User manual

http://www-dssz.informatik.tu-cottbus.de/DSSZ/
High-level simulation for coloured stochastic Petri nets in Snoopy
August 5, 2023

George Assaf, Monika Heiner and Fei Liu
Snoopy@informatik.tu-cottbus.de

http://www-dssz.informatik.tu-cottbus.de
Data Structures and Software Dependability
Computer Science Department
Brandenburg University of Technology Cottbus-Senftenberg

Contents

List of Figures

List of Tables

1 Introduction

This document aims at introducing interested readers into coloured stochastic Petri nets ($\mathcal{SPN}^{\mathcal{C}}$) as they are implemented in our Petri nets framework *Snoopy*. In this context, *Snoopy* [2] supports both modeling and simulation of different kinds of systems using a bunch of Petri net classes, qualitative and quantitative, alike.

Figure 1 sketches the quantitative (coloured) Petri net classes for exploring model behaviour over time. Modelling covers basically the stochastic ($\mathcal{SPN}$), continuous ($\mathcal{CPN}$), and hybrid ($\mathcal{HPN}$) paradigms. The coloured extensions of these net classes are coloured stochastic Petri nets ($\mathcal{SPN}^{\mathcal{C}}$), coloured continuous Petri nets ($\mathcal{CPN}^{\mathcal{C}}$) and coloured hybrid Petri nets ($\mathcal{HPN}^{\mathcal{C}}$). Moreover, a fuzzy modelling approach has been combined with all these mentioned net classes, e.g. $\mathcal{FSPN}^{\mathcal{C}}$ for fuzzy coloured stochastic Petri nets. It is possible to obtain one net class from an other one by *Snoopy's* export feature. It is worth noting that moving from the coloured world to the uncoloured one involves net unfolding, whereas moving from the uncoloured world to the coloured one requires net folding.

Figure 2 presents *Snoopy*'s main graphical user interface (GUI). The main GUI comprises the following parts as referred to by numbers.

1. **Tool Bar**: offers a direct access to the frequently used commands, e.g. *save*.

2. **Menu Bar**: offers a set of commands; for example, the *File* menu provides commands to *create, open, save, save as* ... etc.

3. **Graph elements**: this sub-window gives access to all graph elements, i.e. the modelling elements that belong to the currently opened Petri net file.

4. **Hierarchy**: this sub-window shows the model hierarchy. Each level in the model is represented using either *coarse* places or *coarse* transitions. By clicking a coarse node (double-click), the corresponding view will be shown for further editing.

5. **Declaration**: this sub-window displays the corresponding Petri net declarations including constants, colour sets, colour functions, and observers.

6. **View space**: this sub-window displays the active Petri net view comprising the canvas, in which the model is graphically constructed.

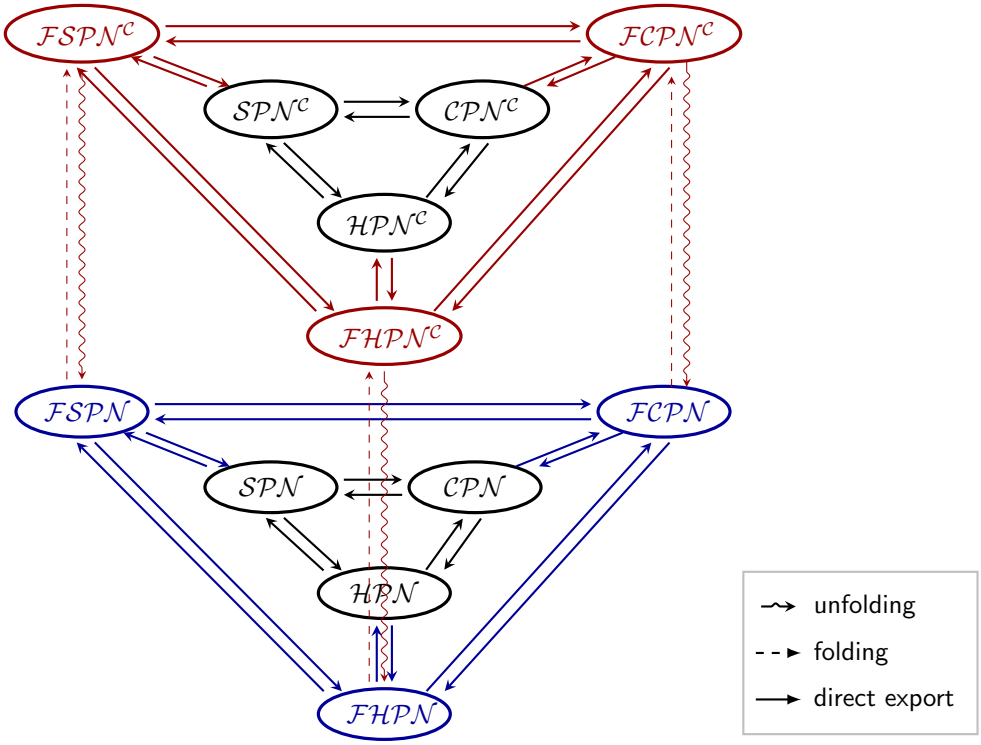

Figure 1: Export relation between some of *Snoopy*'s Petri net classes. Please note that for clarity, three folding/unfolding relations are not shown in this figure ($\mathcal{SPN}$– $\mathcal{SPN}^{\mathcal{C}}$, $\mathcal{CPN}$– $\mathcal{CPN}^{\mathcal{C}}$, $\mathcal{HPN}$– $\mathcal{HPN}^{\mathcal{C}}$).

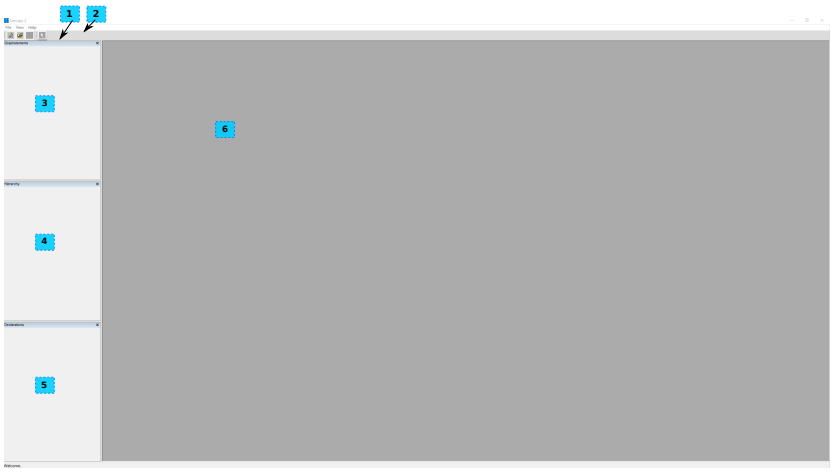

Figure 2: *Snoopy*'s main graphical user interface.

2 Coloured stochastic Petri nets

2.1 Creating a new model

To create a new coloured stochastic Petri net, you must click the new file command either from the tool bar or from the *File Menu*. Then, you must choose the coloured stochastic Petri net class from the list of document templates shown in Figure 3. Once you click on the button *Ok*, an empty $\mathcal{SPN}^{\mathcal{C}}$ template will be created with the default view.

To load an existing Petri net file kept on disk, you can choose the command *open* from the tool bar or from the *File menu*.

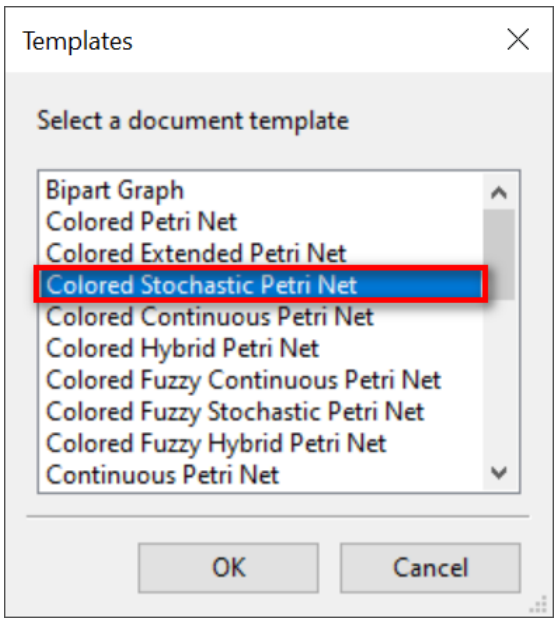

Figure 3: Petri net classes offered by *Snoopy*.

Figure 4 gives the default view of a newly created $\mathcal{SPN}^{\mathcal{C}}$ with a coloured discrete place and a stochastic transition connected by a standard arc. Compare Figure 2, explaining the parts of the window.

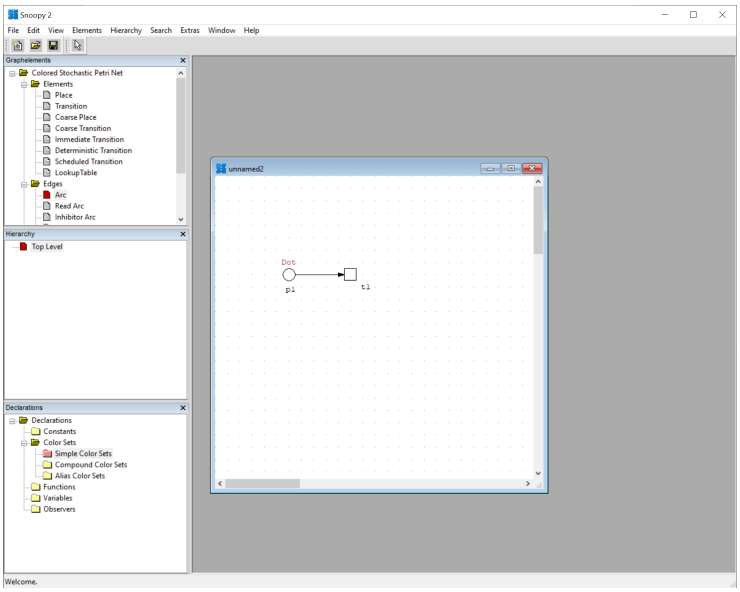

Figure 4: Default view of a newly created stochastic Petri net file. One discrete place and one stochastic transition are drawn on the canvas. The coloured place is assigned the name attribute with the value *p1*, whereas the stochastic transition is assigned the name *t1*.

2.2 Model construction

In the following, we use the *Repressilator* as a running example to demonstrate how to construct an $\mathcal{SPN}^{\mathcal{C}}$ model step by step. Figure 5 shows an $\mathcal{SPN}^{\mathcal{C}}$ of the repressilator.

The repressilator [1, 4] is a synthetic genetic network serving as a model to study gene regulatory networks and the dynamics of genetic oscillations. Furthermore, the repressilator is scaleable by the number of genes involved in the genetic network. Each gene produces a repressor protein that inhibits the expression of the next gene in a cyclic way. The produced protein can also be degraded. Note that the occurrence of each reaction in the system is governed by a Mass-Action kinetics rate function, see Table 1. For teaching yourself about colour expressions, e.g. -x and their interpretations, please check [3].

Table 1: Rate functions of the system's reactions

reaction	***Rate function***	***Kinetic parameter***
generate	$k\_gen \times gene$	k_gen
block	$k\_block \times protein \times gene$	k_block
degrade	$k\_deg \times protein$	k_deg
unblock	$k\_unblock \times blocked$	$k\_unblock$

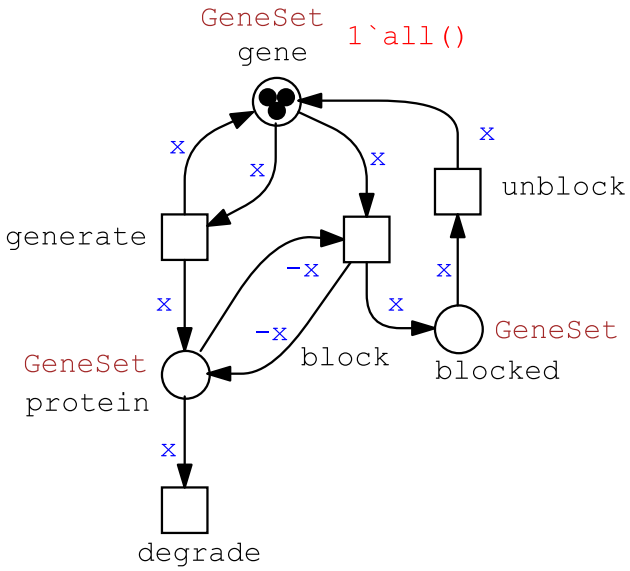

Figure 5: Coloured stochastic Petri net for the repressilator. This model is loaded into *Snoopy's* environment via the open command.

2.2.1 Constant definition

Figure 6 shows the constant definition window in Snoopy. This window comprises the constant grid (referred to by the number 1), displaying the list of defined constants. Each constant has a name, type and value as known from programming languages. Additionally, constants belong to groups. Each group defines a set of constants that have the same usage, for example, constants that are used as kinetic parameters are assigned to the group *parameter*, compare Table 1. Moreover, each group defines a set of values called *value set*. This has a big advantage of running, e.g., simulations with different constant values without having to assign the constants new values. In this window, there are a set of buttons (referred to by the number 2) that permits to add a new constant, deleting existing constants, adding and removing of value sets.

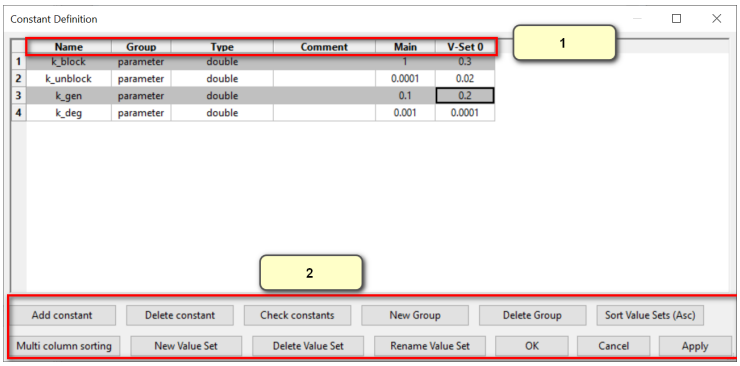

Figure 6: Constant definition window.

2.2.2 Colour set definitions

Figure 7 presents the colour set definitions window, each colour set has a name, type and set of colours. The colour set type can be a simple type such as *enum* and *int* or can be compound colour set. A compound colour sets are defined based on previously defined simple colour sets. For the repressilator, we need only a simple colour set, *Genes* with *enum* type and three colours *a, b, c*. This window comprises a set of buttons for adding/ removing colour sets.

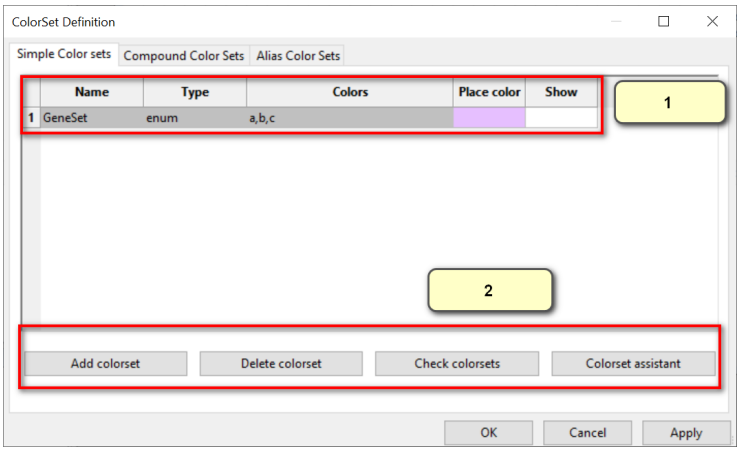

Figure 7: Colour set definitions window.

2.2.3 Coloured places - settings

Figure 8 shows the necessary settings that need to be set. These settings are the following: the colour set of the place (refereed to by number 1). Multi-set colour expression representing the initial marking of the place (referred to by number 2). To define the intial marking of the place, the marking grid of the place has to

be set. the first column of the grid allows to define a colour and the third colour defines the number of tokens of the corresponding colour. The second column is not used for the simple repressilator, as it is used for compound colours, i.e. tuple colours. However, to initialise the net with three genes, we assign the colour function *all()* to the column Colors, which will return all the colours of the colour set *GeneSet* and we assign 1 as a token number. This means this place will be intialised with one token of each colour of the colour set *GeneSet*. This window allows us to add a row to the grid (representing marking)/delete a row from the grid. The name of the place can be set in the name attribute given in the *General* tap.

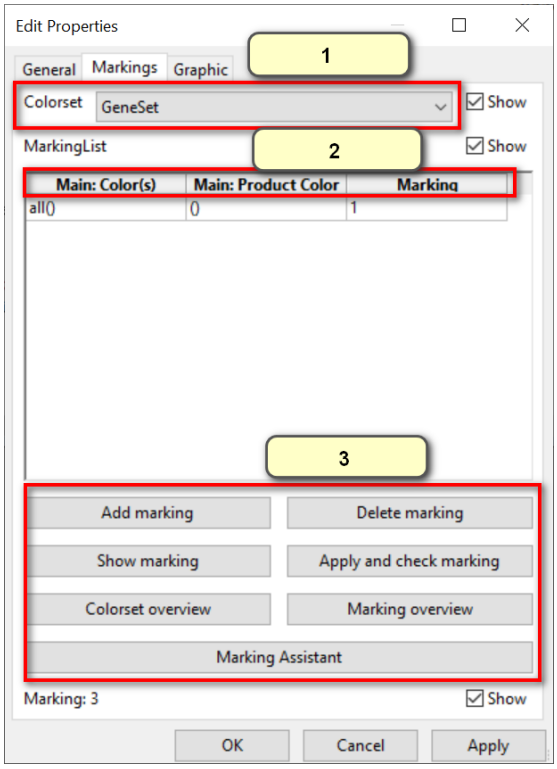

Figure 8: Coloured discrete place - configuration

2.2.4 Coloured stochastic transition - settings

Figure 9 presents the coloured transition properties window. A colour transition is assigned a rate function in the *Functions* tap. in the functions grid. Each function is associated with a predicate (Boolean expression) to define colour-dependent rates. Thsi tap contains a set of assistant functions that help check rate functions syntax, add or remove new functions. The name of the transition is assigned using the name attribute in the *General* tap. A transition can

be assigned a guard using the guard attribute given in the *Guard* tap of the properties window.

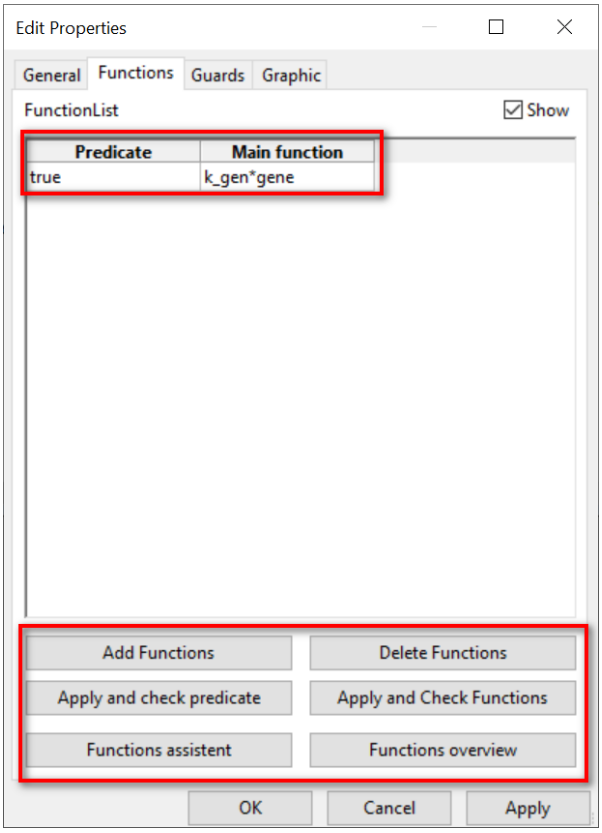

Figure 9: Colour stochastic transition properties.

3 High-level simulation of coloured stochastic Petri nets

After having an $\mathcal{SPN}^{\mathcal{C}}$ model constructed, the high-level simulation mode can be started by selecting the command *Start High-level Simulation/Animation Mode or by pressing F5 button* from the *View* menu, see Figure 10.

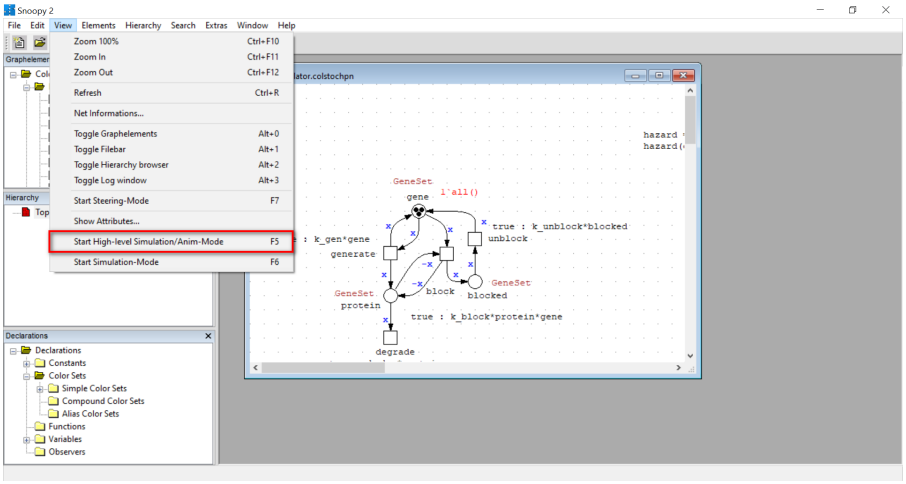

Figure 10: View of the high-level simulation dialogue.

Once High-level simulation mode is toggled, the high-level simulation dialog will be loaded, compare Figure 12. This window is shared between *Snoopy's* coloured animator and the high-level simulation. The configuration section consists of the following configuration sections, as they are labeled by numbers in the taken screenshot:

1. **Animator**: this section comprises a set of buttons that are only used by the user in animation mode. As long as the *Color Simulation Mode* check box is not selected, this set of buttons will be enabled to control the animator, e.g., to start automatic animation of the model.

2. **Step counter**: It is an integer label that is shared between the Animator and high-level simulation, which indicates the current number of steps performed by either the simulator or the animator.

3. **Model configuration**: This section is also shared between the colour animator and the high-level simulation. This section comprises a set of groups of the markings, rate functions, and constants. To explore the model behaviour using different constant values of the model parameters, one can change the value set of the group *parameters* so that there is no need to destroy the simulation window to redefine constant values, see Figure 6.

4. **High-level simulation configuration**: this section is the most important section of the high-level simulation. Note that the following configurations are included within a collapsed window. To view the configurations, you need to expand this window.

    - **Color Simulation Mode**: This option is used to enable high-level simulation, if it is disabled, then the animator is enabled by default.
    - **Interval start**: It is a double value that determines the start time value that the traces will be recorded. Please note that simulation always starts from 0, but this value is to determine the output traces start time.
    - **interval end**: specifies the end simulation time. This value by default is 100 time units.
    - **Interval splitting**: an integer value determines the number of steps that will be recorded. The default value is 100.
    - **Seed Value**: a long integer value that represents the seed value of the random number generator utilised by the high-level simulation algorithm. It is used for reproducing simulation traces using the same settings. Note that the default value will be generated randomly, in case the user does not give an explicit seed value. If the random value is chosen, then the seed value will be shown once, the user starts model simulation (by clicking the start button).
    - **Choose Places**: this button is used to choose which palce instances and/or coloured places that are of interest to be recorded in the output trace file that has previously specified, See Figure 11.
    - **Plot Viewer**: It permits viewing simulation traces by double-clicking the Plot Viewer. Consequently, the Viewer window will be displayed as shown. Currently, It is possible to choose which place to plot as well as to export the simulation traces into CSV format.

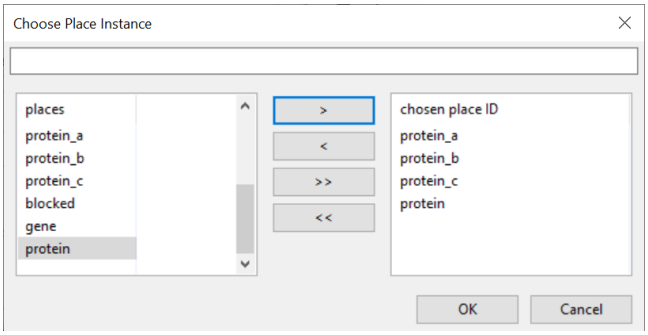

Figure 11: Places selection for output traces. The left sub-window contains the entire place instances and coloured places. The right-hand sub-window contains the selected places that are chosen by the user.

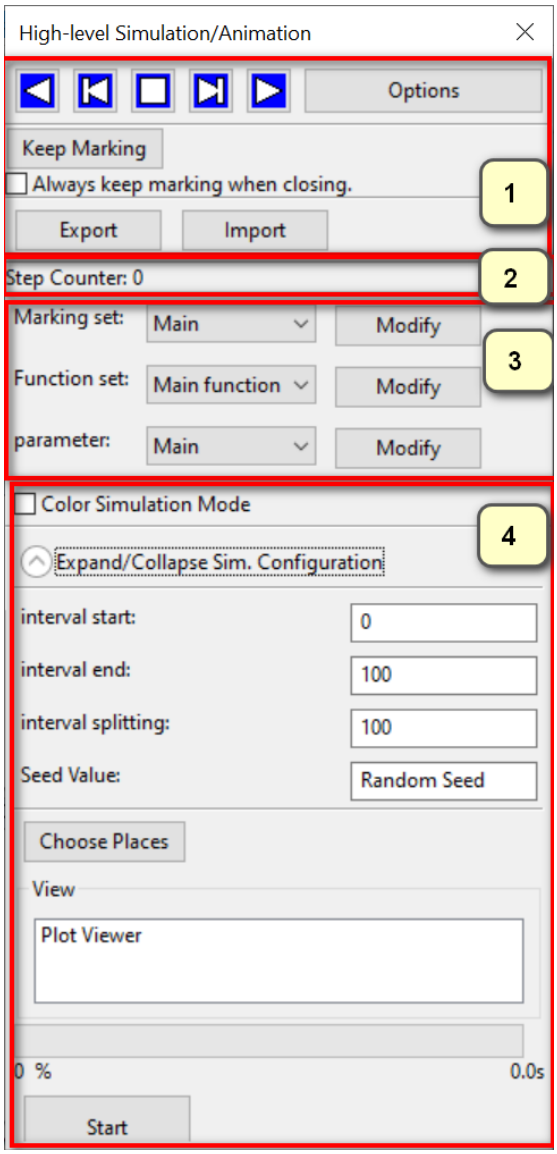

Figure 12: High-level simulation dialogue.

After having the high-level simulation settings configured, the simulation can be started by clicking the start button. The simulation progress will be shown as the simulation proceeds in the progress bar as well as the simulation run time. Figure 13 gives the simulation traces of the place instances of the place *protein* as well as the coloured place *protein*.

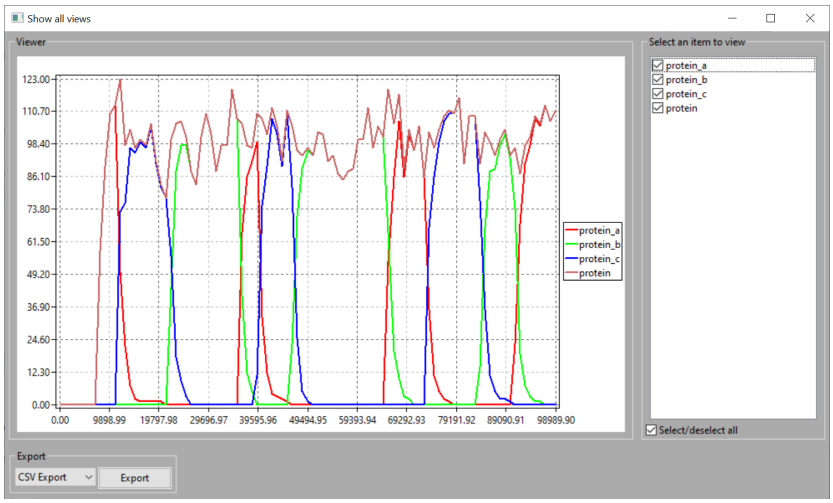

Figure 13: High-level simulation traces of protein places including the coloured place *protein*. Simulation settings: End time: 100,000; interval splitting: 100, seed value: 1895243956.

References

[1] Michael B. Elowitz and Stanislas Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–338, Jan 2000.

[2] Monika Heiner, Mostafa Herajy, Fei Liu, Christian Rohr, and Martin Schwarick. Snoopy – a unifying Petri net tool. In Serge Haddad and Lucia Pomello, editors, *Application and Theory of Petri Nets*, pages 398–407, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[3] F Liu, M Heiner, and C Rohr. Manual for Colored Petri Nets in Snoopy. Technical Report 02-12, Brandenburg University of Technology Cottbus, Department of Computer Science, March 2012.

[4] Fei Liu and Monika Heiner. *Petri Nets for Modeling and Analyzing Biochemical Reaction Networks*, pages 245–272. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.