

Design Issues for Qualitative Modelling of Biological Cells with Petri Nets

Elzbieta Krepska, Nicola Bonzanni,
Anton Feenstra, Wan Fokkink, Thilo Kielmann,
Henri Bal, and Jaap Heringa

Department of Computer Science, Vrije Universiteit
De Boelelaan 1083, 1081 HV Amsterdam, The Netherlands
ekr@cs.vu.nl, {bonzanni,feenstra,wanf}@few.vu.nl,
{kielmann,bal,heringa}@cs.vu.nl

Abstract. Petri nets are a widely used formalism to qualitatively model concurrent systems such as a biological cell. We present techniques for modelling biological processes as Petri nets for further analyses and in-silico experiments. Instead of extending the formalism with „colours” or rates, as is most often done, we focus on preserving the simplicity of the formalism and developing an execution semantics which resembles biology – we apply a principle of maximal parallelism and introduce the novel concept of bounded execution with overshooting. A number of modelling solutions are demonstrated using the example of the well-studied *C. elegans* vulval development process. To date our model is still under development, but first results, based on Monte Carlo simulations, are promising.

1 Introduction

Systems biology [1] is a relatively new field of study which focuses on interactions within and between biological systems. The knowledge about those systems typically is presented in the form of descriptive text, illustrated with diagrams that are often beset with arrows, colourful components and comments. It is not only difficult to locate a particular piece of information, but also to understand it, as there are often unknowns and ambiguities in the description. This problem is inevitable when representing dynamical and concurrent processes in a living cell as flat diagrams.

Furthermore, the amount of biological knowledge is increasing, and has reached the point where the help of machines is becoming indispensable. What is more, *in vitro* (laboratory) experiments tend to be expensive and slow and are often infeasible, whereas *in silico* (computer) experiments could be cheaper, faster and better reproducible. Realistic executable models of biological systems can be used for predictions, preparation and elimination of unnecessary, dangerous or unethical laboratory experiments. This approach would also be applicable, for example, in drug design and testing [2].

Therefore, one of the major questions systems biology is currently trying to answer, is how to represent biological knowledge concisely, unambiguously, without omissions, with well-localised gaps and in a machine-processable way.

There are two typical approaches to cell modelling: the first uses systems of differential equations and the second uses stochastic simulations, see for example [3–5]. They are both quantitative and highly dependent on kinetic constants or reaction rates which are approximate (Sackmann et al. [6] claim that often only 30-50% of data is known). Differential equations work only in cases when many molecules of each protein species are present, and stochastic simulation works only for „well-stirred” chemical soups. However, biological cells are far from being „well-stirred” and examples abound where small amounts, or even single molecules, are crucial to biological processes. In contrast, cell interaction data are available in large amounts [6,7]. Clearly, there is a need for qualitative rather than quantitative modelling.

Petri nets are a well-established technique for modelling concurrent systems. They are simple and powerful in expressing biological knowledge, e.g. binding, signalling, concurrency, nondeterminism, timing. They are extensible and have intuitive visualisation.

As a model organism we picked a well-described and relatively uncomplicated worm, *C. elegans*, and the first phase of its well-studied vulval development process [8]. Additionally, this process has discrete output, which makes it easier to verify correctness of a model.

We started with a well-known basic approach to modelling biological systems using Petri nets, described, for example, in [9]. Throughout the development of our model, we set the following three objectives. (1) Resemble biology. (2) Keep the model homogeneous and simple. (3) Comply to the standard Petri net theory. (4) Keep the model qualitative while trying to reproduce the results of laboratory experiments. These goals quite often conflict and it was necessary to find a consensus between them. As a result, we place our work in-between qualitative and quantitative approaches.

In this paper we present our experiences in modelling the *C. elegans* vulval development process using Petri nets. Instead of extending the formalism with „colours” or rates, as is typically done, we focused on preserving the simplicity of the formalism and changing the execution semantics. Rather than the traditional interleaving execution, we use the maximal parallelism principle and introduce a novel concept of bounded execution with overshooting.

Our network is considerably larger than those typically published in papers on modelling of biological cells using Petri nets – it has about 300 places and 300 transitions and approximately 950 edges. Our model is still under development, but first results, based on Monte Carlo simulations, are promising. So far we are able to correctly simulate 46 out of 48 experiments from [10].

The paper is organised as follows. Section 2 sketches the biological process that we are modelling. Section 3 introduces general Petri net theory. Section 4 presents biological modelling methods and execution semantics. Section 5 describes good practices and techniques in modelling using Petri nets. Section

6 identifies problems we encountered while working on the model. Section 7 overviews related work. Section 8 discusses current results. Section 9 gives conclusions and ideas about continuing this work in future.

2 *C. elegans* Vulval Development

C. elegans is a well-studied nematode (type of worm), living in soil and about 1 mm long. It consists of about 1000 cells which are all numbered, and their destinies at all stages of the development of the worm are well-described. Six of those cells are called Vulval Precursor Cells and numbered P3.p, P4.p, P5.p, P6.p, P7.p and P8.p. Those cells form a line. They participate in the process of vulval development, depicted in Fig. 1.

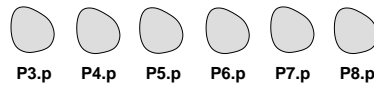


Fig. 1. Cells taking part in vulval development process.

Instructions providing all the information necessary for a living cell to grow and function are present in the form of DNA molecules. The DNA contains a number of genes which are templates for the production of proteins, the cell building blocks. We say that a gene is *wild-type (on)* when the protein is normally produced. It is *knocked-out (off)* when it is not produced, and it is *over-expressed* when it is produced in excess. Proteins present in a cell may have two forms, *active* and *inactive*. Typically, the protein must first be activated to participate in reactions. Proteins may also influence other proteins or themselves. For instance, processes of increasing or decreasing the production of a protein are called, respectively, *up-* or *downregulation*. Certain proteins and their interactions are grouped as *pathways*, „packages” with a coarse-grained function in a living cell.

The vulval development process consists of two interacting pathways. Their significant genes are *lin-12* and *mpk-1*. In this process, all cells have to arrive at a decision which fate to choose: 1, 2 or 3. The cells that choose first fate will divide and create the actual vulva. The cells that choose the second fate become supporters of the vulva. The cells that choose the third fate will fuse with the hypodermis, the „skin” of the worm. Before the process starts, all cells are the same. Provided that all genes are produced normally, the process works as follows (see Fig. 2). A special cell, called the Anchor Cell (AC), comes near cell P6.p and induces a signal leading to first fate in P6.p. Then P6.p sends signals to its sides (lateral signals) resulting in the second fate taken by P5.p and P7.p. Remaining cells do not get any signal and eventually choose the third fate. In case the genes were not produced normally, many other possible patterns of fates can occur. Note that this is an extremely simplistic description. This and much more information about *C. elegans* can be found in the Wormbook [8].

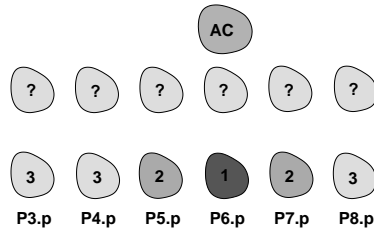


Fig. 2. First stage of *C. elegans* vulval development process. The resulting wild-type fate pattern is (3 3 2 1 2 3).

In this paper we use the following convention: genes are written in lowercase and corresponding proteins in uppercase. The suffix PRO, for example in LIN-12 PRO, denotes the production reaction and suffix DOWN denotes down-regulation. A protein name containing cell number, for example LIN-12/P6.p, denotes protein concentration within this particular cell.

3 Petri Nets

Petri nets [11, 12] are a formalism geared towards modelling and analysis of concurrent systems. A *Place-Transition (PT) Petri net* is a quadruple $(\mathcal{P}, \mathcal{T}, \mathcal{F}, m)$, where \mathcal{P} is a set of places and \mathcal{T} a set of transitions. \mathcal{F} describes weights of arcs which can connect places with transitions or transitions with places. Each place holds zero or more tokens, which represent flow of control through this place. The number of tokens in all places is called a marking of the network, $m: \mathcal{P} \rightarrow \mathbb{N}$, and represents its state.

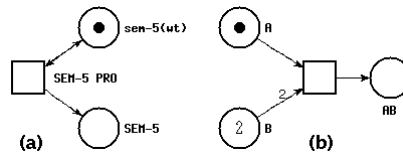


Fig. 3. (a) Production of a protein SEM-5 in the presence of coding gene `sem-5(wt)`. (b) Synthesis of a complex AB from one A and two B.

An example of a Petri net is given in Fig. 3(b). Places are depicted as circles, transitions as rectangles, arcs as arrows and tokens as dots. If no \mathcal{F} -value is given, it is 1.

Incoming arcs at a transition represent its requirements and outgoing arcs represent token production. A transition is enabled when all of the requirements are met. An enabled transition can fire: consume all required tokens and produce new tokens. The interleaving execution semantics of a Petri net is defined as

follows: in each step select randomly one enabled transition, fire it, repeat.¹ If there are no more enabled transitions left, the network deadlocks. This semantics describes totally asynchronous behaviour, i.e. all possible interleavings of transitions.

4 Modelling Biology

Below we present a method to represent biological knowledge as Petri nets. We illustrate it with three examples of typical biological modelling problems. We also introduce the design of our model and explain the adaptation of the Petri net standard that we have developed.

4.1 Basic Translation

The translation of places and transitions into biological entities is straightforward. Places represent genes, protein species and complexes. However, we have encountered many cases when we had to represent a single entity with various characteristics as multiple places. For example, to differentiate between active and inactive LIN-12 proteins, we used two places `LIN-12` and `LIN-12 ACT`. Transitions represent reactions or transfer of a signal. Arcs represent reaction substrates and products. Firing of a transition is execution of a reaction: consuming substrates and creating products.

4.2 Gene Expression

Figure 3(a) depicts a typical example of production of proteins from a gene. In this case the „reaction” `SEM-5 PRO` produces proteins `SEM-5` when the wild-type gene `sem-5(wt)` is present. When the gene is not present, the „reaction” does not take place.

4.3 Downregulation through Production Suppression

Figure 4 depicts downregulation of a protein `LIN-12` through suppressing the expression of a gene `lin-12(wt)`. Normally, if `MPK-1` is not present, the reaction `LIN-12 PRO` is enabled and produces protein `LIN-12`, similarly to the previous example. However, when `MPK-1` is present, the reaction `LIN-12 DOWN` is enabled and has 0.5 chance of firing compared to `LIN-12 PRO`, so the production of `LIN-12` will halve.

¹ In this paper, by random selection we mean that the choice is non-deterministic and all possible choices have equal probability to be selected.

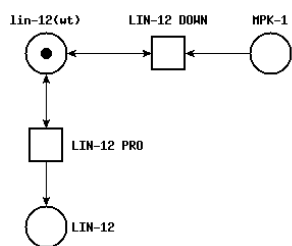


Fig. 4. MPK-1 downregulates LIN-12 by suppression of *lin-12(wt)* gene.

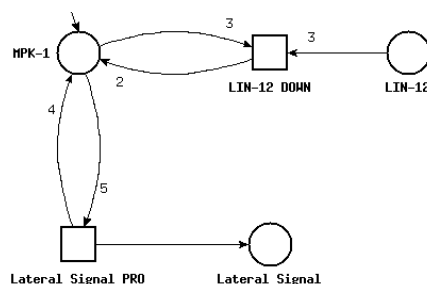


Fig. 5. MPK-1 degradation of LIN-12 through endocytosis and upregulation of a lateral signal by MPK-1.

4.4 Downregulation through Product Removal

Figure 5 illustrates two processes. The first process is another model for decreasing the level of LIN-12 by MPK-1 – the degradation through endocytosis. Here the produced LIN-12 is removed, while in the previous example its production was reduced. Normally, if MPK-1 is not present, the level of LIN-12 does not change. However, when levels of LIN-12 and MPK-1 are high enough (≥ 3), the reaction LIN-12DOWN can execute and decrease the concentration of LIN-12. The second process is sending a lateral signal. When a high (≥ 5) level of concentration of MPK-1 is present, a lateral signal can be initiated.

4.5 Concentration Levels

The number of tokens in our model does *not* represent directly the number of molecules of proteins. In the Petri net model, we interpret it in two ways. In case of a gene: 0–not present and 1–present. And in case of a protein: 0–not present and 1-2, 3-4 and 5-6 – low, medium and high concentrations. The rationale behind this approach is to abstract away from unknown absolute molecule concentration levels, in order to keep the model qualitative.

4.6 Maximal Parallelism

The interleaving semantics of Petri nets describes asynchronous behaviour, cf. Sect. 3. However, this is not realistic for biological cells, where all reactions can happen in parallel. Thus, in our model we use the maximal parallelism execution semantics, which can be summarised informally as *execute greedily as many transitions as possible in one step*. In this semantics, a step \mathcal{S} is a multi-set of transitions, i.e. a transition can occur multiple times in \mathcal{S} . A maximally parallel step is a step that leaves no enabled transitions in the net. A maximally parallel step is enabled if executed transitions are enabled with the required multiplicity.

In case more than one maximally parallel step is possible, the one to execute in a simulation is selected randomly. Figure 6 illustrates an example network and possible maximally parallel steps. The maximal parallelism principle for Petri nets is described for example in [13].

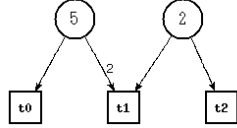


Fig. 6. A network with exactly three possible maximally parallel steps: $\{t_0 \times 5, t_2 \times 2\}$, $\{t_0 \times 3, t_1, t_2\}$, $\{t_0 \times 1, t_1 \times 2\}$.

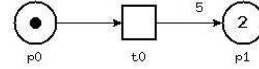


Fig. 7. A bounded network with a bound 6. Transition t_0 cannot fire because it would create 7 tokens in p_1 and maximally 6 tokens are allowed. Saturation in place p_1 is not possible.

Efficient Implementation of Maximal Parallelism. In the implementation of the maximally parallel execution of a Petri net, the key notion is a *conflict*. We say that two transitions t and t' are *directly in conflict* if they have a common parent. They are *in conflict* if there exists a sequence of transitions $t = t_1, t_2, \dots, t_k = t'$ such that each pair (t_i, t_{i+1}) is directly in conflict. For example, in Fig. 6 all three transitions are in conflict.

The computational cost of generating the next maximally parallel step is at least $\Omega(\exp(|\mathcal{T}|))$, as it involves verifying all subsets of \mathcal{T} in the worst case, i.e. when all transitions are in conflict. Fortunately, in our experience so far, in nature the networks tend to be „sparse” and, as a result, this computational cost has not been a bottleneck.

Algorithm 1 nextMaxStep(\mathcal{N} : PetriNet)

- 1: $\mathcal{S} = \emptyset$
 - 2: $\mathcal{D} = \text{generateConflicts}(\mathcal{T})$
 - 3: **for all** $d \in \mathcal{D}$ **do**
 - 4: $A_d = \text{backtrackAllMaxSteps}(d)$
 - 5: $s_d = \text{selectRandom}(A_d)$
 - 6: $\mathcal{S} = \mathcal{S} \cup s_d$
 - 7: **end for**
 - 8: **return** \mathcal{S}
-

Algorithm 1 shows how the next maximally parallel step is created in our implementation. The algorithm proceeds in three stages. In the first stage, the

set of all transitions, \mathcal{T} , is divided into a set of disjoint subsets of transitions in conflict, \mathcal{D} . Existence of enabled transitions in conflict implies that there exists more than one possible step.

In the second stage, for each subset of transitions in conflict, $d \in \mathcal{D}$, all possible maximally parallel steps are generated using a simple backtracking technique. The resulting step, s_d , is then chosen randomly. Note that s_d is a multi-set of transitions that belong to d . In the third stage, the global step is combined out of „local” steps: $\mathcal{S} = \bigcup_{d \in \mathcal{D}} s_d$. This can be done because transitions that belong to different subsets in \mathcal{D} are not in conflict. In our implementation, the first stage of partitioning the network into disjoint transitions sets in conflict is executed once, stored and used for creation of a sequence of maximally parallel steps.

4.7 Bounded Execution

In our first attempt, the execution of the network was not bounded, as it was designed to produce the output into infinity. However, it turned out that numerous nodes reached a high number of tokens, e.g. 800, which was in practice impossible to downregulate or degrade. Furthermore, such a high production of certain proteins is not realistic, as in nature the cell would saturate with the product, and the reaction would slow down or stop. Therefore we introduced a bounded execution with a chosen bound $N = 6$, which means that a place cannot contain more than N tokens.

4.8 Bounded Execution with Overshooting

Bounded execution suffers from two problems. First, it is not possible to execute partial reactions, and therefore sometimes saturation cannot be reached – see the example of a network in Fig. 7.

Second, the partition of transitions used for fast execution (see Sect. 4.6) does not work anymore, as there are more conflicts and the entire network becomes inter-dependent. To overcome both problems, we implemented bounded execution with overshooting. Each transition can overshoot maximally once. In other words, a reaction can produce if all products have at least one free token slot in the output place. Consequently, given a place with n incoming arcs with weights w_1, \dots, w_n , the maximal number of tokens in \mathbf{p} is $(N-1) + \sum_{i=1}^n w_i$.

5 Modelling Solutions

While creating the model of *C. elegans* vulval development process, we developed a number of good practices and techniques for modelling biological knowledge as Petri nets, without extending the formalism. Below we present several of these techniques.

5.1 Reactions Ordering

The fragment of the network depicted in Fig. 5 represented both downregulation of LIN-12 by MPK-1 and sending a lateral signal (see Sect. 4.4). Let us assume that only one token at a time flows into MPK-1. Note that the downregulation of LIN-12 will take place *before* sending a lateral signal. This is because whenever new tokens appear in MPK-1, they first reach the level 3. Only when LIN-12 level drops to < 3 and blocks LIN-12 DOWN, can MPK-1 reach 5 and initiate a lateral signal. The vulval development of *C. elegans* is initiated by the Anchor Cell, as we described in Sect. 2. The P6.p cell receives a strong signal from AC, while its adjacent cells P5.p and P7.p receive weaker signals and an additional lateral signals from P6.p, see Fig. 8.

5.2 Modelling Signal Strength

At first we modelled a stronger signal naively – by sending more tokens: 3 instead of 1, as illustrated in Fig. 9. However, this did not work, as our network, unbounded at that time, would just accumulate high concentrations of certain proteins, and, as a result, there was no difference between weak and strong signals. We increased the probability of production of LIN-3/P6.p with a technique depicted in Fig. 10: using multiple transitions for LIN-3/P6.p. Now the probability of executing LIN-3/P6.p is three times higher than the probability of executing LIN-3/P5.p or LIN-3/P7.p. After applying this change, the pathway behaved like a fast „pipeline”. Note that this way, spatial vicinity has been translated in the model as the probability of execution.

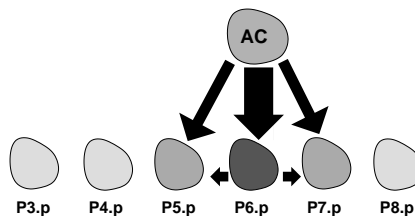


Fig. 8. Anchor Cell initiates the vulval development process by sending stronger and weaker signals.

5.3 Modelling Signal Strength Directly

The method presented in Fig. 10 worked so well that it gave us an idea of directly implementing a transition attribute called **strength**, see Fig. 11. This way we gained clarity in the graphical representation and, additionally, speedup in the maximally parallel step computation, resulting from the reduction of the number of subsets of transitions.

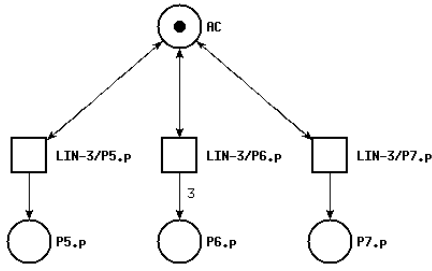


Fig. 9. A simple model for the Anchor Cell signal. Signal strength is modelled as higher production.

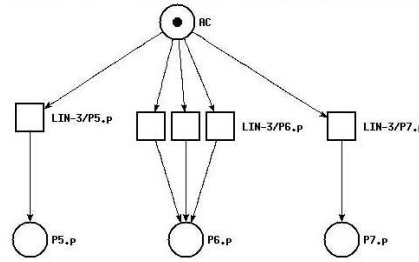


Fig. 10. A method to increase the probability of transition execution.

The execution semantics needed to be adjusted: instead of randomly selecting a maximally parallel step out of all generated possible steps, we compute a probability that a maximally parallel step should be selected. This probability is the sum of strengths of transitions in the step, normalised by the sum of strengths of all transitions. For example, in Fig. 11, the probability that the LIN-3/P6.p transition should be fired is $\frac{3}{1+3+1} = 0.6$. The realisation of this idea was straightforward to implement.

5.4 Modelling Signal Slowdown

After we switched to modelling with saturation, we observed the following phenomenon: the strong AC signalling pathway in the cell P6.p (see Sect. 5.2) would stall, as most of the places were oversaturated. To overcome this problem we used a method depicted in Fig. 12. In the first step, the transition LIN-3 PRO executes. It results in production of LIN-3 down the pathway, but also puts one token in the cycle. In the next two steps, the signal is not generated, but the token continues cycling. After three steps, the signal is back in AC and the process will repeat. Note that the cycle works as a delay or a buffer: the LIN-3 signal will be created every three steps.

By using this method, we reduced the number of tokens sent within the signal, so that the pathway is able to handle the „bandwidth”.

5.5 Good Practice: One Token Moves

In our first attempts to prototype the system, we frequently manually adjusted weights on the arcs. However, this introduced uncertainties and guessed numbers into the model. Thus, we decided to abstract away from them and move just one token wherever possible. (Note, however, that we kept the reaction requirements. If a requirement is k tokens, then we move back $k - 1$ tokens.)

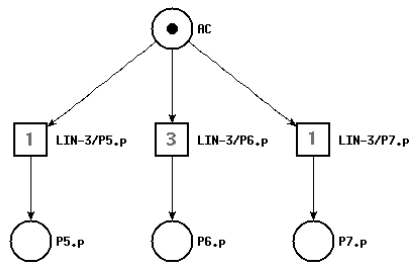


Fig. 11. Transition strength. LIN-3/P6.p is three times stronger than LIN-3/P5.p.

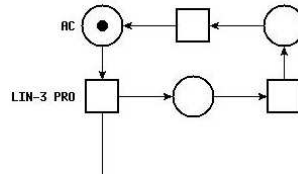


Fig. 12. A method to model signal slowdown. In the first step, the AC signal fires. Afterwards it signals every three steps.

5.6 Unbounded Places

We found that in one important case the places should not be saturated, namely in the environment of the cell. Note, however, that the presence of unbounded places makes the analysis of the net, for example steady state determination, more difficult.

5.7 Model Correctness

Deciding whether a model is correct is difficult and varies from case to case. The *C. elegans* vulval development process has the advantage of discrete output, and for verification we have used 48 experiments published by Fisher et al. [10].

Since the semantics of execution is probabilistic, it is not enough to run one simulation of the network. To verify its behaviour, we ran a Monte Carlo simulation – average multiple runs, typically 3000. We used the DAS-3 cluster [14] for that. For each simulation, we set up genes appropriately and execute the network for a fixed number of steps.

Our observations show that protein concentrations reach „steady” levels after a fixed number of steps. We use these levels to determine cell fates and check whether the output is correct. Our experience so far is that single runs of the simulation are good representatives of average behaviour, that is to say standard deviation of network behaviour is small.

5.8 Debugging Simulations

In our attempts to create a correct model of *C. elegans* vulval development, we found two approaches to debugging Petri net simulation particularly useful. First are the plots of protein concentration levels during a simulation, averaged over various windows. Second, we use a backtracking technique. During simulation we generate statistics about the average numbers of tokens in places,

firings of transitions and executed steps. The statistics aid us to manually go „backward” from output places, and trace which transitions were blocked, and which places were oversaturated or depleted. This method typically reveals a problem in the model, for example too strong downregulation or an incorrect concentration level, and gives good hints as to why the execution did not work as it was supposed to.

6 Problems Identified

While extensively working with large Petri nets we noted several disadvantages of using this formalism for modelling biological cells.

6.1 Drawing tool

A major practical problem when creating the biological model is the lack of a graphical tool supporting handling of large networks (by large we mean more than 100 nodes). The most desirable features of such a tool are:

- Collective operations such as labelling, moving, hiding.
- Clever zooming, for instance zooming into a subset of the network.
- Adding new types of labels to nodes. For example, we needed to add label *strength* to transitions, and we ended up using non-portable ways of doing that.
- Support for the Petri Net Markup Language [15]. Using this standard turned out very convenient, as we were able to switch drawing tools multiple times.
- Advanced visualisation of execution, allowing, for example, to view only the selected nodes and switching of the viewing modes.

TINA [16] is the tool that we had best experiences with.

6.2 Modularity, compositionality

Another disadvantage of using Petri nets for biological modelling is that they offer little support for modularity and compositionality. Our model consists of six identical cells, and each cell consists of two pathways. Applying a modular approach would be very useful in this case. Also, with the view of possibly including several development stages of *C. elegans* in the future, applying a modular approach would be indispensable.

6.3 Synchrony vs. asynchrony

Biology is neither totally synchronous nor totally asynchronous. For example, a chemical reaction, i.e. creation of products and consumption of substrates, can be thought of as immediate, synchronous. By contrast, sending a signal to another cell is a typical asynchronous operation, like sending a letter. And last but not least, time and quantities play an important role, meaning that

communications in different parts of a biological system tend to occur at the same rate. A formalism to model biological systems should be able to express such dependencies.

Petri net interleaving semantics models totally asynchronous behaviour (cf. Sect. 3). We switched to a maximally parallel semantics (cf. Sect. 4.6), which progresses in lockstep, i.e. it models synchronous behaviour. In our experience, this way we achieve a much closer resemblance to biological cells.

However, we lost the possibility to model asynchronous communication. An important open question is: how to introduce asynchrony into a lockstep? Bounded asynchrony is discussed in [17], but has a disadvantage of incorporating a global timer, and [18] focuses on coordination orchestration, which seems heavy-weight for cell modelling.

7 Related Work

Petri nets were introduced by C. A. Petri in his dissertation in 1962 [11]. An article by Murata [12] contains a good introduction to general Petri net theory. Reddy was the first to use Petri nets for modelling of biochemical reactions. He presented a basic translation of a biological narrative into a Petri net and an analysis with invariants [9]. Since that time, numerous papers applying Petri nets to biological modelling were published, for example [6, 19–21]. All of these papers use interleaving execution semantics and do not model saturation. Resulting nets are also much smaller than ours. A different approach take Genrich et al. [2], who create their network by querying biological databases, such as KEGG [22]. The amount of data is huge and the resulting network is at first gigantic, but then simplified to only 8 medium-size pathways.

Many extensions to Petri nets have been developed and used in modelling of biological systems: Coloured Petri Nets (tokens have colours) [2], Stochastic Petri Nets (transitions have rates) [21, 23], Timed Petri Nets (transitions have delays) [20], Hybrid Petri Nets (places can be continuous) [24], Functional Petri Nets (transitions have functions) [25], and Hybrid Functional Petri Nets (everything mixed) [26]. There is also a recurring approach of representing a biological pathway in logic and translating it automatically into a Petri net [6, 27, 28]. The reader may find recent survey papers concerning modelling of biological systems in [7, 26, 29].

The most common way of validation of Petri net models of biological systems are simulations and comparing the evolution of concentration levels or reached steady state to data in the literature, e.g. [20]. More advanced structural analysis involves determining invariants or checking for deadlocks [9, 19] and model checking, e.g. [28].

Besides Petri nets, there are numerous formalisms to model and analyse biological pathways, such as process calculi, Boolean networks, statecharts, REO. The reader might consult [18, 26, 30]. It is still not clear though which of those techniques are best suited for biological modelling.

8 Current Results

At the time of writing this article, our model is still under development. Our network is considerably larger than those typically published in papers on modelling of biological cells using Petri nets – it has about 300 places and 300 transitions and approximately 950 edges. We localised a lot of the necessary biological information in the literature, for example [8, 31, 32].

First results look promising. We developed an implementation of the execution method, and are able to correctly repeat 46 out of 48 experiments from [10], using Monte Carlo simulations.

9 Conclusions

In this paper we presented the technique of modelling biological cells as distributed systems represented by Petri nets. We base our findings on the *C. elegans* vulval development model that we have built. We propose to model with the basic Petri net formalism, which is easy to understand and visualise, and to adapt its execution semantics to resemble biology. Our execution semantics is based on the maximal parallelism principle (execute in parallel as much as possible) and bounded execution with overshooting (saturation).

Petri net models that we use are both qualitative and quantitative. Qualitative – because network structure represents static knowledge about interactions within a cell. Quantitative – because places and transitions can contain or require multiple tokens and we exploit this to express different signal strengths and concentration levels. For instance, we can express that one protein is produced five times faster than another protein, or that a concentration level for downregulation has been reached.

There is currently a need for executable biological models, but no consensus on what is the best biological modelling language [30]. In our experience so far, quantitative Petri nets with maximal parallelism and bounded execution could be a good choice. They very naturally feature concurrency, nondeterminism, synchronisation and visualisation, they allow to model systems that resemble the actual behaviour of biological cells, and a Monte Carlo simulator can be implemented efficiently.

In the near future we intend to work on issues such as modularity, synchrony versus asynchrony, system robustness, steady states, and model checking.

References

- [1] Alon, U.: An Introduction to Systems Biology: Design Principles of Biological Circuits. Chapman & Hall/CRC (2006)
- [2] Genrich, H., Küffner, R., Voss, K.: Executable Petri net models for the analysis of metabolic pathways. International Journal on Software Tools for Technology Transfer (STTT) **3**(4) (2001) 394–404
- [3] E-Cell: <http://www.e-cell.org>

- [4] Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry* **81**(25) (1977) 2340–2361
- [5] Srivastava, R., You, L., Summers, J., Yin, J.: Stochastic vs. deterministic modeling of intracellular viral kinetics. *Journal of Theoretical Biology* **218**(3) (2002) 309–21
- [6] Sackmann, A., Heiner, M., Koch, I.: Application of Petri net based analysis techniques to signal transduction pathways. *BMC Bioinformatics* **7** (2006) 482+
- [7] Chaouiya, C.: Petri net modelling of biological networks. *Briefings in Bioinformatics* **8**(4) (2007) 210–219
- [8] Wormbook: <http://www.wormbook.org>
- [9] Reddy, V., Mavrovouniotis, M., Liebman, M.: Qualitative analysis of biochemical reaction system. *Computers in Biology and Medicine* **26**(1) (1996) 9–24
- [10] Fisher, J., Piterman, N., Hajnal, A., Henzinger, T.A.: Predictive modeling of signaling crosstalk during *C. elegans* vulval development. *PLoS Computational Biology* **3**(5) (2007) e92+
- [11] Petri, C.A.: *Kommunikation mit Automaten*. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2 (1962)
- [12] Murata, T.: Petri nets: Properties, analysis and applications. *Proceedings of the IEEE* **77**(4) (1989) 541–580
- [13] Kleijn, J.H., Koutny, M., Rozenberg, G.: Towards a Petri net semantics for membrane systems. In: *Membrane Computing*. Volume 3850 of *Lecture Notes in Computer Science.*, Springer (2005) 292–309
- [14] DAS-3, D.A.S.: <http://www.cs.vu.nl/das3>
- [15] Language, P.N.M.: <http://www.informatik.hu-berlin.de/top/pnml>
- [16] TINA: <http://www.laas.fr/tina>
- [17] Fisher, J., Henzinger, T.A., Mateescu, M., Piterman, N.: Bounded asynchrony: A biologically inspired notion of concurrency. Technical report, MTC (2007)
- [18] Clarke, D., Costa, D., Arbab, F.: Modelling coordination in biological systems. In: *International Symposium on Leveraging Applications of Formal Methods (ISoLA)*. Volume 4313 of *Lecture Notes in Computer Science.*, Springer (2004) 9–25
- [19] Koch, I., Junker, B.H., Heiner, M.: Application of Petri net theory for modelling and validation of the sucrose breakdown pathway in the potato tuber. *Bioinformatics* **21**(7) (2005) 1219–1226
- [20] Li, C., Ge, Q.W., Nakata, M., Matsuno, H., Miyando, S.: Modelling and simulation of signal transductions in an apoptosis pathway by using timed Petri nets. *Journal of Biosciences* **32** (2007) 113–127
- [21] Goss, P., Peccoud, J.: Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. *Proceedings of the National Academy of Sciences of the United States of America (PNAS)* **95**(12) (1998) 6750–6755
- [22] KEGG: <http://www.genome.jp/kegg>
- [23] Srivastava, R., Peterson, M.S., Bentley, W.E.: Stochastic kinetic analysis of the *Escherichia coli* stress circuit using σ -32-targeted antisense. *Biotechnology and Bioengineering* **75**(1) (2001) 120–129
- [24] Matsuno, H., Doi, A., Nagasaki, M., Miyano, S.: Hybrid Petri net representation of gene regulatory network. *Proceedings of Pacific Symposium on Biocomputing* **5** (2000) 341–352
- [25] Hofestädt, R., Thelen, S.: Quantitative modeling of biochemical networks. In *Silico Biology* **6** (1998) 39–53
- [26] Matsuno, H., Li, C., Miyano, S.: Petri net based descriptions for systematic understanding of biological pathways. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science* **E89-A**(11) (2006) 3166–74

- [27] Simao, E., Remy, E., Thieffry, D., Chaouiya, C.: Qualitative modelling of regulated metabolic pathways: application to the tryptophan biosynthesis in *E. Coli*. *Bioinformatics* **21**(2) (2005) 190–196
- [28] Steggles, L., Banks, R., Wipat, A.: Modelling and analysing genetic networks: From Boolean networks to Petri nets. In: International Conference on Computational Methods in Systems Biology (CMSB). Volume 4210 of Lecture Notes in Bioinformatics., Springer (2006) 127–141
- [29] Peleg, M., Rubin, D., Altman, R.B.: Using Petri net tools to study properties and dynamics of biological systems. *Journal of the American Medical Informatics Association (JAMIA)* **12**(2) (2005) 181–199
- [30] Fisher, J., Henzinger, T.A.: Executable cell biology. *Nature Biotechnology* **25**(11) (2007) 1239–1249
- [31] Yoo, A.S., Bais, C., Greenwald, I.: Crosstalk between the EGFR and LIN-12/Notch pathways in *C. elegans* vulval development. *Science* **303**(5658) (2004) 663–666
- [32] Shaye, D.D., Greenwald, I.: Endocytosis-mediated downregulation of LIN-12/Notch upon Ras activation in *Caenorhabditis elegans*. *Nature* **420**(6916) (2002) 686–690