Reggie DAVIDRAJUH[1], Bożena SKOŁUD[2]
[1]University of Stavanger, Norway
[2]Politechnika Śląska

IMPLEMENTACJA KOLOROWANYCH SIECI PETRIEGO ZA POMOCĄ GPENSIM

Streszczenie. Sieci Petriego są użytecznym narzędziem do modelowania i symulacji rzeczywistych systemów dyskretnych zdarzeniowych. Kolorowe sieci Petriego są użytecznym rozszerzeniem sieci Petriego, które zwiększa moc modelowania sieci Petriego. W tym artykule przedstawiamy symulator ogólnego zastosowania (GPenSIM) do implementacji kolorowych sieci Petriego. GPenSIM działa na platformie MATLAB. Dzięki GPenSIM można rozwijać klasy i rozszerzenia sieci Petriego.

IMPLEMENTING COLORED PETRI NETS WITH GPENSIM

Summary. Petri Nets are a useful tool for the modeling and simulation of real-world discrete-event systems. Colored Petri Nets are a useful Petri Net extension that increases the modeling power of Petri Nets. In this paper, we introduce the General-purpose Petri Net simulator (GPenSIM) for implementing Colored Petri Nets. GPenSIM is a new simulator that runs on the MATLAB platform. GPenSIM provides a Petri Net language, with which Petri Net classes and extensions can be developed.

1. Introduction

Modeling, analysis, and performance evaluation of discrete-event systems are conducted in order to find out useful information about the behavior of the systems, such as the productivity (flow rate) and the existence of bottlenecks and deadlocks. Petri Net is useful for the performance evaluation of discrete-event systems because of its useful properties such as self-documentation and explicit state information [12].

General-purpose Petri Net simulator (GPenSIM) is a new tool for the modeling, simulation, and performance analysis of discrete-event systems [4]. GPenSIM, developed by the first author of this paper, is a toolbox on the MATLAB platform. GPenSIM is being used by some universities around the world because of its simplicity, flexibility, and extensibility. To build Petri Net models, GPenSIM provides a Petri Net language, with which a variety of Petri Net classes and extensions can be developed. GPenSIM also provides some functions for the analysis of Petri Nets.

In P/T Petri Net, tokens residing inside a place are homogeneous. Colored Petri Net is an extended Petri Net that allow the distinction between tokens [3]. In Colored Petri Net, a token can has a data packet attached to it, and this data packet is called the token color. In some Petri Net tools like CPN, data packets (color) can be any data type. However, tokens usually contain one data type only, referred to as the 'color set' of the place.

GPenSIM allows only one type of 'color set', the set of ASCII text strings. Thus, compared to CPN, GPenSIM offers only a rudimentary facility for coloring tokens. However, this simple coloring mechanism when combined with the enabling functions and global variables, usually facilitates modeling any complex, large-scale, real-world discrete-event systems.

In this paper: section-II introduces Colored Petri Nets. Section-III introduces GPenSIM. Section-IV presents an example to show how easily Colored Petri Nets can be implemented with GPenSIM.

2. Colored PEtri Nets

Colored Petri Net is defined as follows [8]:

A Colored Petri Net is a nine-tuple $CPN = (P, T, F, S, C_f, N_f, A_f, G_f, I_f)$, where:

- P is a set of places.
- T is a set of transitions, $P \cap T = T \cap P = \emptyset$,
- F is a set of flows (arcs), from $p_i \in P$ to $t_j \in T$ and from $t_i \in T$ to $p_j \in P$
- S is a set of color, containing the colors (c_i) and the operations on the colors.
- C_f is the color function that maps $p_i \in P$ into colors $c_i \in S$.
- N_f is the node function that maps F into $(P \times T) \cup (T \times P)$.
- A_f is the arc function that maps each flow (arc) $f \in F$ into the expression e.
- G_f is that guard function that maps each transition $t_i \in T$ to a guard expression g. The output of the guard expression should evaluate to Boolean value: true or false.
- I_f is the initialization function that maps each place $p_i \in P$ into an initialization expression. The initialization expression must evaluate to multiset of tokens with a color corresponding to the color of the place C(p).

A. Colored Petri Net: GPenSIM realization

In comparison with CPN tool, realization of Colored Petri Net in GPenSIM is somewhat simpler. For example:

- In GPenSIM, the set of colors are limited to set of ASCII text strings whereas in CPN, colors of any datatype can be added to tokens.
- Also in GPenSIM, the functions C_f, N_f, A_f, G_f, and I_f are all fused together and becomes the enabling function that is coded in the pre-processor files.
- In CPN, logical conditions can be imposed on places, transitions, and arcs. In GPenSIM, only transitions can process logical expressions.

- In CPN, the arc weights can dynamically change due to the value of the logic conditions attached to it. However, in GPenSIM, there is a clear separation of static and dynamic details. Once the static details are coded in the Petri Net Definition File (PDF) file, the arcs weights remains fixed as declared in the PDF.

Even with these simplifications (or perhaps, because of these simplification), GPenSIM is being used to solve many industrial problems as described in the section on discussions.

2. GEPenSIM

General-purpose Petri Net simulator (GPenSIM) is a toolbox on MATLAB platform. GPenSIM is for modeling, simulation, and performance analysis of discrete-event systems. GPenSIM can also be used for control of discrete-event systems. GPenSIM (the current version is v10) is being used by some universities around the world, e.g., in Australia, China, Korea, and the USA [1,2,9,10]. The reasons for the acceptance being the simplicity of learning and using, and its flexibility to incorporate newer functionality [1,2,4,9,10].

Implementing a Petri Net model with GPenSIM usually happens via four MATLAB files (M-files) [4]:

1. Petri Net Definition File (PDF): A PDF declares the static Petri Net graph: the set of places, the set of transitions, and the set of arcs are declared in this file.
2. Main Simulation File (MSF): The MSF declares the initial dynamics (e.g., initial tokens in the places, firing times of the transitions, firing costs of the transitions) and runs the simulations. When the simulation terminates, the code for plotting and printing the simulation results are also coded in this file.
3. The pre-processor file (COMMON_PRE): If there are additional conditions for the enabled transitions to satisfy before firing, these conditions are coded in the COMMON_PRE file.

The post-processor file (COMMON_POST): If there are any post-firing actions to be performed after firing of transitions, these actions can be coded in the COMMON_POST file.

For colored Petri Nets, the functions C_f, N_f, A_f, G_f, and I_f are all fused together and becomes the enabling function that is coded in the COMMON_PRE file.

A. Implementing Colored Petri Nets with GPenSIM

In GPenSIM, each token can become a unique one, identifiable with a unique token identification number (tokID). Also, some tags (‘colors’) can be added to each token. When using colors in GPenSIM, the following issues are important:

1. **Only transitions can manipulate colors:** in the pre-processor COMMON_PRE, one can add, delete, or alter colors of the output tokens.
2. **By default, colors are inherited**: when a transition fires, it collects all the colors from the consumed (input) tokens and then it passes these colors to the deposited (output) tokens. However, color inheritance can be prevented by **overriding**.

3. An enabled transition can select **specific input tokens based on preferred colors**.
4. An enabled transition can also select **specific input tokens based on time**; e.g., the time the tokens are created.
5. The structure of tokens: tokens have a unique identity number (tokID), creation time, and a set of colors.

B. Structure of a Token

A token has a structure that consists of three elements:

1. **tokID** (integer value): a unique token identification number.
2. **creation_time** (real value): the time the token was created by a transition. Please note that this time may be different from (less than or equal to) the time the token was actually deposited into an output place by the transition.
3. **t_color** (set of text strings): a set of colors.

E.g.:

```
              tokID: 101
      creation_time: 30.25
            t_color: {'Tamil', 'Norwegian', 'English',
'German'}
```

C. GPenSIM functions for selection of tokens based on their colors

The table 1. below shows the GPenSIM functions that are used for color manipulation:

Table 1

GPenSIM functions for manipulation of token color

Function	***Description***
tokenAllColor	Select only the tokens that have all of the specified colors.
tokenAny	Select any tokens (without any preference on color).
tokenAnyColor	Select tokens with any of the specified colors; selected tokens must have at least one of the specified color.
tokenArrivedBetween	Select tokens that were deposited into a place between the stated time intervals.
tokenArrivedEarly	Select tokens that were deposited earliest into a place.
tokenArrivedLate	Select tokens that were deposited latest into a place.
tokenColorless	Select only the colorless tokens (tokens with NO color).
tokenEXColor	Select tokens with **exact** colors (no more or no less).
tokenWOAllColor	Exclude a token ONLY if it has all of the specified colors.
tokenWOAnyColor	Exclude a token ONLY if it has ANY of the specified

	colors.
tokenWOEXColor	Exclude a token ONLY if it has **exact** colors as specified
tokIDs	Returns a set of tokIDs of tokens in a place; if the second argument 'nr_tokIDs_wanted' is not specified, then tokIDs of all the tokens in the place is returned.
prnfinalcolors	This function returns colors of the final tokens; final tokens are the tokens that are left in places when the simulation was stopped or completed. In addition to the first input argument (which is the simulation results), the optional second input argument limits the places we are interested. E.g.: `prnfinalcolors(sim, {'p2', 'pNUM1'});`
Prncolormap	This function returns colors of all of the tokens (final tokens as well as previous ones) that were in different places during the simulations; the optional second input argument limits the places we are interested. E.g.: `prncolormap(sim, {'p2', 'pNUM1'});`

3. Coloring in GPenSIM: An Application Example

This section presents an example for implementing Colored Petri Nets with GPenSIM. This example is purposely made to be simple so that the basics of coloring in GPenSIM can be explained. Also, this example is taken from the unpublished user manual Part-II for GPenSIM, written by the first author of this paper.

Figure-1 shows that two transitions **t1** and **t2** are in conflict as they try to grab the same token from the input place **pS**. The cold start transition **tS** deposits token into **pS** at a slower rate (firing time = 10 TU) and this token is being sought by the two transitions **t1** and **t2**. To avoid conflict, let us say that **t1** is allowed to fire 90% of the time, and **t2** for the rest 10%. To realize this, **tS** will add color 't1' to the output token 90% of the time, and the color 't2' for the rest of the time. This means **t1** and **t2** can only take token that bears the respective color.

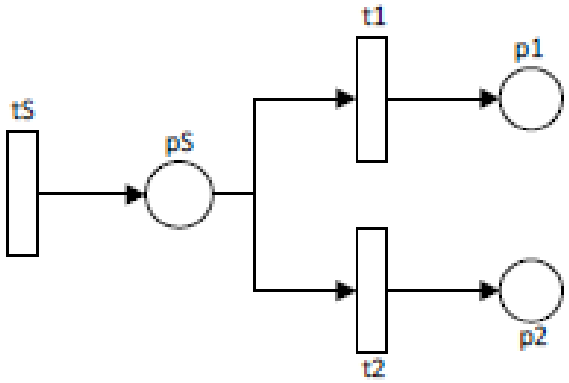

Fig.1. t1 and t2 are in conflict

The static Petri Net graph details are coded in the Petri Net Definition File (PDF). The PDF is given below:

```
% Applicatio Example: Resolving Conflict with color
function [png] = conflict_pdf()
png.PN_name = 'Resolving Conflict with color';
png.set_of_Ps = {'pS', 'p1', 'p2'};
png.set_of_Ts = {'tS','t1','t2'};
png.set_of_As = {'tS','pS',1, ...                % tS
        'pS','t1',1, 't1','p1',1, ... % t1
        'pS','t2',1, 't2','p2',1}; % t2
```

The Main Simulation File (MSF) is for declaring the initial dynamics, to start the simulation, and to plot the results once the simulations are complete. The MSF is given below:

```
% Application Example: Resolving conflict with color
% t1 and t2 are in conflict. t1 has 90% chance, whereas t2 has
10%
clear all; clc;
global global_info
global_info.STOP_AT = 1000; % stp after 1000 TU
png = pnstruct('conflict_pdf');

dyn.m0 = {'pS', 1}; % pS has one token initially
dyn.ft = {'tS', 10, 'allothers', 1};  % firing times of t1 &
t2 is 1 TU
pni = initialdynamics(png, dyn);
sim = gpensim(pni);
plotp(sim, {'p1', 'p2'});
prnstate();
```

The pre-processor file COMMON_PRE is the one in which the logical expressions for color manipulations are coded. These logical expressions serve two purposes:

1. Allow an enabled transition to start firing, if the enabling conditions are satisfied.
2. Allow that transition to manipulate colors of the tokens.

In COMMON_PRE (given below), **tS** adds color 't1' to the output token 90% of the time (color 't2' for the 10% of the time). **t1** selects token with color 't1' only, and similarly, **t2** selects token with color 't2' only.

```
function [fire, transition] = COMMON_PRE (transition)
tname = transition.name;
switch tname
    case 'tS'
        random_num = rand; % 0 to 1
        %%%%   t1 has 90% chance, whereas t2 has only 10%
        if and(ge(random_num,0), lt(random_num,0.9))
            color = 't1'; % t1 can fire
        else
            color = 't2'; % t2 can fire
        end
        transition.new_color = color;
        fire = 1; % always fire tE as it is not in conflict

    case 't1'
        % From pS, t1 takes only the token with color 't1'
       tokID = tokenAnyColor('pS',1, {'t1'});%select token
with color 't1'
        fire = tokID; % fire only if the token has color 't1'

    case 't2'
        % From pS, t2 takes only the token with color 't2'
        tokID = tokenAnyColor('pS',1,{'t2'});% select token
with color 't2'
        fire = tokID; % fire only if the token has color 't2'

    otherwise
        disp('Unknown method.')
end
```

The simulation results (print the final states with '**prnstate**') show that **t1** has fired about 90% of the time (as **p1** has 90% of the tokens).

```
90p1 + 9p2 + pS
```

4. Discussion

GPenSIM supports many well-known Petri Net extensions and subclasses such as Petri Nets with Inhibitor Arcs, Petri Nets with Priority, enabling functions, and Colored Petri Net. Due to flexibility, it is also easy to implement newer extensions with GPenSIM (e.g., Cohesive Place-Transition Nets with Inhibitor Arcs [6]). This paper describes the implementation of Colored Petri Nets with GPenSIM.

As shown in the section-II, the implementation of Colored Petri Nets in GPenSIM is much simplified in comparison with the CPN tool. Perhaps, because of these simplification, many users are starting to adopt GPenSIM as their modeling tool.

GPenSIM offers only a crude set of functionality for color manipulations. However, even with these crude functionalities, GPenSIM is being used to solve many industrial problems (e.g., Airport capacity modeling [5], Norwegian Atlantic Salmon fish supply chain [11], and modeling Flexible Manufacturing Systems [7]).

REFERENCES

1. Cameron A., Stumptner M., Nandagopal N., Mayer W., Mansell T.: Rule-based peer-to-peer framework for decentralised real-time service oriented architectures. Sci. Comput. Program. 2015, 97, 202–234.
2. Chang H.: A Method of Gameplay Analysis by Petri Net Model Simulation. J. Korea Game Soc. 2015, 15, 49–56, doi:10.7583/JKGS.2015.15.5.49.
3. Colored Petri Net Tool (CPN): http://cpntools.org/
4. Davidrajuh R.: Modeling Discrete-Event Systems with GPenSIM: An Introduction; Springer, 2018.
5. Davidrajuh R., Lin B.: Exploring airport traffic capability using Petri net based model. Expert Systems with Applications, 38(9), 2011, 10923-10931.
6. Davidrajuh R., Saadallah, N.: Implementation of "Cohesive Place-Transition Nets with Inhibitor Arcs" in GPenSIM. In IEEE 2016 Asia Multi Conference on Modelling and Simulation. 2016, p. 4-6.
7. Davidrajuh R., Skolud B., Krenczyk D.: Performance Evaluation of Discrete Event Systems with GPenSIM. Computers, 7(1), 2018, 8.
8. Jyothi S.D.: Scheduling Flexible Manufacturing System Using Petri-Nets and Genetic Algorithm; Department of Aerospace Engineering, Indian Institute of Space Science and Technology: Thiruvananthapuram, India, 2012.
9. Jensen K.: Coloured Petri Nets (2 ed.). Berlin: Heidelberg. ISBN 3-540-60943-1, 1996.
10. Lopez F.; Barton K.; Tilbury D.: Simulation of Discrete Manufacturing Systems with Attributed Hybrid Dynamical Nets. Unpublished work, 2017.
11. Melberg R., Davidrajuh R.: Modeling Atlantic salmon fish farming industry. In Industrial Technology, 2009. ICIT 2009. IEEE International Conference on IEEE, pp 1-6.
12. Peterson J.L.: Petri Net Theory and the Modeling of Systems; Prentice-Hall: Englewood Cliffs, NJ, USA, 1981