Petri nets in Snoopy: A unifying framework for the graphical display, computational modelling, and simulation of bacterial regulatory networks

Wolfgang Marwan[1*], Christian Rohr[1,2], Monika Heiner[2]

[1] Otto von Guericke University & Magdeburg Centre for Systems Biology, c/o Max Planck Institute for Dynamics of Complex Technical Systems, Sandtorstr. 1, 39106 Magdeburg, Germany.

[2] Department of Computer Science, Brandenburg University of Technology, Postbox 10 13 44, 03013 Cottbus, Germany.

Email: marwan@mpi-magdeburg.mpg.de, rohr@mpi-magdeburg.mpg.de; monika.heiner@informatik.tu-cottbus.de;

[*]Corresponding author

Abstract

Using the example of phosphate regulation in enteric bacteria, we demonstrate the particular suitability of stochastic Petri nets to model biochemical phenomena and their simulative exploration by various features of the software tool Snoopy.

keywords

Petri nets, stochastic Petri nets, non-Markovian Petri nets, deterministic delay, animation, Gillespie simulation, stochastic simulation analysis, continuous time Markov chain.

1 Introduction

1.1 Executable Petri nets as a unifying framework for systems biology

Systems biology is based on a transdisciplinary joint effort to understand the complex mechanisms of life at the molecular systems level. For good reasons, experimentalists and theoreticians traditionally use different languages specific to their respective disciplines, thinking and expressing themselves in different

ways. Experimentalists think in molecules and molecular mechanisms, which they illustrate with pictures, qualitative schemes, and biochemical reaction pathways. Theoreticians communicate by using equations and mathematical symbols, which are difficult to read for the majority of experimentalists, who frequently do not have any significant mathematical background. A true understanding of each other would be greatly facilitated by establishing a communication platform (and language) which is equally easy to use for both experimentalists and theoreticians. Such common language should be of unequivocal expressiveness and directly refer to the traditional languages of both.

With Snoopy, we provide a tool that supports the use of Petri nets as

- a common communication platform (modelling language) for experimentalists and theoreticians, together with
- a unifying framework for the graphical display, computational modelling, simulation, and bioinformatic annotation of biochemical networks, such as bacterial regulatory networks.

Petri nets as executable models. A Petri net is a mathematical graph with a strictly defined syntax. A Petri net is directly executable, if it is represented with the help of an appropriate tool like Snoopy. Snoopy translates automatically and thus reproducibly the graphical scheme into a set of equations used by the program to run simulations. In other words, a graphical representation of a Petri net drawn in Snoopy can be executed, i.e. simulations can be run with a mouse click; no special additional encoding is required.

Petri nets represent molecular and other mechanisms with a strictly defined syntax. The syntax of the Petri net language is simple and therefore very easy to learn (see below). Because the syntax is strictly defined, there is no ambiguity in what a graphical representation of the Petri net model means in terms of reaction mechanism. The Petri net formalism is ideal to naturally represent chemical reactions and their mechanisms, and any type of biochemical interactions. As defined by the user, Petri net components may represent molecules and reactions, or even more complex entities and processes. This allows to describe and represent biological processes at arbitrary level of abstraction (i.e. with arbitrary resolution in detail), but always mechanistically unambiguous, and to join those processes into a coherent, executable model. This option is extremely useful for signal transduction networks or gene regulatory networks as usually not all processes involved in a phenomenon of interest are and will be known with the same resolution in detail.

Petri nets provide a unifying framework for modelling and simulation in systems biology. Petri nets can be used to perform all major modelling and simulation approaches central to systems biology. Briefly,there are several reasons suggesting the deployment of Petri nets to investigate biochemical networks.

1. Petri nets [1] enjoy an intuitive bipartite graphical representation, which makes them easily comprehensible and facilitates the communication between wet-lab experimentalists and computational theoreticians.

2. Petri nets allow the unambiguous representation of various types of biological processes at different levels of abstraction (with different resolution of detail) in one and the very same model ranging from the conformational change of a single molecule to the macroscopic response of a cell [2], the development of a tissue, or even the behaviour of a whole organism; for a recent survey of case studies employing Petri nets see [3].

3. Petri nets come along with an explicit notion of concurrency. This allows to distinguish unmistakably between alternative and concurrent behaviour and means an adequate representation of inherently concurrent behaviour, such as the concurrent dephosphorylation in the individual stages of a signalling cascade [4] (see also Note 2).

4. Petri nets are directly executable by an appropriate tool like Snoopy [5]. This execution allows a time-free animation in the case of qualitative Petri nets (these are the standard Petri nets), and time-dependent animation and simulation in the case of quantitative Petri nets (in our case, stochastic Petri nets).

5. Petri nets can be explored by a substantial body of mathematically founded analysis techniques, covering structural and behavioural properties as well as their relations [4,6]; see also Notes 3-5.

6. Petri nets may serve as a kind of umbrella formalism integrating qualitative and quantitative (i.e. stochastic, continuous, or hybrid) modelling and analysis techniques. A related unifying framework is demonstrated in [4,6].

7. Petri nets in all their flavours and related analysis techniques are supported by several reliable tools, developed by an international community of computer scientists [7-10].

As we will show below, Snoopy provides several features that allow to comfortably handle large, complex models.

2 Material

2.1 Software

In this chapter we use Snoopy [5], a software tool to design and animate or simulate hierarchical graphs, among them the Petri net classes: standard Petri nets (PN), extended Petri nets (xPN), and extended stochastic Petri nets (xSPN). The Snoopy software has three distinguished features.

1. It is extensible; its generic design facilitates the implementation of new graph types.

2. It is adaptive by supporting the simultaneous use of several graph types, while the GUI adopts dynamically to the graph type in the active window.

3. It is platform-independent. The tool runs on Mac OS X, Windows, and Linux.

The software is freely available for non-profit academic research purposes at `http://www-dssz.informatik.tu-cottbus.de/software/snoopy.html`.

2.2 File formats

Petri nets which have been designed with Snoopy can be saved and reloaded in a Snoopy-specific file format applying XML technology. The file formats vary slightly for the individual Petri net classes and are typically indicated by specific file extensions, such as spped (PN), spept (xPN), spstochpn (xSPN). Snoopy offers build-in animation and stochastic simulation, which are used in this chapter.

Supplementary, Snoopy provides continuous simulation of continuous Petri nets and export to various analysis tools as well as import and export of the standard exchange format SBML, Level 2, Version 3 (`http://sbml.org`). The Petri nets and simulation plots can be saved in Encapsulated Postscript (eps) format as well. For more information see Snoopy's website, and Notes 3, 4.

2.3 Sample files

All Petri nets used in this chapter are available at `http://www-dssz.informatik.tu-cottbus.de/examples/mimb/`. To repeat the computational experiments discussed in this chapter, you just need to download Snoopy, to open these example files, and to follow the steps described in the Methods section.

3 Methods

In this section we describe briefly three net classes and the main features of their animation and simulation, which will later be used to perform the case study, the phosphate network of enteric bacteria. See also Note 1.

3.1 Petri nets (PN)

The standard notion of Petri nets does not involve any timing aspects. Those nets are purely qualitative, i.e. time-free models. Technically speaking, Petri nets are weighted, directed, bipartite graphs which consist of the following basic ingredients (compare Figures 1-2).

1. There are two types of nodes, which are called **places**, graphically represented by circles, and **transitions**, graphically represented by rectangles. Places usually model passive system components like species (substrates) playing the role of precursors and products of chemical reactions, while transitions stand for active system components like chemical reactions, transforming precursors into products or simply transporting species from

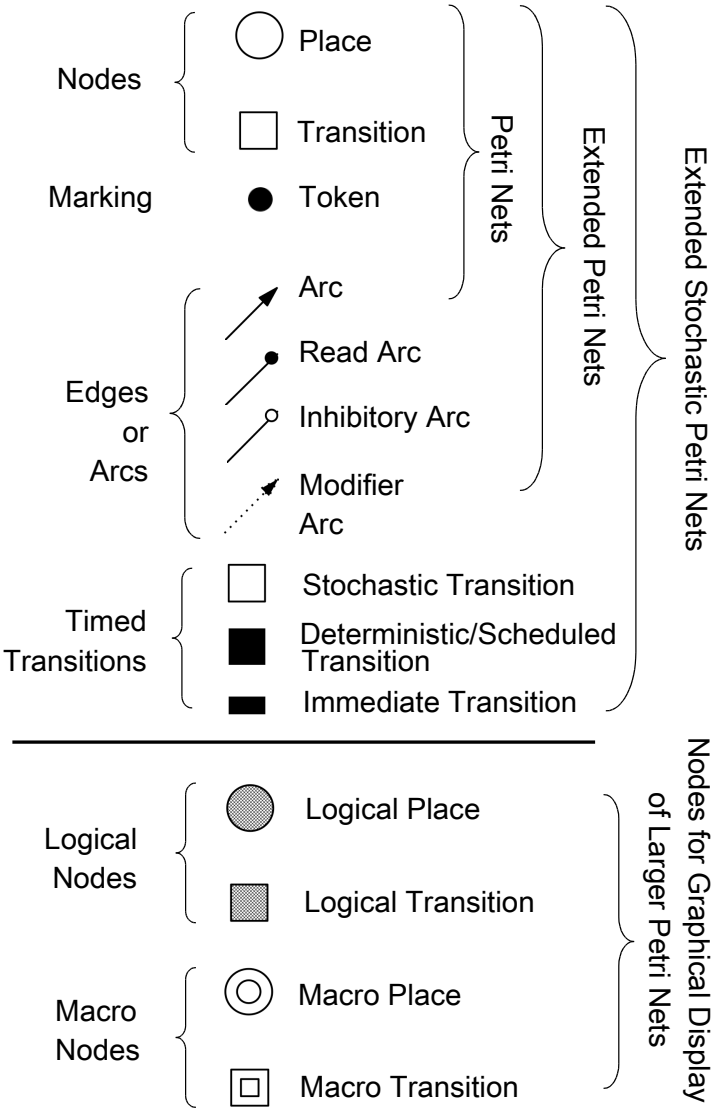

Figure 1: Petri net elements and their graphical representation. Petri nets are weighted, directed, bipartite graphs consisting of nodes and arcs. The nodes of a Petri net, the places and transitions, are interconnected by arcs. An arc always connects a place with a transition or vice versa, but never two places or two transitions with each other. Places may contain (be marked with) tokens, while transitions can not contain any tokens. Standard Petri nets are composed of places, transitions and (standard) arcs. The other Petri net components are introduced in the text.

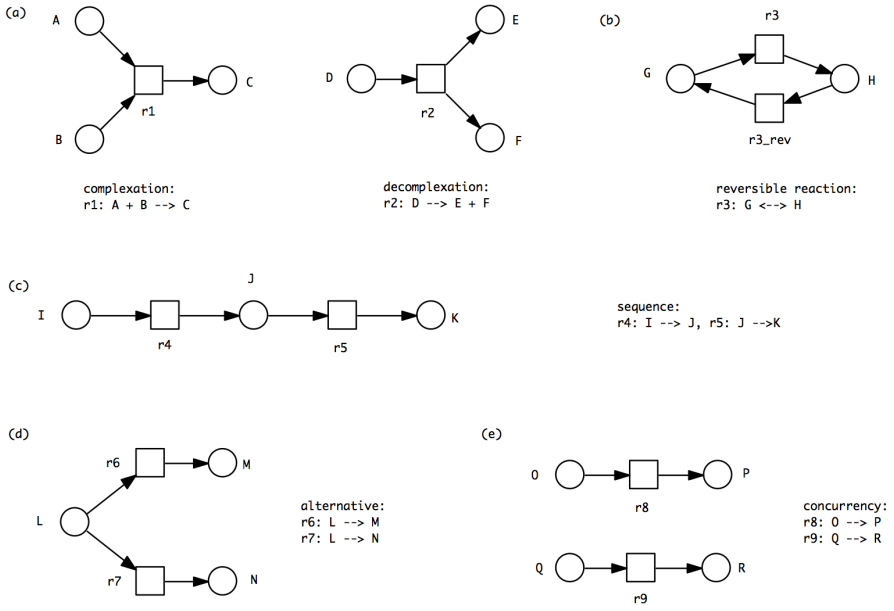

Figure 2: Some typical basic structures of biochemical networks. (a) formation and decay of a macro-molecular complex; (b) reversible reactions; (c) sequential reactions; (d) alternative reactions; (e) concurrent, i.e. independent reactions.

a site A to a site B. Reversible chemical reactions are modelled by two opposite transitions (see Figure 2-b). Due to their strong correspondence, we often use the two terms 'transition' and 'reaction' interchangeably.

2. The directed **arcs** (edges) connect always nodes of different type. They go from precursors to reactions, and from reactions to products. In other words, the pre-places of a transition correspond to the reactions precursors, and its post-places to the reactions products.

3. Arcs are weighted by natural numbers. The arc weight may be read as the **multiplicity** of the arc, reflecting known stoichiometries or other semi-quantitative assumptions. The arc weight 1 is the default value and is usually not given explicitly.

4. A place carries an arbitrary amount of **tokens**, represented as black dots or a natural number. The number zero is the default value and usually not given explicitly. Tokens can be interpreted as the available amount of a given species in number of molecules or any abstract, i.e. discrete concentration level. The tokens on all places establish together the marking of the net, which represents the current state of the system.

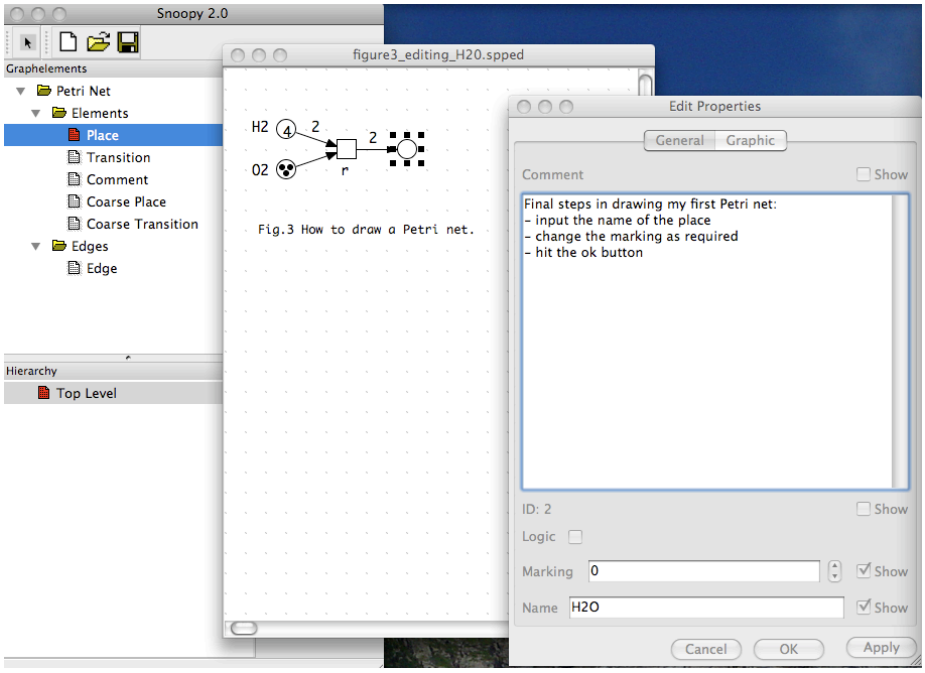

Figure 3: How to draw a Petri net with Snoopy. The menu panel on the left allows to select the type of graph elements to be created, the hierarchy panel to open macro nodes to show their sub-nets (not used here). The drawing window in the middle shows the Petri net under development. The window on the right belongs to the place that is selected in the drawing window and allows to edit the properties of this place.

How to draw a Petri net with Snoopy. To draw your first Petri net corresponding to Figure 3, repeat the following steps.

1. Create a new net and chose the appropriate net class. You get a new drawing window with the name "*unnamed*".
2. Select the graph element *Place* in the menu panel on the left-hand side, and make a left mouse click on the drawing window at those positions where you want to get a place. Each mouse click creates a new place.
3. Similarly, select the graph element *Transition* in the menu panel, and create transitions.
4. To connect two nodes, select *Edge* in the menu panel, click on the source node, and move the mouse pointer, while keeping the mouse button pressed, to the target node, where the mouse button is released.
5. A double click on a graph element (place, transition, edge) opens an attribute window, which allows to edit the element-specific properties.

6. Finally, do not forget to save your work under a new name (File → Save as).

The tokens may move through the net driven by the firing of transitions (see Figure 4). The rules of the token game are defined by the firing rule. It consists of two parts: the precondition and the firing itself.

1. A transition is **enabled**, if all its pre-places carry at least as many tokens as required by the weights of the corresponding ingoing arcs.

2. An enabled transition **may fire** (may occur), i.e. an enabled transition is never forced to fire. Upon firing a transition removes from all its pre-places as many tokens as specified by the weights of the ingoing arcs, and adds to all its post-places as many tokens as specified by the weights of the outgoing arcs. The firing happens atomically (i.e. there are no states in between), and firing does not consume any time.

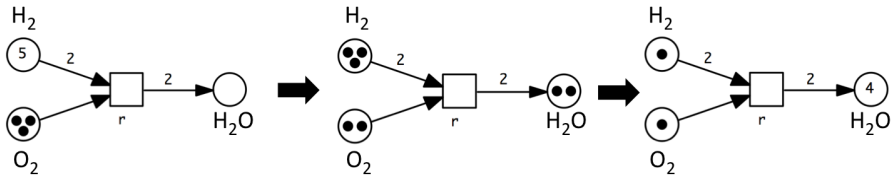

Figure 4: Petri net model of a simple chemical reaction: the formation of water from oxygen and hydrogen. The reaction's reactants and products are represented as places, the number of available molecules of each chemical compound is represented by tokens that mark the respective place. The stoichiometry of the reaction $2H_2 + O_2 \rightarrow 2H_2O$ is represented by the arc weights, indicated as numbers proximal to the respective arc. If no number is given, the arc weight is always 1. Transition r can fire two times to produce four molecules of water. Subsequently the transition cannot fire anymore as the stoichiometry of the reaction is not fulfilled.

How to animate a Petri net with Snoopy. Snoopy visualizes the token flow, allowing for an easy comprehension. To animate your first Petri net, repeat the following steps.

1. Open the net, and open the animate window (View → Animation Mode).

2. A left mouse click on a transition triggers the firing of this transition, if it is enabled.

The repeated atomic firing of enabled transitions establishes the discrete behaviour of the qualitative Petri net. This behaviour may include (compare Figure 2c-e):

1. **sequential** reactions reflecting causality (e.g., reaction r5 can not happen before r4 happened);
2. **alternative** reactions competing for tokens on shared pre-places and therefore branching into alternative behaviour; in Petri net terminology, the transitions are said to be in conflict (e.g., reactions r6 and r7 share the pre-place L; a token on L can either be consumed by r6 or by r7);
3. **concurrent** reactions, which are neither in causal nor conflict relation; thus, they are independent and can fire in any order or even concurrently (e.g., reactions r8 and r9).

In addition to a basic toolkit for drawing Petri nets, Snoopy supports two distinguished features for the design and systematic construction of larger Petri nets (see Figures 1 and 5).

1. Places and transitions can be specified as **logical nodes** (also called fusion nodes). They are automatically coloured in grey. Logical nodes with the same name are identical, i.e., graphical copies of a single node (place or transition). They are often used for compounds involved in several reactions or reactions involving distributed compounds.
2. There are two types of **macro nodes**. Macro transitions (drawn as two centric squares) help to hide transition-bordered subnets (i.e. subnets having only transitions as interface to the super-net). Likewise, macro places (drawn as two centric circles) can be used to hide place-bounded subnets (i.e. subnets having only places as interface to the super-net). Both types of macro nodes allow a hierarchical structuring of Petri nets.

Logical nodes and macro nodes help to deal with larger networks. They may contribute to a net's readability and, therefore, are crucial for non-trivial net examples. However, they do not extend the expressiveness of the modelling language. Contrary, the features introduced in the next section do extend the expressiveness.

How to draw an hierarchical Petri net with Snoopy. To hierarchically structure a Petri net, repeat the following steps.

1. Draw your flat net (or portion of it) as you wish to have it.
2. Select the sub-graph which you want to be abstracted by a macro node, and select Hierarchy → Coarse. Chose the appropriate coarse element and hit the OK button.
3. A double click on the macro node opens its attribute window and allows to assign a suitable name, a click on the entry with this name in the hierarchy panel on the left-hand side opens the sub-graph in a separate window. The blue net parts have been automatically generated and represent the connection of the sub-graph (macro node) with the neighbouring nodes on the next higher hierarchy level.

4. Finally, do not forget to save your work under a new name (File → Save as).

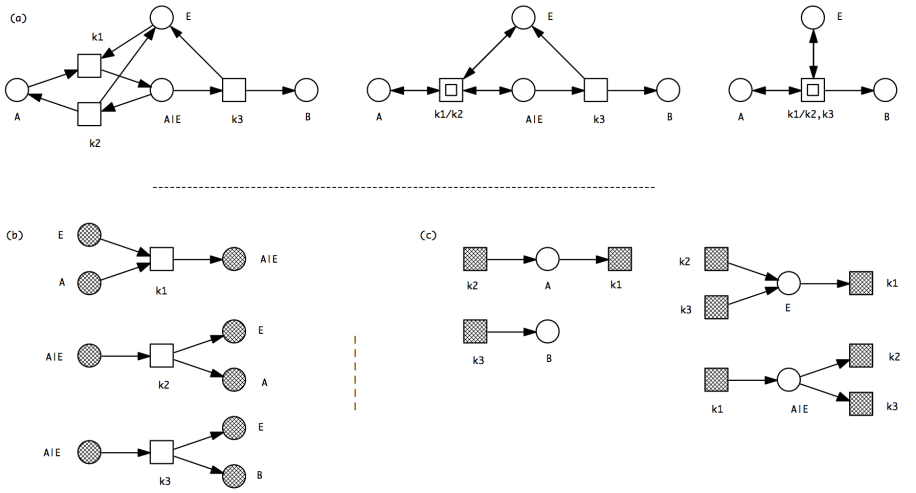

Figure 5: The use of macro transitions and logical nodes. Different graphical representations of one and the same model representing the enzymatic reaction $A + E \leftrightarrow A|E \rightarrow B + E$, where $A|E$ is the enzyme-substrate complex. (a) Macro transitions yield hierarchically structured models; (b) reaction-centric representation by the use of logical places; (c) species-centric representation by the use of logical transitions.

3.2 Extended Petri nets (xPN)

Extended Petri nets build on Petri nets and additionally provide special arc types, which go always from a place to a transition, see Figure 1 and 6. They can also carry multiplicities. The two most popular special arcs are:

1. **Read arcs** (also called test arcs), graphically represented by a black dot as arc head, query for tokens in a place representing positive side-conditions, e.g., the conformation of a protein complex that may determine whether a reaction can occur, or a specific physiological state of a cell that may determine whether a cell is responsive to a certain stimulus. The tested place needs at least as many tokens as given by the read arc's multiplicity to enable a transition. The firing of the transition does not change the number of tokens on the tested place.

   A read arc may be simulated by two opposite arcs. Although a read arc and two opposite arcs have the same total effect, there is a subtle semantic difference, which is sometimes useful to precisely represent molecular mechanisms within a biochemical network. Assume that an enzyme

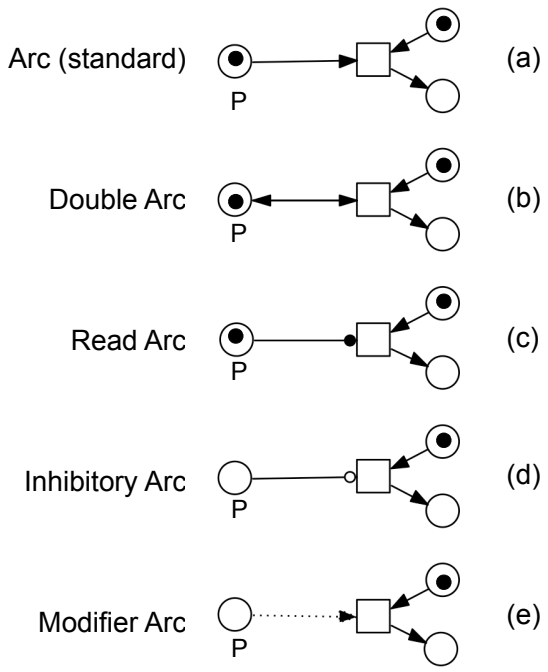

Figure 6: Different types of arcs and their influence on the firing of a transition. (a) Standard arc; the transition fires whereby the token in P is consumed. (b) Double arc; the transition fires if there is at least one token in P; upon firing, the token is transiently consumed and put back into P. (c) Read arc; the transition fires only if there is at least one token in P; the token is not consumed upon firing of the transition. (d) Inhibitory arc; the transition fires only if there is no token in place P. (a)-(d) may also carry weights. (e) Modifier arc; used with stochastic transitions only and always without weights; the probability per unit of time for the transition to fire depends on the number of tokens in place P; the transition even switches if there is no token in P. These different types of arcs can be used to model phenomena that typically occur in biochemical networks: (b) Double arcs can be used to model a catalytic reaction by neglecting formation and decay of the enzyme-substrate complex. In dynamic models the effect of substrate saturation will not occur if this representation is chosen. (c) Read arcs can be used, e.g., to model the influence of the conformation of a protein molecule on the occurrence or rate of a biochemical reaction. (d) Inhibitory arcs can be used, e.g., to model the inhibitory influence of the conformation of a protein molecule on the occurrence or rate of a biochemical reaction, e.g. upon binding of an inhibitory subunit to an enzyme. (e) Modifier arcs can be used, e.g., to model the influence of a covalent modification on the activity of a protein. The covalent modification may be represented by a token in P. If there is no token in P, the protein displays its basal activity.

E catalyses the reaction of substrate X to product Y. Then E is transiently consumed by forming the enzyme-substrate complex and reformed as the enzyme-product complex decays to form Y. Such an enzymatic reaction is represented in a mechanism-oriented style by two opposite arcs if formation and decay of the enzyme-substrate complex are not explicitly modelled. On the other hand, using the read arc is appropriate if, for example, an unphosphorylated receptor (Y) autophosphorylates (Y-P) as the consequence of a conformational change to R (the token in R represents the active conformation of the receptor). Upon autophosphorylation, the active conformation of the receptor is not transiently consumed and accordingly, the token stays in R (compare the two panels of Figure 7; see also Note 2).

2. **Inhibitory arcs**, graphically represented by an hollow dot as arc head, indicate negative side-conditions in an abstract way, e.g., if the presence of a given protein (inhibitory subunit of a protein complex) or condition inhibits a specific reaction. The inhibiting place must have less tokens than given by the inhibitory arc's multiplicity to enable a transition. The firing of the transition does not change the number of tokens on the inhibiting place. Inhibitory arcs can only be simulated by standard Petri nets, if the inhibiting place is bounded (the number of tokens never exceeds a given finite number). So they strictly increase the expressiveness.

There are two other special arc types, not used in this chapter: equal arcs and reset arcs (for more details, see [11]).

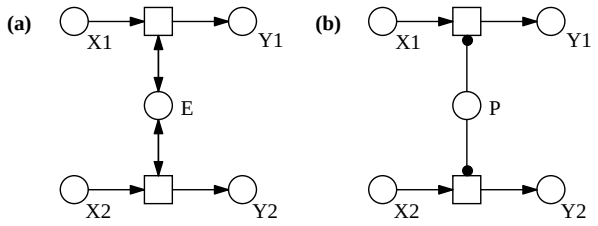

Figure 7: Semantic differences between two opposite arcs and a read arc. E catalyses the reaction r1 and r2. Thus, r1 and r2 compete for the tokens on E and can only use them alternatively. P stands for an active confirmation, allowing r3 and r4 to happen concurrently, thus sharing the tokens on P. The two arc types are not distinguishable in interleaving semantics, but in partial order semantics (see also Note 2).

3.3 Animation of qualitative Petri nets (PN, xPN)

Having obtained a new Petri net, the next step often aims at a better understanding of the net behaviour. As we have already seen, we can execute the

Petri net by playing the token game to experience the net behaviour. Each execution exemplifies some possible net behaviour. The animation can be triggered manually by clicking on an enabled transition. It can also be done in automatic mode by clicking on the control panel. In each step of the automatic mode the set of all enabled transitions is determined. There are three strategies to choose the transition(s) to be fired in the next execution step among all enabled transitions.

1. **Single step** – one single transition is randomly chosen.
2. **Intermediate step** – an arbitrary subset of concurrent transitions is randomly chosen.
3. **Maximal step** – a maximal set of concurrent transitions is randomly chosen.

Steps just done can be played backwards up to a depth of 10; this default value can be changed in the Global Preferences dialogue.

By playing the token game, we can produce and observe any reachable state. All states, which can be reached from a given state by any firing sequence of arbitrary length, constitute the set of reachable states. Each execution run of an xPN corresponds to a walk through the state space defined by this xPN. The set of states reachable from the initial state is said to be the state space of a given system. Often, this qualitative state space is infinite, caused by considering all possible behaviour of a structurally unbounded Petri net under any timing constraints. We call a Petri net bounded, if the number of tokens on all places is bounded by a constant independently of what happens. Then we get a finite state space, which however can still be too huge to be explored exhaustively.

Executing a Petri net generally needs to make decisions between alternative behaviour. Encountered alternatives (conflicts) are taken non-deterministically (automatic mode) or user-guided (manual mode). To get an exhaustive picture, we have to consider all possible execution sequences. Obviously, that's not possible for systems with infinite behaviour, which can be caused by cyclic behaviour and/or infinite state space. Thus we have to confine ourselves to a subset of execution sequences, which are considered to be representative.

In the next section we will introduce time and we will see how specific kinetic assumptions will typically restrict the qualitatively infinite state space to a quantitatively finite state space, if we are only interested in states with a probability above a certain threshold.

How to change the net class in Snoopy. A (qualitative) Petri net can be converted into a stochastic Petri net, which allows to re-use the structure. Only the additional attributes have to be set.

1. Open the qualitative Petri net, and go to the export Window (File → Export), chose the appropriate target(s) and hit the OK button.
2. Open the stochastic Petri net, just generated. It looks exactly the same as the qualitative net.

Now you are ready to specify the attributes specific for stochastic Petri nets, which are discussed in the next section.

3.4 Stochastic Petri nets (SPN)

Stochastic Petri nets build on standard Petri nets. As in the qualitative case, a stochastic Petri net maintains a discrete number of tokens on its places. But contrary to the time-free case, a stochastic firing rate is associated with each transition, determining a stochastic waiting time before an enabled transition actually fires, provided it did not loose its license to fire in between. The waiting times are random variables following an exponential probability distribution (with the parameter lambda). Therefore, all transition firing sequences of the qualitative Petri net can theoretically still occur, but the probability depends on the individual firing rates of the transitions involved (see Figure 8). Snoopy provides the following features to specify state-dependent firing rates (i.e. of the parameter lambda of the exponential probability distribution).

1. **Rate functions** are arbitrary mathematical functions, which are stored in lookup tables, if necessary. However, to keep a close relation to the net structure, only the transition pre-places are allowed as variables in the transition rate function. Popular kinetics (mass-action semantics, level semantics, see [4]) are supported by pre-defined function patterns. Of course, each transition gets its own rate function, making up together a list of rate functions. Moreover, several of such rate functions lists can be maintained, allowing for quite flexible models.

2. **Parameters**, in figures represented by ellipses, are real-valued constants, which are used for the rate functions. Parameters can be collected in macro parameter nodes, in figures represented as two centric ellipses. Again, several sets of parameters can be maintained.

3. **Modifiers** are particular arcs, in figures represented by dashed lines, which always go from a place to a transition. Pre-places connected with a transition by a modifier arc may modify the transition's firing rate, but do not have an influence on the transition's enabledness (contrary to the special arcs).

The firing itself of a stochastic transition does not consume time and follows the standard firing rule of qualitative Petri nets. Stochastic Petri nets with exponentially distributed firing delays for all transitions fulfil the Markov property; thus their semantics is described by a continuous time Markov chain (CTMC); see [4] for more details. However, in this chapter we confine ourselves to simulative approaches. Each simulation run of an SPN corresponds to a walk through the CTMC defined by this SPN, see Subsection 3.6.

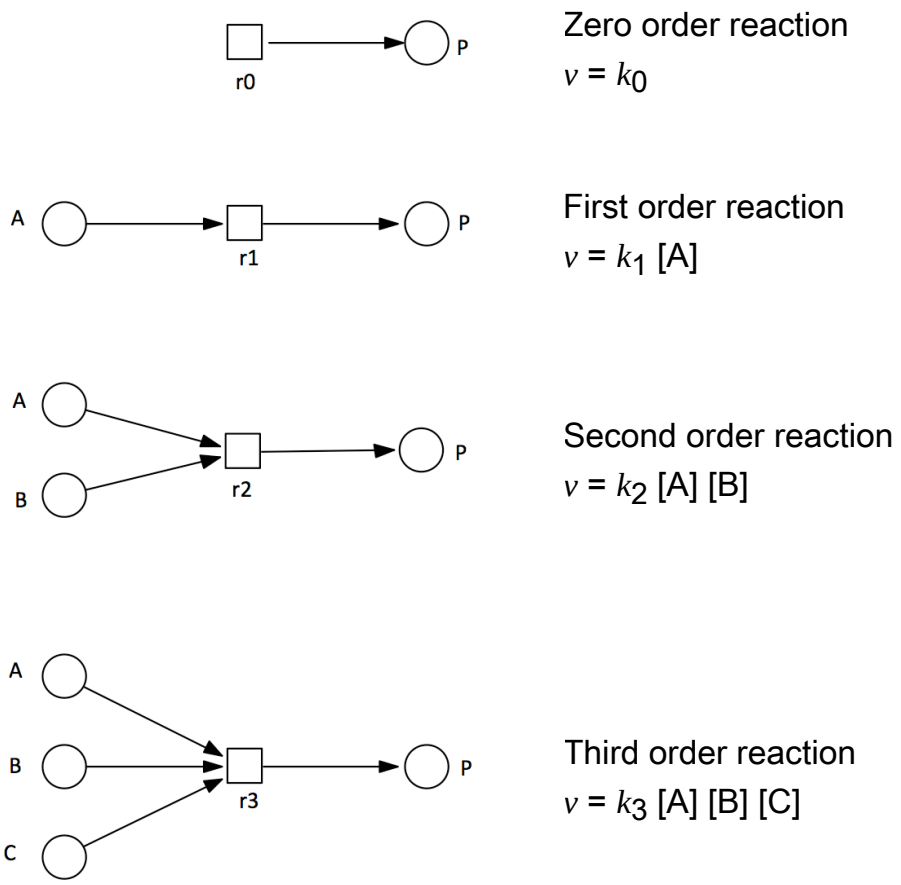

Figure 8: Automatic generation of reaction rate equations with stochastic transitions in extended stochastic Petri nets. Using the MassAction function, Snoopy automatically generates the correct reaction rate equation depending on whether a reaction is zero, first, second, or third order. Here, [x] stands for the concentration of the place x in the case of continuous Petri nets, and for the number of tokens on x in the case of stochastic Petri nets.

3.5 Extended Stochastic Petri nets (xSPN)

Extended stochastic Petri nets combine stochastic Petri nets with the special arc types of extended Petri nets. Additionally, there are deterministically timed transitions, which come in three flavours.

1. **Deterministic transitions** have contrary to stochastic transitions – a deterministic firing delay which is specified by an integer constant. The delay is always relative to the time point where the transition gets enabled. The transition may lose its enabledness while waiting for the delay to expire. Then, the transition will not fire. This behaviour is also known as the so-called pre-emptive firing rule. As for stochastic transitions, the firing itself of a deterministic transition does not consume time and follows the standard firing rule of qualitative Petri nets. Deterministic transitions may be useful to reduce networks, and thus speed-up simulations, e.g. by replacing a linear sequence of stochastic transitions by one deterministic transition with the delay set to the expectation value of the sum of the sequence.

2. **Immediate transitions** are a popular special case of deterministic transitions. They have a zero delay and always highest priority. The latter creates a subtle difference between an immediate transition and a deterministic transition with zero firing delay: if there is a conflict between the two, .i.e. a situation, where two transitions compete for tokens, the immediate transition gets priority. Immediate transitions may help to avoid stiff systems by using them for transitions with extremely high rates (non-significant delay), compared to the other transitions in the system.

3. **Scheduled transitions** are another special case of deterministic transitions. The deterministic firing occurs according to a schedule specifying absolute points of the simulation time. A schedule can specify just a single time point, or equidistant time points within a given interval, triggering the firing once or periodically. However, transitions only fire at their scheduled time points if they are enabled at this time. Scheduled transitions can dramatically restrict the (qualitative) net behaviour. The crucial point is that they allow to disturb the core model at well-defined time points as it is done experimentally with the actual biological system under investigation in the wet-lab.

An unrestricted use of deterministic transitions destroy the Markov property, which precludes an analytical evaluation by constructing and analysing the CTMC. If we consider stochastic Petri nets without deterministic transitions, the probability of two transitions to fire at the same time is practically zero. Contrary, in stochastic Petri nets with deterministic transitions, it is possible that two transitions want to fire simultaneously. To analyse such a system, all possible choices have to be considered, while in the simulation a random choice takes place, see next section (for more details and related examples see [12]).

In most practical cases, extended stochastic Petri nets are the obvious choice. Therefore, Snoopy does not distinguish between stochastic Petri nets (SPN) and extended stochastic Petri nets (xSPN).

3.6 Simulation of quantitative Petri nets (SPN, xSPN)

The simulation of stochastic Petri nets can be read as a timed version of the qualitative token game, taking into account the stochastic and deterministic delays of enabled transitions. Snoopy builds upon the Gillespie algorithm [13]. This exact method does a step-by-step simulation of possible states of the stochastic Petri net. Consequently, the result represents always a valid state of the underlying stochastic process at any time point of the simulation. The standard stochastic simulation algorithm works in the following way:

1. Initialise the SPN network with the chosen initial marking.
2. Calculate the firing rates of all enabled transitions using their rate functions.
3. Calculate the combined rate by summing up all transition rates.
4. Determine the time interval until the next state change takes place. This is done by computing an exponentially distributed random number depending on the current combined transition rate of the net.
5. Increase the simulation time by this time interval.
6. Determine the next system state change. For this purpose, a weighted random selection of the transition is made which gets the license to fire.
7. Let the selected transition fire and update the marking of the SPN.
8. Go back to step 2, if the simulation time has not yet reached its end point.

The simulation of xSPN requires some straightforward modifications of the standard simulation algorithm. Deterministic transitions require a higher priority than stochastic transitions to ensure their correct handling. Immediate transitions need to have the highest priority (over deterministic and scheduled transitions) because they have to fire instantaneously when they get enabled. Consequently, the simulation algorithm has to check for enabled immediate transitions after any change of the marking. In the case of more than one enabled immediate transitions, the selection is done randomly, but uniformly distributed. The system may run into a time deadlock, if always an immediate transition is enabled. Then, time will not progress anymore. Time deadlocks are an indication of inconsistent time assignments. The user has to take care of avoiding them.

Deterministic and scheduled transitions are treated differently. If a deterministic transition gets enabled, its timer starts running until it expires. If

the deterministic transition is still enabled, it fires. The timer will be simply switched-off, if the transition should have lost in between the enabledness. The firing of a transition in conflict does not change the timer value. The scheduled transitions fire at the specified absolute time points, if they are enabled at that time. Otherwise the intention to fire is not used. All conflicts between deterministic and/or scheduled transitions are resolved randomly, by uniformly distributed selection.

Stochastic animation. The progress of a stochastic simulation run can be observed in the automatic animation mode, which shows the corresponding token flow.

Repeated stochastic simulation. Each simulation run is one possible trace of state changes over time. Therefore, when doing stochastic simulation, a significant number of simulation runs is usually required to get meaningful results. For this purpose the traces of the simulation runs are averaged.

The simulation can be parametrised in four ways.

1. Different sets of markings, rate functions and parameters can be chosen.
2. The simulation time of interest can be specified, specifically the interval start and interval end, e.g. from 0 to 100 or from 250 to 300.
3. The number of output step points in the specified simulation interval can be declared. They define together with the interval start and end values the grid of the recorded simulation data.
4. The number of simulation runs can be specified.

Having started the simulation, the progress bar indicates how far the simulation has been gone until now, and the time consumed by the simulation is displayed too.

The user can select one out of three export functions:

1. **Direct export**: the averaged result of the selected places or transitions is exported.
2. **Single trace export**: the result of every simulation run is exported separately.
3. **Exact trace export**: every change of the marking of the selected places or the transition rates at any time point is exported.

The simulation results can be shown as

1. **Table**, where each column represents a selected place (transition) and each row shows the averaged marking (firing times) at one output step point.
2. **Plot**, where each curve stands for one selected place (transition); the x-axis holds the time and the y-axis holds the number of tokens (firing times).

Different tables and plots can be created to switch conveniently between different views on the simulation results. Each table is characterised by a set of selected places (transitions). If a node was coloured, its corresponding curve gets the same colour in the plot. Simulation plots can also be saved in eps format.

4 Case study

4.1 A case study: the phosphate regulation network in enteric bacteria

Let us now consider the phosphate utilisation gene regulatory network in Escherichia coli (see Figure 9 discussed in [14]). Note that this case study is neither meant as a scientific contribution to the understanding of the gene regulatory network of phosphate utilisation nor does it provide any comprehensive representation of the knowledge on this subject; for a recent survey see [15]. Instead, the case study is intended to show how a typical biochemical model, a gene

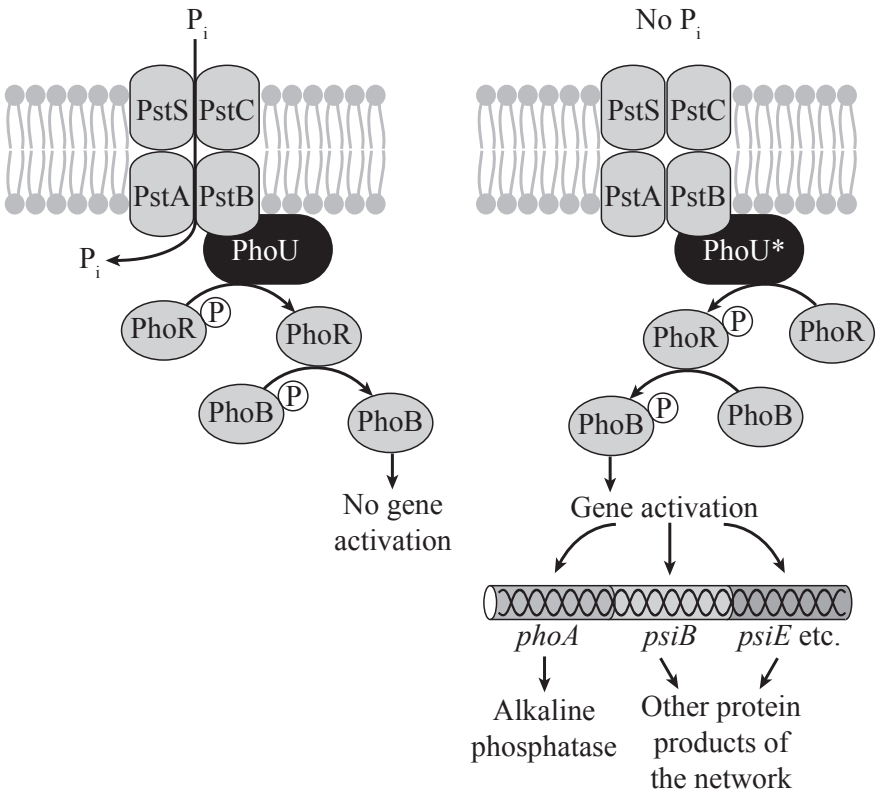

Figure 9: Biochemical model of the phosphate regulatory network in enteric bacteria. The scheme is adapted from Neidhardt et al. (1990).

regulatory network, may be formally translated into a Petri net. As we will show, the Petri net can serve as a qualitative scheme representing the molecular reaction mechanisms, but it can also be used for dynamic simulations.

For different reasons, inorganic phosphate may become a growth-limiting factor for a bacterial cell population. Under these conditions cells synthesise alkaline phosphatase (PhoA), an enzyme which is secreted into the periplasm (i.e. into the space between cytoplasmic membrane and outer membrane of a Gram-negative bacterium), where it degrades organic phosphate esters into inorganic phosphate to be taken up and recycled by the cell. Transport of inorganic phosphate is mediated by an uptake system composed of four proteins, PstS, PstC, PstA and PstB that form a transmembrane protein complex. Experimental evidence suggests that phosphate transport is sensed by the PhoU protein. If the phosphate transport system is active, the PhoU protein resides in its inactive state. Phosphate limitation renders the transport system inactive, which activates PhoU and causes phosphorylation of the PhoR protein which in turn phosphorylates PhoB. Phosphorylated PhoB (PhoB-P) is a positive regulator which binds to the promotor region of certain operons. Upon PhoB-P binding, among others, the phoA gene is expressed and alkaline phosphatase (PhoA protein) is synthesised and exported into the periplasm. While active PhoU causes the phosphorylation of PhoR, inactive PhoU is thought to act as a PhoB-P phosphatase which switches off the DNA binding activity of the PhoB protein. A summary of the molecular components involved in the regulation of the phosphate utilisation network is shown in Table 1.

Table 1: Molecular components involved in phosphate regulation.

Abbreviation	Molecular Component
PhoA	Alkaline phosphatase enzyme degrading organic phosphate compounds to inorganic phosphate
Pi	Inorganic phosphate
Po	Organic phosphate
PstSCAB	Transmembrane protein complex, transporter of inorganic phosphate
PhoU	Signal transducer relaying the PstSCAB complex activity
PhoR	Phosphorylatable regulatory protein
PhoB	Phosphorylatable regulatory protein

4.2 Application of xSPN to study case: the phosphate network in enteric bacteria

Qualitative modelling (xPN). The regulatory mechanism illustrated in Figure 9 is implemented in the Petri net shown in Figure 10. The molecular compo-

nents in their different states (e.g. active, inactive, phosphorylated, unphosphorylated) are represented as places (Table 2) and their biochemical reactions or functional interactions are represented as stochastic transitions (Table 3). Additionally, there are deterministically timed transitions (immediate and scheduled transitions) to model the experimental addition of inorganic/organic phosphate. There are different arc types. Standard arcs represent the mass flow. Read arcs and inhibitory arcs are used to model regulatory interactions between proteins through physical protein-protein interaction. Double arcs (a sort-hand notation for two opposite arcs) represent catalytic reactions which may be complex.

The graphical representation of the cytoplasmic membrane and of some proteins of interest is put underneath the Petri net in order to explain the modular structure of the net and to highlight which reactions occur inside or outside the cell, respectively. Note that these graphics are not a functional element of the Petri net as generated in Snoopy. The PstSCAB transmembrane protein complex has a dual function as transporter and as a sensor of inorganic phosphate. When inorganic phosphate is present in the periplasm, the PstSCAB complex is phosphorylated through reaction r7. The phosphorylated form, PstSCAB-P is active in transporting inorganic phosphate through the cytoplasmic membrane into the cytoplasm (r5) where it is used for biosynthetic reactions (r6). The PstSCAB protein allosterically controls the activity of the PhoU protein, modelled by read arcs that control the transitions r9 and r10 representing the

Table 2: Petri net places represent molecular components in their different states.

Place	Molecular component
Pi_PeriPlasm	Inorganic phosphate in the periplasm
Pi_Cytoplasm	Inorganic phosphate in the cytoplasm
Po_PeriPlasm	Organic phosphate in the periplasm
PhoA_Periplasm	PhoA protein in the periplasm
PhoA	PhoA protein in the cytoplasm
PhoA mRNA	PhoA gene mRNA
PstSCAB	Transporter of inorganic phosphate, inactive form
PstSCAB-P	Transporter of inorganic phosphate, phosphorylated, active form
PhoU_inactive	Signal transducer, inactive form
PhoU_active	Signal transducer, active form
PhoR	PhoR protein, dephosphorylated form
PhoR-P	PhoR protein, phosphorylated form
PhoB	PhoB protein, dephosphorylated form
PhoB-P	PhoB protein, phosphorylated form
switch_on, switch_off	Auxiliary places to model the experiments

Table 3: Petri net transitions represent biochemical reactions or functional interactions. The transitions are stochastic if not specified otherwise.

Transition	Reaction or Process
r1	Experimental addition of inorganic phosphate (immediate transition)
r2a, r2b	Switch on/off the experimental addition by r1 (scheduled transitions)
r3	Environmental source of organic phosphate (immediate transition)
r4	Degradation of organic phosphate
r5	Transport of inorganic phosphate into the cytoplasm
r6	Consumption of inorganic phosphate by biosyntheses
r7	Phosphorylation of PstSCAB complex
r8	Dephosphorylation of PstSCAB complex
r9	Activation of PhoU by PstSCAB complex
r10	Deactivation of PhoU by PstSCAB-P complex
r11	Phosphorylation of PhoR by active PhoU
r12	Phosphorylation of PhoB by PhoR-P, involving dephosphorylation of PhoR-P
r13	Dephosphorylation of PhoB-P by inactive PhoU
r14	Transcription of the phoA gene upon binding of PhoB-P to its promoter
r15	Decay of the phoA mRNA
r16	Translation of the phoA mRNA into PhoA protein
r17	Transport of the PhoA protein into the cytoplasm
r18	Denaturation and decay of the periplasmic PhoA protein

activation (r9) and the deactivation (r10) of the PhoU protein, respectively. As long as inorganic phosphate is in the periplasm, the PstSCAB complex is kept in its phosphorylated, transport-active state, modelled by the inhibitory arc that prevents dephosphorylation via transition r8. In its active form, the PhoU protein causes the phosphorylation of PhoR (r11) which in turn phosphorylates PhoB. This direct transfer reaction of the phosphate group between the two protein molecules is considered as a second order reaction and represented by transition r12. PhoB-P then binds to DNA to promote the transcription of several genes including the phoA gene. The complex process of regulator binding and gene transcription is represented by transition r14. Translation of the phoA mRNA into the PhoA protein occurs through transition r16 which also summarizes a complex set of biochemical reactions involving many additional components. The double arrow linking r16 and the phoA mRNA place indicates that from each mRNA molecule many copies of the PhoA protein can be synthesized. The PhoA protein is finally transported into the periplasm (r17) where it

degrades organic phosphate to supply the cell with inorganic phosphate which in turn switches off the signaling cascade and subsequently the biosynthesis of the PhoA protein, as long as sufficient inorganic phosphate enters the cell.

We have already seen how to draw a simple Petri net, so you will not have any trouble in drawing the Petri net for your first case study. Coarse transitions and coarse places may be used to structure large networks into modules in order to improve the readability of the net. Since the Petri net presented for the case study is rather small, we did not use coarse nodes here.

Animation of the xPN. To explore the net behaviour, do the following steps:

1. load the net (File → Open),
2. start the animation (View → Start Animation Mode),
3. run the net in manual or automatic mode.

Specifically you should check the following scenarios:

1. Which reactions are necessary in which (partial) order to fire the transitions Biosynthesis (r6), Decay (r15), or Denaturation&Decay (r18)?
2. Which net behaviour is possible if the inorganic supply is switched on (place switch_on is marked), which net behaviour is triggered if the inorganic supply is switched off?
3. Try to figure out the maximal token numbers you can get on each place!
4. Is it possible to reach a marking where none of the transitions is enabled?
5. Having played with the net for a while, is it always possible to come back to the given initial marking?

Quantitative modelling (xSPN). All stochastic reactions follow mass-action kinetics with all parameters initially set to 0.1. Afterwards, the parameters were adjusted so that the steady state concentration of inorganic phosphate in the periplasm is approximately the same no matter whether the external source is inorganic or organic phosphate, respectively: the parameter of r4 is set to 0.2, and the parameter of r15 to 0.075.

The transition t_off is scheduled to fire at time point 100, and the transition t_on is scheduled to fire at time point 1000.

Stochastic animation of the xSPN. Follow the stochastic token flow in the automatic animation mode. Which reaction sequences do you observe while the inorganic supply is switched on? What is the difference to the reaction and state sequences, which you have observed in the animation of the corresponding xPN? The answer lies in the immediate transitions (here r1), which have now always highest priority.

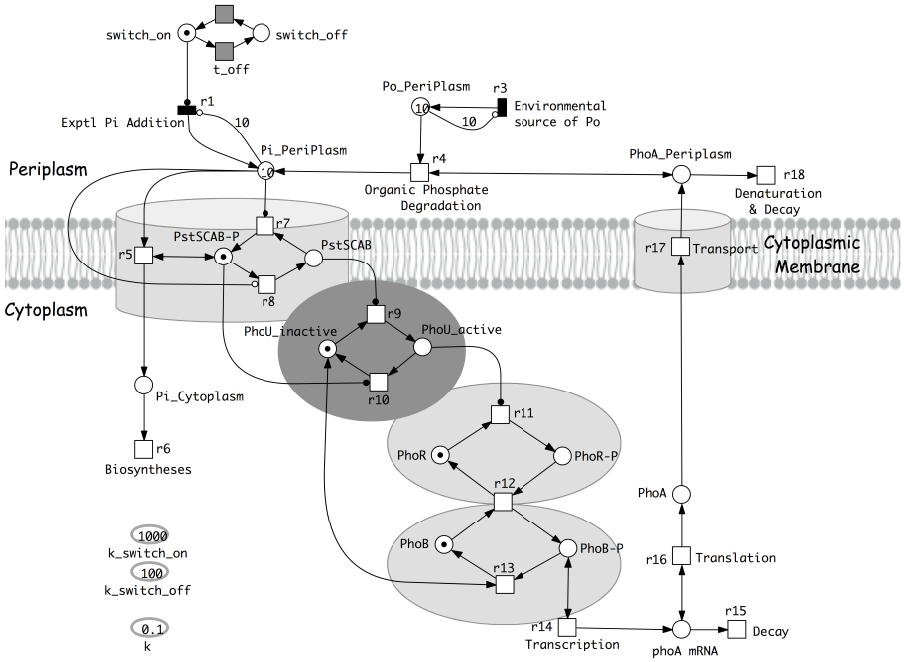

Figure 10: Petri net model of the phosphate regulatory network, along with a schematic representation of the membrane and of some relevant proteins. Biomolecular components and their functional states are represented as places, (bio-)chemical reactions are represented as transitions. Stoichiometric reactions are represented as standard arcs, allosteric interactions of proteins by read arcs and inhibitory arcs. Organic and inorganic phosphate is supplied by the environment of the cell at constant concentration. In the simulations, the constant concentration of these phosphate compounds is obtained by using transitions with immediate firing behaviour (r1, r3) delivering tokens to the corresponding places (Pi_PeriPlasm; Po_PeriPlasm). The firing of r1 and r3 is controlled by inhibitory arcs. These inhibitory arcs shut the transitions off if the pre-defined number of tokens is in the pre-place of the respective transition. Experimental addition and removal of inorganic phosphate to the cell is modelled with the help of the switch_on place which is connected by a read arc to r1. Therefore r1 delivers inorganic phosphate as long as (1) there is a token in the switch_on place and (2) the number of tokens in the Pi_PeriPlasm place is less than ten. The supply with inorganic phosphate may be switched off and on at defined time points by switching the token in the switch_on place with the deterministically timed (scheduled) transitions t_on and t_off. In the simulations, organic phosphate in the periplasm remains constant at all times. The Petri net represents the processes involved in phosphate regulation with different resolution of details. Only part of the net represents molecular processes in terms of interactions of individual molecules. Complex reaction mechanisms, like biosyntheses using inorganic phosphate (r6), transcription of the phoA gene (r14) or translation of the phoA mRNA (r16) are represented as individual transitions that condense multiple individual steps into one single reaction, assuming first order rate constants in the example provided. Note however, that the quantitative behaviour of a transition in the xSPN version of Snoopy can be programmed individually so that even complex and non-linear kinetic mechanisms can be modelled.

Stochastic simulation of the xSPN. In the stochastic simulation, the reaction of each individual molecule is considered in the form of individual tokens that move through the net. When the number of molecules of each biochemical component per cell is known, one can obtain realistic traces of how the number of molecules develops over time. The result may be a time-dependent change in the concentration of the compound if the number of molecules per cell is sufficiently high or the number of molecules may be subjected to stochastic fluctuations over time if it is low. When a stochastic simulation is run many times and the simulation results are averaged, one can approach the result of a deterministic simulation as obtained, for example, by solving a set of ordinary differential equations. Each way of simulation, stochastic as performed here, or deterministic, e.g. by solving ordinary differential equations, may be of particular advantage dependent on the question under consideration. For studying and simulating the behaviour of individual cells, stochastic simulations may be essential.

In order to provide an example of a simulation, we have run the model 50,000 times to show how the phosphate regulation network may behave dynamically. The concentration of the sources, inorganic and organic, were assumed to be constant over time. The system was first allowed to equilibrate in the presence of constant external inorganic phosphate (Figure 11). After the steady state was reached, external inorganic phosphate was removed (switched off) by a step-down to zero concentration and the system was allowed to approach the new steady state. The oscillations in the concentration of the components obtained in the simulation are due to the feed-back loops in the system. Note that the precise dynamic behaviour of the system depends on the ratio of the rate constants which we do not know. Finally, the external source of inorganic phosphate is switched on again and the system equilibrates into its pre-stimulus state. The results of the stochastic simulation are shown as a panel obtained by importing the simulation results into the plot program *Kaleidagraph* (Figure 11) and as screen shots of the simulation results window of Snoopy (Figure 12-13).

You can continue by trying the following simulation scenarios:

1. Which rates influence amplitudes, frequencies and damping of the oscillations caused by feedback regulation?

2. Check how the dynamic system behaviour is influenced by the decay rates of mRNA and PhoA protein.

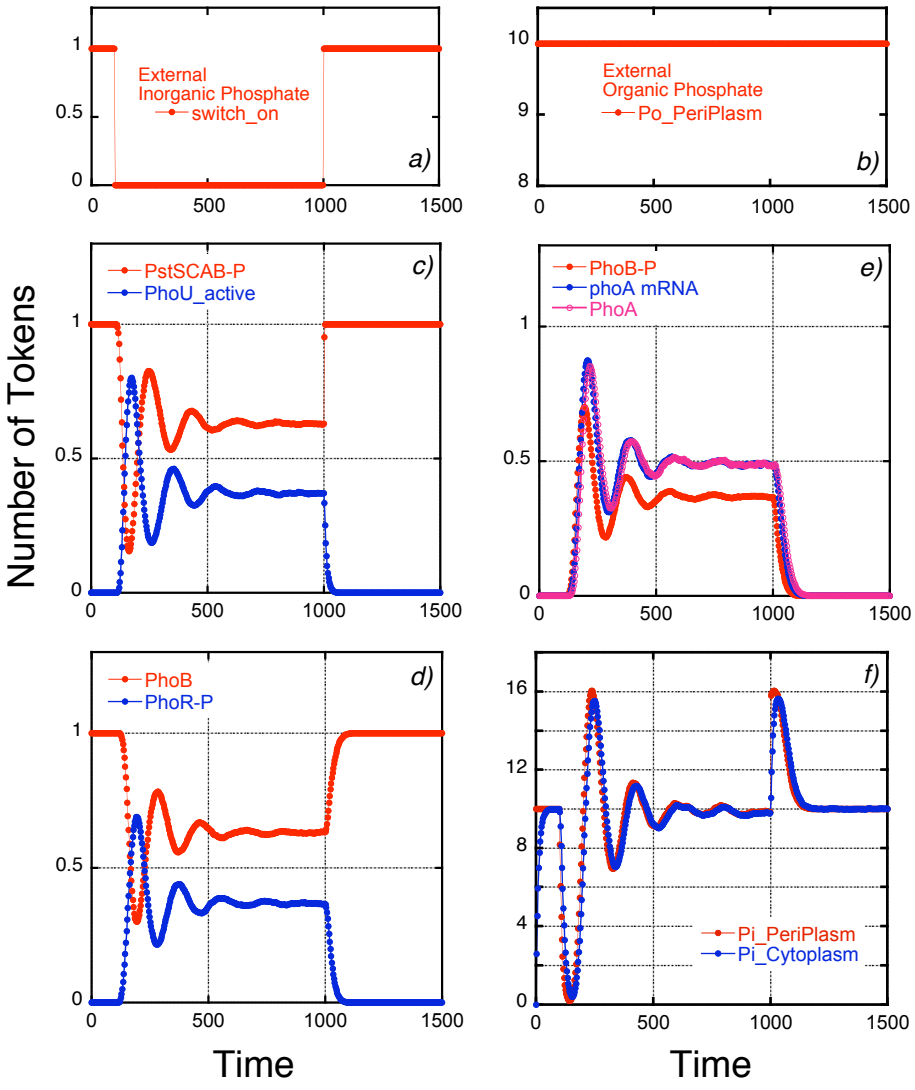

Figure 11: Response of the phosphate regulatory network to step-wise changes in the supply of external inorganic phosphate. Traces were obtained by averaging 50,000 simulation runs. The rate constants of the stochastic transitions are: r4 0.2, r15 0.075, else 0.1. Initial marking as given in Figure 9. Simulation results were exported from Snoopy as csv file and imported into *Kaleidagraph*, see `http://www.kaleidagraph.com/`, for display.

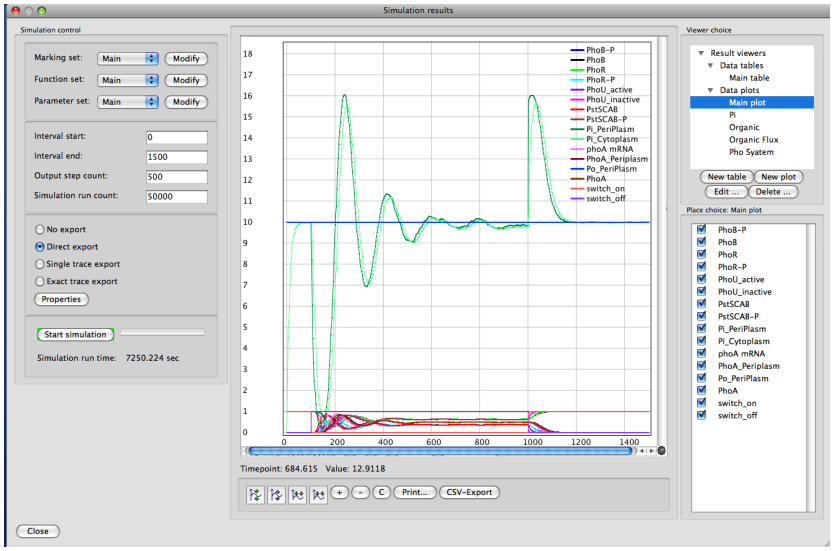

Figure 12: Screen shot of the simulation dialogue window of Snoopy displaying simulation results shown in Figure 10. Simulation run parameters: Interval start: 0; Interval end: 1500; Output step count: 500; Simulation run count: 50,000. Snoopy is a modelling and simulation tool. Petri nets can be graphically edited, parameter lists edited and simulations run and results graphically displayed without leaving the program.

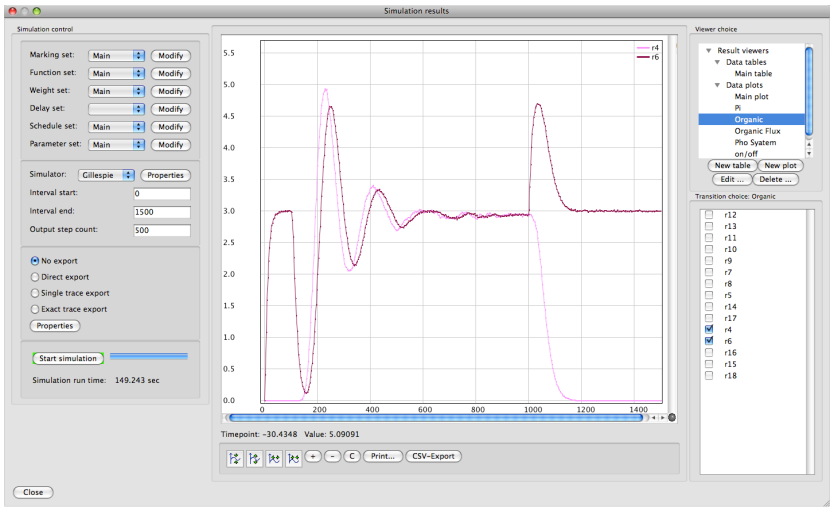

Figure 13: Simulation dialogue showing transition firing times (amount of firing in the last grid interval).

5 Notes

1. In this chapter, we deliberately omitted formal definitions. The interested reader finds them together with accompanying examples of the basic notions of biochemically interpreted Petri nets in [4]; for extended Petri nets see specifically [11], for extended stochastic Petri nets [12], and for general textbooks introducing into Petri nets and stochastic Petri nets the references therein.

2. Petri nets represent an inherently concurrent modelling paradigm. They allow to distinguish precisely between alternative and concurrent (i.e. independent) behaviour. However for analysis purposes, the partial order (true concurrency) semantics is often reduced to the interleaving semantics, where concurrency of reactions is described by all interleaving sequences of these reactions. We get a state transition system (known in the Petri net community as reachability graph or marking graph), which is basically a finite automaton, i.e. a sequential model, and as such does not support the differentiation between alternative and concurrent behaviour. This disadvantage is (partially) compensated by the numerous sophisticated algorithms and advanced data structures which have been developed for their efficient analysis.

   Nevertheless, often it is advisable to preserve all the concurrency information, which a biochemical network model enjoys, e.g., to distinguish the subtle difference between read arcs and two opposite arcs (Figure 7). The partial order semantics yields a behaviour description comprising all (infinite) partial order runs. It is often given as labelled condition/event net. Here, transitions represent events, labelled by the name of the reaction taking place, while places stand for binary conditions, labelled by the name of the species, set or reset by the event, respectively. Partial order runs, which can be read as the unfolding of a net structure, give further insight into the dynamic behaviour of a network, which may not be apparent from the standard net representation. For examples see [4,6], where the behaviour of subnets induced by T-invariants is given as a partial order run.

3. Snoopy supports standard analysis techniques of Petri net theory, such as static and dynamic analyses, by export to various analysis tools, see [5]. Additionally, Snoopy's Petri nets are directly read by the Petri net analyser Charlie [16]. One of the standard static analysis techniques comprises the computation of place and transition invariants, which are often helpful for network validation, see e.g. [4,6,17-19], and (Behre et al., this volume). Sets or multi-sets of nodes, such as place and transition invariants, Parikh vectors, structural deadlocks or traps (see [4] for explanations of these terms), which have been produced with Charlie, are read by Snoopy to visualize the sub-networks induced by them.

   The Petri net of our case study is covered by transition invariants. There are five place invariants, and such conservation laws, which are obvi-

ous in the given case: (switch_on, switch_off), (PstSCAB, PstSCAB-P), (PhoU_inactive, PhoU_active), (PhoR, PhoR-P), and (PhoB, PhoB-P). The following five places are not covered by place invariants: Pi_PeriPlasm, Po_PeriPlasm, Pi_CytoPlasm, PhoA_Periplasm, PhoA, phoA mRNA; their maximal token numbers are determined by the timing constraints.

4. An established analysis technique for special behavioural properties is model checking. It checks the Petri net behaviour against properties formally specified in temporal logics, see (chapter Batt et al., this volume). Charlie provides some standard explicit model checkers, basically for teaching purposes. Snoopy allows also to export the designed Petri nets to quite a number of model checkers, among them the symbolic model checkers BDD-CTL, IDD-CTL [20] and IDD-CSL [21]. Simulation traces generated by Snoopy's (stochastic or continuous) simulation engines can be checked againts PLTLc properties with the Monte Carlo Model Checker MC2 [22]. See [4, 6] for case studies demonstrating a systematic and seamless model checking approach in the qualitative, stochastic and continuous modelling paradigms.

5. A given Petri net may also be read as a continuous Petri net, if it is amenable to continuization and the population semantics allows to consider just the averaged case. Continuous Petri nets define uniquely a system of ordinary differential equations (ODEs) [4], but not vice versa. The paper [23] demonstrates the structured design of ODEs by the step-wise composition of hierarchically structured continuous Petri nets. A family of related Petri net models builds the core of an integrative approach comprising qualitative, stochastic and continuous Petri nets, which is demonstrated by a running example each in [4,6]. It also provides an adequate framework to reason about the behavioural relation of models, sharing structure, but having different kinetics; see, e.g., [24]. For a general outline of BioModel Engineering see [25].

Acknowledgements

Christian Rohr is funded by the International Max Planck Research School for Analysis, Design and Optimization in Chemical and Biochemical Process Engineering Magdeburg.

References

1. Petri, C. A., Reisig, W. (2008) Petri net. Scholarpedia 3(4):6477 `http://www.scholarpedia.org/article/Petri_net`.

2. Marwan, W., Wagler, A., and Weismantel, R. (2009) Petri nets as a framework for the reconstruction and analysis of signal transduction pathways and regulatory networks. J. Nat. Comput., in press [DOI 10.1007/s11047-009-9152-x].

3. Baldan, P., Cocco, N., Marin, A. and Simeoni, M. (2010) Petri nets for modelling metabolic pathways: a survey. J. Nat. Comput., in press [DOI 10.1007/s11047-010-9180-6].

4. Heiner, M., Gilbert, D., and Donaldson, R. (2008) Petri nets in systems and synthetic biology. Lect. Notes Comput. Sci.5016: 215-264.

5. Rohr, C., Marwan, W., and Heiner, M. (2010) Snoopy - a unifying Petri net framework to investigate biomolecular networks; Bioinformatics 26, 7: 974-975.

6. Heiner, M., Donaldson, R., and Gilbert, D. (2010) Petri Nets for Systems Biology. In MS Iyengar (ed.): Symbolic Systems Biology: Theory and Methods, Chapter 3, Jones & Bartlett Publishers, LLC.

7. Petri Nets World: Online Services for the International Petri Nets Community, `http://www.informatik.uni-hamburg.de/TGI/PetriNets/`.

8. SBML Software Summary: `http://sbml.org/SBML_Software_Guide/SBML_Software_Summary`.

9. Bitesize Bio Free Online Bioinformatics tools, `http://bitesizebio.com/2009/01/06/free-online-bioinformatics-tools/`.

10. Systems Biology Software, `http://systems-biology.org/software/`.

11. Tovchigrechko, A. (2008) Efficient symbolic analysis of bounded Petri nets using Interval Decision Diagrams; Ph.D. Thesis, BTU Cottbus, Germany.

12. Heiner, M., Lehrack, S., Gilbert, D., and Marwan, W. (2009) Extended Stochastic Petri Nets for Model-Based Design of Wet-lab Experiments. Lect. Notes Bioinformatics 5750: 138-163.

13. Gillespie, D. T. (1977) Exact stochastic simulation of coupled chemical reactions; J/ Phys. Chem. 81(25): 2340-2361.

14. Neidhardt, F. C., Ingraham, J. L. et al. (1990). Physiology of the Bacterial Cell - A Molecular Approach. Sunderland, Massachusetts, Sinauer Associates. p. 370.

15. Yi-Ju Hsieh, Y.-J. and Wanner, B.L. (2010) Global regulation by the seven-component Pi signaling system. Curr. Opin. Microbiol. 13:198-203.

16. Franzke, A. (2009) Charlie 2.0 - a multi-threaded Petri net analyser. Diploma Thesis, BTU Cottbus, Germany.

17. Heiner, M., and Koch, I. (2004) Petri Net Based System Validation in Systems Biology. Lect. Notes Comput. Sci. 3099: 216-237.

18. Palsson, B.O. (2006) Systems Biology: Properties of Reconstructed Networks. Cambridge University Press.

19. Heiner, M. (2009) Understanding Network Behaviour by Structured Representations of Transition Invariants - A Petri Net Perspective on Systems and Synthetic Biology. In A Condon, D Harel, JN Kok, A Salomaa, E Winfree (eds.) Algorithmic Bioprocesses. Springer, Natural Computing Series, 367-389.

20. Heiner, M., Schwarick, M., and Tovchigrechko, A. (2009) DSSZ-MC - A Tool for Symbolic Analysis of Extended Petri Nets. Lect. Notes Comput. Sci. 5606: 323-332.

21. Schwarick, M., and Heiner, M. (2009) CSL model checking of biochemical networks with Interval Decision Diagrams. Lect. Notes Bioinformatics 5688: 296-312.

22. MC2(PLTLc) - Monte Carlo Model Checker for PLTLc properties, University of Glasgow, `http://www.brc.dcs.gla.ac.uk/software/mc2/`.

23. Breitling, R., Gilbert, D., Heiner, M., and Orton, R. (2008) A structured approach for the engineering of biochemical network models, illustrated for signalling pathways. Brief. Bioinformatics 9: 404-421.

24. Heiner, M., and Sriram, K. (2010) Structural Analysis to Determine the Core of Hypoxia Response Network. PLoS ONE 5(1): e8600.

25. Breitling, R.; Donaldson, R.A.; Gilbert, D.; Heiner, M. (2010) Biomodel Engineering - From Structure to Behavior. Lect. Notes Bioinformatics 5945, 1-12.