, in fact uses

only transitions collected, after removing avoids, by the backward followed by the forward transformations.

Definition: Minimal Paths. Let I (initial state), G (goals), A (avoids) be occurrence sets such that $(I \cup G) \cap A = \emptyset$. The set $P(T, I, G, A)$ is the set of Petri net computations, π, that start from the initial state I, and reach a state containing all occurrences in G, using transitions in T without ever marking A.

$$\pi = O_0 \xrightarrow{\tau_1} \ldots \xrightarrow{\tau_k} O_k \in P(T, I, G, A) \;\Leftrightarrow\; O_0 = I \;\wedge\; G \subseteq O_k \;\wedge \bigwedge_{0 \leq i \leq k} O_i \cap A = \emptyset$$

π is minimal if there is no computation π' that uses a proper subset of the transitions used in π, and we let $mP(T, I, G, A)$ be the set of transitions of $P(T, I, G, A)$ that are minimal.

Lemma: Path Monotonicity. The set of minimal paths monotonically increases with increasing initial state and decreasing goals and avoids. Specifically, if $T \subseteq T'$, $I \subseteq I'$, $G' \subseteq G$, $A' \subseteq A$, then

$$P(T, I, G, A) \subseteq P(T', I', G', A')$$

and

$$mP(T, I, G, A) \subseteq mP(T', I', G', A')$$

Definition: Removing Avoids. Assume given T and A as above. The result of removing rules that mention an element of A is defined by

$$T/A = \{\tau \in T \;\big|\; (pre(\tau) \cup post(\tau)) \cap A = \emptyset\}$$

Lemma: Removing Avoids is Safe. If $\pi \in mP(T, I, G, A)$, then $\pi \in mP(T/A, I, G, A)$. **Proof.** Since by definition no transition in $T - T/A$ could be used in π.

Definition: Backward collection. Assume given T, G as above. The backward collection T^b_G of T relative to G is defined by

$$T^b_G = \bigcup_{j \in \mathbf{Nat}} T^b_j \quad \text{where}$$

$$G_0 = G \qquad G_{j+1} = G_j \cup \bigcup_{\tau \in T^b_j} pre(\tau))$$

$$T^b_j = \{\tau \in T \;\big|\; out(\tau) \cap G_j \neq \emptyset\}$$

Note that for some n, $G_j = G_{j+1}$ for $j > n$ since T is finite and thus only finitely many increments can be made.

Lemma: Backward Monotonicity. Backwards collection is monotonic in transitions and goals. That is, if $T \subseteq T'$ and $G \subseteq G'$, then $T^b_G \subseteq (T')^b_{G'}$.

The lemma **Backwards 1** captures the essence of the reason that a transition that appears in some minimal path for a set of goals is one produced by backwards collection.

Lemma: Backwards 1. If $O \xrightarrow{\tau_1} O_1 \xrightarrow{\tau_2} O_2$ and $pre(\tau_2) \cap out(\tau_1) = \emptyset$ then we can find O_2' such that $O \xrightarrow{\tau_2} O_2'$. Furthermore, for any occurrence set G^*, if $out(\tau_1) \cap G^* = \emptyset$, then $G^* \cap O_2 \subseteq G^* \cap O_2'$.

Proof. With the assumptions of the lemma, $pre(\tau_2) \subseteq O$, letting $O_2' = (O - in(\tau_2)) \cup out(\tau_2)$ we have, by definition of transition, the desired transition. Also by definition of transition, $O_2 = ((O - in(\tau_1) \cup out(\tau_1)) - in(\tau_2)) \cup out(\tau_2)$. Assuming $out(\tau_1) \cap G^* = \emptyset$ we have $G^* \cap O_2 = G^* \cap ((O - in(\tau_1) - in(\tau_2)) \cup out(\tau_2)) \subseteq G^* \cap O_2'$.

The lemma **Backwards 2** identifies conditions under which a sequence of transitions can be restarted at a new state. For backwards collection, the state of interest is one resulting from deleting an irrelevant transition, such as τ_1 in **Backwards 1**.

Lemma: Backwards 2. If $O \cap G \subseteq O' \cap G$, $pre(\tau) \subseteq G$ and $O \xrightarrow{\tau} O_1$, then we can find O_1' such that $O_1 \cap G \subseteq O_1' \cap G$ and $O' \xrightarrow{\tau} O_1'$.

Proof. By the assumptions, $pre(\tau) \subseteq O'$, so letting $O_1' = (O' - in(\tau)) \cup out(\tau)$ we have the desired transition. Since $O_1 = (O - in(\tau)) \cup out(\tau)$, if $g \in O_1$ either $g \in out(\tau)$ or $g \in O - in(\tau) \subseteq O' - in(\tau)$. Thus $g \in O_1'$.

Theorem: Backward safety. If $\pi \in mP(T, I, G, A)$, then $\pi \in mP(T_G^b, I, G, A)$.

Proof Sketch. Let $\pi = I \xrightarrow{\tau_1} O_1 \ldots \xrightarrow{\tau_k} O_k \in mP(T, I, G, A)$. We show that $\tau_j \in T_G^b$ for $1 \leq j \leq k$. Suppose not. Let G^* be the union of the G_js in the definition of T_G^b, and let j be the largest number such that $\tau_j \notin T_G^b$. Thus $out(\tau_j) \cap G^* = \emptyset$. We construct $\pi' \in P(T, I, G, A)$ using fewer transitions, contradicting minimality of π. If $j = k$ then $G \subseteq O_{k-1}$ and π' is the first $k-1$ transitions of π. If $j < k$ then let $O_j' = O_{j-1}$, and $O_{i+1}' = (O_i' - in(\tau_{i+1})) \cup out(\tau_{i+1})$ for $j \leq i < k$. By maximality of j, $\tau_{i+1} \in T_G^b$ and thus $pre(\tau_{i+1}) \subseteq G^*$ for $j \leq i < k$. We claim that $O_{i+1} \cap G^* \subseteq O_{i+1}' \cap G^*$ and $O_i' \xrightarrow{\tau_{i+1}} O_{i+1}'$ for $j \leq i < k$. For $i = j$ this follows by backwards lemma 1 and for $i > j$ it follows by backwards lemma 2. Thus taking $\pi' = I \xrightarrow{\tau_1} O_1 \ldots \xrightarrow{\tau_{j-1}} O_j' \xrightarrow{\tau_{j+1}} \ldots \xrightarrow{\tau_k} O_k'$ we are done.

Definition: Forward collection. The forward collection T_I^f of T relative to I is defined by

$$T_I^f = \bigcup_{j \in \mathbf{Nat}} T_j^f \qquad I^f = \bigcup_{j \in \mathbf{Nat}} I_j \quad \text{where}$$

$$I_0 = I \qquad I_{j+1} = \bigcup_{\tau \in T_j^f} post(\tau)$$

$$T_j^f = \{\tau \in T \mid pre(\tau) \subseteq I_j\}$$

Again, we have that for some n, $I_j = I_n$ for $j \geq n$.

Lemma: Forward Monotonicity. If $T \subseteq T'$ and $I \subseteq I'$, then $T_I^f \subseteq (T')_{I'}^f$

Theorem: Forward safety. If $\pi \in mP(T, I, G, A)$, then $\pi \in mP(T_I^f, I, G, A)$.

Proof. This is because for each transition τ_{j+1} in π, using the notation of the definition, $pre(\tau_{j+1}) \subseteq I_j$, and thus $\tau_{j+1} \in T_j^f$ for $0 \leq j < k$.

Corollary: Relevant Subnet. If $\pi \in mP(T, I, G, A)$ is non-empty, then

$$\pi \in mP(((T/A)_G^b)_{I,A}^f, I, G, A)$$

Thus search for such paths can be carried out in the relevant subnet $(T^b_{G,A})^f_{I,A}$. Note that if $G \not\subseteq I^f$ then $P(T, I, G, A)$ is empty.

5 Summary and Future Work

As context we presented an overview of the capabilities that the Pathway Logic Assistant provides for computing with Pathway Logic models and for using different representations of networks and computations for analysis and interactive visualization. The main contributions of the paper are: a definition of mappings between rewriting logic and petri net representations of biological processes (and similar concurrent processes); proof that these mappings preserve properties of interest; and definition of a relevant subnet transformation that reduces the number of transitions that must be considered in search for a path satisfying a goals-avoids property.

As models grow in size, we expect to need to explore alternative path finding algorithms. Possibilities include employing more highly tuned model checkers, discovering new simplification and abstraction transformations, and developing constraint solving approaches. Another big challenge is refining PLMaude models to incorporate semi-quantitative information about expression levels and relative preference for competing reactions, and to be able compute with and visualize the refined models in ways that are meaningful to working biologists.

Acknowledgments. The authors would like to thank the Pathway Logic team and the anonymous reviewers for helpful criticisms. The work was partially funded by NIH/NIGMS grant GM68146.

References

[Bio04] BioSpice, 2004. `https://community.biospice.org/`.

[CCRF+05] Laurence Calzone, Nathalie Chabrier-Rivier, Francois Fages, Lucie Gentils, and Sylvain Soliman. Machine learning bio-molecular interactions from temporal logic properties. In G. Plotkin, editor, *Proceedings of the Third International Conference on Computational Methods in System Biology (CMSB 2005)*, 2005.

[CDE+03a] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, and C. Talcott. Maude 2.0 Manual, 2003. `http://maude.cs.uiuc.edu`.

[CDE+03b] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott. The Maude 2.0 system. In Robert Nieuwenhuis, editor, *Rewriting Techniques and Applications (RTA 2003)*, volume 2706 of *Lecture Notes in Computer Science*, pages 76–87. Springer-Verlag, 2003.

[CRCD+04] N. Chabrier-Rivier, M. Chiaverini, V. Danos, F. Fages, and V. Schächter. Modeling and querying biomolecular interaction networks. *Theoretical Computer Science*, 351(1):24–44, 2004.

[CRFS04] N. Chabrier-Rivier, F. Fages, and S. Soliman. The biochemical abstract machine BIOCHAM. In V. Danos and V. Schächter, editors, *Proceedings of the Second International Conference on Computational Methods in System Biology (CMSB-04)*, Lecture Notes in Bioinformatics. Springer, 2004.

[CVGO05] Muffy Calder, Vladislav Vyshemirsky, David Gilbert, and Richard Orton. Analysis of signalling pathways using the PRISM model checker. In G. Plotkin, editor, *Proceedings of the Third International Conference on Computational Methods in System Biology (CMSB 2005)*, 2005.

[EHC03] S. Efroni, D. Harel, and I.R. Cohen. Towards rigorous comprehension of biological complexity: Modeling, execution and visualization of thymic t-cell maturation. *Genome Research*, 2003. Special issue on Systems Biology, in press.

[EKL+02a] 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.

[EKL+02b] 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`.

[FSCR04] F. Fages, S. Soliman, and N. Chabrier-Rivier. Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of Biological Physics and Chemistry*, 4(2):64–73, 2004.

[gen04] geneticXchange, 2004. `http://midas-10.cs.ndsu.nodak.edu/bio/ppts/chung.pdf`.

[HFS+02] M. Hucka, A. Finney, H. Sauro, H. Bolouri, J. Doyle, and H. Kitano. The ERATO systems biology workbench: Enabling interaction and exchange between software tools for computational biology. In *Proceedings of the Pacific Symposium on Biocomputing*, 2002.

[Hof94] R Hofestädt. A Petri net application to model metabolic processes. *Syst. Anal. Mod. Simul.*, 16:113–122, 1994.

[IBM04] IBM Discoverylink, 2004. `http://publib-b.boulder.ibm.com/Redbooks.nsf/0/3c7a635147cf20d785256a540064e287?OpenDocument&Highlight=0,DiscoveryLink`.

[KHK+03] N. Kam, D. Harel, H. Kugler, R. Marelly, A. Pnueli, J. Hubbard, and M. Stern. 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*, pages 4–20. Springer, 2003.

[LoL04] LoLA: Low Level Petri net Analyzer, 2004. `http://www.informatik.hu-berlin.de/~kschmidt/lola.html`.

[Mes92] J. Meseguer. Conditional Rewriting Logic as a unified model of concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.

[MT04] I. A. Mason and C. L. Talcott. IOP: The InterOperability Platform & IMaude: An interactive extension of Maude. In *Fifth International Workshop on Rewriting Logic and Its Applications (WRLA'2004)*, Electronic Notes in Theoretical Computer Science. Elsevier, 2004.

[NNPR03] F. Nielson, H. R. Nielson, C. Priami, and D. Rosa. Control flow analysis for bioambients. In *BioConcur*, 2003.

[PJRC05] M.J. Prez-Jimnez and F.J. Romero-Campero. Modelling EGFR signalling cascade using continuous membrane systems. In G. Plotkin, editor, *Proceedings of the Third International Conference on Computational Methods in System Biology (CMSB 2005)*, 2005.

[PNN05] Henrik Pilegaard, Flemming Nielson, and Hanne Riis Nielson. Static analysis of a model of the ldl degradation pathway. In G. Plotkin, editor, *Proceedings of the Third International Conference on Computational Methods in System Biology (CMSB 2005)*, 2005.

[PRSS01] C. Priami, A. Regev, E. Shapiro, and W. Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 2001. in press.

[RLM96] V. N. Reddy, M. N. Liebmann, and M. L. Mavrovouniotis. Qualitative analysis of biochemical reaction systems. *Comput. Biol. Med.*, 26:9–24, 1996.

[RPS+03] A. Regev, E. Panina, W. Silverman, L. Cardelli, and E. Shaprio. Bioambients: An abstraction for biological compartments, 2003. to appear TCS.

[RSS01] A. Regev, W. Silverman, and E. Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In R. B. Altman, A. K. Dunker, L. Hunter, and T. E. Klein, editors, *Pacific Symposium on Biocomputing*, volume 6, pages 459–470. World Scientific Press, 2001.

[SH02] A. Schmidt and A. Hall. Guanine nucleotide exchange factors for rho gtpases: turning on the switch. *Genes Dev.*, 16:1587–15609, 2002.

[STBK05] Carron Shankland, Nam Tran, Chitta Baral, and Walter Kolch. Reasoning about the ERK signal transduction pathway using BioSigNet-RR. In G. Plotkin, editor,

Proceedings of the Third International Conference on Computational Methods in System Biology (CMSB 2005), 2005.

[Ste00] M.-O. Stehr. A rewriting semantics for algebraic nets. In C. Girault and R. Valk, editors, *Petri Nets for System Engineering – A Guide to Modelling, Verification, and Applications*. Springer-Verlag, 2000.

[TEK+04] 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.

[ZOS03] I. Zevedei-Oancea and S. Schuster. Topological analysis of metabolic networks based on Petri net theory. *In Silico Biology*, 3(0029), 2003.