Understanding Signalling Networks as Collections of Signal Transduction Pathways

Robin Donaldson
Dept. of Computing Science
University of Glasgow
Glasgow, UK
radonald@dcs.gla.ac.uk

Carolyn Talcott
Computing Science Lab
SRI International
Menlo Park, CA
carolyn.talcott@sri.com

Merrill Knapp
Biosciences Division
SRI International
Menlo Park, CA
merrill.knapp@sri.com

Muffy Calder
Dept. of Computing Science
University of Glasgow
Glasgow, UK
muffy@dcs.gla.ac.uk

ABSTRACT

A signalling network is a network of reactions that govern how a cell responds to its environment. A pathway is a dynamic flow of "signal" through the network (signal transduction), for example from a receptor to a transcription factor that enables expression of a gene. In this paper we introduce a method to compute all pathways in a signalling network that satisfy a simple property constraining initial, final and intermediate states. This method, concerned with signal transduction, is compared to the steady state view underlying Petri net place/transition invariants and flux balance analysis. We apply the method to the signalling network model being developed in the Pathway Logic project and identify knockout/inhibition targets and common (pathway) events. This approach also allows us to better understand and formalise the interaction between pathways in a network, for example to identifying pathway inhibition targets that limit the effect on unrelated pathways.

Categories and Subject Descriptors

I.6 [**Simulation and Modeling**]: Model Validation and Analysis; J [**Computer Applications**]: Life and Medical Sciences

General Terms

Algorithms

Keywords

Signalling Networks, Pathways, Reaction Minimal Paths, T Invariants

1. INTRODUCTION

Pathway Logic (PL) [20, 19] is a formal framework for modelling biological processes based on Rewriting Logic [12, 1]. The objective of the Pathway Logic project is to help biologists to

- organise and represent experimental data in a computable form (linked to publications and/or lab notebooks, controlled vocabularies and relevant databases)
- build and compute with models, based on data and conjectures, that describe behaviours of interest
- understand data and models in a larger context.

To this end, the PL framework provides a formal computable representation of experimental data and possible events (using rewrite rules); executable models of behaviour; and tools to visualise and explore knowledge bases and model behaviours.

Computable representations with appropriate visualisation enable biologists to work with much larger models, to combine models and data in new ways, and to discover possible interactions among multiple processes. The point of computable models is that they can be *executed*. That is, they provide a direct (abstract) representation of a system's activities and can be used to sample possible behaviours (simulation), as well as analysing properties of the system in isolation or in larger contexts.

Here we focus on the use of PL for modelling cellular signalling processes. Signalling processes transmit information that controls cellular decisions: for example, what proteins to express and differentiation or phenotype choices. A cell senses its environment by way of receptors that bind molecules present in the immediate neighbourhood. This event causes a change in the receptor, which in turn may enable further events that change the state of proteins (protein modification, complex formation, or location) and other biochemical entities. These states encode information that is propagated via signalling events. Possible events are described by rules that specify the conditions for an event to occur (the starting state of participating entities) and the outcome of the event (the state after the event has happened). A rule knowledge base (RKB) is a set of rules.

The rules form a network where rules and entity states are nodes, and edges connect input entity states of a rule to that rule and the rule to its output entity states. An executable model is obtained by specifying an initial state (entities and their states) and a RKB. A pathway is a (multi)set of rules that can fire in some order (an execution). The model's behaviour is the set of possible pathways.

We are interested in signal transduction pathways in which the flow of the "signal" is indicated by transitory local changes of state. More specifically, we are not interested in the signalling events of predefined pathways (i.e. predefined by biologists) but rather we are interested in all the possible signalling events in the *set* of pathways in a network of rules. Given a model, some of the questions of interest concerning pathways are

- What are all the pathways leading from a state satisfying property P_A to a state satisfying property P_B? For example P_A might be the presence of one or more signals (ligands) and P_B might be activation of a transcription factor.
- Is there an entity that is required in all these pathways? This would be a knockout. Are there pairs of entities such that every pathway uses at least one of the pair (double knockouts)?
- Are there events that are common to all the pathways?

Pathways of interest are characterised by specifying an initial state along with goals (desired final states of some entities) and avoids (entity states that should not occur during execution). A subset of rules within the global network called the relevant subnet is defined in [20] and shown to contain all the pathways leading from the initial state and satisfying the goals and avoids. The Pathway Logic Assistant (PLA) provides an interactive graphical interface to browse and query an RKB and associated models. Using PLA one can

- explore a network of rules to discover what is up/down stream of some entity X, or to see if X and Y can be connected
- specify simple properties – to reach or avoid particular entity states
- find the *relevant subnet* for a specified property
- find *a pathway* within the relevant subnet
- compare pathways – what is shared, what is different.

PLA uses model checking to find pathways (counter examples to the assertion that no pathway exists). What it currently can not do is automatically find all pathways. That is a key aspect of the work presented here.

We adopt Petri nets [15, 5] as our underlying computational model, to formalise rule networks, models, executions, and pathways. Petri nets provide graphical representations of biochemical systems with places (corresponding to biochemical entities), transitions (corresponding to reactions), and tokens (corresponding to concentrations of molecules). Analysis of (independent) pathways within a network is typically carried out by derivation of minimal T invariants [9], which are vectors of transitions that achieve a steady state. However, in some cases analysis by T invariants is insufficient. The main contribution of this paper is to identify those situations and to develop a complete algorithm that computes pathways, that produce (goal) outputs, avoiding designated (avoid) outputs, and are minimal with respect to reactions. Our algorithm, concerned with signal transduction, is compared to the steady state view underlying T invariant analysis. We apply the algorithm to a Pathway Logic knowledge base of cellular signalling response to a variety of stimuli, to find all pathways activating two choices of goals, and to then compute knockout sets and essential transition sets.

The remainder of this paper is organised as follows. Section 2 introduces the Petri net notation used. Section 3 reviews how pathways are currently computed using T invariants (the steady state view) and shows that in some cases, the steady state view is not sufficient. Section 4 introduces an algorithm to compute all pathways in a Petri net that satisfy given constraints on initial, final and intermediate states. Section 5 is concerned with the application of the new method to Pathway Logic models. Section 6 compares our work to the current work in the literature, both model checking and steady state techniques. We suggest future lines of research and conclude in Section 7.

2. PETRI NETS

We follow the notation of Heiner et al. [7]. A *Petri net*, or *net* for short, is a tuple (T, P, f). P is a finite set of places (biochemical entities) and T is a finite set of transitions (reactions) such that $P \cap T = \emptyset$. $f : ((P \times T) \cup (T \times P)) \to \mathbb{N}$ is the set of (non-negatively) weighted directed arcs between places and transitions. A *marked Petri net* is a tuple $\mathcal{M} = (T, P, f, m_0)$. m_0 is the initial marking on the net where a marking $m : P \to \mathbb{N}$ is a mapping of places to tokens. The number of tokens on a place $p \in P$ in a marking m is $m(p)$. A Petri net is represented graphically by circles (places), rectangles (transitions), arcs with arrows (directed arcs) and dots or numbers within places (tokens).

The set of pre- and post-places of a transition t is ${}^\bullet t = \{p \in P \mid f(p,t) > 0\}$ and $t^\bullet = \{p \in P \mid f(t,p) > 0\}$ respectively. Likewise the set of pre- and post-transitions of a place p is ${}^\bullet p = \{t \in T \mid f(t,p) > 0\}$ and $p^\bullet = \{t \in T \mid f(p,t) > 0\}$ respectively.

The dynamic behaviour of a net is defined by the firing of transitions in T. A transition t is enabled in a marking m, written $m[t\rangle$, if $\forall p \in {}^\bullet t : m(p) \geq f(p,t)$. The set of transitions enabled in m, $\{t \in T \mid m[t\rangle\}$, is written $Enabled(T, m)$. A transition t that is enabled in m may fire to produce a new marking m', written $m \to_t m'$ where $\forall p \in P : m'(p) = m(p) - f(p,t) + f(t,p)$.

An *execution* is a sequence of transitions $R = t_1, \ldots, t_k$ from m reaching m', written $R \vdash m \to m'$, if $m \to_{t_1} m_1 \ldots \to_{t_k} m'$. An execution is reaction minimal, with respect to a given set of output places, if there is no proper subsequence of the transitions in R that can be fired to mark the given output places. There may be more than one reaction minimal execution for a set of outputs.

The *incidence (stoichiometric) matrix* of a net (T, P, f) is a matrix $C : P \times T \to \mathbb{Z}$, indexed by P and T, such that $C(p,t) = f(t,p) - f(p,t)$. Hence, $C(p,t)$ is the change in marking on p by firing t.

Place and transition invariants formalise a steady state view of Petri nets.

A *T invariant* is a transition vector $y : T \to \mathbb{Z}$ such that y is a nontrivial nonnegative integer solution of $C \cdot y = 0$. Hence a transition vector is a T invariant if there exists some marking (not necessarily reachable) such that the sequential firing of the transitions in the T invariant reproduces the marking (cyclic behaviour). For a T invariant y, there is an execution R (an ordering of y) such that $R \vdash m \to m$.

The *support* of an invariant x is the set of nodes corresponding to the non-zero entries of x, written $supp(x)$. An invariant x is minimal if there is no invariant z such that $supp(z) \subset supp(x)$ and the greatest common divisor of all nonzero entries of x is 1. In the following text we consider only minimal invariants and henceforth omit the word minimal.

A *P invariant* is a place vector $x : P \to \mathbb{Z}$ such that x is a nontrivial nonnegative integer solution of $x \cdot C = 0$. The weighted sum of the tokens on the places in a P invariant is constant for any marking reachable by the firing of transitions (mass conserving places).

A Petri net is k-bounded (has a finite set of reachable markings) if there is some k such that no place in the net can have more than k tokens. A Petri net is guaranteed to be k-bounded if all places belong to at least one minimal P invariant (all places are mass conserving).

3. T INVARIANT ANALYSIS

In this section we explore how to compute T invariants that in some cases correspond to pathways in a Petri net. We then characterise three network structures that cause this approach to produce incorrect results.

The usual approach to compute T invariants that correspond to pathways is to apply transformations to the net prior to computing the T invariants. The purpose of these transformations is to repeat the empty marking (all places have no tokens), such that the transitions in the T invariant capture the production of tokens, their flow through the net and their consumption. We follow the approach taken by [8, 11] and apply the following three transformations (illustrated in Figure 1):

(1) T invariant analysis is not concerned with the initial marking m_0 of a Petri net. To encode the initial marking, any place p that is initially marked $m_0(p) > 0$ is given a *source transition* that can generate an infinite number of tokens on p.

(2) To allow the possibility of repeating the empty marking, places with no post-transitions $p^\bullet = \emptyset$ are given a *sink transition* that can consume an infinite number of tokens on the place.

(3) Again, to allow the possibility of repeating the empty marking, places that are both pre- and post-places ${}^\bullet t \cap t^\bullet$ of a transition t are changed to be only pre-places of the transition $t^\bullet = t^\bullet - ({}^\bullet t \cap t^\bullet)$. Hence all bidirectional arcs are changed to unidirectional arcs from the place to the transition.

These transformations allow us to use T invariant analysis to compute pathways in a network. The pathways are computed from inputs (places with source transitions) to outputs (places with sink transitions). If we wish to designate a place as an output that was not given a sink transition through transformation (2), we can explicitly add a sink transition to the place.

In some cases T invariant analysis computes pathways very efficiently. However we have found three structure patterns that can cause incorrect results. We describe the patterns (place traps, consumption conflicts and protein degradations) below.

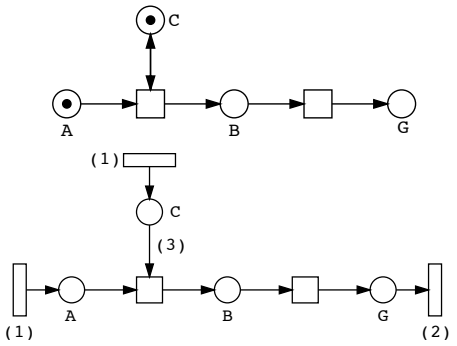

Figure 1: A marked Petri net before (top) and after (bottom) applying the transformations required to compute pathways. Source and sink transitions are denoted by long rectangles. Labels (1), (2) and (3) denote the transformation applied to the net. Petri net images are produced by Snoopy [18].

3.1 Place Traps

A *place trap* (often referred to as trap) is a set of places that once marked cannot become unmarked [7]. A set of places $Q \subseteq P$ is a place trap if $Q^\bullet \subseteq {}^\bullet Q$ (every transition that subtracts tokens from the place trap also puts tokens into the place trap). Place traps are found in many models of biological systems, for example protein phosphorylation, protein ubiquitination and enzymes often involve place traps.

A model that contains a place trap cannot always repeat an empty marking because once the place trap is marked, it cannot become unmarked. Because of this, there can be no T invariant that places a token onto the place trap. However, a T invariant can include a place trap by repeating a marking that is empty for all places except at least one place in the place trap.

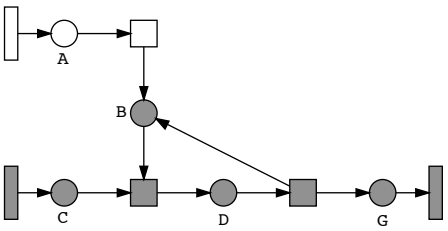

Figure 2: A Petri net containing a place trap on {B, D}. The single T invariant and all related places are highlighted in grey.

Consider the problem of finding a pathway from $\{A, C\}$ to $\{G\}$ in the Petri net in Figure 2. The Petri net has a single T invariant that starts with the place trap $\{B, D\}$. The T invariant repeats a marking that is empty for all places except B. The T invariant is not an execution because it is not realisable – there is no ordering of the transitions in the T invariant such that all the transitions can fire. The T

invariant misses the transitions to produce B.

We note that the place trap in Figure 2 could be caused by an enzymatic reaction where B is the enzyme, C is the substrate, D is the enzyme-substrate complex and G is the product. The enzyme and enzyme-substrate complex is the place trap in this case.

Enzymes are often abstracted such that the enzyme-substrate complex is removed and hence an enzyme is then a place with a bidirectional arc to a transition. The enzyme is a place trap (with a single place) if there are no unidirectional outgoing arcs from the place. Transformation (3) removes place traps with a single place, however this can cause extra consumption conflicts as described below. Place traps with multiple places can be handled similarly by removing (by hand) an arc that will break the trap, as per [11]. However, we wish to avoid manual alteration of models.

3.2 Consumption Conflicts

Consider the problem of finding a pathway from $\{A, D\}$ to $\{G\}$ in the Petri net in Figure 3. There is no marking that can be repeated by a T invariant. The production of E and F is coupled because E and F are produced by the same transition (perhaps a decomplexation or protein cleavage reaction). The number of tokens produced on E and F is always the same however there is a difference in the number of outgoing arcs from E and F, 2 and 1 respectively. These arcs are required to produce a token on place G. We call this scenario a consumption conflict. Producing a token on G will always leave one more token on F than E, therefore it is never possible to repeat any marking.

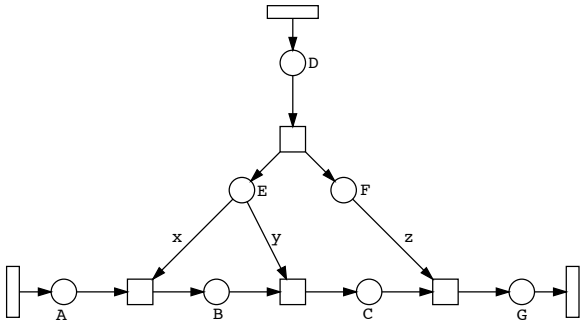

Figure 3: A Petri net containing a consumption conflict between places E and F. There are no T invariants in this Petri net because of the conflict.

Note that arcs x, y and z in Figure 3 can either be unidirectional or bidirectional arcs (E and/or F can be an enzyme) because transformation (3) will convert all bidirectional arcs to unidirectional arcs. Hence, transformation (3) can cause extra consumption conflicts in a model.

3.3 Protein Degradations

Consider the problem of finding a pathway from $\{A, B\}$ to $\{G\}$ in the Petri net in Figure 4. The minimal sequence of transitions to produce G will leave a token on D. The token on D must be consumed because the empty marking must be repeated. An extra two transitions must fire to consume the token on D, hence the T invariant contains extra transitions that are not part of the (minimal) pathway from $\{A, B\}$ to $\{G\}$. The T invariant in this net is actually a pathway from $\{A, B, E\}$ to $\{G, F\}$.

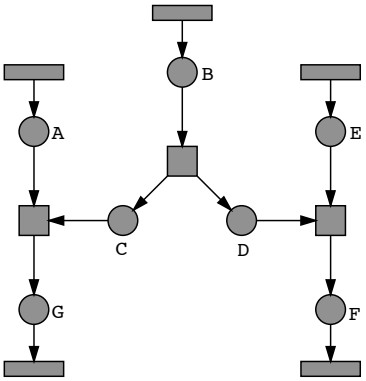

Figure 4: A Petri net with the single T invariant and all related places are highlighted in grey. The T invariant does not correspond to a (minimal) pathway from A to G because unrelated transitions are included.

To summarise, the three patterns above illustrate types of Petri nets where T invariant analysis is inappropriate to compute pathways. Place traps are detectable, however it is not immediately obvious whether consumption conflicts and protein degradations are always detectable.

3.4 Alternative Approach

An alternative approach is to use a different set of transformations. We change transformation (3), the consumption of enzymes, to

(3') All places that are both pre- and post-places $^\bullet t \cap t^\bullet$ of a transition t are given a sink transition that can consume an infinite number of tokens on the place.

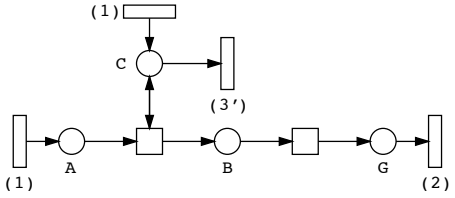

Figure 5: The alternative set of transformations applied to the Petri net from Figure 1. Notice the sink transition added by transformation (3').

Rather than consume the enzyme in the transition, we allow the enzyme to be consumed by a sink transition (shown in Figure 5). This removes consumption conflicts caused by transformation (3). However this is not a complete fix because not all consumption conflicts are caused by transformation (3), unrelated transitions may be included due to protein degradation and place traps may exist.

If the model contains no bidirectional arcs, there will be the same T invariants as before otherwise there will be a larger number of smaller T invariants. In this case we must compose T invariants to find (minimal) pathways, however composing T invariants is a non trivial task (discussed briefly in [5]). Also note that composing T invariants to find (minimal) pathways is similar to the initial problem, composing transitions to find (minimal) pathways. The alternative approach is therefore not suitable.

4. REACTION MINIMAL PATHS

We introduce the notion of a reaction minimal path that is well suited to describe pathways in a signalling network. Specifically, we are interested in reaction minimal goal/avoid paths that produce the goal (outputs) from the initial marking of a model (inputs) without using an avoid. This approach is based on a direct analysis of the possible executions/pathways of a (marked) Petri net rather than a steady state analysis of a modified Petri net. Therefore this approach does not face challenges described in the previous section and requires no semantic transformations of the Petri net.

A multiset is a pair (A, f) where A is the underlying set of elements and $f : A \to \mathbb{N}^+$ is the (positive) multiplicity of each element in A. The multiplicity of $a \in A$ is written $f(a)$. Given a multiset $M = (A, f)$, the elements of M are written $\{f(a_1) * a_1, \ldots, f(a_n) * a_n\}$ where $n = |A|$ and if $f(a) = 1$ then $f(a)*$ is omitted. The cardinality of M, written $|M|$, is $\sum_{a \in A} f(a)$. An element a belongs to M, written $a \in M$, iff $a \in A$. An element a is added to M, written $Add(M, a)$, returning $M' = (A', f')$ where $A' = A \cup \{a\}$, $\forall b \in (A - \{a\}) : f'(b) = f(b)$ and if $a \in A$, $f'(a) = f(a) + 1$ else $f'(a) = 1$. Given two multisets $M_1 = (A_1, f_1)$ and $M_2 = (A_2, f_2)$, M_1 is a proper submultiset of M_2, written $M_1 \subset M_2$, if $M_1 \neq M_2$ and $\forall a \in A_1 : a \in A_2$ and then $f_1(a) \leq f_2(a)$.

An execution R is a sequence of transitions whereas a path $\overline{R}$ is a multiset of transitions. A path $\overline{R}$ from m reaching m', written $\overline{R} \vdash m \rightsquigarrow m'$, is a multiset representation of an execution R such that $R \vdash m \to m'$.

LEMMA 1. *All executions R of $\overline{R}$ starting at m reach the same final marking.*

Proof. An execution R of $\overline{R}$ starting at m reaches m' where $\forall p \in P : m'(p) = m(p) + \sum_{t \in R} f(t, p) - \sum_{t \in R} f(p, t)$. *$m'$ is independent of the order of transitions in R because $\sum_{t \in R} f(t, p)$ and $\sum_{t \in R} f(p, t)$ are independent of the order of the transitions in R.*

Avoids: An avoid set A is a set of places $A \subseteq P$ to be avoided. A transition t satisfies the avoid constraint, written $t \models A$, if the transition does not have a pre- or post-place in the avoid set, $(\bullet t \cup t\bullet) \cap A = \emptyset$. A path $\overline{R}$ from m to m' satisfies the avoid constraint, written $\overline{R} \vdash_A m \rightsquigarrow m'$, if $\forall t \in \overline{R} : t \models A$.

Goals: A goal set G is a set of places $G \subseteq P$ that we wish to have marked. A marking m satisfies the goal constraint, written $m \models G$, if $\forall g \in G : m(g) \geq 1$. A path $\overline{R}$ from m to m' satisfies the goal constraint, written $\overline{R} \vdash^G m \rightsquigarrow m'$, if $m' \models G$.

A goal/avoid path is a path from the initial marking in a model satisfying both the goal and the avoid constraints, written $\overline{R} \vdash^G_A m_0 \rightsquigarrow m'$. A goal/avoid path $\overline{R}$ is reaction minimal if there is no goal/avoid path $\overline{R'}$ that is a proper submultiset $\overline{R'} \subset \overline{R}$. Clearly no path can have a place as both a goal and an avoid. Thus we require that the sets of goals and avoids are disjoint: $G \cap A = \emptyset$.

4.1 Algorithm

We present an algorithm for computing all reaction minimal goal/avoid paths in a marked Petri net $\mathcal{M} = (T, P, f, m_0)$ that is k-bounded. We use multiset semantics as we are not concerned with the order of the firing of transitions. The reaction minimal property ensures that all transitions in the path are required to reach the goal set.

The set of reaction minimal goal/avoid paths from m_0 reaching G without using A can be found by generating pairs of markings and paths. The pairs are generated in stages following a breadth first search such that Stage(n) contains pairs $(m, \overline{R})$ where $|\overline{R}| = n$ and $\overline{R} \vdash m_0 \rightsquigarrow m$.

Given $(m, \overline{R})$ reached by an execution R in Stage(n), $(m, \overline{R})$ could be reached again by following a different execution $R' \neq R$. We ignore subsequent $(m, \overline{R})$ as these represent different interleavings of the same multiset of transitions.

We say that $(m, \overline{R})$ is subsumed by $(m', \overline{R'})$ relative to G if $\overline{R'}$ is a proper submultiset of $\overline{R}$, $\overline{R'} \subset \overline{R}$, and
(1) $m' = m$ ($\overline{R'}$ reaches the same marking)
or
(2) $m' \models G$ ($\overline{R'}$ satisfies the goal constraint)

DEFINITION 1. *Stage(0) contains one pair, the initial marking and the empty path $(m_0, \emptyset)$. Stage(n) for $n \geq 1$ contains all pairs $(m, \overline{R})$ with $\overline{R} \vdash m_0 \rightsquigarrow m$ and $|\overline{R}| = n$ such that $(m, \overline{R})$ is not subsumed by a member of Stage(j) for $j < n$. This ensures that all goal/avoid paths are reaction minimal.*

We use breadth first search because checking whether $(m, \overline{R})$ in Stage(n) is subsumed by some $(m', \overline{R'})$ requires checking $(m', \overline{R'})$ in Stage(j), $j < n$. Hence, $\overline{R'} \subset \overline{R}$ requires $|\overline{R'}| < |\overline{R}|$.

LEMMA 2. *Termination: there exists an n such that Stage(n) is empty.*

Proof. The set of possible markings in a k-bounded Petri net is finite. The set of paths to each marking such that there is no proper submultiset that reaches the same marking is finite because the set of transitions is finite. Therefore, the set of (marking, path) pairs is finite and hence there must be some n such that Stage(n) is empty.

LEMMA 3. *Completeness: if there exists a reaction minimal goal/avoid path $\overline{R} \vdash^G_A m_0 \rightsquigarrow m$ then $(m, \overline{R})$ is in Stage(n) where $n = |\overline{R}|$.*

Proof. Definition 1 ensures that all goal/avoid paths found in the stages are reaction minimal. The set of stages contains all markings except those markings reachable only by a non reaction minimal goal/avoid path. Therefore if there exists a reaction minimal goal/avoid path $\overline{R} \vdash^G_A m_0 \rightsquigarrow m$ then $(m, \overline{R})$ is in Stage(n) where $n = |\overline{R}|$.

THEOREM 1. *For any k-bounded Petri net, all executions to a goal set avoiding an avoid set are found by generating the set of stages.*

Proof. Multiset semantics are sufficient to describe an execution (Lemma 1). The set of stages is finite for any k-bounded Petri net (Lemma 2). If there exists a reaction minimal goal/avoid path then it is in some Stage(n) (Lemma 3). Therefore, all executions to a goal set avoiding an avoid set are found by generating the set of stages.

Pre-processs: To make all paths satisfy the avoid constraint, we remove any transition that has a pre- or post-place in the avoid set: $T* = \{t \in T \mid (\bullet t \cup t\bullet) \cap A = \emptyset\}$.

The algorithm for computing stages assuming a pre-processed network works as follows:

Stage 0:
$(m_0, \emptyset)$
Stage n: Given Stage(n-1)
for $(m, R) \in$ Stage(n-1) **do**
  for $t_i \in Enabled(T*, m)$ **do**
    m' such that $m \rightarrow_{t_i} m'$
    $R' = Add(R, t_i)$
    Add (m', R') to Stage(n) if it is not subsumed by a pair in Stage(j) for some $j < n$
  end for
end for
if Stage(n) is Empty **then**
  Exit
end if

Reaction Minimal Paths: The set of reaction minimal goal/avoid paths reaching G without using A in a k-bounded Petri net $\mathcal{M}$ is given by:
$RMP(\mathcal{M}, G, A) = \{\overline{R} \mid \exists n : (m, \overline{R}) \in \text{Stage(n) and } m \models G\}$

Algorithm correctness. In the algorithm above, paths are multisets, stages are generated as per Definition 1 (due to the breadth first search and subsumption rule) and the set of stages is finite for any k-bounded Petri net. Therefore, Theorem 1 holds for this algorithm.

Relevant Subnet Optimisation. To optimise the computation we can apply the relevant subnet function with respect to G and A, $Subnet(T, m_0, G, A)$, as discussed in Section 1. This function removes any transition that has a pre- or post-place in the avoid set and transitions that do not contribute to reaching the goal set.

Approximation. An *unbounded Petri net* is a Petri net where there does not exist a k such that all places have at most k tokens in any reachable marking. An unbounded Petri net has an infinite set of reachable markings and therefore the algorithm in the previous section may not terminate. To obtain approximate results, we can compute the set of reaction minimal goal/avoid paths with an upper bound on the number of Stages used, N. Hence the algorithm will terminate after Stage(N) and the paths will have a maximum cardinality of N. Note that this is the same approach taken by bounded model checking.

4.2 Example 1

Consider the model in Figure 6 with a goal set $\{G\}$ and avoid set $\emptyset$. The algorithm produces 3 stages as below:
Stage 0: $(\text{AXF}, \emptyset)$
Stage 1: $(\text{BXF}, \{r1\})$ $(\text{AYF}, \{r2\})$
Stage 2: $(\text{BXG}, \{r1, r3\})$ $(\text{BYF}, \{r1, r2\})$ $(\text{AYG}, \{r2, r4\})$
The set of reaction minimal goal/avoid paths is:
$\{\{r1, r3\}, \{r2, r4\}\}$
Stage 3 is empty because all (marking, path) pairs are subsumed by some pair in a previous Stage. $(\text{BYG}, \{r1, r2, r3\})$ is subsumed by $(\text{BXG}, \{r1, r3\})$ because BXG satisfies the goal constraint. Likewise, $(\text{BYG}, \{r1, r2, r4\})$ is subsumed by $(\text{AYG}, \{r2, r4\})$ because AYG satisfies the goal constraint.

4.3 Example 2

Consider the model in Figure 7 with a goal set $\{G\}$ and avoid set $\emptyset$. The algorithm produces 3 stages as below:

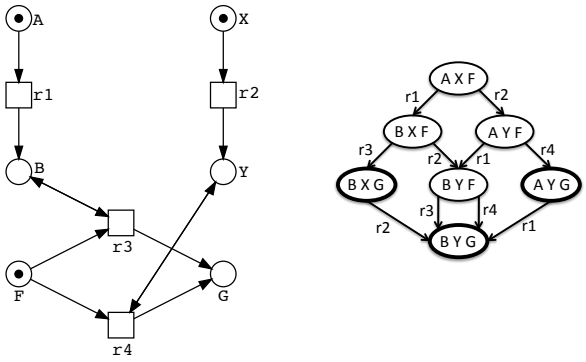

Figure 6: An example model (left) and related statespace (right). States that satisfy the goal constraint are marked with a thick line.

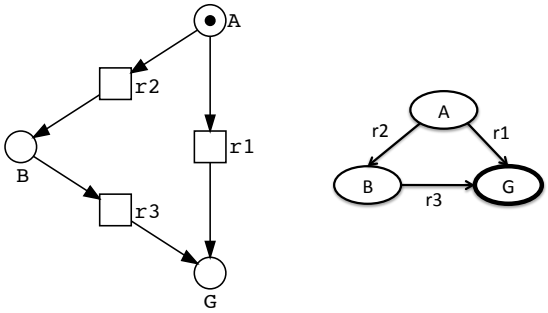

Figure 7: An example model (left) and related statespace (right). States that satisfy the goal constraint are marked with a thick line.

Stage 0: $(\text{A}, \emptyset)$
Stage 1: $(\text{G}, \{r1\})$ $(\text{B}, \{r2\})$
Stage 2: $(\text{G}, \{r2, r3\})$
The set of reaction minimal goal/avoid paths is:
$\{\{r1\}, \{r2, r3\}\}$.
Although $\{r2, r3\}$ is a longer path to marking G than $\{r1\}$, it is distinct and all transitions are required to reach G, therefore it is reaction minimal.

4.4 Example 3

Consider the model in Figure 8 with a goal set $\{G\}$ and avoid set $\emptyset$. The algorithm produces 7 stages as below:
Stage 0: $(\text{A B C}, \emptyset)$
Stage 1: $(\text{A1 B C}, \{r1\})$
Stage 2: $(\text{A B1 C}, \{r1, r2\})$ $(\text{A B C1}, \{r1, r3\})$
Stage 3: $(\text{A1 B1 C}, \{2 * r1, r2\})$ $(\text{A1 B C1}, \{2 * r1, r3\})$
Stage 4: $(\text{A B1 C1}, \{2 * r1, r2, r3\})$
Stage 5: $(\text{A1 B1 C1}, \{3 * r1, r2, r3\})$
$(\text{A G}, \{2 * r1, r2, r3, r4\})$
Stage 6: $(\text{A1 G}, \{3 * r1, r2, r3, r4\})$
The set of reaction minimal goal/avoid paths is:
$\{\{2 * r1, r2, r3, r4\}\}$
Even though this is a 1-bounded Petri net, the transition $r1$ must fire more than once to reach $\{G\}$. This is the rationale for using multiset representations of executions even in models with only presence/absence of biochemical species (e.g. Petri nets from Pathway Logic).

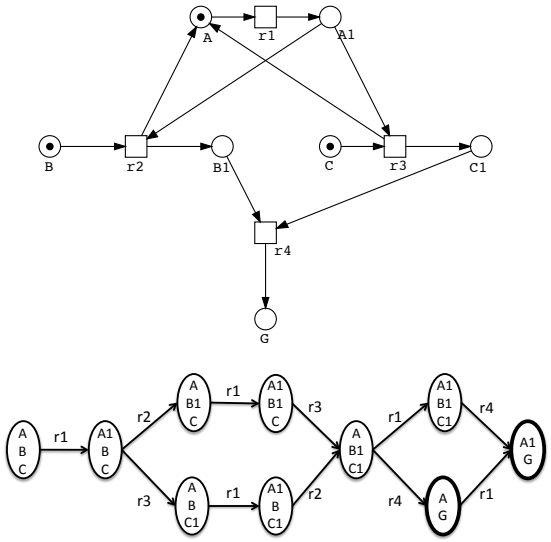

Figure 8: An example model (top) and related statespace (bottom). States that satisfy the goal constraint are marked with a thick line.

5. RESULTS

We have used the algorithm presented in Section 4.1 to compute reaction minimal goal/avoid paths for two Petri net models generated from the Pathway Logic knowledge base of cellular signalling response: activation of Erk and activation of Rela. These results are summarised in Table 1 and show that T invariant analysis does not compute the set of pathways in one of the models.

For the Erk activation model the initial event is Egf (Epidermal Growth Factor) binding to its receptor, EgfR, and the goal is activation of Erk1 and Erk2 (Erks) in the EgfR complex (EgfRC): G=Erks–act–EgfRC. We first generated the relevant subnet for G using the Pathway Logic Assistant (using the relevant subnet algorithm in [20]). The relevant subnet for this goal contains none of the problematic network structures discussed in Section 3, and in this case, the set of T invariants corresponds exactly to the set of pathways activating Erks. It is interesting to note that T invariant analysis had a significantly quicker execution time of less than 1s.

For the Rela activation model there are two potential stimuli IL1 (Interleukin 1) and Tnf (Tumor Necrosis Factor) and the goal is activation of Rela in the nucleus: G=Rela–act–Nuc. The relevant subnet for this goal contains several place traps due to the ubiquitination reactions, for example the E2 ubiquitin ligase Traf5 causes phosphorylated Irak1 to become ubiquitinated. The standard set of transformations required for T invariant analysis results in consumption conflicts, hence no T invariants are found. We have also applied the alternative T invariant approach outlined in Section 3.4. This resulted in a large number of small T invariants and the set of T invariants did not fully cover the relevant subnet, therefore it was not possible to compose T invariants to find the pathways.

We now proceed with an analysis of what reaction minimal paths tell us about these models. In the following we assume

	Erks–act–EgfRC	**Rela–act–Nuc**
Places	54	92
Transitions	38	57
Reachable Markings	149,014	95,096
Explored (marking, path) Pairs	618,861	171,237
Number of Stages	24	34
Execution Time (s)	217	23
Reaction Minimal Paths	144	39
T Invariants	144	0

Table 1: The result of computing the reaction minimal goal/avoid paths for G=Erks–act–EgfRC and G=Rela–act–Nuc contrasted with T invariants.

a marked Petri net $\mathcal{M} = (P, T, f, m_0)$, a set of goals G and a set of avoids A. Let $RMP = RMP(\mathcal{M}, G, A)$ be the set of reaction minimal goal/avoid paths which can be easily computed using the algorithm presented in Section 4.

5.1 Essential Transitions

A transition t is essential, written $ess[\mathcal{M}, G, A](t)$, if there is no path $\overline{R}$ using only transitions in $T' = T - t$ such that $\overline{R} \vdash_A^G m_0 \rightarrow m'$ for some m'. We can compute the set of essential transitions using the set of reaction minimal goal/avoid paths RMP as follows:

$$ess[\mathcal{M}, G, A] = \{t \mid \forall \overline{R} \in RMP : t \in \overline{R}\}$$

More generally, a set of transitions T' is essential, $ess[\mathcal{M}, G, A](T')$, if every path satisfying G and A contains a member of T', and no proper subset of T' has this property. This can be checked using the set of reaction minimal paths for G,A as follows:

$$ess[\mathcal{M}, G, A](T') \Leftrightarrow$$
$$\forall \overline{R} \in RMP \;\; \exists t \in T' : t \in \overline{R} \;\wedge$$
$$\forall T'' \subset T' \;\; \exists \overline{R} \in RMP : T'' \cap \overline{R} = \emptyset$$

5.2 Used Places

A path $\overline{R} \vdash_A^G m_0 \rightsquigarrow m'$ uses a place p, $uses[\mathcal{M}, G, A](\overline{R}, p)$, if there is no path $\overline{R''} \vdash_A^G m_0 \rightsquigarrow m'$ where $\overline{R''} \subseteq \overline{R'}$ and $\overline{R'}$ is the result of removing from $\overline{R}$ any transition t with p as a pre-place, $p \cap {}^\bullet t \neq \emptyset$. This holds just if $\exists \overline{R} \in RMP \;\; \exists t \in \overline{R} : p \cap {}^\bullet t \neq \emptyset$. The reduction to reaction minimal paths is valid because if $uses[\mathcal{M}, G, A](\overline{R}, p)$ then $uses[\mathcal{M}, G, A](\overline{R'}, p)$ for every reaction minimal path $\overline{R'} \subseteq \overline{R}$. Furthermore, if $\overline{R}$ is reaction minimal then $uses[\mathcal{M}, G, A](\overline{R}, p)$ iff $\exists t \in \overline{R} : p \cap {}^\bullet t \neq \emptyset$.

The set of all used places in a path $\overline{R}$ is:

$$uses[\mathcal{M}, G, A](\overline{R}) = \{p \in P \mid uses[\mathcal{M}, G, A](\overline{R}, p)\}$$

5.3 Knockouts

A place p is called a (single) knockout for G, A, written $KO[\mathcal{M}, G, A](p)$, if there is no path $\overline{R} \vdash_A^G m_0 \rightsquigarrow m$ using only transitions that do not use p, hence only transitions in $T' = \{t \in T \mid p \cap {}^\bullet t = \emptyset\}$. To check if p is a knockout, we need only check that all reaction minimal paths use p:

$$KO[\mathcal{M}, G, A](p) \Leftrightarrow \forall \overline{R} \in RMP : uses[\mathcal{M}, G, A](\overline{R}, p)$$

	Erks–act–EgfRC	Rela–act–Nuc
Signals	1	2
Essential Transitions	8	0
Essential Transition Pairs	11	183
Used Places	53	84
Single Knockouts	26	9
Double Knockouts	20	674

Table 2: The results of computing essential transitions, used places and knockouts for G=Erks–act–EgfRC and G=Rela–act–Nuc.

More generally, $P' \subseteq P$ is called a knockout set, $KO[\mathcal{M}, G, A](P')$, if every path $\overline{R} \vdash_A^G m_0 \rightsquigarrow m$ uses some element of P' and there is no proper subset of P' with this property. This can be checked using RMP as follows:

$$KO[\mathcal{M}, G, A](P') \Leftrightarrow$$
$$\forall \overline{R} \in RMP \;\; \exists p \in P' : uses[\mathcal{M}, G, A](\overline{R}, p) \;\wedge$$
$$\forall P'' \subset P' \; \exists \overline{R} \in RMP \; \forall p \in P'' : \neg uses[\mathcal{M}, G, A](\overline{R}, p)$$

5.4 Multi-signal Cellular Responses

Often cellular response requires more than one signal/ stimulus to be present. Activation of effector cells of the immune system is one example. Thus it is interesting to ask:

"Are there paths reaching G from m_0 that require more than one stimulus?"

Let $S \subseteq P$ be the places considered stimuli. Then $\overline{R}$ uses more than one stimulus if $S \cap uses[\mathcal{M}, G, A]\ (\overline{R})$ has more than one element. Again, this can be checked using only the paths in RMP.

5.5 Analysis of Erks and Rela Activation

As seen in Table 1 the relevant subnet for activation of Erks has many more reaction minimal paths than the relevant subnet for activation of Rela. This can be partly explained by the presence of multiple GTP-binding proteins each of which has several associated Guanine Nucleotide Exchange Factors (GEFs) that can serve to exchange GTP for GDP. This also partly explains why the computation for Erks involved exploring over four times as many (marking, path) pairs than reachable markings, whereas the computation for Rela involved less than two times.

From Table 2 we see that Rela has no essential transitions while Erks has several. This is consistent with the fact that the Rela relevant subnet has two possible stimuli, which give rise to mostly different pathways, while the Erks relevant subnet has a single stimulus. In contrast, the Rela relevant subnet has many more essential transition pairs than the Erks relevant subnet. The essential pairs for Rela are formed by taking one from a Tnf initiated pathway and one from an IL1 stimulated pathway. The same block of 18 IL1 pathway transitions is repeated with several Tnf pathway transitions, leading to a modest combinatorial explosion. Using the Pathway Logic Assistant we compared the IL1 and Tnf pathways activating Rela, and noticed that the proteins that occur in both pathways correspond to the single knockouts computed using reaction minimal paths. In the Erks case single knockouts correspond closely to the essential transition while the single knockouts in the Rela case are used in different transitions in response to the different stimuli. As with pairs of essential transitions, there are many more double knockouts for Rela than for Erks. Again this is partly due to combinatorial pairing of groups of stimulus specific knockouts.

The enumeration of essential transition and knockout sets has "discovered" difference in structure of the two relevant subnets. For example the Rela relevant subnet is composed of two "über pathways" one for each stimulus, while the Erks relevant subnet is basically one pathway with many local variations.

6. RELATED WORK

We discuss two alternative approaches to finding all pathways: model checking, which treats pathways as processes, and steady state analysis, which treats pathways as structures.

6.1 Model Checking Techniques

The notion of error traces in model checking is similar to reaction minimal goal/avoid paths, however there are some important differences. Given an LTL property ϕ, an error state S_E is a state that violates ϕ. An error trace is a sequence of transitions $t_1, \ldots, t_j$ from the initial state S_0 to an error state S_E such that $S_0 \rightarrow_{t_1} S_1 \ldots \rightarrow_{t_j} S_j$ where $S_j = S_E$ and $S_1, \ldots, S_{j-1} \neq S_E$. To compute error traces to the goal set G, we use a temporal logic property that asserts that the goal set is not reachable. For example, in LTL the property would be:

$\phi = \neg\Diamond(g_1 \geq 1 \wedge \ldots \wedge g_v \geq 1)$

with $g_1 \ldots g_v \in G$. This property states that it is not possible $\neg$ to reach a state $\Diamond$ where $g_1, \ldots, g_v$ are all marked. An error state for ϕ is a state where $g_1, \ldots, g_v$ are all marked.

The Pathway Logic Assistant [20] allows the user to generate a single pathway to a goal set without using an avoid set in a model. Avoids are implemented by the relevant subnet transformation as before. The LoLA (Low Level Analyzer) Petri net analysis tool [16] is used to generate paths to the goal set. LoLA, using stubborn set reduction (a partial order reduction technique), can efficiently answer reachability queries such as ϕ. If ϕ is reachable, then LoLA will return an error trace for ϕ. The error trace is not guaranteed to be reaction minimal, however transitions in the error trace that are not required to reach the goal state can be removed automatically. Unfortunately, LoLA returns only one path (an error trace) to the goal set. Subsequent paths can be found by manually removing transitions in the path from the network, however this can be time consuming and inefficient.

The SPIN model checker [10] can return all state error traces in a model for ϕ. A state error trace is a sequence of states $S_0 \rightarrow S_1 \ldots \rightarrow S_j$ where $S_j = S_E$, $S_1, \ldots, S_{j-1} \neq S_E$. We map a state error trace to the set of error traces that could generate the state error trace. SPIN permits breadth first search of the state space, which produces minimal length error traces. However, consider example 1 in Figure 6. The error states in the model are BXG, AYG and BYG and the minimal length error traces are $r1, r3$, $r2, r4$ and $r1, r2, r3/r1, r2, r4$ respectively. Because minimal length error traces are produced for *all* error states, there is no guarantee that all transitions are required to reach G. Hence error traces $r1, r2, r3$ and $r1, r2, r4$ for BYG contains a redundant transition $r2$ and $r1$ respectively. Furthermore,

consider example 2 in Figure 7. There is one error state G and the minimal length error trace is $r1$. Because only one error trace is returned for each error state, the algorithm misses an error trace to G that is reaction minimal, $r2, r3$.

Stories [3] are the most closely related work to reaction minimal paths. A story in a rule-based language captures the events that are required to reach an event of interest. A story as a sequence of events that; starting from the initial state, reaches an event of interest called the *observable*; consists only of events that are required to reach the observable; and, contains no event subsequence that has the same property. The authors however do not discuss in detail the method used to compute stories. Though not strictly a model checking technique, stories are generated from paths through the state space (stochastic simulations) [2]. The story sampler converts a stochastic simulation into a story. To compute subsequent stories, more stochastic simulations are required, however there is no guarantee that all stories are generated. Only by exploring (at least parts of) the state space can we guarantee to find all stories.

6.2 Steady State Analysis

T invariants (discussed above) and flux balance analysis (FBA) [13] are forms of steady state analysis, both being concerned with solving steady state equations. FBA is concerned with steady state behaviour of a metabolic system, analysing solutions V to

$S \cdot V = 0$ where $V \geq O$ (i.e. $v_i \geq 0$ for at least one i, $1 \leq i \leq r$)

where S is a $m \times r$ matrix and V is $r \times 1$ vector of flux levels. $v_i \geq 0$ says that reaction i can "use reactants and create products", but cannot "use products and create reactants". Bidirectional reactions must be replaced by pairs of reactions. Some reactions represent metabolic uptake (input) or secretion (output). Additional constraints may be added corresponding to bounds on these and other metabolic fluxes: $v_i \leq c$, where c is some constant.

Cycles in a metabolic network cause problems for flux balance analysis, leading to unrealisable or non-robust solutions. The usual approach is to selectively break the cycles (by hand). An alternative solution is to relax the steady state constraint and allow increase in some metabolites. This can be automated but the result requires solution of boolean combinations of inequality constraints, a more complex problem. In contrast, cycles are not a problem for dynamic pathway analysis such as our algorithm proposed above.

Finite descriptions of the solution space are important for analysis [14]. An example is the set of elementary modes: $M = \{V_1, ..., V_k\}$. This set has the following properties:

1. Any solution V is a positive linear combination of vectors in M.
2. Each V_i in M is a maximally zero solution (as for minimal T invariants).
3. Every maximally zero solution is in M.

Elementary modes can be used for analyses including [14, 17]:

- Predictions of minimal nutrient sets (media) by computing the inputs (nutrients) required for given outputs (biomass precursors).
- Finding unutilised reactions that can point to gaps in the model.
- Pathway redundancy – a measure of how many independent pathways have equivalent input and output fluxes. This is related to tolerance to single-gene knockouts or drug inhibition.
- Finding correlated reaction sets, reactions that are always "on" and "off" together; suggesting that the corresponding enzyme sets may reside on the same operon (regulon) and thus be co-regulated.

Steady state analysis answers different questions than dynamic pathway analysis. They are complementary methods to study process networks. Steady state analysis concerns "flows/firing rates" that maintain a given state. Dynamic pathway analysis, such as the algorithm introduced in this paper, is concerned with changing state and how information (local state change) propagates through a network, to control other processes, for example metabolism or gene expression.

Steady state techniques have been applied to signalling networks. FBA has been used to modularise the Toll-like receptor signalling network into "distinct input/output signalling (DIOS) pathways" [11]. The network was decomposed into 10 DIOS pathways and resulted in the identification of novel inhibition targets. Also, T Invariants were used to analyse the Pheromone response pathway in Yeast [6]. Clustering of the T invariants revealed functional modules that allowed a better understanding of the pathway. Finally, T invariant analysis of the apoptosis network in [9] discovered that some T invariants describe basic pathways and some describe cross-talk between pathways.

7. CONCLUSION & FUTURE WORK

In this paper we have explored understanding the behaviour of a signalling network by considering signal transduction pathways in which the flow of the "signal" is indicated by transitory local changes of state. We are interested in pathways that satisfy certain properties on start, final and intermediate states, for example to compute pathways from a signal to the activation of some transcription factor.

The notion of signal transduction, of concern to this paper, was compared to the steady state view in which constant information is studied, often applied to metabolic processes. We have characterised three network structures in which the steady state view is insufficient to characterise signal transduction. In each case we have demonstrated the network structure with T invariant analysis of a small Petri net model.

We have introduced our method to compute all pathways in a network that satisfy given constraints on initial, final and intermediate states. The method is then applied to the Pathway Logic knowledge base of cellular signalling response to stimuli. The pathway set is used to find knockout targets and common signalling events for an output as well as to identify those outputs which require/use multiple signals.

Several future directions have been identified which will allow us to better understand and formalise the interaction between pathways in a signalling network. We aim to identify key targets to control cellular signalling for therapeutic advance by applying these techniques to the Pathway Logic knowledge base.

A significant number of states in our Pathway Logic models are due to the independent firing of transitions. We can reduce the computational expense of our method to compute pathways by reducing the number of these states using a partial order reduction technique. However, such a technique must not compromise the correctness of the algorithm. We will explore this idea in future research.

Of additional interest is the interactions among different relevant subnets/pathways. For example, what are the knockouts for one relevant subnet/pathway that do not occur in or are not knockouts for another relevant subnet/pathway. Hence, can we find knockout targets for a cellular output that minimise the effect on other cellular outputs? We wish to extend the idea of relevant subnet/pathway interaction using graphical patterns. We aim to develop a set of patterns that formalise common types of interaction in signalling networks, for example using the categories of interaction in [4]. This will allow us to better understand the interacting nature of signal transduction in signalling networks to identify better knockout/inhibition targets.

An open-source Java program that computes all reaction minimal goal/avoid paths in Pathway Logic models as well as the models used in this paper can be found at: `www.dcs.gla.ac.uk/~radonald/cmsb2010/`. The Pathway Logic Assistant, knowledge bases, and documentation can be found at `pl.csl.sri.com`.

Acknowledgments.

This research was performed while Robin Donaldson was an international fellow at SRI International. The authors wish to thank the Jim Gatheral travel scholarship, without which this research would not be possible. This research is also supported by the project *The Molecular Nose*, funded by the Engineering and Physical Sciences Research Council (EPSRC), the *Biologica* project funded by National Science Foundation grant IIS-0513857, and the *A Computational Model of Aberrant Signaling Networks in Cancer* project funded by National Institutes of Health National Cancer Institute *Integrative Cancer Biology* program grant CA-112970.

8. REFERENCES

[1] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude: A High-Performance Logical Framework*. Springer, 2007.

[2] V. Danos, J. Feret, W. Fontana, R. Harmer, and J. Krivine. Rule-based modelling and model perturbation. *T. Comp. Sys. Biology*, 11:116–137, 2009.

[3] V. Danos, J. Feret, W. Fontana, R. Harmer, J. Krivine, P. Biosystems, E. N. Supérieure, and E. Polytechnique. Rule-based modelling of cellular signalling. In *Proceedings of CONCUR'07, LNCS*, volume 4703, pages 17–41, 2007.

[4] R. Donaldson and M. Calder. Modelling and Analysis of Biochemical Signalling Pathway Cross-talk. In *EPTCS 19*, pages 40–54, 2010.

[5] D. Gilbert, M. Heiner, and S. Lehrack. A Unifying Framework for Modelling and Analysing Biochemical Pathways Using Petri Nets. In *Proc. CMSB 2007*, pages 200–216. LNCS/LNBI 4695, Springer, 2007.

[6] E. Grafahrend-Belau, F. Schreiber, M. Heiner, A. Sackmann, B. H. Junker, S. Grunwald, A. Speer, K. Winder, and I. Koch. Modularization of biochemical networks based on classification of Petri net t-invariants. *BMC Bioinformatics*, 9, 2008.

[7] M. Heiner, D. Gilbert, and R. Donaldson. Petri Nets in Systems and Synthetic Biology. In *Schools on Formal Methods (SFM)*, pages 215–264. Springer LNCS 5016, 2008.

[8] M. Heiner and I. Koch. Petri Net Based Model Validation in Systems Biology. In J. Cortadella and W. Reisig, editors, *ICATPN*, volume 3099 of *LNCS*, pages 216–237. Springer, 2004.

[9] M. Heiner, I. Koch, and J. Will. Model Validation of Biological Pathways Using Petri Nets – Demonstrated for Apoptosis. In *Proceedings of CMSB '03*, page 173, London, UK, 2003. Springer-Verlag.

[10] G. J. Holzmann. *The SPIN Model Checker : Primer and Reference Manual*. Addison-Wesley Professional, September 2003.

[11] F. Li, I. Thiele, N. Jamshidi, and B. Palsson. Identification of Potential Pathway Mediation Targets in Toll-like Receptor Signaling. *PLoS Comput Biol*, 5(2):e1000292+, February 2009.

[12] J. Meseguer. Conditional Rewriting Logic as a Unified Model of Concurrency. *Theoretical Computer Science*, 96(1):73–155, 1992.

[13] J. D. Orth, I. Thiele, and B. O. Palsson. What is flux balance analysis? *Nat Biotech*, 28(3):245–248, 2010.

[14] J. A. Papin, N. D. Price, S. J. Wiback, D. Fell, and B. O. Palsson. Metabolic pathways in the post-genome era. *Trends in Biochemical Sciences*, 28(5):250–258, 2003.

[15] C. A. Petri. Introduction to general net theory. In W. Brauer, editor, *Net Theory and Applications, Proceedings of the Advanced Course on General Net Theory of Processes and Systems, Hamburg, 1979*, volume 84 of *LNCS*, pages 1–19, Berlin, Heidelberg, New York, 1980. Springer-Verlag.

[16] K. Schmidt. LoLA: A Low Level Analyser. In M. Nielsen and D. Simpson, editors, *Application and Theory of Petri Nets, 21st International Conference (ICATPN 2000)*, volume 1825 of *LNCS*, pages 465–474. Springer-Verlag, 2000.

[17] S. Schuster, T. Dandekar, and D. A. Fell. Detection of elementary flux modes in biochemical networks: a promising tool for pathway analysis and metabolic engineering. *Trends in Biotechnology*, 17(2):53–60, 1999.

[18] Snoopy Website. A Tool to Design and Animate/Simulate Graphs. BTU Cottbus, `http://www-dssz.informatik.tu-cottbus.de/software/snoopy.html`, 2008.

[19] C. Talcott. Pathway logic. In *Formal Methods for Computational Systems Biology*, volume 5016 of *LNCS*, pages 21–53. Springer, 2008. 8th International School on Formal Methods for the Design of Computer, Communication, and Software Systems.

[20] C. Talcott and D. L. Dill. Multiple Representations of Biological Processes. *Transactions on Computational Systems Biology*, 2006.