A Petri Net Library for Modeling Hybrid Systems in OpenModelica

Sabrina Proß Bernhard Bachmann

University of Applied Sciences Bielefeld
Am Stadtholz 24
33609 Bielefeld, Germany

sabrina.pross@fh-bielefeld.de bernhard.bachmann@fh-bielefeld.de

Abstract

For modeling continuous and hybrid Petri Nets with dynamic edge weightings, the already existing Petri Net Libraries were further developed. The new library was implemented in OpenModelica using the SimForge GUI, however it works also with Dymola. With the extensions it is possible to model complex biological as well as production or traffic systems.

1. Introduction

The Petri Nets formalism was first introduced by Carl Adam Petri in 1962 [1]. Today Petri Nets can be found in many different areas. Modeling traffic light crossings, production processes or metabolisms of bacteria are only a few examples. In the last years, they were more and more extended for using them for different kinds of problems.

The first Petri Net Library in Modelica was developed by Mosterman et al. and has been further improved by Otter et al. [2; 3]. Herewith the modeling of "normal" Petri Nets or so-called statecharts is possible. "Normal" Petri Nets are bounded and the Places have the capacity one. No time is associated with their behavior. An external signal can enable or disable the Transitions.

This Petri Net Library was further developed by Fabricius [4]. The extensions are:

- Places can contain an integer number of Tokens.
- The Transitions can be timed. They can have either deterministic or stochastic delays.

The Petri Net Library of the present paper bases on the previous ones. The improvements are:

- Continuous Petri Nets with real numbers of Tokens and continuous firing.
- Continuous and discrete Petri Net elements can be connected to model hybrid Petri Nets.
- The edges can have integer weightings in the discrete case and real ones in the continuous case.
- The edge weightings can be functions.
- The Places can contain a maximum and a minimum amount of Tokens.
- Each edge can have an upper and a lower boundary. The number of Tokens of the respective Place must be between these values so that the connected Transition can fire.
- In the discrete case: If a Place does not contain enough Tokens to fire in all possible Transitions, a random variable decides in which Transitions the Place fires. It is the same if a Place cannot gain Tokens from all possible Transitions because of its maximum value.

Firstly, the new Petri Net Library was developed to model biological systems. Metabolites, enzymes and genes are modeled with Places and Transitions represent the reactions between them [5].

Biochemical reactions, which convert one substance to another, proceed continuously. In order to model these, continuous Petri Net elements were implemented.

Furthermore, the speed of these reactions depends mostly on the current concentration of specific substances which can be now displayed by dynamic edge weightings [6].

Additionally, it should be possible to model gene regulation which contains discrete processes as well as continuous ones. Hybrid Petri Nets, which comprise both discrete and continuous Petri Net elements, are now able to model this [7].

DOI: 10.3384/ecp09430014

The edges can also have upper and lower boundaries. This is necessary for modeling substances which only react when a specific concentration is reached.

In the further development of this library, it became clear that these extensions are not only specific for biological systems. They are also useful in other areas. The present paper illustrates this with an example of the steel production process.

2. Petri Nets

A Petri net is a graphical construction to describe and analyze concurrent processes and non-deterministic procedures. It is a graph with two different kinds of nodes: Places and Transitions, whereas only a Place can be connected with a Transition or a Transition with a Place. A Place is symbolized with a circle and a Transition with a rectangle (see Figure 1).

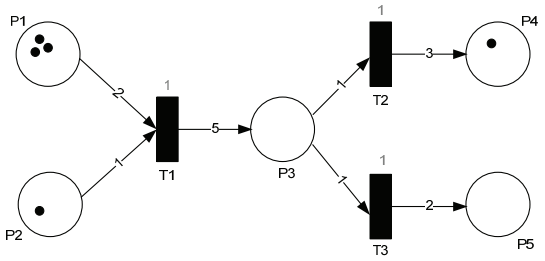

Figure 1: A Petri Net

Every Place contains an integer number of Tokens. In Figure 1, they are shown as black dots. The edges of a Petri Net are provided with weightings and the Transitions with delays. In Figure 1, these are the black numbers at the edges and the grey numbers over the Transitions, respectively. A delay represents the time units that a certain process takes.
A Transition *T* is ready to fire when every Place in its previous area has at least as much Tokens as the edge weighting from the certain Place to *T*. In Figure 1, Transition *T1* is ready to fire because *P1* has three Tokens and must have at least two Tokens, whereas *P2* must contain at least one Token which it actually has. *T2* and *T3* are not ready to fire.
A Transition *T,* that is ready to fire, fires by removing so many Tokens dependent on the respective edge weighting from all of the Places in its previous area. In addition, a specific number of Tokens is laid down in relation to the edge weighting to all of the Places in its past area. After firing *T1* in Figure 1 *P1* has one Token, *P2* zero, *P3* five, *P4* one and *P5* zero (see Figure 2).

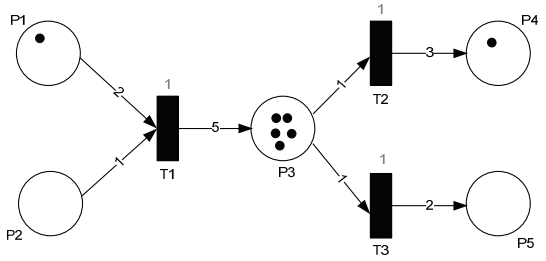

Figure 2: The Petri Net of Figure 1 after firing *T1*

Assumed a Place has only one Token, like in Figure 3, this Token can be either fired in Transition *T1* or in Transition *T2*. Therefore, a decision is necessary which Transition is chosen. One possible solution is that a uniform distributed random variable decides whether *T1* or *T2* gets the Token. It is also thinkable that the edges are weighted. In the example shown in Figure 3, Transition *T1* is chosen with a probability of 80% and *T2* with 20%, respectively.

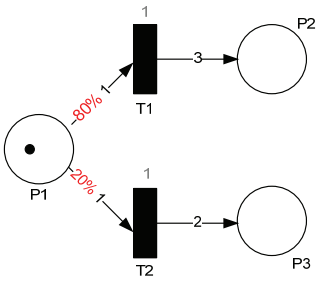

Figure 3: Example for output weightings

One possible extension are Petri Nets with capacities: each Place can only contain a maximum amount of Tokens and must always have a minimum amount of Tokens.
Furthermore, the edges can have threshold and inhibition values. In Figure 4 these are the red numbers in brackets. The first value is the threshold and the second is the inhibition. A Transition is only ready to fire if the connected Places have more or as much Tokens as the threshold value and less or as much as the inhibition value. In Figure 4 is the Transition *T1* ready to fire because this Tansition is only connected with *P1* and *P1* contains three Token. This is more than two and less than five. The Transition *T2* is not ready to fire because the threshold value of the connecting edge between *P1* and *T2* is not achieved.

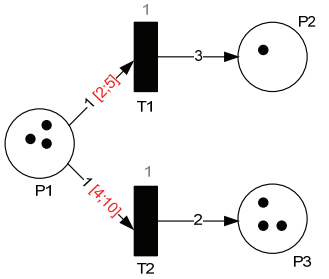

Figure 4: Example for threshold and inhibition values

An additional extension are self modifying Petri Nets which were firstly introduced by Valk [8]. The edge weightings are now functions, which can depend on the current number of Tokens of the respective Places (see Figure 5).

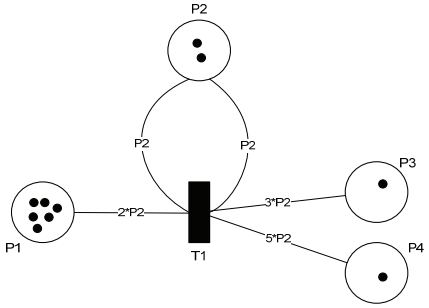

Figure 5: A self modifying Petri Net

2.1. Stochastic Petri Nets

The only difference between Petri Nets described above and stochastic Petri Nets is that the delay is a random variable instead of a fixed value. This random variable can be for example exponential or normal distributed. In the latter case it has to be avoided that the random variable is negative.

2.2. Continuous Petri Nets

A continuous Petri Net is a graphical representation of a differential equation system with the properties of a Petri Net.

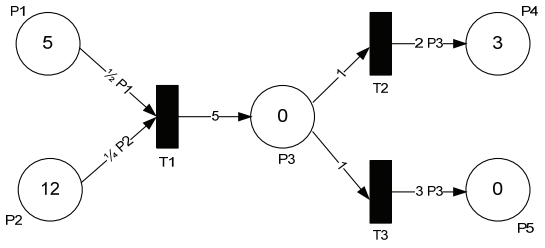

Figure 6: A continuous Petri Net

The number of Tokens in each Place can be real and changes continuously. The edge weightings represent the firing speed for the different branches. The sum of the incoming and outcoming speeds is proportional to the change of Tokens.

The continuous Petri Net in Figure 6 is the graphical representation of the following differential equation system:

$$\frac{dP1}{dt} = -\frac{1}{2} \cdot P1$$

$$\frac{dP2}{dt} = -\frac{1}{4} \cdot P2$$

$$\frac{dP3}{dt} = 5 - 1 - 1 = 3$$

$$\frac{dP4}{dt} = 2 \cdot P3$$

$$\frac{dP5}{dt} = 3 \cdot P3.$$

The difference between the continuous Petri Net and the differential equation system is that if one of the Places in the previous area of the Transition is empty, the Places in the past area will not gain Tokens anymore. If for example *P1* in Figure 6 is empty, then *P3* will not gain any Tokens.

2.3. Hybrid Petri Nets

Hybrid Petri Nets contain discrete and continuous elements. A discrete Transition can be connected with a continuous Place or a continuous Place with a discrete Transition. Connections between discrete Places and continuous Transitions are forbidden.

If a continuous Place is connected with a discrete Transition, the Transition fires by decreasing Tokens continuously in the time of the delay. The slope of the graph is calculated by dividing the edge weighting by the delay. If a discrete Transition is connected with a continuous Place, the Transition fires by adding Tokens continuously in the time of the delay.

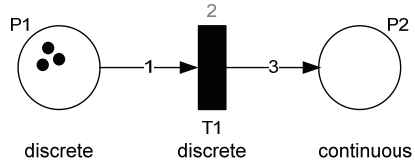

Figure 7: A Hybrid Petri Net

Figure 7 is an example of a hybrid Petri Net. *P1* and *T1* are discrete and *P2* is continuous. After Transition *T1* is ready to fire, *P1* waits two time units before firing one Token. In

these two time units, *P2* receives three Tokens continuously. Figure 8 shows these Token progressions.

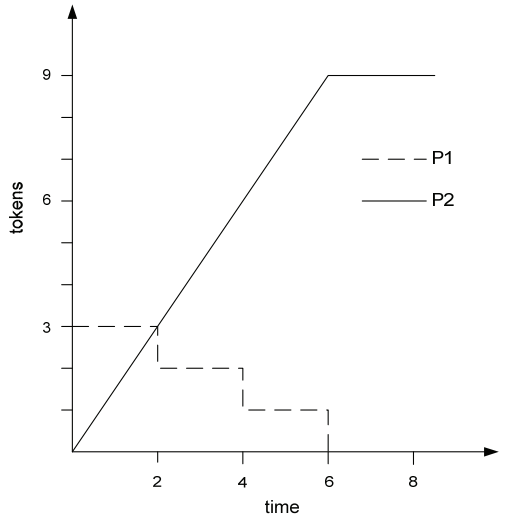

Figure 8: Token progressions of the hybrid Petri Net in Figure 7

3. Petri Net Library

The Petri Net Library is structured in four sub-libraries: Discrete, Continuous, Stochastic and Reactions (see Figure 9). The Reactions Library is discussed in detail in [9].

Figure 9: Structure of the Petri Net Library

Figure 10 shows the icons for the discrete, stochastic and continuous elements of the Petri Net Library. A discrete Place is represented by a turquoise circle and a discrete Transition by a turquoise rectangle. A stochastic Transition is yellow with a turquoise margin and the continuous elements have thick blue margins.

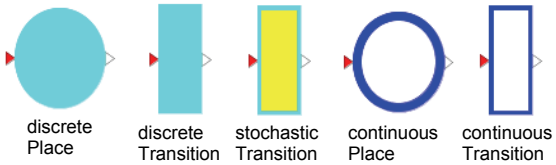

Figure 10: Icons of the Petri Net Library

Every sub-library has general models for Places and Transitions (package partialModels in Figure 11) which are extended to models with fix numbers of input and output connectors. TD21 is for example the denotation for a discrete Transition with two input connectors and one output connector (see Figure 11).

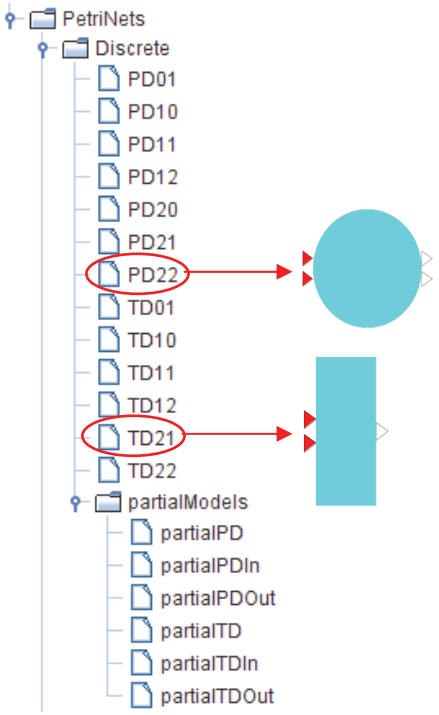

Figure 11: Structure of the discrete Petri Net Library

3.1. Place

In the property-dialog of the discrete and contiuous Place the user can insert the number of Tokens at the beginning of the simulation and the minimum and maximum amount of Tokens that the Place is able to contain (see Figure 12).

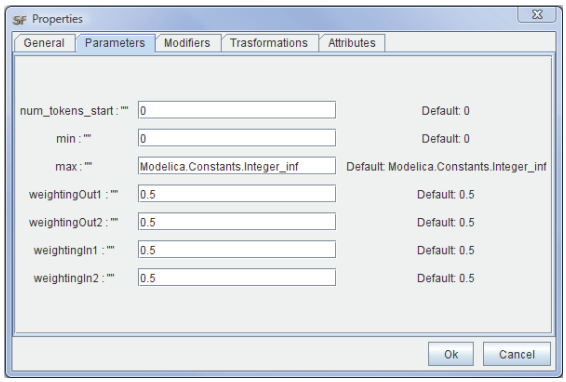

Figure 12: Property-dialog of a discrete Place with two inputs and two outputs

In the discrete Place, it is also possible to determine input and output weightings. If a Place is not able to fire in all activated Transitions due to its lack of Tokens, the edges can be weighted. By means of a uniform distributed random variable, it is decided which Transitions receive Tokens. The same principle is applied if a Place is not able to gain Tokens from all ready to fire Transitions due to its maximum value (cf. section 2).

The current number of Tokens is determined in the Place. To do that, two sums are calculated at first. One is the sum of all Tokens that leave the Place and the other one of all Tokens that come inside the Place. In the discrete case, the new number of Tokens is calculated as follows:

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

In the continuous case, the new number of Tokens is calculated by a differential equation. Of course, additional conditions are considered, i.e. the right hand side of the differential equation may not be negative if the Tokens are equal to the minimum.
After this computation, it is checked whether the Place is empty or full. The current state is reported to the connected input and output Transitions (see Figure 13).
The inState of a continuous Place is only true if the Place is full and the outState is only false if the Place is empty.
The inState of a discrete Place is only true if the Place is full or the Place has just gained Tokens or both. The outState is only false if the Place is empty or the Place has just gained Tokens or both.

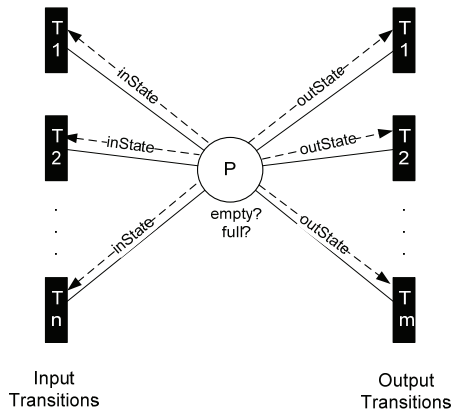

Figure 13: The states of a Place

The inState is the state that is reported to the input Transitions and the outState is the state that is reported to the output Transitions (see Figure 13).

3.2. Transition

In the property-dialog of the discrete Transition, a *delay* can be entered (see Figure 14). If the corresponding Transition is activated, it will take as much time units as keyed in until the Transition fires. In the stochastic case, these are the characteristic values of the corresponding distribution. For example the expectation value *lambda* of an exponential distribution or the expectation value *m* and the standard deviation *s* of a normal distribution. Therefore, the random numbers are calculated by an external C-function.

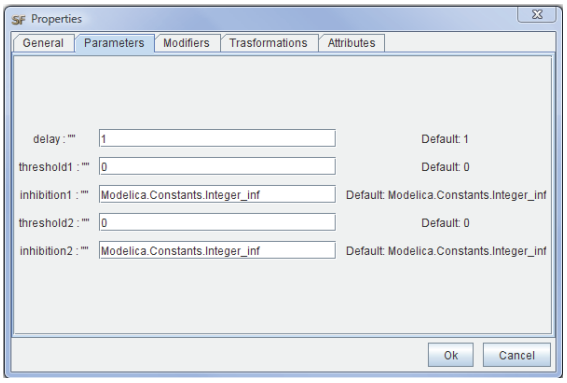

Figure 14: Property-dialog of a discrete Transition with two inputs

There is also the possibility to determine a condition which has to be true so that the Transition is ready to fire. This condition can be entered in 'Modifiers'. For example: con = time>5. In the discrete case a delay can be determine additionally.

The weightings for each edge, which goes in or out of the respective Transition, can be entered in the property-dialog. This has to be done with the aid of 'Modifiers', as functions are also allowed in this case (cf. Section 2). The weightings from the edges that go into the Transition are denoted by *sub1*, *sub2*, … from the top to the bottom. The weightings for the edges, that go out of the Transition, are called *add1*, *add2*, … (see Figure 15).

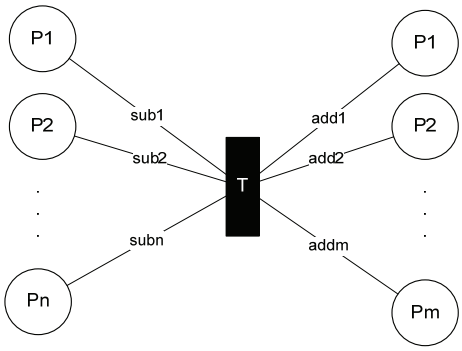

Figure 15: The denotation of the edges weightings

The weightings for Transition *T1* in Figure 6 are keyed in as follows:
sub1 = ½*P1.t
sub2 = ¼*P2.t
add1 = 5.

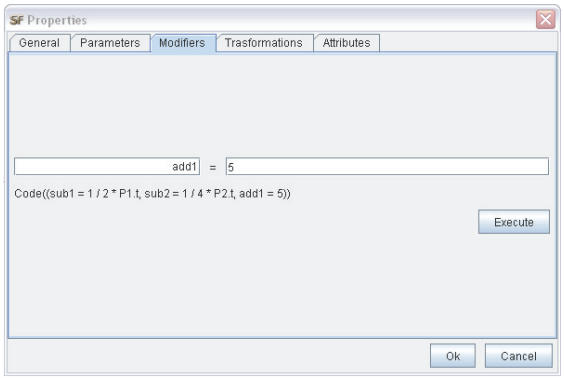

Figure 16: The entry of *sub* and *add* in 'Modifiers'

If functions are entered as edge weightings, the name of the respective Place has to be typed in with the ending '.t' (stands for Tokens).

In the property-dialog the *threshold* and *inhibition* values for each edge, which goes into the respective Transition, can be entered, too (see Figure 14 and Figure 4). The denotation is like the edge weightings. The boundaries for the first input edge from the top are called *threshold1* and *inhibition1*, for the second these are *threshold2* and *inhibition2* and so on.

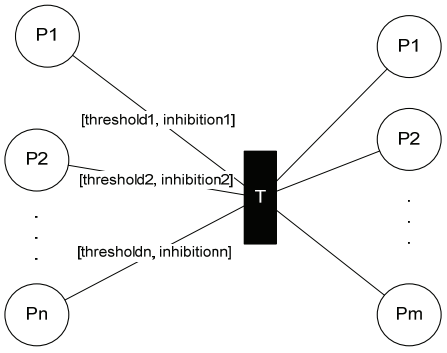

Figure 17: The denotation of the inhibition and threshold values

The Transition decides whether it fires or not. For that, all Transitions check the states of all connected Places. If all states of the input Places are true and none state of the output Places is true and in addition the entered condition is true, the Transition is activated (see Figure 18):

```
activated = if Functions.allTrue(inState)
       and not Functions.anyTrue(outState)
       and con;
```

In the continuous case the activation is equivalent to firing. In the discrete and stochastic case, the activation time is saved and the Transition fires when the corresponding *delay* is passed.

```
when edge(activated) then
     last_activation_time = time;
end when;
delay_passed = activated and
       time - delay >last_activation_time;
```

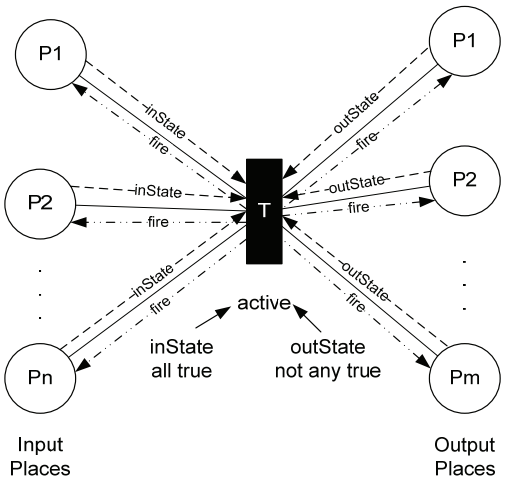

Figure 18: The activation of a Transition

If the Transition fires or not, is reported to the connected input and output Places.

4. Example

Figure 19 shows a simplified example of the production process of crude steel, compare [10].
At first, the iron ore is transported per ship from Brasilia to a stock at the port of Rotterdam. This trip takes generally 14 days. Every 24 days a ship arrives at the port of Rotterdam. But the exact time of arrival is uncertain. The trip can take a little bit longer or shorter because of nature or other conditions. This is modeled with the aid of a stochastic Transition (Transition *ship*). The time of arrival is a normal distributed random variable

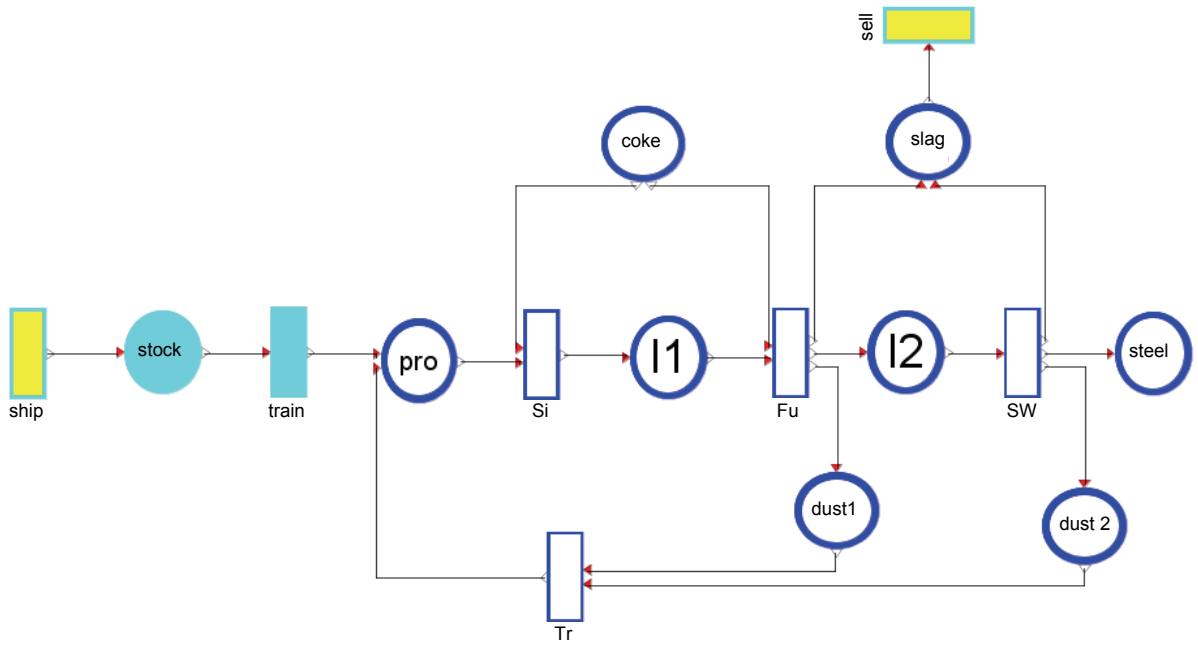

Figure 19: Steel production process

with the expectation value m = 24 and the standard deviation s = 1. A shipload contains 360.000 t iron ore. For this reason *add1* of Transition *ship* is equal to 360.000. The stock at the port can contain at most 720.000 t iron ore. Therefore, the maximal value of the Place *stock* is fixed to 720.000. The start value of this Place is 360.000.

At the next level the iron ore is loaded from the stock to several trains. A train can contain 5000 t iron ore and the drive to the steel production in Duisburg takes 8 hours. The iron ore is delivered "just in time" to the production process. Hence, no other stock is needed. The discrete Transition *train* represents the transport from Rotterdam to Duisburg. The delay is 1/3 day (= 8 hours) and *sub1* = *add1* = 5000. The iron ore (Place *pro*) and the coke (Place *coke*) are mixed in the sintering plant. It accrues the intermediate product sinter (Place *I1*). For one ton employed iron ore 0.2 t coke is needed and 0.73 t sinter is produced. This production step is modeled continuously by means of the Transition *Si*. The edge weightings are the following:

$$sub1 = 0.2 \cdot pro.t$$

$$sub2 = pro.t$$

$$add1 = 0.73 \cdot pro.t.$$

The sinter is further processed in the blast furnace to hot metal (Place *I2*). In addition, the by-products slag (Place *slag*) and blast furnace dust (Place *dust1*) are produced. For one ton employed sinter 0.2 t coke is needed and 0.1 t slag, 0.65 t hot metal and 0.01 t blast furnace dust are produced. The Transition *Fu* displays this. The edges weightings are:

$$sub1 = 0.2 \cdot I1.t$$

$$sub2 = I1.t$$

$$add1 = 0.1 \cdot I1.t$$

$$add2 = 0.65 \cdot I1.t$$

$$add3 = 0.01 \cdot I1.t.$$

The by-product slag is sold to building industry. When 50.000 t slag are produced the company is informed but it is uncertain when the company arrives to pick up the slag and how long this procedure takes. This is modeled with a stochastic Transition with a normal distributed delay (m = ½ and s = 1/8) and *sub1* = 50.000.

In the last production step the hot metal is processed to crude steel (Place *steel*) in the steel works. Slag (Place *slag*) and converter dust (Place *dust2*) are the by-products here.

For one ton employed hot metal 0.13 t slag, 0.8 t crude steel and 0.05 t converter dust are produced. The Transition *SW* represents the steel works. The edge weightings are:

$$sub1 = I2.t$$

$$add1 = 0.13 \cdot I2.t$$

$$add2 = 0.8 \cdot I2.t$$

$$add3 = 0.05 \cdot I2.t.$$

Iron ore can be substituted by blast furnace dust (Place *dust1*) and converter dust (Place *dust2*). This is modeled with the Transition *Tr* and the edges weightings are:

$$sub1 = dust1.t$$

$$sub2 = dust2.t$$

$$add1 = 0.1 \cdot (dust1.t + dust2.t).$$

Following, some simulation results are shown. Figure 20 displays three possible progressions of the stock of iron ore at the port of Rotterdam. Every progression is different because of the stochastic modeling. The stock is limited to 720.000 t iron ore. Hence, this border is not exceed. The iron ore is loaded to trains. Every 8 hours a train drives with 5000 t iron ore to Duisburg. These are the discrete stages in the magnification.

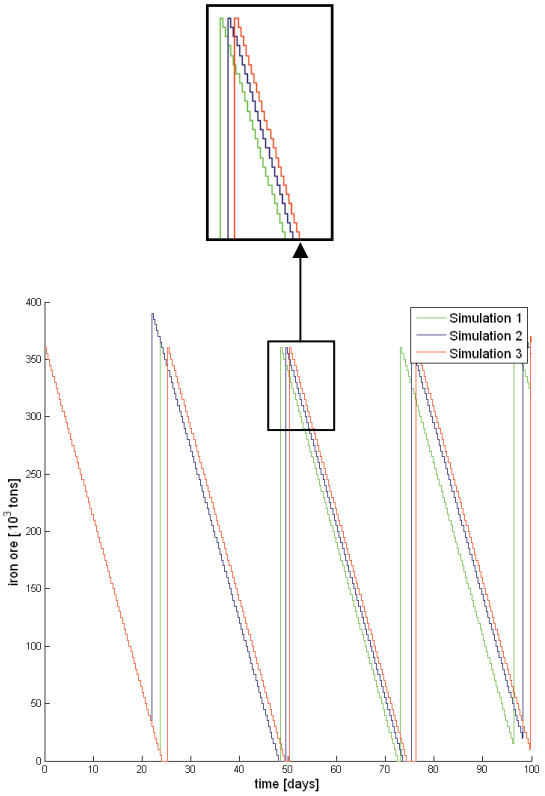

Figure 20: Three simulation results of the iron ore stock at the port of Rotterdam

The iron ore is exhausted in all simulations at specific time points:

Simulation 1 [days]	Simulation 2 [days]	Simulation 3 [days]
48 – 48.5	48 – 49.5	24 – 25.25
72.5 - 73	73.5 – 75.4	49.25 – 50.31
		74.31 – 76.28

This causes bottlenecks in the production process.

The next figure shows the progression of iron ore, sinter and hot metal of simulation 3. The decrease after day 24.3, 49.6 and 74.4 is caused by the exhaused stocks.

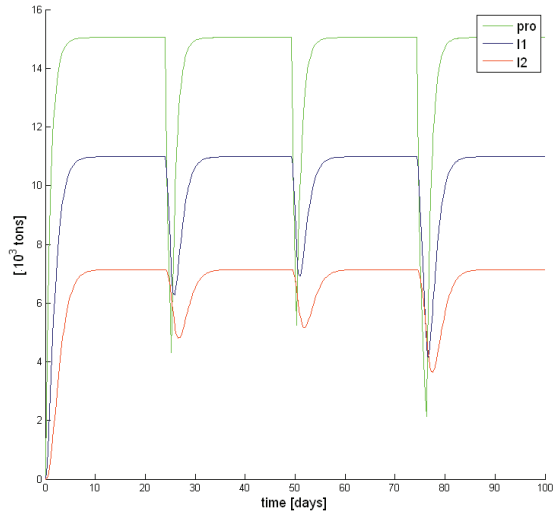

Figure 21: The progressions of iron ore (*pro*), sinter (*I1*) and hot metal (*I2*) (simulation 3)

Figure 22 illustrates the bottleneck in the production process of simulation 3, too. The exhausted stocks are reflected in the amount of crude steel. The production is decreased after every empty stock period.

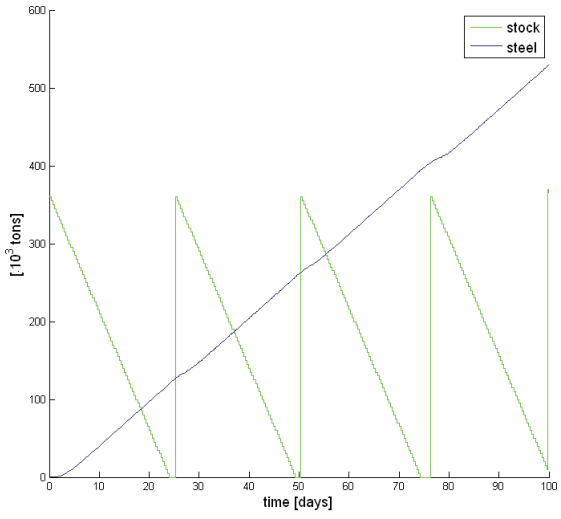

Figure 22: Stock of iron ore (*stock*) and produced crude steel (*steel*) by comparison (simulation 3)

Figure 23 displays the slag progression of simulation 3. When 50.000 t slag are achieved, it is sold to the building industry. The bottlenecks are here visible, too.

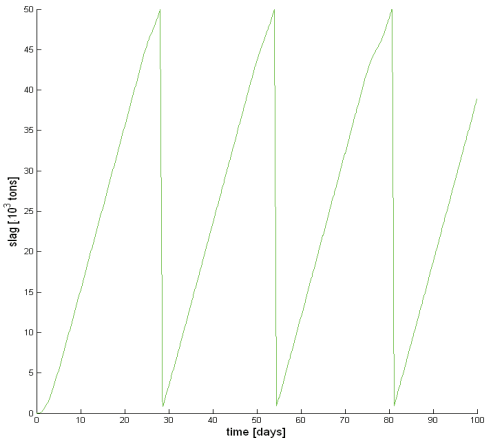

Figure 23: Slag (*slag*) progression of simulation 3

The conclusion of these simulations is that the delivery period of iron ore has to be reduced. The new period has to be big enough that only small bottlenecks appear and small enough that no high stocks accumulate. If for example a period of 22.5 days is chosen, the probability of a bottleneck is 6.7 % and the probability that this bottleneck takes longer than one day is 0.62 %. Now is the task to find the "optimal" solution between bottlenecks and stock costs. Figure 24 shows three simulation results of the progression of the iron ore stock if the delivery period is 22.5 days.

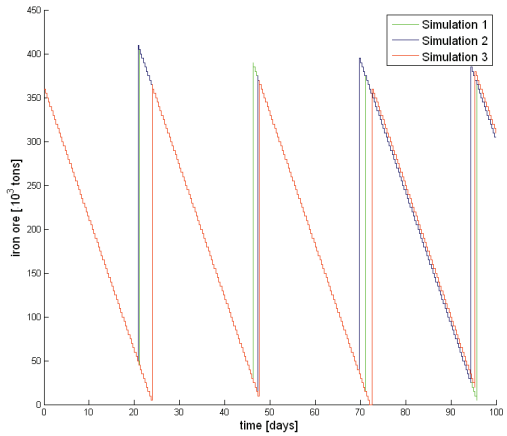

Figure 24: Three simulation results of the iron ore stock with a delivery period of 22.5

5. Conclusions

This paper has shown the new extensions of the Petri Net Library. Now it is possible to model continuous and hybrid Petri Nets with dynamic edge weightings in OpenModelica. These innovations can be applied in different kinds of areas. The paper has demonstrated this with an example of the steel production process. But the Petri Net Library is also useful for the modeling of biological systems [9].

References

[1] **Petri, Carl Adam.** *Kommunikation mit Automaten.* Bonn: Institut für Instrumentelle Mathematik , 1962.

[2] **Mosterman, Pieter J., Otter, Martin and Elmqvist, Hilding.** *Modeling Petri nets as Local Constraint Equations for Hybrid Systems Using Modelica.* Reno, USA , 1998. Summer Computer Simulation Conference .

[3] **Otter, Martin, Arzèn, K.-E. and Dressler, I.** *SateGraph-A Modelica Library for Hierarchical State Machines.* Hamburg , 2005. Modelica Conference. pp. 569-578.

[4] **Fabricius, Stefan M. O.** *Extensions to the Petri Net Library.* 2001.

[5] **Reddy, Venkatramana N., Liebman, Michael N. and Mavrovouniotis, Michael L.** *Qualitative Analysis of Biochemical Reaction Systems*. Compu. Biol. Med. 1996, pp. 9-24.

[6] **Hofestädt, R. and Thelen, S.** *Quantitative Modeling of Biochemical Networks*. In Silico Biology. 1998, 1, pp. 39-53.

[7] **Doi, Atsushi, et al.** *Constructing biological pathway models with hybrid functional Petri nets*. In Silico Biology. 2004.

[8] **Valk, Rüdiger.** *Self-Modifiying Nets: A natural Extension of Petrinets*. *LNCS.* 1978, 62, pp. 464-476.

[9] **Proß, Sabrina, et al.** *Modeling a Bacterium's Life: A Petri-Net Library in Modelica.* Como, Italy , 2009. Modelica conference.

[10] **Dyckhoff, Harald and Spengler, Thomas.** *Produktionswirtschaft.* Berlin Heidelberg : Springer-Verlag, 2005.

