

An Advanced Environment for Hybrid Modeling of Biological Systems Based on Modelica

Sabrina Proß and Bernhard Bachmann

University of Applied Sciences Bielefeld, Am Stadtholz 24, 33609 Bielefeld, Germany,
<http://www.fh-bielefeld.de/ammo>

Summary

Biological systems are often very complex so that an appropriate formalism is needed for modeling their behavior. Hybrid Petri Nets, consisting of time-discrete Petri Net elements as well as continuous ones, have proven to be ideal for this task. Therefore, a new Petri Net library was implemented based on the object-oriented modeling language Modelica which allows the modeling of discrete, stochastic and continuous Petri Net elements by differential, algebraic and discrete equations. An appropriate Modelica-tool performs the hybrid simulation with discrete events and the solution of continuous differential equations. A special sub-library contains so-called wrappers for specific reactions to simplify the modeling process.

The Modelica-models can be connected to Simulink-models for parameter optimization, sensitivity analysis and stochastic simulation in Matlab.

The present paper illustrates the implementation of the Petri Net component models, their usage within the modeling process and the coupling between the Modelica-tool Dymola and Matlab/Simulink. The application is demonstrated by modeling the metabolism of Chinese Hamster Ovary Cells.

1 Introduction

The modeling of biological systems demand often a combination of continuous and discrete processes, a differential equation system sole is often not sufficient. Examples are gene regulation and processes where the organism switches from substance production to consumption or vice versa when special environmental conditions occur. The structure of the differential equation for a substance depends on specific conditions and changes within time. This kind of process takes place within the metabolism of Chinese Hamster Ovary Cells (CHO-Cells) which switches from lactate and ammonium production to consumption after a specific change of environmental conditions. The concrete mechanism is part of section 3.

The realization of the CHO-model requires an appropriate formalism. Hybrid Petri Nets consisting time-discrete as well as continuous Petri Net elements have proven to be ideal. The biological pools, e.g. metabolites, genes, proteomes and signals are represented by Places and the reactions between them can be modeled by Transitions. This transfer of biological systems to Petri Nets was first introduced by Reddy [1] and an introduction in the basic Petri Net concepts is given in section 1.1.

For the modeling and simulation of hybrid Petri Nets a new library was developed with the object-oriented modeling language Modelica [2], whereby the Petri Net component models consist of differential, algebraic and discrete equations. The Modelica language and the available simulation tools are introduced in section 1.2 and the concrete implementation and usage of these Petri Net components are part of section 2. A short introduction in other hybrid Petri Net simulators is given in section 1.3.

1.1 Petri Nets

The Petri Net formalism for graphical modeling of concurrent and nondeterministic processes, was first introduced by Carl Adam Petri in 1962 [3]. A Petri Net is mathematically a directed, 2-colored and bipartite graph. The property 2-colored implies the division in two unique node sets which are called *Transitions* and *Places* and only Places can be connected to Transitions or Transitions to Places according to the bipartite attribute. The Places are represented graphically by circles and Transitions by rectangles. Places model states for example of objects or conditions while Transitions model the changes of these states for example activities or events.

Definition 1: A *Petri Net* is the tuple (P, T, F, G, f) of a finite set of Places $P = \{P_1, P_2, \dots, P_p\}$, a finite set of Transitions $T = \{T_1, T_2, \dots, T_t\}$, where $P \cap T = \emptyset$, an edge set $F \subseteq (P \times T)$, an edge set $G \subseteq (T \times P)$ and an edge weighting function $f: (F \cup G) \rightarrow \mathbb{N}^n$, where $f_{p,t}$ is the weighting of the edge from Place $p \in P$ to Transition $t \in T$ and $f_{t,p}$ is the weighting of the edge from t to p .

Every Place can contain an integer number of *Tokens*. These Tokens are represented graphically by little, black dots or numbers inside the Places. A concrete determination of the Token number of a Place is called state of the Place and a concrete determination of the Token numbers of every Place is called the state of the Petri Net. A Transition can fire these Tokens if all Places in its previous area (*previous Places*) have at least as much Tokens as the respective edge weighting. It is said that the Transition is *ready-to-fire*. A Transition that is ready-to-fire, *fires* by removing as much Tokens as the respective edge weighting determines from all its previous Places and by adding as much Tokens as the edge weighting specifies to all Places in its past area (*past Places*).

Figure 1 shows on the top an example of a Petri Net where the Transitions $T1$ and $T2$ are ready-to-fire and the others not. The Petri Net at the bottom displays the new state after firing the Transitions $T1$ and $T2$.

In the last years, the basic Petri Net concept, described above and defined in Definition 1 has been more and more extended in order to model different kind of applications (e.g. biological systems). The first extension is that every Place in a Petri Net can have a lower and upper limit of Tokens. These Petri Nets are called *Petri Nets with capacity*.

Not only the Places can have lower and upper limits. It is also thinkable that the edges from Places to Transitions are bounded with threshold (lower bound) and inhibition (upper bound) values. In the biological sense these parameters are useful to model reactions that are inhibited or activated by a specific substance concentration.

The edge weighting function can be modified for modeling dynamic edge weightings that depend on the actual Token number of several Places. Thus, not only positive integers can be written at the edges but also the names of the Places. This Petri Net extension is called *self-modified Petri Net* and was first introduced by Valk [4].

These self-modified Petri Nets can be further expanded to *functional Petri Nets*. The edge weightings can be functions depending on the Token numbers of several Places [5].

For the simulation of a Petri Net it is necessary to associate time with its behavior. One possibility to do this is that every Transition gets a *delay*. A delay is the time period that the respective state change takes.

This concept can be also modified by random delays, i.e. the fixed values are replaced by random numbers that change at every activation point in time. The delays are exponential

distributed random numbers, whereby the characteristic parameter λ can depend functionally on the Token numbers of several places. This modification is called *stochastic Petri Net* (see e.g. [6], [7]).

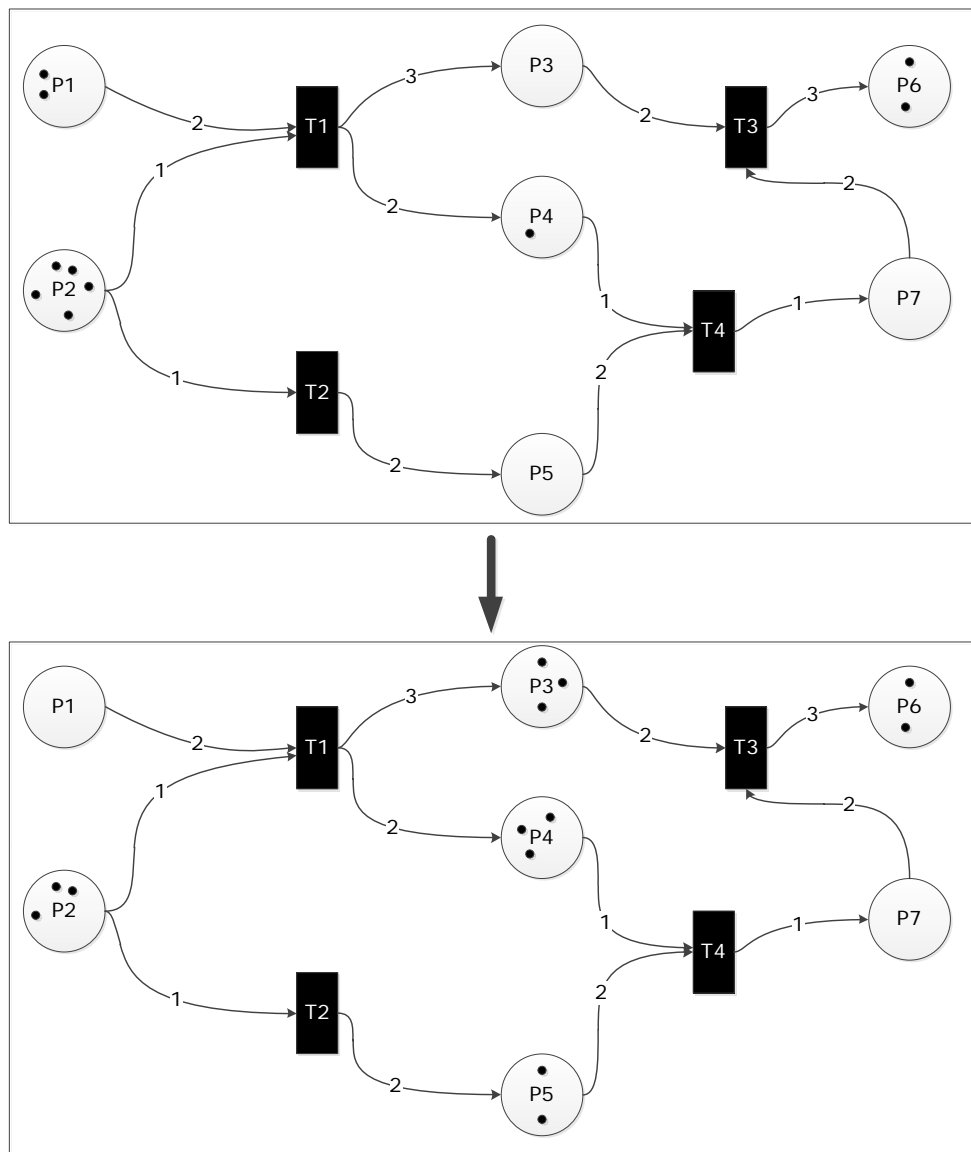


Figure 1: Petri Net example, top: the Transitions $T1$ and $T2$ are ready to fire, bottom: the new state of the Petri Net after firing Transition $T1$ and $T2$.

Biochemical reactions occur in the most of the cases continuously. In order to model these reactions, the discrete Petri Net concept has to be transferred to a continuous one [8]. The most serious difference between discrete and continuous Petri Nets is that the Token numbers are real and that the Transitions fire continuously. A function is assigned to every edge of a continuous Petri Net depending on the Token numbers of several Places just like the functional Petri Nets. These functions specify the speed of the firing process and are the right sides of differential equations.

Definition 2: A *continuous Petri Net* is the tuple (P, T, F, G, f) of a finite set of Places $P = \{P_1, P_2, \dots, P_p\}$, a finite set of Transitions $T = \{T_1, T_2, \dots, T_t\}$, where $P \cap T = \emptyset$, an edge set $F \subseteq (P \times T)$, an edge set $G \subseteq (T \times P)$ and an edge weighting function $g: (F \cup G) \rightarrow \mathcal{H}$ which assigns every edge a function $h \in \mathcal{H}: \mathbb{R}_+^p \rightarrow \mathbb{R}_+$ depending on

a subset of Place states, whereby $h_{p,t}$ is the function assigned to the edge $f_{p,t}$ from a Place $p \in P$ to a Transition $t \in T$ and $h_{t,p}$ is the function assigned to $f_{t,p}$. The *state change* of the Place $p \in P$ can be recalculated by the following differential equation

$$\frac{dM(p)}{dt} = - \sum_{t \in TF_{past}(p)} f_{p,t} + \sum_{t \in TF_{pre}(p)} f_{t,p}$$

where $TF_{past}(p) \subseteq T$ is the set of all firing past Transitions and $TF_{pre}(p) \subseteq T$ is the set of all firing previous Transitions.

A continuous Petri Net is an ordinary differential equation system whose structure can change within time.

Additionally, the modeling of biological systems demands often a combination of discrete processes and continuous ones. One example is gene regulation or the metabolic reactions of substances which switch from production to consumption or vice versa when specific conditions appear. Hybrid Petri Nets which contain discrete as well as continuous Petri Nets elements accomplish this [9].

Definition 3: A *hybrid Petri Net* is the tuple $(PD, PC, TD, TC, F, G, fd, fc, d, M_0)$, whereby

- $PD = \{PD_1, PD_2, \dots, PD_{pd}\}$ is a finite set of discrete Places
- $PC = \{PC_1, PC_2, \dots, PC_{pc}\}$ is a finite set of continuous Places
- $TD = \{TD_1, TD_2, \dots, TD_{td}\}$ is a finite set of discrete Transitions
- $TC = \{TC_1, TC_2, \dots, TC_{tc}\}$ is a finite set of continuous Transitions
- with $PD \cap TD = \emptyset$, $PC \cap TC = \emptyset$, $PD \cap PC = \emptyset$, $TD \cap TC = \emptyset$, $PD \cap TC = \emptyset$ and $PC \cap TD = \emptyset$
- $F \subseteq (PD \times TD) \cup (PC \times TC) \cup (PC \times TD)$ is the set of edges
- $G \subseteq (TD \times PD) \cup (TC \times PC) \cup (TD \times PC)$ is the set of edges
- $fd: (F \cup G) \setminus \{(PC \times TC) \cup (TC \times PC)\} \rightarrow \mathcal{G}$ is an edge weighting function which assigns every edge a function $g \in \mathcal{G}: \mathbb{N}^{pd} \rightarrow \mathbb{N}$ depending on a subset of discrete Place states
- $fc: (F \cup G) \setminus \{(PD \times TD) \cup (TD \times PD) \cup (PC \times TD) \cup (TD \times PC)\} \rightarrow \mathcal{H}$ is an edge weighting function which assigns every edge a function $h \in \mathcal{H}: \mathbb{R}_+^{pd+pc} \rightarrow \mathbb{R}_+$ depending on a subset of discrete and continuous Place states
- $d: TD \rightarrow \mathbb{R}_+^{td}$ is a delay function
- $M_0: PD \rightarrow \mathbb{N}_+^{pd}$, $PC \rightarrow \mathbb{R}_+^{pc}$ is the initial state.

The following connections are allowed within hybrid Petri Nets

- ✓ discrete Place \rightarrow discrete Transition
- ✓ discrete Transition \rightarrow discrete Place
- ✓ continuous Place \rightarrow continuous Transition
- ✓ continuous Transition \rightarrow continuous Place
- ✓ continuous Place \rightarrow discrete Transition
- ✓ discrete Transition \rightarrow continuous Place

Not allowed are the connections

- ✗ discrete Place \rightarrow continuous Transition
- ✗ continuous Transition \rightarrow discrete Place

The change-over from a discrete to a continuous process and vice versa is described by the examples in Figure 2.



Figure 2: Hybrid Petri Net examples, left: a discrete Transition (delay=2) is connected to a continuous Place, right: a continuous Place is connected to a discrete Transition (delay=7).

Left: If Transition $TD1$ is ready to fire, it waits two time units before one Token is removed from $PD1$. In these two time units five Tokens are added to $PC1$ continuously, i.e. its state change is described by the differential equation

$$\frac{dM(PC1)}{dt} = \frac{f_{TD1,PC1}}{d_{TD1}} = \frac{5}{2}.$$

Thus, the Token number of $PC1$ increases linearly with the slope $\frac{5}{2}$. Right: If Transition $TD2$ is ready to fire, it fires by removing three Tokens in seven time units continuously from $PC2$

$$\frac{dM(PC2)}{dt} = -\frac{f_{PC2,TD2}}{d_{TD2}} = -\frac{3}{7}$$

and by adding nine Tokens to $PD2$ after seven time units. Thus, the Token number of $PC2$ is in the range of the delay a line with the slope $-\frac{3}{7}$. The change-over interpretation is also displayed in Figure 3.

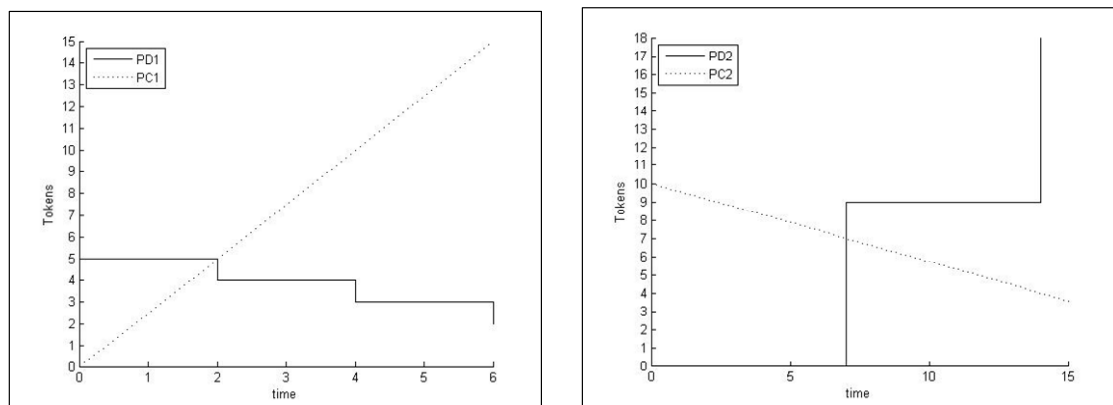


Figure 3: The change-over interpretation from discrete to continuous (left) and vice versa (right) of the examples in Figure 2.

1.2 Modelica

The basic Petri Net concept and its extensions described above have been implemented with the aid of the object-oriented modeling language Modelica [2]. Modelica is a free language and developed by the Modelica Association since 1996. Furthermore, several libraries for simulating technical systems have been developed over the last decade. Since the year 2000

Modelica is used successfully in industry, which is documented in the proceedings of many Modelica conferences and journals. The development of the language and libraries is ongoing and driven by several European projects (EUROSYSLIB, MODELISAR, and OPENPROD). The Modelica-models can be described by differential, algebraic and discrete equations on the textual level and by schematics on the graphical level. The schematics consist of connected components which are defined by other components or on the lowest level by equations in the Modelica syntax. The components have connectors which describe the interaction between them. By drawing lines between the connectors of the components these components are connected and can interact. In this manner a model is constructed. The components can be structured in libraries, called packages, which make hierarchical modeling possible. For the graphical modeling and the simulation an appropriate environment is needed. Here are some examples of free and commercial Modelica simulation environments.

Free:

- OpenModelica (Compiler, www.openModelica.org) and SimForge (graphical environment, <https://trac.ws.dei.polimi.it/simforge/>)
- JModelica (www.jmodelica.org)

Commercial:

- Dymola from Dassault Systemes (www.3ds.com/products/catia/portfolio/dymola)
- MapleSimTM from Maplesoft (www.maplesoft.com/products/maplesim/index.aspx)
- MathModelica System Designer from MathCore (www.mathcore.com/products/mathmodelica)

For the examples in this paper the commercial tool Dymola – Dynamic Modeling Laboratory – version 7.4 has been used.

The first Petri Net Library in Modelica was developed by Mosterman and others [10]. Herewith the modeling of a special case of discrete Petri Nets is possible where all Places have one Token as the maximum capacity and zero as minimum capacity. Additionally, all edges have the weighting one. No time was associated with their behavior. An external signal is associated with the Transitions which can enable or disable them. This Petri Net Library was further developed by Fabricius [11]. The extensions are that the Places can contain an integer number of Tokens and can have minimal and maximal capacities. The Transitions can be timed or stochastic. The first Petri Net concept of Mosterman and other was also further developed by Otter and others [12] to so-called StateGraphs. The StateGraph Library is a part of the Modelica standard library.

1.3 Petri Net Simulation Tools

There are already a lot of Petri Net simulators available. An overview can be found in [13]. But only a few of them are applicable for the hybrid Petri Net simulation. The most common tool is the Cell Illustrator which is a hybrid Petri Net simulator especially for biological systems ([14], [15]). It is commercial and bases on the tools Visual Object Net of the TU Ilmenau [16] and Genomic Object Net of the Yamaguchi University [17]. The Cell Illustrator has a good user interface and is easy to handle but the simulation is like a black box. No definitions of the hybrid simulation are available and no choice of the solver for the ODEs is possible, additionally it is not known which solver is really used. Furthermore, it is not possible to make neither a Monte Carlo simulation to get knowledge about the parameter sensitivity nor a parameter optimization.

To close this gap a Petri Net Library was developed with the object-oriented modeling language Modelica. The code behind the several Petri Net component models is visible, changeable and expendable. Additionally, this library will be free available. Dymola 7.4 consists 16 different ODE-solver (e.g. Dassl, Euler, Rkfix2) which are useable for the hybrid Petri Net simulation. Moreover, the Modelica-models can be connected to Simulink-models, in this manner all the Matlab power can be used for the parameter optimization, sensitivity analysis and stochastic simulation.

2 Petri Net Library in Modelica

The Petri Net Library described in this paper bases on the previous ones developed in Modelica ([10], [11], [12]). The improvements are [18]:

- Discrete Petri Nets
 - The edges can have integer or functional weightings depending on the Token numbers of several Places
 - The edges can have integer bounds (threshold and inhibition values)
 - If a Place has a bottleneck, the connected Transitions are enabled randomly with different probabilities (see Figure 9)
- Continuous Petri Nets
 - Transfer of the discrete Petri Net concept to a continuous one
 - The edges can have functional weightings depending on the Token numbers of several Places
 - The Places can have minimum and maximum capacities and the edges can have bounds (threshold and inhibition values)
- Hybrid Petri Nets
 - Combination of discrete and continuous Petri Net elements to hybrid Petri Nets

The Petri Net library is structured in five sub-libraries: Discrete, Continuous, Stochastic, Reactions and Global. Additionally, there are packages for Interfaces, Constants, Functions and Blocks which are used within the component models (see Figure 4). The individual developed models can be stored in the sub-package Models.

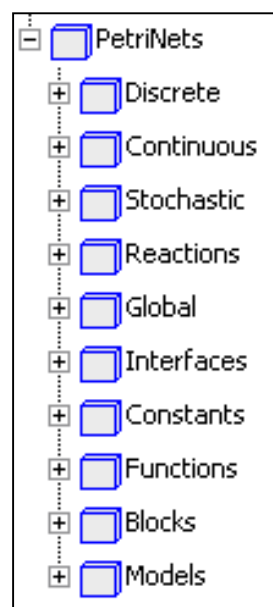


Figure 4: Structure of the Petri Net library.

The Petri Net elements of the library are represented by the icons in Figure 5. The little white and red triangles at the Petri Net icons are the connectors. Transitions and Places can be connected graphically by drawing a line from a white to a red triangle. Not allowed are connections between discrete Places and continuous Transitions (cp. Definition 3). Figure 6 shows an example of a hybrid Petri Net modeled by the Petri Net library in Dymola.

A connector defines the variables that are part of the communication interface. Two components that are connected via their connectors can exchange the variables that are defined within their connectors. Thus, they can interact with each other. The Petri Net Library has four different connectors `FirePortIn`, `SetPortIn`, `FirePortOut` and `SetPortOut`. Both output connectors, `FirePortOut` and `SetPortOut`, are represented graphically by white triangles and both input connectors, `FirePortIn` and `SetPortIn`, by red triangles (see Figure 7). The `FirePortOut`-connector can be only connected to the `FirePortIn`-connector and the `SetPortOut`-connector can be only connected to the `SetPortIn`-connector. In this manner a Place can be only connected to a Transition or a Transition to a Place, like it is demanded. A connection can be graphically constructed by drawing a line from the connector of one component to a connector of another component or textually by the equation `connect(connector1, connector2)`. The connector itself can be implemented in Modelica with the aid of the following construct:

```
connector <name>
  <data type> variable1;
  <data type> variable2;
  ...
end <name>;
```

The connectors can be found in the Interfaces sub-library.


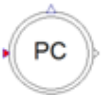



	Discrete Place
	Continuous Place
	Discrete Transition
	Stochastic Transition
	Continuous Transition

Figure 5: Icons of the Petri Net library.

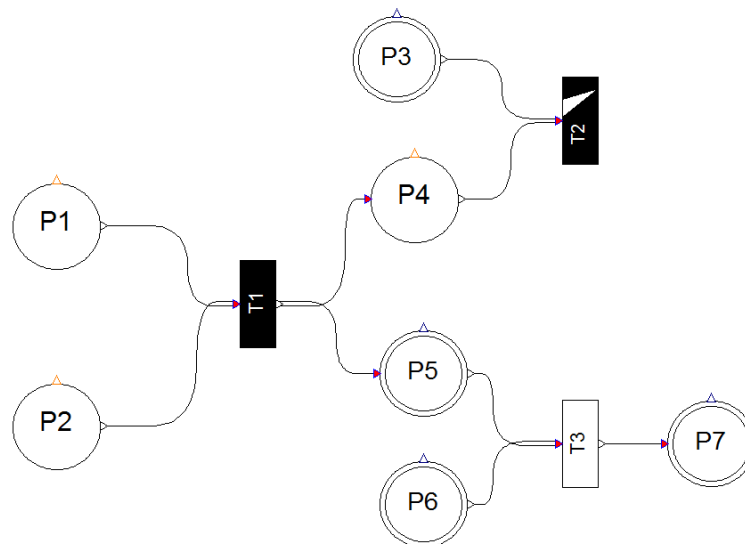


Figure 6: Hybrid Petri Net modeled by the Petri Net library in Dymola where *P1*, *P2*, *P4* and *T1* are discrete, *T2* is stochastic and *P3*, *P5*, *P6*, *P7* and *T3* are continuous Petri Net elements.

The Token number of a Place is one example of a connector variable. It is calculated in the Place but also needed in the connected Transitions to check whether they are ready to fire or not. The keywords `input` and `output` determine if the equations of these variables are in the component where the connector is used or if they are in a connected component. The equation for Token number is provided in the Place, i.e. it is an output variable of the `FirePortOut`-connector and an input of the `FirePortIn`-connector. The keywords `input` and `output` in the connectors guarantee that the Place and Transition models are balanced, i.e. they have the same number of equations than variables (see balanced model definition in [2]).

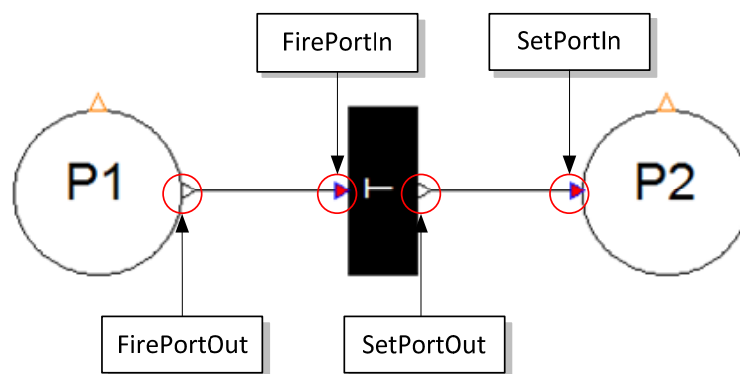


Figure 7: Connectors of the Petri Net library.

2.1 The Place

The parameters of a Place can be entered to a property dialog in Dymola or textually. The property dialog appears by double clicking on the Place icon (see Figure 8).

Table 1 contains the parameters which can be set in all Places (discrete and continuous) and Table 2 those that are only part of discrete Places. The parameter `weightingOut` is explained in Figure 9.

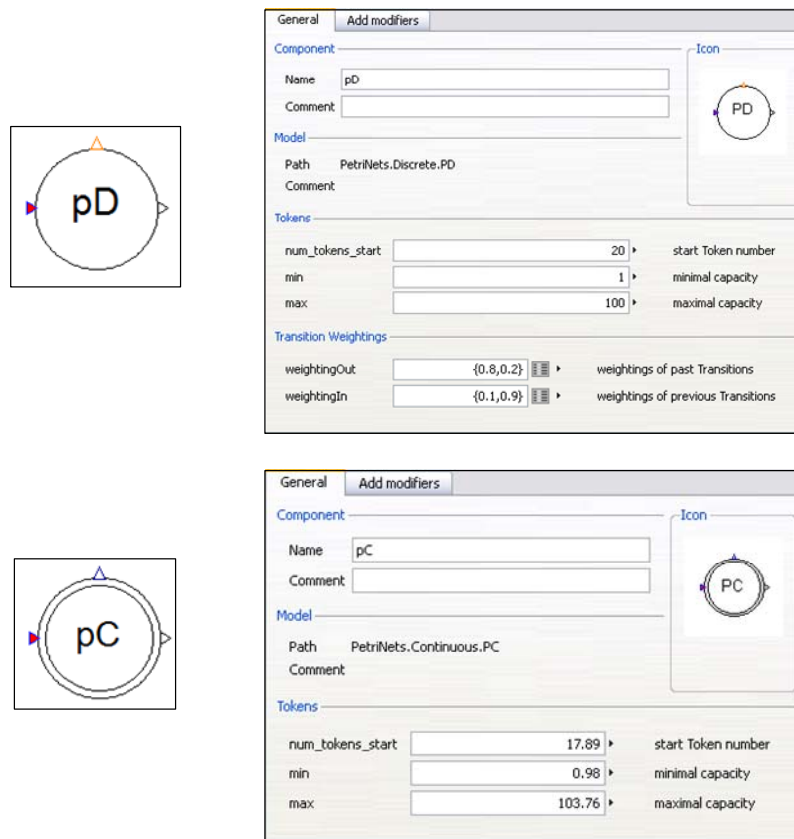


Figure 8: Property dialogs of discrete and continuous Places.

Table 1: Parameters of both Places (discrete and continuous).

Identifier	Description	Default value
num_tokens_start	The number of Tokens that the Place contains at the beginning of the simulation. In the discrete case integer and in the continuous case real numbers can be entered.	zero
min	The minimum number of Tokens that the Place must always contain. In the discrete case integer and in the continuous case real numbers can be entered.	zero
max	The maximum number of Tokens that the Place can contain. In the discrete case integer and in the continuous case real numbers can be entered.	infinite

Table 2: Parameters only of a discrete Place.

Identifier	Description	Default value
weightingOut	Weighting of the Transitions in the past area of a Place. If a Place has not enough Tokens to enable all connected Transitions, a random decision must be made, whereby the respective Transition is chosen with the entered probability. The sum of all weightings has to be one (see Figure 9).	$1/n_{\text{Out}}$ n_{Out} = number of connected past Transitions
weightingIn	Weighting of the Transitions in the previous area of a Place. If a Place cannot gain Tokens from all connected Transitions in its previous area due to its maximum value, a random decision must be made, whereby the respective Transition is chosen with the entered probability. The sum of all weightings has to be one.	$1/n_{\text{In}}$ n_{In} = number of connected previous Transitions

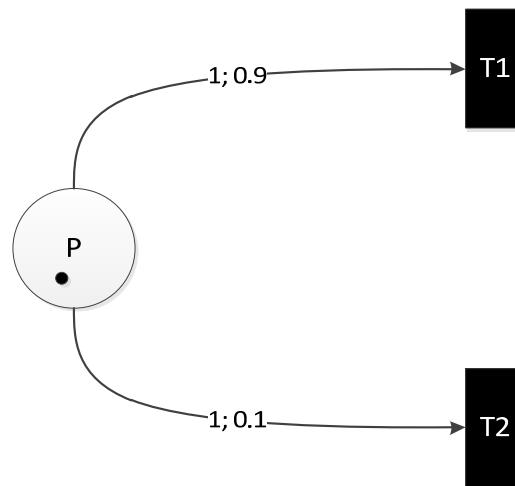


Figure 9: Petri Net example for the weightingOut-parameter: The Place *P* has one Token not enough for firing simultaneously in *T1* and *T2* since both edge weightings are one. The Place has a bottleneck and a random decision is applied, whereby *T1* is chosen with the probability 0.9 and *T2* with the probability 0.1. The sum of all weightings of a Place must be equal to one.

The Place parameters can be also entered textually by adding the names of the parameters and their values within brackets after the component type:

PetriNets.Discrete.PD	pD	(num_tokens_start=20, min=1, max=100, weightingOut={0.8,0.2}, weightingIn={0.1,0.9});
PetriNets.Continuous.PC	pC	(num_tokens_start=17.89, min=0.98, max=103.76);
Component Type	Component Name	Parameter Values

2.1.1 Implementation of the discrete Place

The Token number is determined in the Place model by calculating two sums. One is the sum of all Tokens that are removed from the Place at a certain point in time (*sumOut*) and the other is the sum of all Tokens that are added to the Place at a certain point in time (*sumIn*). The Token number has only to be recalculated when one or more connected Transitions fire i.e. one of the sums or both are greater than zero. This is realized by a discrete equation in Modelica, the *when*-equation. The equation within the *when* statement is only active when the corresponding condition, here the Boolean variable *tokeninout*, becomes true. At this point in time an event is triggered and the discrete equation for the Token number *t* becomes active and the Token number *t* is recalculated by the pre-value of *t*, the value immediately before the event, and the sums *sumIn* and *sumOut*.

```

tokeninout = sumIn > 0 or sumOut > 0;
when tokeninout then
  t=pre(t) + sumIn - sumOut;
end when;

```

After every Transition firing, it has to be checked if the Place is empty or full, i.e. if their minimum or their maximum capacity is reached. The actual state is then reported to the

connected Transitions via the connector variables `outstate` and `instate` (see Figure 10). The connected Transitions can decide with them if they can fire or not.

```
outstate = not pre(empty) and not fire;
instate = pre(full) or set;
```

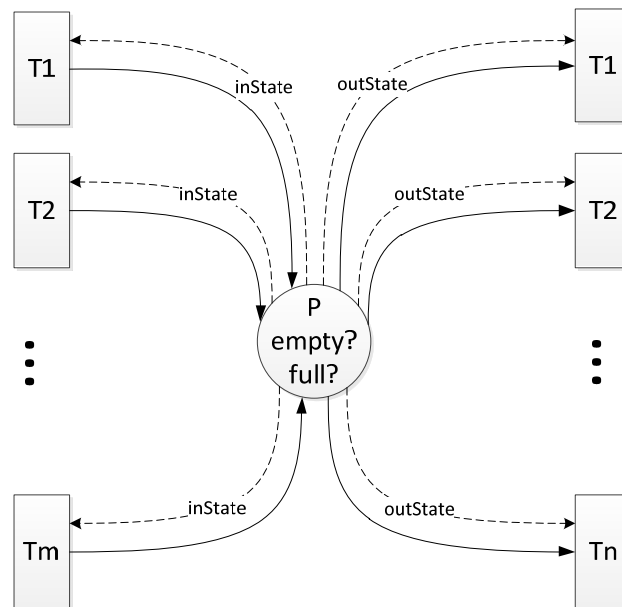


Figure 10: The connector variables `inState` and `outState` of a Place are reported to the connected Transitions.

2.1.2 Implementation of the continuous Place

The Token number of a continuous Place is calculated by a differential equation in contrast to the discrete when-equation of the discrete Place model. In the most of the cases this is the following differential equation, whereby `der` is the keyword for the differential operator with respect to time in Modelica

```
der(t) = sumIn - sumOut;
```

But also some special cases have to be trapped to guarantee an error-free simulation e.g. what happens when the Place has zero Tokens and `sumOut` is greater than `sumIn`. Besides, the variables `full` and `empty` are calculated within the Place model. With the aid of these variables the state of the continuous Place can be determined and reported to the connected Transitions, like in the discrete case (see Figure 10).

2.2 The Transition

The parameters of a Transition can be entered to the Dymola property dialog which appears by double clicking on the Transition icon (see Figure 11).



Figure 11: Property dialogs of discrete, stochastic and continuous Transitions.

The parameters settable in all Transition models (discrete, stochastic and continuous) are summarized in Table 3, the parameters only part of the discrete Transition contains Table 4 and those that can be only set in stochastic Transitions are in Table 5.

Table 3: Parameters of all Transitions (discrete, stochastic and continuous).

Identifier	Description	Default value
sub	Weightings of the edges from the Places in the previous area to the Transition. In the discrete and stochastic case integers and functions can be entered and in the continuous one real numbers and functions are allowed. With the “.t”-notation one can access the Tokens of a Place for the edge weightings, e.g. <code>sub={2*P1.t}</code> .	1
add	Weightings of the edges from the Transition to the Places in its past area. In the discrete and stochastic case integer numbers and functions can be entered and in the continuous one real numbers and functions are allowed. With the “.t”-notation one can access the tokens of a Place for the edge weightings, e.g. <code>add={2*P1.t}</code> .	1
inhibition	Upper bounds of the edges from the Places in the previous area to the Transition. Integers can be entered in the discrete and stochastic case and real ones in the continuous case.	infinite
threshold	Lower bounds of the edges from the Places in the previous area to the Transition. Integer numbers can be entered in the discrete and stochastic case and real ones in the continuous case.	zero
con	Additional fire condition which has to be true so that the Transition become active and fires (e.g. <code>time > 9.7</code>)	true

Table 4: Parameter only of a discrete Transition.

Identifier	Description	Default value
delay	The time units that a discrete Transition waits after its activation before it fires.	1

Table 5: Parameters only of a stochastic Transition.

Identifier	Description	Default value
c	Constant for the pre-defined lambda functions	1
lambdafunc	Pre-defined function for the lambda calculation; choice between stochastic mass action hazard function and stochastic level hazard function (see [19])	Stochastic mass action
lambda	User-defined function for lambda instead of the pre-defined lambda functions (Stochastic mass action hazard function and Stochastic level hazard function)	

2.2.1 Implementation of the discrete Transition

In the discrete Transition it is decided with the aid of the Place-reported variables `inState` and `outState` if it is ready to fire, activated and if it fires.

A discrete Transition is active if the states of all Places in its previous area are true and none of the states of the Places in its past area is true. Additionally, the user-defined condition has to be true.

```
activated = allTrue and not anyTrue and con;
```

The pre-value of the variable `activated` is also needed to avoid an algebraic loop. An algebraic loop is a kind of deadlock between different when-equations. It is present if there are mutual dependencies [20].

```
preactivated = pre(activated);
```

When the variable `preactivated` switches from false to true, an event is triggered which activates the equations of the variables `last_activation_time` that saves the activation time.

```
when preactivated then  
    last_activation_time = time;  
end when;
```

When the Transition is timed, the Transition waits the user-defined time units before it fires. The variable `delay_passed` saves whether the delay is passed or not.

```
delay_passed = preactivated and  
    time-delay > last_activation_time;
```

For the determination when a Transition fires, is a differentiation between discrete and continuous Places necessary. The output connector variables `fire` and `set` are needed in the connected Places for the calculation of their Token numbers (see Figure 12).

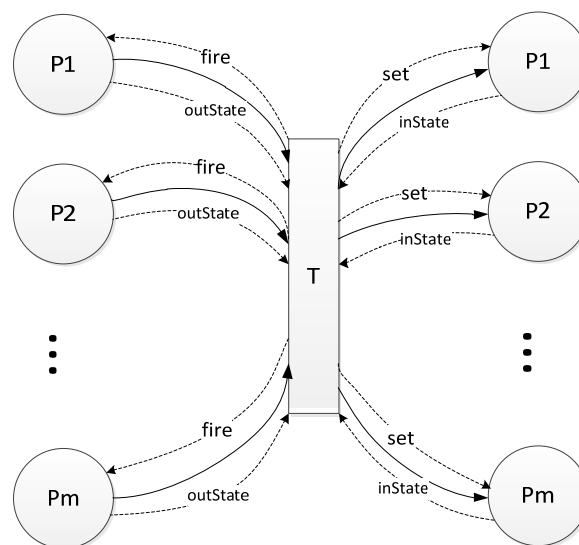


Figure 12: The input and output connector variables of Transitions.

2.2.2 Implementation of the stochastic Transition

The only difference between a discrete and a stochastic Transition is that the delay is a random number instead of a fixed value. In the implementation this is reflected by an additional equation which becomes active when the variable `preactivated` switches from false to true. At this point in time, an exponential distributed random number is generated with the aid of the function `randomexp`. The characteristic parameter λ of the exponential distribution is the value of the entered lambda-function at this point in time. This can be either a pre-defined (stochastic mass action hazard function or stochastic level hazard function) or a special user-defined function.

```

when preactivated then
  last_activation_time = time;
  delay = Functions.random.randomexp(pre(lambda));
end when;

```

The uniform distributed random numbers that are transformed to exponential distribution are generated by an external C-function.

2.2.3 Implementation of the continuous Transition

The continuous Transition model is, compared to the discrete or stochastic one, simple since when a continuous Transition is ready to fire, it is also active and fires, i.e. only the states of the connected Places have to be checked. A continuous Transition fires if all states of its previous Places are true and none state of its past Places. Additionally, the user-defined condition `con` has to be true.

```

fire = alltrue and not anytrue and con;

```

The output connector variable `fire` is then reported to the connected Places for the calculation of the new Token number (see Figure 12).

2.3 The Reactions Sub-Library

The Petri Net models of the Discrete, Continuous and Stochastic sub-libraries can be wrapped to models for different kinds of biological reactions to simplify the modeling process. These model components are organized in the sub-library Reactions which is also divided in several sub-libraries for different reaction types. Till now there are:

- Reaction kinetics
- Enzyme kinetics
- Growth kinetics
- Culture strategies
- Process activation

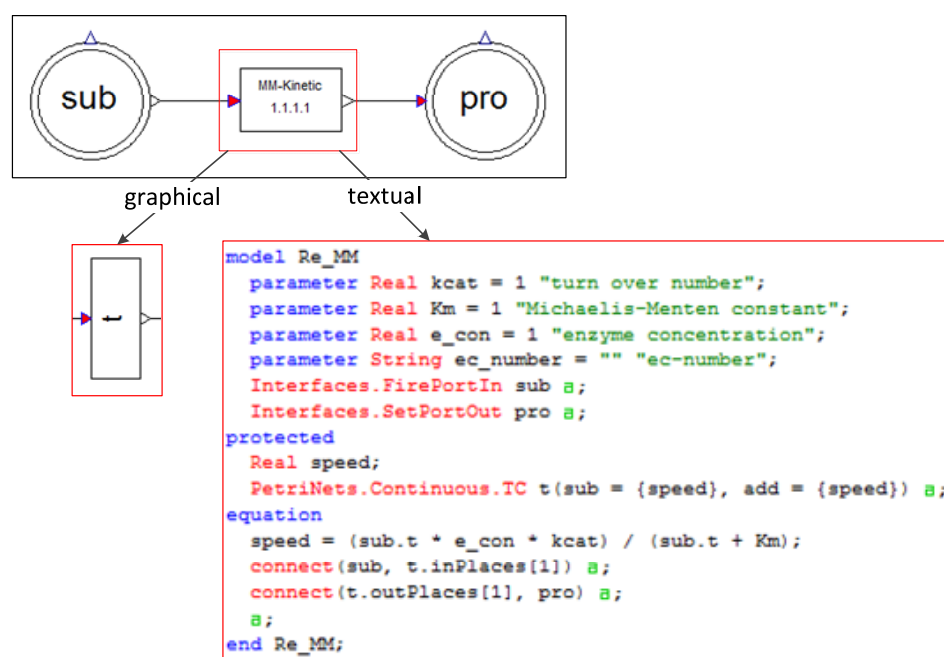


Figure 13: Wrapper of the Michaelis-Menten Kinetics.

The wrapping process is illustrated by the Michaelis-Menten Kinetics in Figure 13. Behind the “MM-Kinetic”-icon, which is between two continuous Places for substrate and product, is a continuous Transition with the Michaelis-Menten Kinetics as edge weighting. In this manner, the user has only to enter the parameters of the Michaelis-Menten Kinetics in the property dialog of the “MM-Kinetic”-icon instead of the whole equation every time (see Figure 14). All the other wrappers work in the same manner. The user can easily add his individual reactions in this package to make the specific modeling process much faster and easier. For a detailed description of the wrapping process see [21].

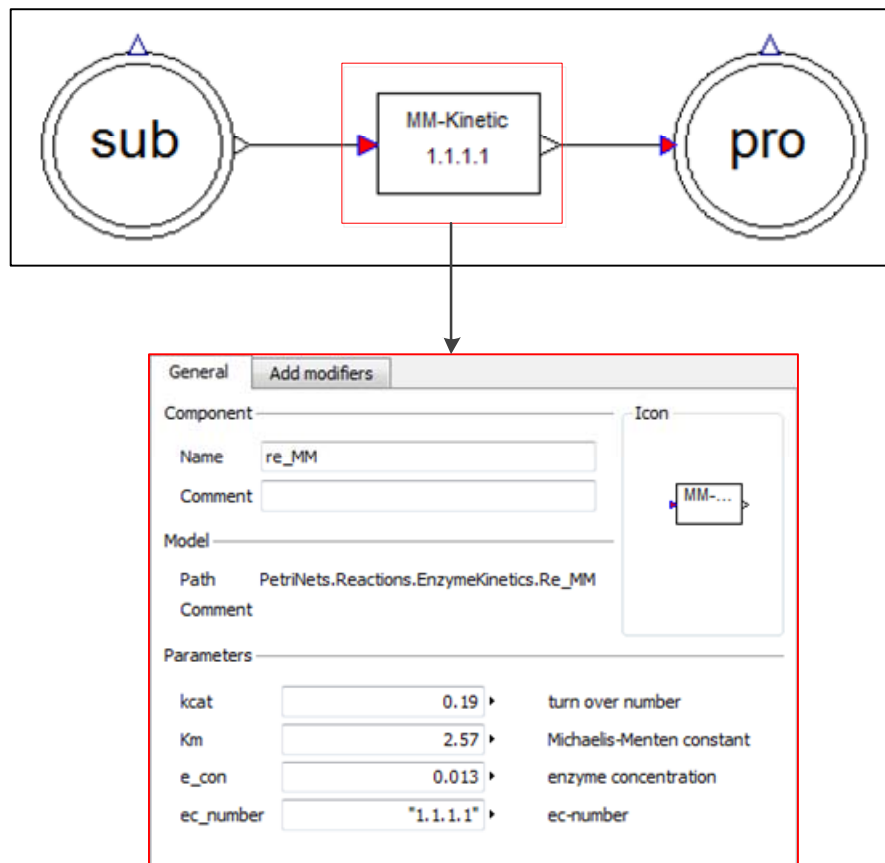


Figure 14: Property dialog of the Michaelis-Menten Kinetics wrapper.

2.4 Modeling, Simulation and Animation in Dymola

A Petri Net model can be constructed easily in Dymola by drag the respective Petri Net icon from the Package Browser to the diagram (see Figure 15).

If the model is constructed graphically, it is also textual available. The textual model of the Petri Net in Figure 15 is given in Figure 16. In this manner, it is possible to parse hybrid Petri Net models of other tools (e.g. in XML-format) to Modelica-text and compile and simulate it with a Modelica-tool.

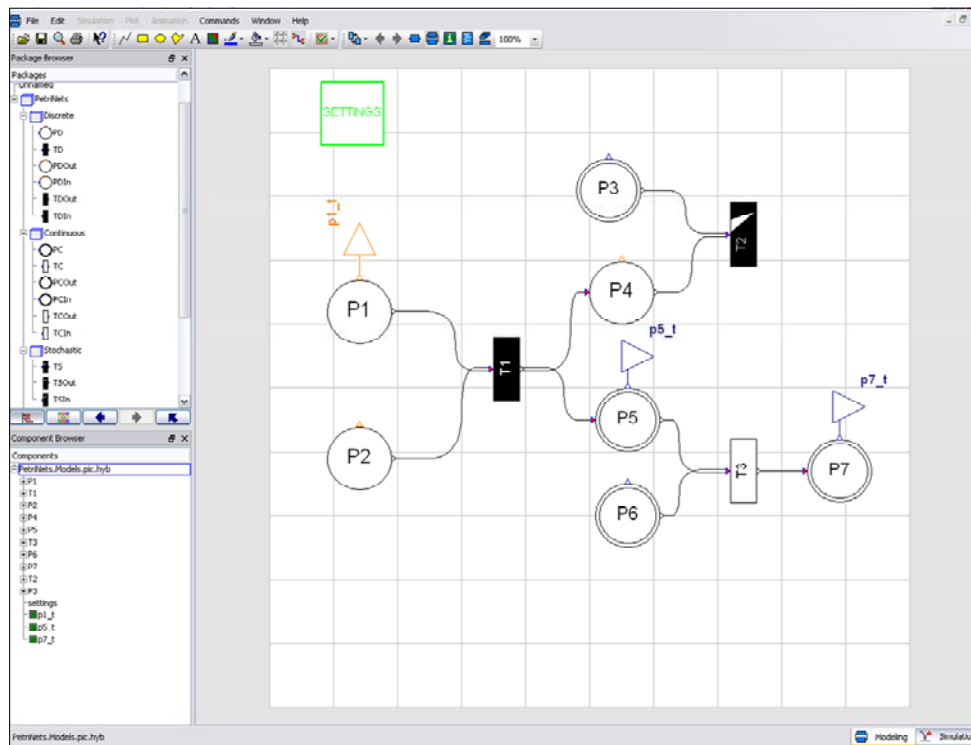





Figure 15: Graphical Modeling of Petri Nets in Dymola.

```

model hyb
  Discrete.PDOut P1(nOut=1, num_tokens_start=100) a;
  Discrete.TD T1(nIn=2, nOut=2, add={3,2}) a;
  Discrete.PDOut P2(nOut=1, num_tokens_start=200) a;
  Discrete.PD P4(nIn=1, nOut=1, num_tokens_start=30, min=2) a;
  Continuous.PC P5(nIn=1, nOut=1, num_tokens_start=10) a;
  Continuous.TC T3(nIn=2, nOut=1, sub={1,P6.t*0.05}, add={P6.t*0.1}) a;
  Continuous.PCOut P6(nOut=1, num_tokens_start=100) a;
  Continuous.PCIn P7(nIn=1) a;
  Stochastic.TSIn T2(nIn=2) a;
  Continuous.PCOut P3(nOut=1, num_tokens_start=100, min=10) a;
  inner Global.settings settings(scale=0.5) a;
  Modelica.Blocks.Interfaces.IntegerOutput p1_t a;
  Modelica.Blocks.Interfaces.RealOutput p5_t a;
  Modelica.Blocks.Interfaces.RealOutput p7_t
  a;
equation
  connect(P1.outTransition[1], T1.inPlaces[1]) a;
  connect(P2.outTransition[1], T1.inPlaces[2]) a;
  connect(T1.outPlaces[1], P4.inTransition[1]) a;
  connect(T1.outPlaces[2], P5.inTransition[1]) a;
  connect(P5.outTransition[1], T3.inPlaces[1]) a;
  connect(P6.outTransition[1], T3.inPlaces[2]) a;
  connect(T3.outPlaces[1], P7.inTransition[1]) a;
  connect(P4.outTransition[1], T2.inPlaces[1]) a;
  connect(P3.outTransition[1], T2.inPlaces[2]) a;
  connect(P1.pd_t, p1_t) a;
  connect(P5.pc_t, p5_t) a;
  connect(P7.pc_t, p7_t) a;
a;
end hyb;

```

Figure 16: Textual Modeling of Petri Nets in Dymola.

For the simulation, one has to switch from the Modeling view (see Figure 15 and Figure 16) to the Simulation view (see Figure 17). There the model can be translated to C-code with the translation button  in the toolbar and afterwards it can be simulated with the simulation button . Dymola offers a lot of properties for tuning the simulation by clicking on simulation properties button , i.e. one can choose among others the start and stop time and the ODE-solver, whereby 16 different are available.

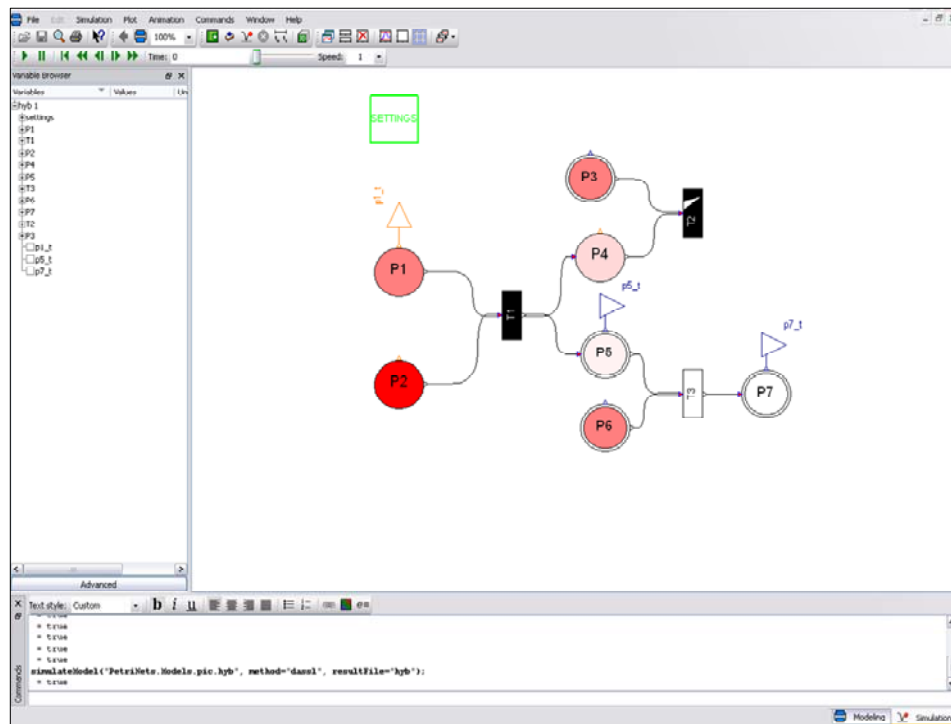


Figure 17: Simulation view of a Petri Net model in Dymola.

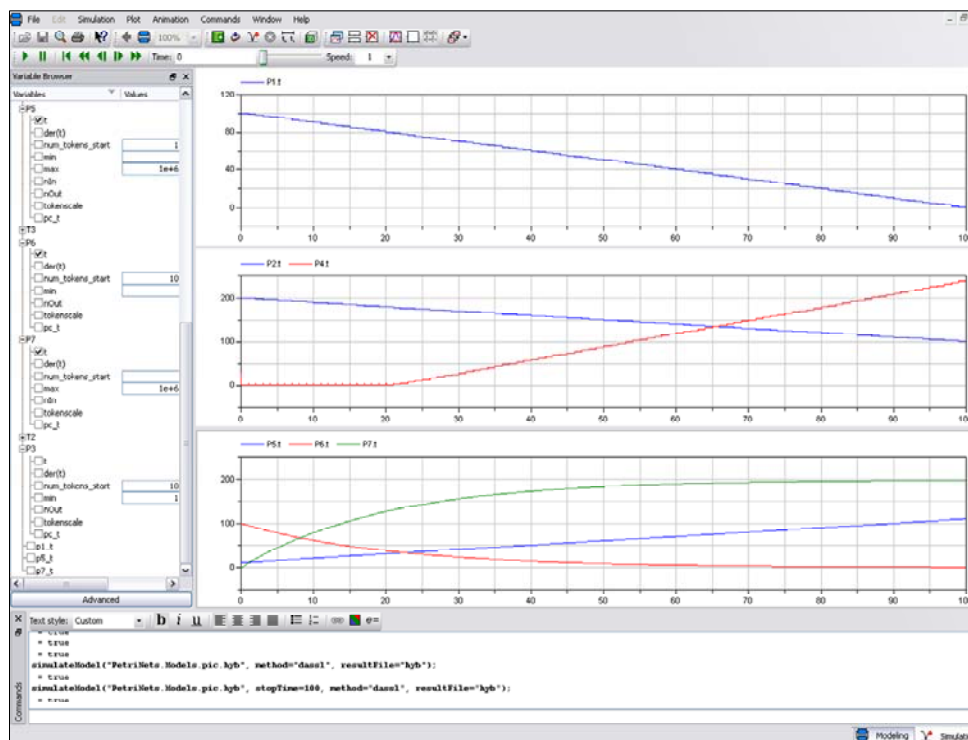


Figure 18: The representation of the simulation results by several plots with selected Token numbers.

The simulation results can be displayed by either plots of selected Token numbers (see Figure 18) or by an animation. By the latter one the degree of redness changes during time according to the Token number of the Place, i.e. a red Place has many Tokens and a white Place is empty. The redness degree can be scaled from 0 to 100 by the green Settings-box which is a

component of the Global sub-library (see Figure 19). This animation is realized by the annotation `DynamicSelect`

```
annotation(fillColor = DynamicSelect ({255,255,255},if scale<100
    then {255,255-2.55*tokenscale,255-2.55*scale} else {255,0,0}))
```

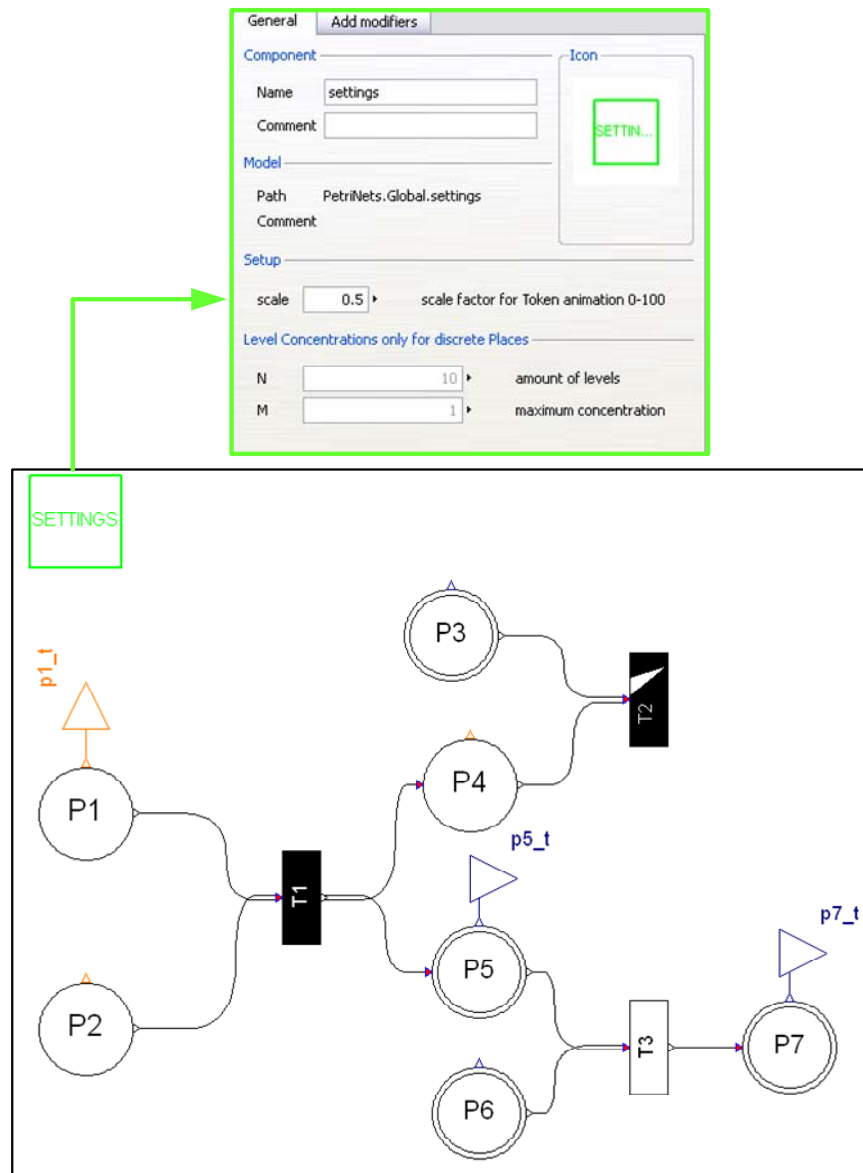


Figure 19: Scaling of the redness degree by the Settings-component of the Global sub-library.

Figure 20 shows the redness change of a Petri Net example during time. This animation offers a good way for analyzing large and complex Petri Nets.

2.5 Connection Dymola and Matlab/Simulink

For parameter optimization, sensitivity analysis and stochastic simulation according to Gillespie [22], it is necessary to simulate the model several times with different parameter settings. Dymola offers a possibility to connect a Modelica-model to Matlab by a Simulink interface (DymolaBlock) and a set of Matlab m-files [23].

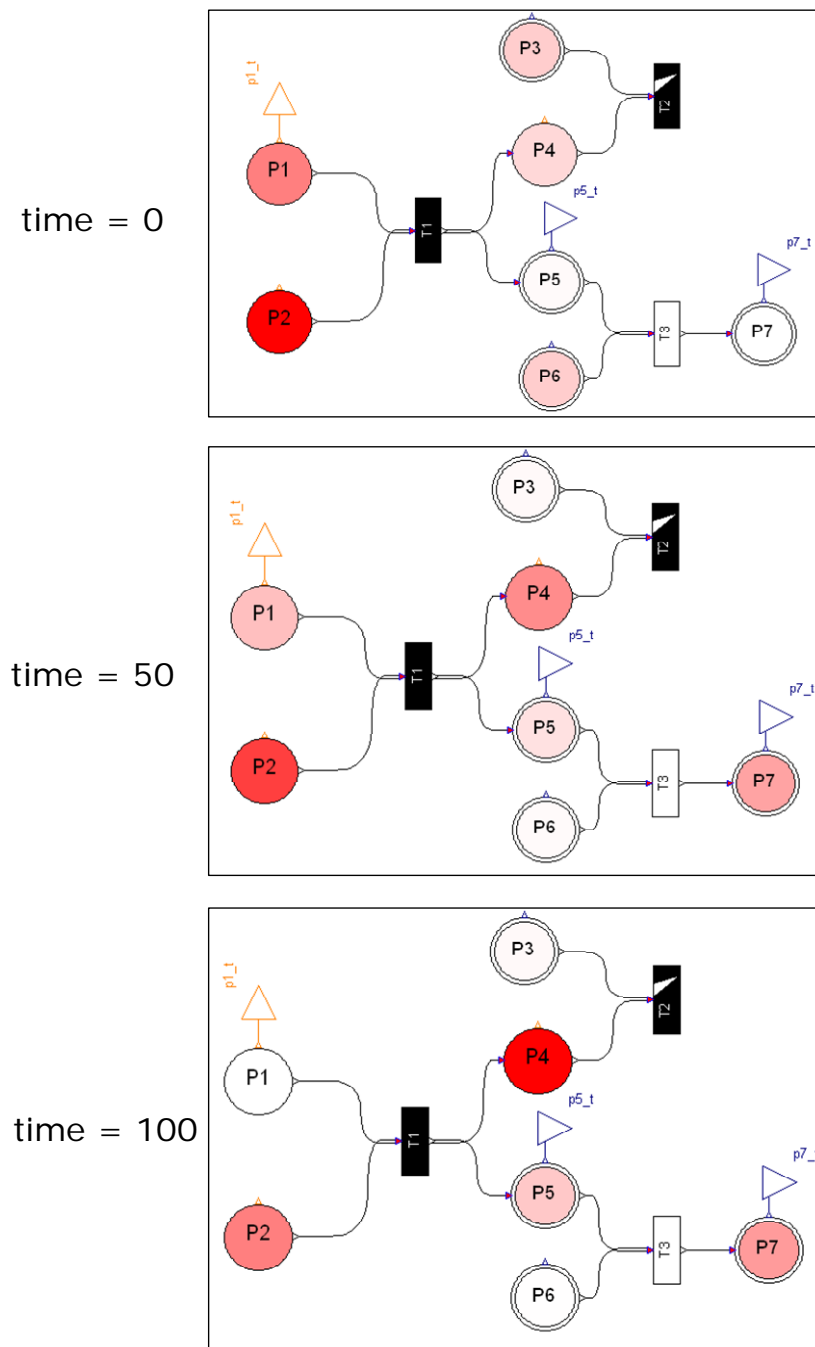


Figure 20: Animation of a Petri Net in Dymola; the Token distribution of the Petri Net example, top: at the beginning of the simulation, middle: after a simulation of 50 time units, bottom: after a simulation of 100 time units; the degree of redness corresponds to the Token numbers, i.e. a red Place has many Tokens and a white Place is empty.

Figure 21 displays on the left a Petri Net modeled by the Petri Net library in Dymola and on the right the corresponding Simulink model. If the Token number of a Place over the time is needed in Matlab for further calculations, one has to create a connector above the respective Place. This is an orange IntegerOutput connector in the case of a discrete Place or a blue RealOutput connector if it is a continuous Place. In the Petri Net example of Figure 21 the Token numbers of the Places *P1*, *P5* and *P7* are needed in Matlab where *P1* is a discrete Place with an IntegerOutput connector and *P5* and *P7* are continuous with a RealOutput connector. The DymolaBlock in Simulink generates a connector for all Places connected with an Output connector in Dymola. These connectors can be connected via Bus to an Outputport so that these simulation results are saved within a matrix and are available in the Matlab

environment for further calculations. In the same manner it is also possible that the Petri Net model gets inputs from Matlab via a connection between a Simulink source and a Modelica IntegerInput or RealInput connector.

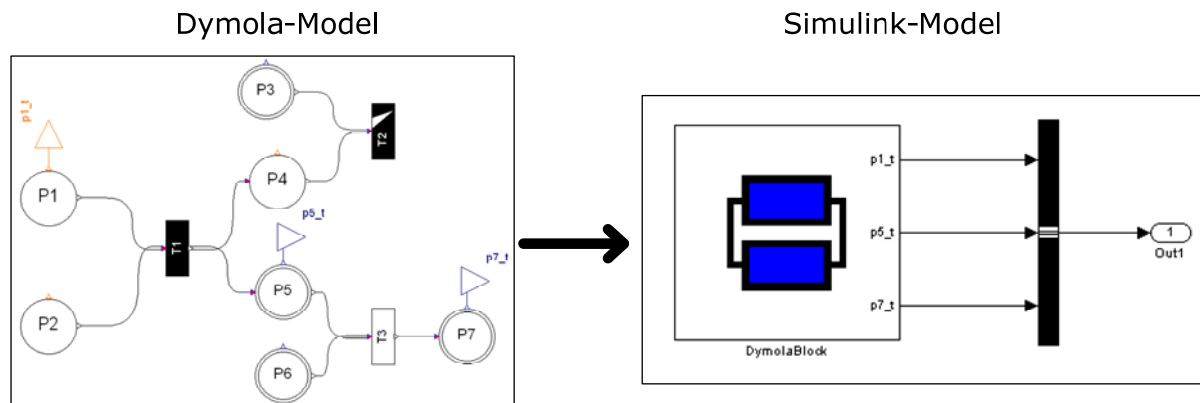


Figure 21: Connection Dymola and Matlab/Simulink by a Simulink Interface (DymolaBlock), left: Petri Net modeled by the Petri Net library in Dymola, right: Simulink interface of the Dymola-model in Matlab/Simulink.

To connect a Dymola-model with Simulink, one has to enter the model name and its path in the property dialog of the DymolaBlock (see Figure 22). After that, the model can be compiled and the parameters can be set. The parameters can be also set within Matlab by special m-files and the model can be simulated by the prompt

```
sim(model,timespan,options,ut)
```

For a detailed description see [23].

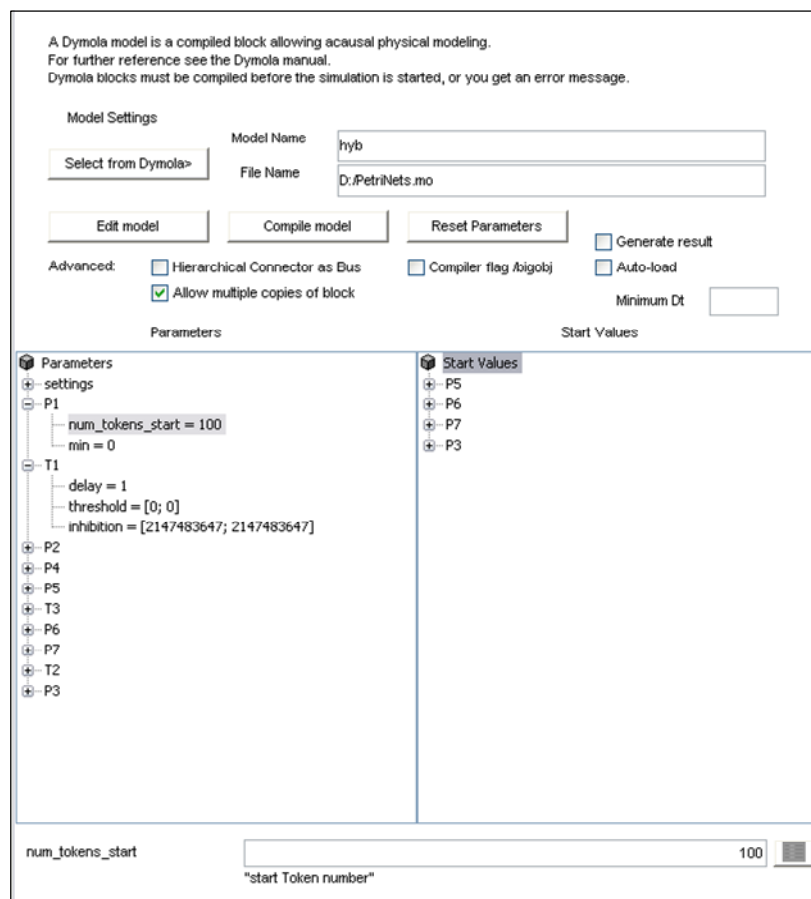


Figure 22: Property dialog of the DymolaBlock in Simulink.

3 Example: Antibody production of the Chinese Hamster Ovary Cells

The Chinese Hamster Ovary Cells (CHO-cells) produce antibodies which are part of many pharmaceuticals [24]. Additionally, they produce the waste-products lactate and ammonium which can inhibit their growth and antibody production when specific concentrations are exceeded ([25], [26], [27]). The main metabolism of CHO-cells is displayed in Figure 23. Experiments were performed by growing the CHO-Cells in shaking flasks. They are fed with the nutrients glucose and glutamine and they produce antibodies, ammonium and lactate. By the latter ones it is assumed that they cannot only be produced by the CHO-Cells but also consumed when the environmental conditions are appropriate ([28], [29]).

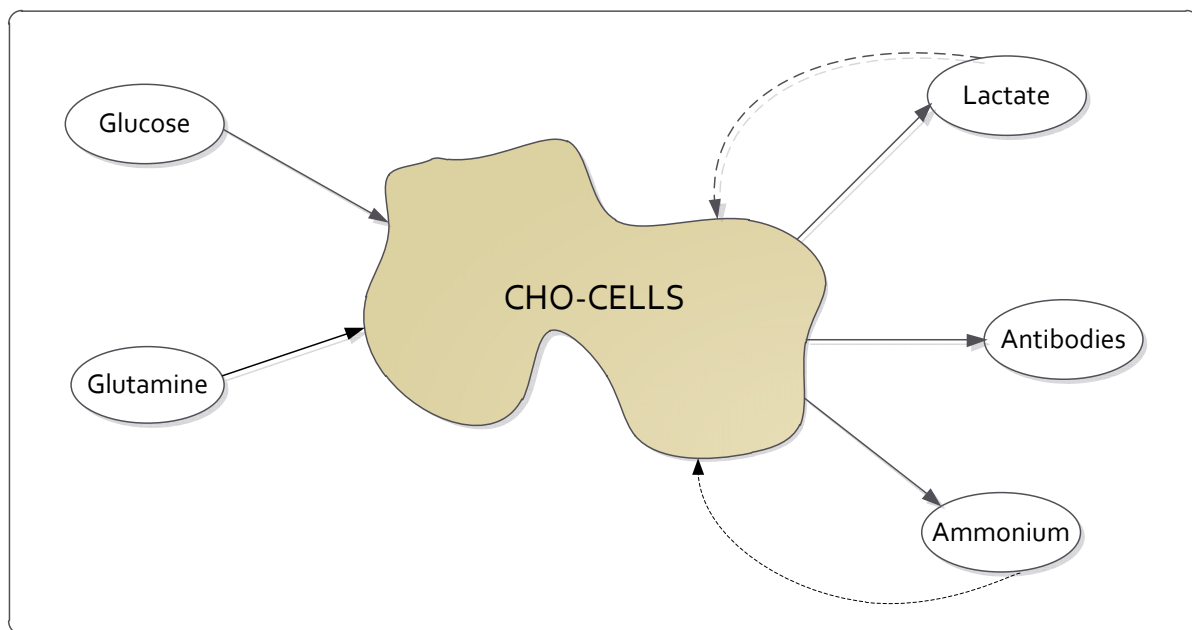


Figure 23: The main metabolism of CHO-cells.

Figure 24 represents the experimental data of CHO-Cells growing in shaking flasks. The experiments were performed by the University of Applied Sciences Bielefeld, Instrumental Biotechnology Institute [30]. The exponential growing phase of the cells ends at day 4 and the cells pass over to a stationary phase which takes approximately 2 days. Afterwards more cells die than new ones grow thus the curve of living cells decreases and the curve of death cells increases (death phase). The nutrient Glucose is exhausted at the end of the experiment and the waste-product lactate is produced till day 4 and afterwards it is consumed by the CHO-Cells. They convert it back to pyruvate which enters the citric acid cycle (TCA-cycle) [29]. Here, it is assumed that they start the lactate consumption when a specific lactate concentration is exceeded. Additionally, the ammonium concentration decreases after 4 days and the glutamine concentration increases. In this conjunction, it seems likely that the CHO-cells can convert ammonium back to glutamine when the glutamine concentration falls below a specific value. The Antibody production starts first after 2.5 days and not stops until the end of the experiment. At this point the supposition is that the cells start the production first when the glucose becomes limiting.

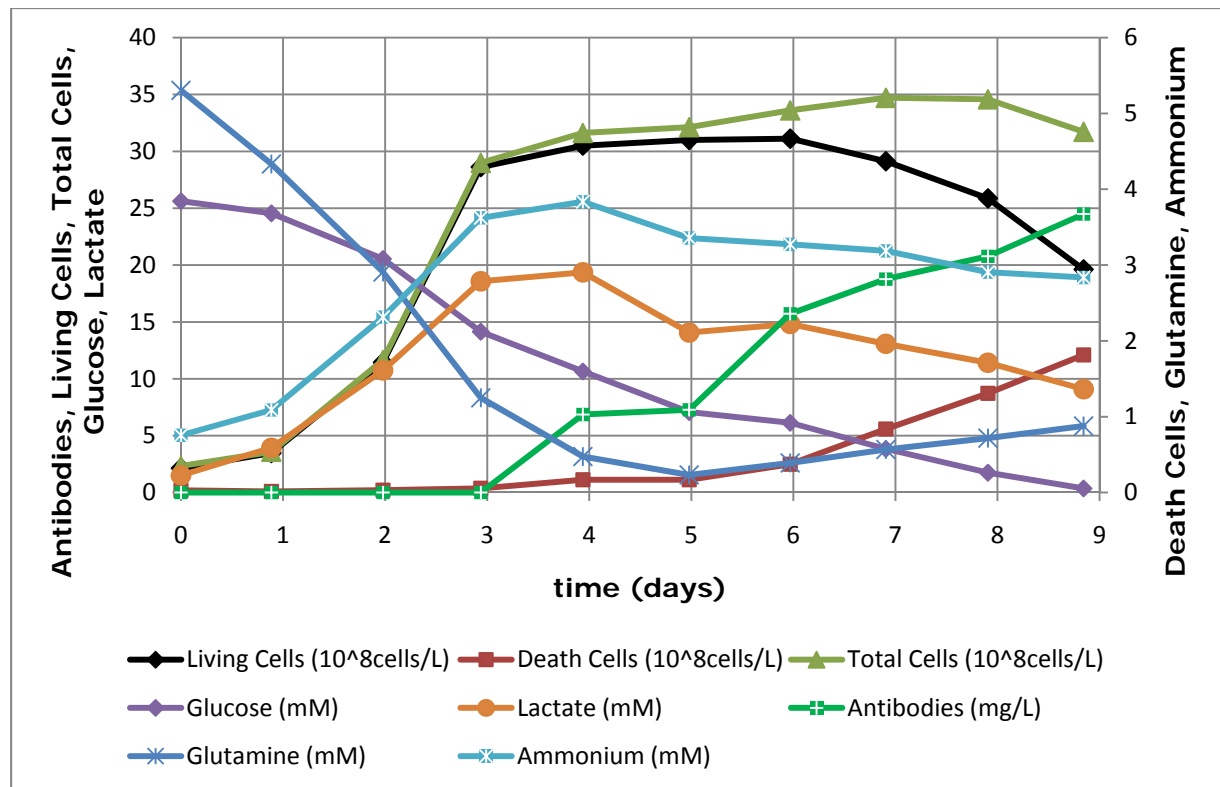


Figure 24: Experimental data of CHO-Cells growing in shaking flasks.

A continuous Petri Net models the dynamics of the CHO-cells (see Figure 25). This Petri Net covers a lot of different differential equation systems. Which of them is chosen depends on the environmental conditions. At the beginning of the experiment, it represents the following ODEs

$$\frac{dX_t}{dt} = \mu \cdot X_v \quad \text{Eq. 1}$$

$$\frac{dX_d}{dt} = \mu_d \cdot X_v \quad \text{Eq. 2}$$

$$\frac{dX_v}{dt} = (\mu - \mu_d) \cdot X_v \quad \text{Eq. 3}$$

$$\frac{dGlc}{dt} = -q_{glc} \cdot X_v \quad \text{Eq. 4}$$

$$\frac{dGlu}{dt} = -q_{glu} \cdot X_v - k_{sd} \cdot Glu \quad \text{Eq. 5}$$

$$\frac{dLac}{dt} = q_{lac} \cdot X_v \quad \text{Eq. 6}$$

$$\frac{dAmm}{dt} = q_{amm} \cdot X_v + k_{sd} \cdot X_v \quad \text{Eq. 7}$$

$$\frac{dAb}{dt} = 0 \quad \text{Eq. 8}$$

$$X_t(0) = X_{t0}, X_d(0) = X_{d0}, X_v(0) = X_{v0}, Glc(0) = Glc_0, \\ Glu(0) = Glu_0, Lac(0) = Lac_0, Amm(0) = Amm_0, Ab(0) = Ab_0 \quad \text{Eq. 9}$$

where X_t is the concentration of total cells (10^8 cells/L), X_d is the concentration of death cells (10^8 cells/L), X_v is the concentration of living cells (10^8 cells/L), Glc is the glucose concentration (mM), Glu is the glutamine concentration (mM), Lac is the lactate concentration (mM), Amm is the Ammonium concentration (mM), Ab is the Antibody concentration (mg/L), μ is the specific growth rate (1/d), μ_d is the specific death rate (1/d), q_{glc} is the specific glucose uptake rate (mmol/ 10^8 cells/d), q_{glu} is the specific glutamine uptake rate (mmol/ 10^8 cells/d), k_{sd} is the constant for the spontaneous degradation of glutamine, q_{lac} is the specific lactate production rate (mmol/ 10^8 cells/d), q_{amm} is the specific ammonium production rate (mmol/ 10^8 cells/d) and X_{t0} , X_{d0} , X_{v0} , Glc_0 , Glu_0 , Lac_0 , Amm_0 and Ab_0 are the start concentrations.

The conversion from glutamine to ammonium can take place in two different ways: the CHO-cells can perform it ($q_{glu} \cdot X_v$, $q_{amm} \cdot X_v$) and it can occur within the medium by spontaneous decomposition ($k_{sd} \cdot Glu$) [31]. No antibodies are produced at the beginning of the experiment thus the differential equation is set to zero. After a specific change of the environmental conditions, the Antibody production starts and Eq. 8 has to be changed to

$$\frac{dAb}{dt} = q_{ab} \cdot X_v \quad \text{Eq. 10}$$

where q_{ab} is the antibody production rate (mg/ 10^8 cells/d). The supposition is that the decreasing glucose concentration initiates the antibody production. In terms

$$\frac{dAb}{dt} = \begin{cases} 0, & Glc \geq 14 \text{ mM} \\ q_{ab} \cdot X_v, & Glc < 14 \text{ mM} \end{cases} \quad \text{Eq. 11}$$

A similar switching situation occurs by the lactate concentration. At the beginning the dynamics are represented by Eq. 6 and after a specific change of the environmental conditions, especially the lactate concentration passes a threshold, the dynamics are described by

$$\frac{dLac}{dt} = \begin{cases} q_{lac} \cdot X_v - q_{lacs} \cdot X_v, & Lac \geq 19 \text{ mM} \\ q_{lac} \cdot X_v, & Lac < 19 \text{ mM} \end{cases} \quad \text{Eq. 12}$$

where q_{lacs} is the specific lactate consumption rate (mmol/ 10^8 cells/d). The glutamine consumption and production, respectively, leads to the following switching equation, whereby the change is initiated by the decreasing glutamine concentration

$$\frac{dGlu}{dt} = \begin{cases} -q_{glu} \cdot X_v - k_{sd} \cdot Glu, & Glu \geq 0.4 \text{ mM} \\ -q_{glu} \cdot X_v - k_{sd} \cdot Glu + q_{glus} \cdot X_v, & Glu < 0.4 \text{ mM} \end{cases} \quad \text{Eq. 13}$$

where q_{glus} is the specific glutamine production rate (mmol/ 10^8 cells/d) and the corresponding dynamics for the ammonium concentration are

$$\frac{dAmm}{dt} = \begin{cases} q_{amm} \cdot X_v + k_{sd} \cdot Glu, & Glu \geq 0.4 \text{ mM} \\ q_{amm} \cdot X_v + k_{sd} \cdot Glu - q_{amms} \cdot X_v, & Glu < 0.4 \text{ mM} \end{cases} \quad \text{Eq. 14}$$

where q_{amms} is the specific ammonium consumption rate (mmol/ 10^8 cells/d).

Figure 25 displays the Petri Net modeling the discussed conditions above (Eq. 1 - Eq. 4, Eq. 11 - Eq. 14). All Places and Transitions are continuous. Table 6 contains the Places and their

corresponding substances and Table 7 summarizes the information of the Transitions. The orange Activation-boxes are wrappers of the Reactions sub-library and they work like a discrete switch. When the Token number of the connected Place exceeds the entered value of the parameter *tres* or fall below the entered value of the parameter *inhi*, the connected Transition becomes active and remain active until one of the connected Places becomes empty in contrast to the threshold and inhibition values of the Transitions. Everything inside the blue box with a macroscopic picture of the CHO-cells occurs within the cells and outside of the blue box are the reaction for the spontaneous decomposition of glutamine and the substances that the cells releases to the medium. The total amount of cells, the sum of living cells and death cells, is modeled by an algebraic equation

$$Xt_t = Xv_t + Xd_t.$$

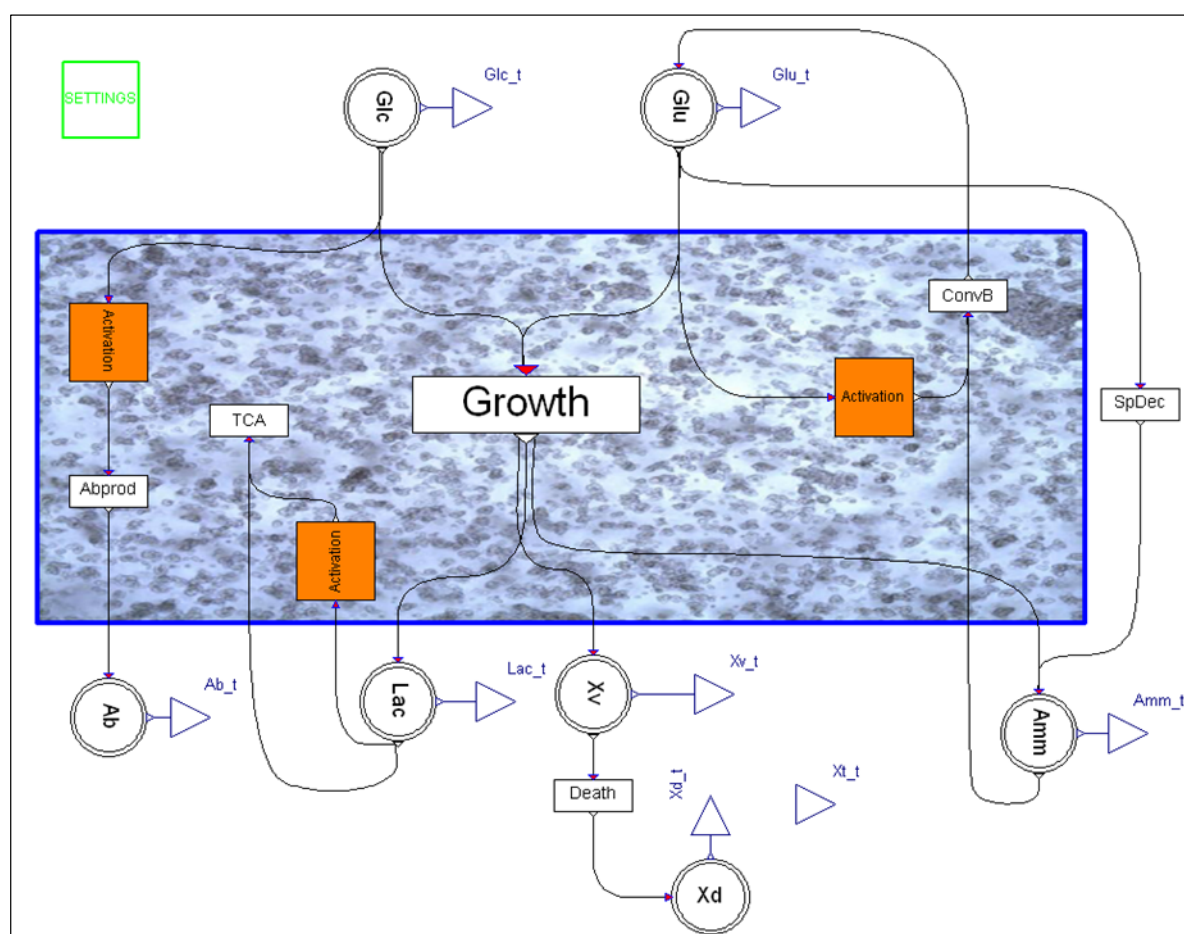


Figure 25: Petri Net model of the main CHO-metabolism in Dymola.

Table 6: Places of the CHO-model in Figure 25 and the corresponding substances.

Place	Substance
Xv	Concentration of living CHO-Cells
Xd	Concentration of death CHO-Cells
Glc	Glucose concentration
Glu	Glutamine concentration
Lac	Lactate concentration
Amm	Ammonium concentration
Ab	Antibody concentration

Table 7: Transitions of the CHO-model in Figure 25, the corresponding reactions and the edge weighting functions (Eq. 1 - Eq. 4, Eq. 11 - Eq. 14).

Transition	Reaction	Weightings		Conditions
Growth	Cell growth	Glc → Growth	$q_{glc} \cdot Xv.t$	
		Glu → Growth	$q_{glu} \cdot Xv.t$	
		Growth → Xv	$\mu \cdot Xv.t$	
		Growth → Lac	$q_{lac} \cdot Xv.t$	
		Growth → Amm	$q_{amm} \cdot Xv.t$	
Death	Cell death	Xv → Death	$\mu_d \cdot Xv.t$	
		Death → Xd	$\mu_d \cdot Xv.t$	
TCA	Lactate consumption	Lac → TCA	$q_{lacs} \cdot Xv.t$	Orange Activation Box thres=19
Abprod	Antibody production	Abprod → Ab	$q_{ab} \cdot Xv.t$	Orange Activation Box inhi=14
SpDec	Spontaneous decomposition of glutamine to ammonium	Glu → SpDec	$k_{sd} \cdot Glu.t$	
		SpDec → Amm	$k_{sd} \cdot Glu.t$	
ConvB	Conversion of ammonium back to glutamine	Amm → ConvB	$q_{amms} \cdot Xv.t$	Orange Activation Box inhi=0.4
		ConvB → Glu	$q_{glus} \cdot Xv.t$	

The experimental data of Figure 24 are approximated by smoothing splines to get further insight to the relations between the respective specific rates. The rates at the beginning of the simulation can be calculated by the following equations

$$\mu = \frac{1}{X_v} \cdot \frac{dX_t}{dt} \quad \text{Eq. 15}$$

$$\mu_d = \frac{1}{X_v} \cdot \frac{dX_d}{dt} \quad \text{Eq. 16}$$

$$q_{glc} = -\frac{1}{X_v} \cdot \frac{dGlc}{dt} \quad \text{Eq. 17}$$

$$q_{glu} = -\frac{1}{X_v} \cdot \left(\frac{dGlu}{dt} + k_{sd} \cdot Glu \right) \quad \text{Eq. 18}$$

$$q_{lac} = \frac{1}{X_v} \cdot \frac{dLac}{dt} \quad \text{Eq. 19}$$

$$q_{amm} = \frac{1}{X_v} \cdot \left(\frac{dGlu}{dt} - k_{sd} \cdot Glu \right) \quad \text{Eq. 20}$$

The specific antibody production rate can be calculated after day 2.5 when the cells start the production

$$q_{ab} = \frac{1}{X_v} \cdot \frac{dAb}{dt} \quad \text{Eq. 21}$$

The relations analysis yields the following equation structures for the specific rates

$$\mu = \mu_{max} \cdot \frac{Glu}{K_{Glu} + Glu} \quad \text{Eq. 22}$$

$$\mu_d = \mu_{dmax} \cdot \frac{KD_{Glc}}{KD_{Glc} + Glc} \quad \text{Eq. 23}$$

$$q_{glc} = \frac{1}{Y_{X,Glc}} \cdot \mu \quad \text{Eq. 24}$$

$$q_{glu} = \frac{1}{Y_{X,Glu}} \cdot \mu \quad \text{Eq. 25}$$

$$q_{lac} = Y_{Lac,Glc} \cdot q_{glc} = Y_{lac,Glc} \cdot \frac{1}{Y_{X,Glc}} \cdot \mu \quad \text{Eq. 26}$$

$$q_{amm} = Y_{Amm,Glu} \cdot q_{glu} = Y_{Amm,Glu} \cdot \frac{1}{Y_{x,Glu}} \cdot \mu \quad \text{Eq. 27}$$

$$q_{ab} = k_{ab} \quad \text{Eq. 28}$$

$$q_{lacs} = k_{lacs} \quad \text{Eq. 29}$$

$$q_{amms} = k_{amms} \quad \text{Eq. 30}$$

$$q_{glus} = Y_{Glu,Amm} \cdot q_{amms} \quad \text{Eq. 31}$$

with the parameters μ_{max} (1/d) as maximum specific growth rate and K_{Glu} as constant of the Monod kinetics, μ_{dmax} (1/d) as maximum specific death rate, KD_{Glc} as constant of the death kinetics (mM), $Y_{X,Glc}$ (10^8 cells/mmol), $Y_{X,Glu}$ (10^8 cells/mmol), $Y_{Lac,Glc}$ (mol/mol), $Y_{Amm,Glu}$ (mol/mol) and $Y_{Glu,Amm}$ (mol/mol) as yield coefficients, k_{ab} (mg/ 10^8 cells) as constant of the antibody production, k_{lacs} (mmol/ 10^8 cells) as constant of the lactate consumption and k_{amms} as constant of the ammonium consumption.

For performing a sensitivity analysis and afterwards a parameter optimization for the 13 model parameters, the Petri Net model in Dymola (Figure 25) has to be connected to Matlab via a Simulink interface as described in section 2.5. The corresponding Simulink-model is displayed in Figure 26. The simulation results of all Token numbers are needed in Matlab, thus all Places have a blue RealOutput connector (Figure 25) so that a corresponding port at the Simulink interface is provided.

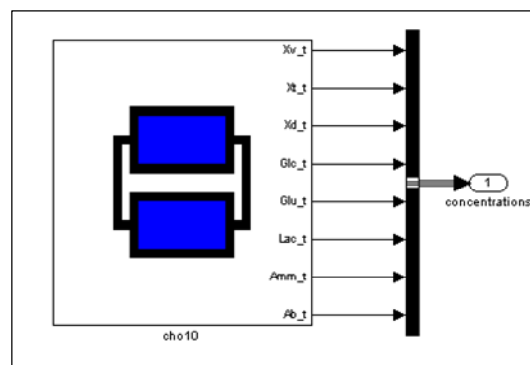


Figure 26: Simulink-model of the Petri Net model in Figure 25.

Before the 13 parameters of the model are estimated a global sensitivity analysis is performed to get further insight in the parameter characteristics. This analysis is the basis of the following parameter optimization since less sensitive parameters can be fixed during the

optimization process to increase the chance of a converging optimization algorithm. The global sensitivity analysis is performed by Matlab with a specific method (extended Fourier Sensitivity Analysis Test (eFAST) see e.g. [32]). Therefore, the model is simulated several times with different parameter settings and each time the following objective function is evaluated (least square approach)

$$Q(\mathbf{p}, t) = \sum_{i=1}^n \sum_{j=1}^r \left(\frac{y_i(\mathbf{p}, t_j) - d_i(t_j)}{d_i(t_j)} \right)^2 \quad \text{Eq. 32}$$

where $y_i(\mathbf{p}, t_j)$ is the model output of the i th concentration at time t_j with the parameter values \mathbf{p} and $d_i(t_j)$ is the experimental concentration at time t_j . The eFAST-method measures the contribution of each parameter to the variance of this objective function, whereby the parameters are varied in a specific range. If a parameter contributes less to the variance, this parameter cannot be identified with an optimization procedure and has to be fixed and if a parameter contributes much to the variance of the objective function this parameter is identifiable within the optimization process.

The results of the global sensitivity analysis, i.e. the contribution of each parameter to the objective function variance, is displayed in Figure 27. It becomes clear that 7 of 13 parameters contribute 91 % of the variance so that 6 parameters

$$Y_{X,Glu}, \mu_{dmax}, KD_{glc}, k_{lacs}, Y_{Lac,Glc}, Y_{Amm,Glu}$$

can be fixed during the optimization process and 7

$$k_{sd}, Y_{Glu,Amm}, k_{amms}, Y_{X,Glc}, \mu_{max}, K_{Glu}, k_{ab}$$

have to be optimized.

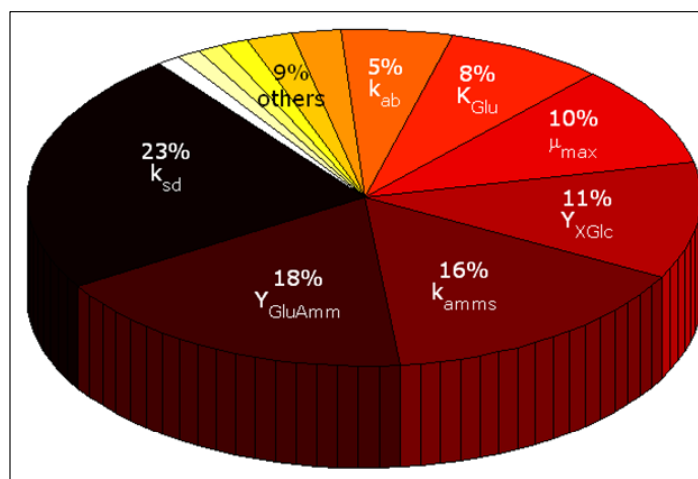


Figure 27: Model variance contribution of every parameter according to the eFAST method.

The parameter optimization is performed by a special kind of an Evolution Strategy (Covariance Matrix Adaption Evolution Strategy [33]). Local methods fail due to the non-differentiability of the objective function (Eq. 32) which is a result of the discrete switches (events) between the different ODEs. The optimization procedure takes place in Matlab via a

Simulink interface. Figure 28 displays the results of this optimization procedure which shows a good agreement with the experimental data.

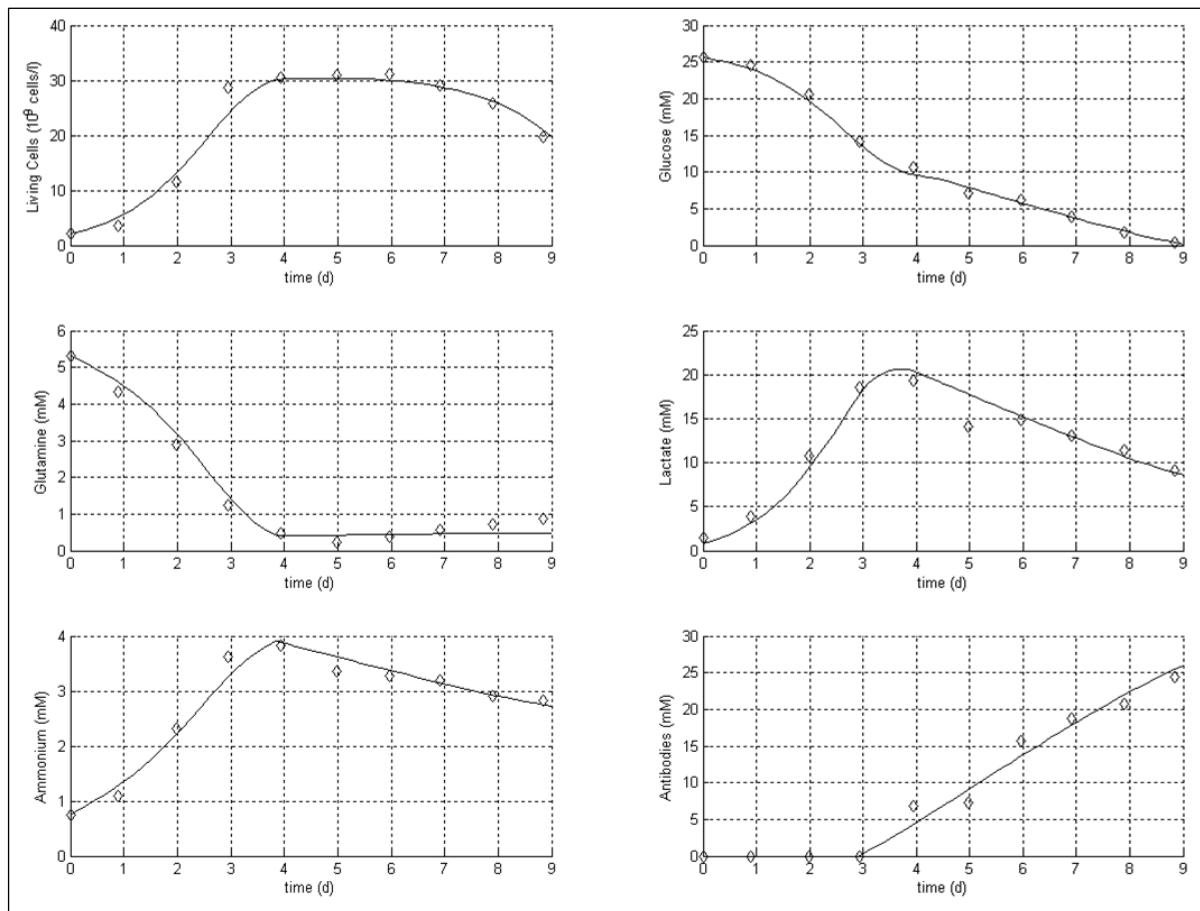


Figure 28: Results of the parameter optimization procedure.

To achieve a good model of the CHO-metabolism, it is also possible to choose a stochastic approach, i.e. a stochastic Petri Net model and a stochastic simulation according to Gillespie's algorithm ([22], [19]). The edge weightings of the continuous approach in Table 7 are now the dynamic values of the characteristic parameter λ of the exponential distribution by which the delay of the stochastic Transition is chosen randomly at every activation point in time (cp. Sections 1.1 and 2.2.2). The transformation of the parameters of the continuous model to the stochastic one is well studied and can be found in [34]. Figure 29 displays the CHO-metabolism modeled by a stochastic Petri Net, whereby the Places are discrete and the Transitions are stochastic. The Tokens represent here different concentration levels like it is presented in [19]. One Token equates to 0.5 (mM, 10^8 Cells/l, mg/l), thus there are $N + 1 = 90 + 1$ different levels since the maximum concentration (M) is set to 45. The values of M and N can be entered in the green settings-box which has to be a part of every model and can be found in the Global-library. This stochastic Petri Net model is also connected to a Simulink interface in Matlab so that the stochastic simulation can take place within an m-file. The results are displayed in Figure 30 where 500 Simulation are accomplished and the means were built each with 10 simulations.

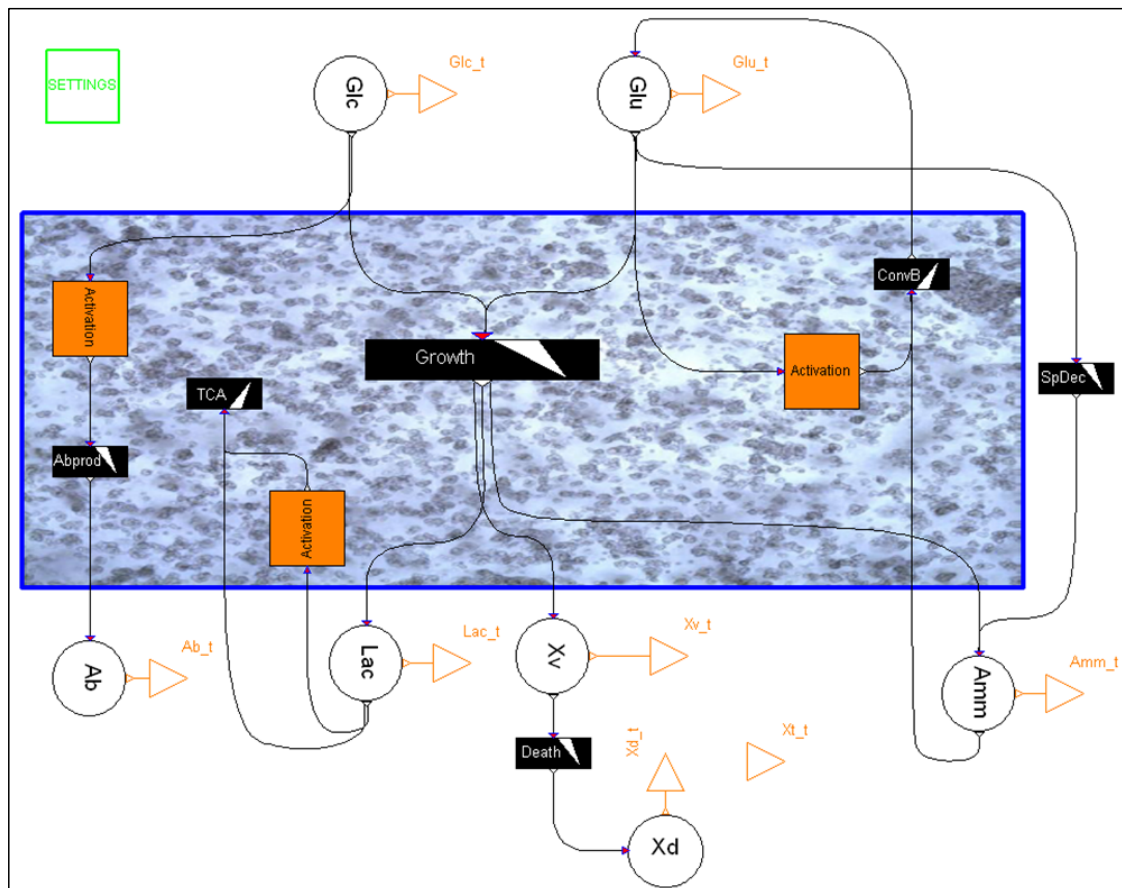


Figure 29: Stochastic Petri Net model of the main CHO-metabolism.

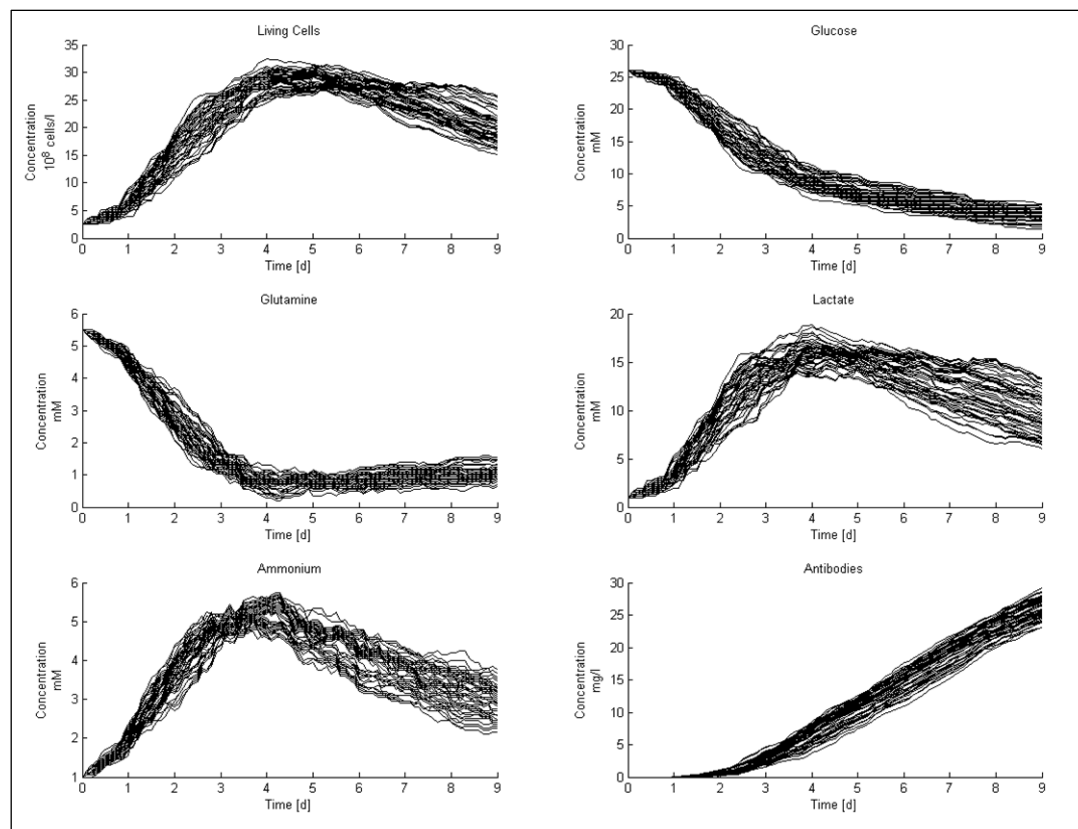


Figure 30: The stochastic simulation results according to Gillespie's algorithm of the stochastic Petri Net model in Figure 29.

4 Discussion

The Petri Net library in Modelica is a good instrument for hybrid modeling of biological systems. The advantages of this approach are:

- The object-oriented modeling language Modelica is able to model discrete Places and Transitions as well as stochastic and continuous ones. The Places and Transitions are models that easily can be changed, modified, or expanded so that further Petri Net extensions can be implemented fast.
- The language allows the realization of hybrid models by combining discrete and continuous processes. The hybrid simulation with discrete events and the solution of continuous differential equations is then performed by the Dymola tool or by another Modelica-tool.
- The Reactions sub-library offers a fast and simple way to build up a model and further personal reactions can be easily added to it.
- The hierarchical modeling concept of Modelica enables a structuring of the models on different levels which is useful when the model is complex and used by different persons with different aims.
- The Petri Net animation of Dymola offers a way to get insight of the Token distribution by large and complex Petri Nets.
- The coupling of Dymola-models and Simulink-models allows the simulation of a model many times and use the arising simulation results for subsequent calculations so that the performing of stochastic simulation, sensitivity analysis and parameter identification in Matlab is possible.
- The Petri Net library can be integrated in other Petri Net modeling tools by parsing the Petri net of the respective tool (e.g. XML-format) to Modelica-text and simulate it via a batch process where the simulation results are saved in a data file.

In this manner the new Petri Net library close the gaps of the Cell Illustrator and other hybrid Petri Net simulation tool and leads to a complete environment for hybrid modeling of biological systems.

References

- [1] Reddy VN, Mavrovouniotis ML, Liebman MN (eds) (1993) Petri net representations in metabolic pathways. Proc Int Conf Intell Syst Mol Biol, vol. 1
- [2] Modelica Association (2010) Modelica - A Unified Object-Oriented Language for Physical Systems Modeling Language Specification Version 3.2
- [3] Petri CA (1966) Communication with automata. Rome Air Development Center, Research and Technology Division
- [4] Valk R (1978) Self-modifying nets, a natural extension of Petri nets. Automata, Languages and Programming:464–476
- [5] Hofestädt R, Thelen S (1998) Quantitative modeling of biochemical networks. In Silico Biology 1(1):39–53
- [6] Bause F, Kritzinger PS (2002) Stochastic Petri Nets. Vieweg
- [7] Heiner M, Gilbert D, Donaldson R (2008) Petri nets for systems and synthetic biology. Formal Methods for Computational Systems Biology:215–264
- [8] Gilbert D, Heiner M (2006) From Petri nets to differential equations-an integrative approach for biochemical network analysis. Petri Nets and Other Models of Concurrency-ICATPN 2006:181–200
- [9] Doi A, Fujita S, Matsuno H, Nagasaki M, Miyano S (2004) Constructing biological pathway models with hybrid functional Petri nets. In Silico Biology 4(3):271–291

- [10] Mosterman PJ, Otter M, Elmqvist H (1998) Modeling Petri nets as local constraint equations for hybrid systems using Modelica
- [11] Fabricius SM (2001) Extensions to the Petri Net Library in Modelica. ETH Zurich, Switzerland
- [12] Otter M, Årzén KE, Dressler I (eds) (2005) StateGraph-a Modelica library for hierarchical state machines. 4th International Modelica Conference
- [13] Uni-Hamburg (2010) Petri Net World. <http://www.informatik.uni-hamburg.de/TGI/PetriNets/>
- [14] Nagasaki M, Saito A, Doi A, Matsuno H, Miyano S (2009) Foundations of Systems Biology: Using Cell Illustrator and Pathway Databases. Springer-Verlag New York Inc
- [15] Nagasaki M, Doi A, Matsuno H, Miyano S (2005) Petri net based description and modeling of biological pathways. Algebraic Biology-Computer Algebra in Biology:19–31
- [16] Dr. Rainer Drath Visual Object Net++. http://www.r-drath.de/Home/Visual_Object_Net++.html
- [17] Matsuno H, Tanaka Y, Aoshima H, Doi A, Matsui M, Miyano S (2003) Biopathways representation and simulation on hybrid functional Petri net. In Silico Biology 3(3):389–404
- [18] Proß S, Bachmann B (eds) (2009) A Petri Net Library for Modeling Hybrid Systems in OpenModelica. Modelica Conference, Como, Italy
- [19] Gilbert D, Heiner M, Lehrack S (eds) (2007) A unifying framework for modelling and analysing biochemical pathways using Petri nets. Proceedings of the 2007 international conference on Computational methods in systems biology. Springer-Verlag
- [20] Fritzson PA (2004) Principles of object-oriented modeling and simulation with Modelica 2.1. Wiley-IEEE Press
- [21] Proß S, Bachmann B, Hofestädt R, Niehaus K, Ueckerdt R, Vorhölter FJ, Lutter P (2009) Modeling a Bacterium's Life: A Petri-Net Library in Modelica. Modelica Conference, Como, Italy
- [22] Gillespie DT (1977) Exact stochastic simulation of coupled chemical reactions. The journal of physical chemistry 81(25):2340–2361
- [23] Dynasim AB (2010) Dymola-Dynamic Modeling Laboratory-User Manual Volume 2, Lund/Sweden
- [24] Birch, JR, Racher AJ (2006) Antibody production. Advanced drug delivery reviews 58(5-6):671–685
- [25] Kurano N, Leist C, Messi F, Kurano S, Fiechter A (1990) Growth behavior of Chinese hamster ovary cells in a compact loop bioreactor. 2. Effects of medium components and waste products. Journal of biotechnology 15(1-2):113–128
- [26] Lao MS, Toth D (1997) Effects of ammonium and lactate on growth and metabolism of a recombinant Chinese hamster ovary cell culture. Biotechnology progress 13(5):688–691
- [27] Ozturk SS, Riley MR, Palsson BO (1992) Effects of ammonia and lactate on hybridoma growth, metabolism, and antibody production. Biotechnol. Bioeng. 39(4):418–431
- [28] Tsao YS, Cardoso AG, Condon RG, Voloch M, Lio P, Lagos JC, Kearns BG, Liu Z (2005) Monitoring Chinese hamster ovary cell culture by the analysis of glucose and lactate metabolism. Journal of biotechnology 118(3):316–327
- [29] Provost A, Bastin G, Agathos SN, Schneider YJ (2006) Metabolic design of macroscopic bioreaction models: application to Chinese hamster ovary cells. Bioprocess and biosystems engineering 29(5):349–366
- [30] Link J (2010) Charakterisierung der Prozessparameter tierischer Zellkulturen in Schüttelinkubatoren. Bachelor Thesis, Bielefeld

- [31] Ozturk SS, Palsson BO (1990) Chemical decomposition of glutamine in cell culture media: effect of media type, pH, and serum concentration. *Biotechnology progress* 6(2):121–128
- [32] Saltelli A, Bolado R (1998) An alternative way to compute Fourier amplitude sensitivity test (FAST). *Computational Statistics & Data Analysis* 26(4):445–460
- [33] Hansen N (2006) The CMA evolution strategy: a comparing review. *Towards a new evolutionary computation*:75–102
- [34] Wilkinson DJ (2006) *Stochastic modelling for systems biology*. Chapman & Hall/CRC