Revisiting Petri Net Modeling of the Cigarette Smokers' Problem: A GPenSIM Approach

Davidrajuh, Reggie

Abstract:
Petri Nets is a family of modeling formalisms, with different interpretations and abstraction levels; the Petri nets family of modeling formalisms includes ordinary Petri Net, generalized Petri Net, and also all other extensions e.g. Colored Petri Net, Petri Net with priority, and Petri Net with inhibitor arcs. The availability of different formalisms of the Petri Net family is considered as its main strength, since a discrete system can be modeled with different Petri Net formalism, in order to achieve different abstraction levels and different interpretations. This work presents an application of a new Petri Net simulator known as GPenSIM; GPenSIM is used to model the classical Cigarette Smokers' Problem, using the different Petri Net extensions implemented in GPenSIM. Thus, this work can be considered as a benchmark to test the strengths and weakness of GPenSIM, in modeling and simulation of discrete event systems.

INTRODUCTION

Petri Net was introduced in 1962 as a mathematical tool for modeling and simulation of discrete event systems [1]. Petri Net, being a simple bipartite graph, was successfully used for modeling of discrete systems in different fields, like computer science, chemistry, biological science, and social science [2].

In this paper, the classical Cigarette Smokers' Problem (CSP) is modeled once again, using a new Petri Net simulator known as General Purpose Petri Net Simulator (GPenSIM) [4]. The different Petri Net extensions that are implemented in the GPenSIM (such as inhibitor arcs, priorities, and color) are used to model the CSP problem.

Literature study shows that the CSP is already solved many times, using a variety of Petri Net extensions [5]; [6]; [7];[8]. Hence, this work providing the yet another solution to the classical CSP, serves only as a benchmark to show the strengths and weakness of GPenSIM, in modeling and simulation of discrete event systems.

In this work: section-II describes the CSP. Section-III presents a very concise introduction to GPenSIM. Section-IV presents a number of solutions to the CSP, using different Petri Net extensions implemented in GPenSIM.

THE CIGARETTE SMOKERS' PROBLEM

Suhas Patil originally described the cigarette smokers' problem (CSP) in a technical report at MIT in 1971 [3].

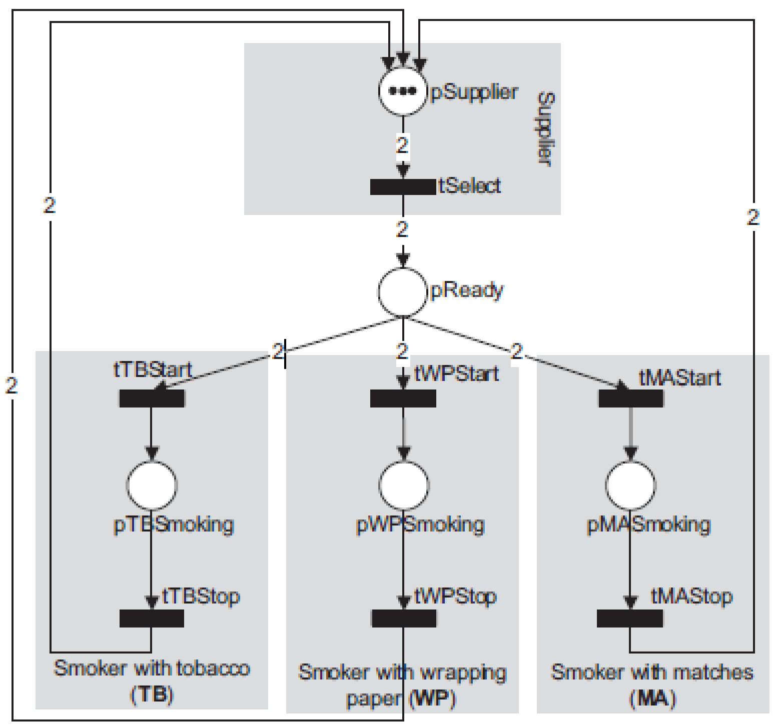


Figure 1: P/T Petri Net model

A. The four agents of the CSP

The CSP involves four agents (or processes); one of the agents is the supplier and the rest three are smokers:

1. Supplier: supplier possesses infinite amount of the three basic ingredients for making cigarettes: tobacco, wrapping paper, and match
2. Smoker with tobacco (TB): this agent possesses infinite amount of tobacco and expects the other two ingredients (wrapping paper and match) from the supplier,
3. Smoker with wrapping paper (WP): this agent possesses infinite amount of wrapping paper and expects the other two ingredients (tobacco and match) from the supplier, and
4. Smoker with match (MA): this agent possesses infinite amount of matches and expects the other two ingredients (tobacco and wrapping paper) from the supplier.

B. The procedure for smoking

Let us imagine that the supplier and the three smokers are sitting around a table. The procedure consists of the following steps:

1. Though the supplier has an infinite supply of all the three ingredients for making cigarettes, the supplier arbitrarily selects only two of the ingredients and places them on the table.
2. The smoker who has the remaining ingredient can take the two ingredients, and then make and smoke a cigarette. Upon completion of smoking, the supplier will be signaled.
3. For the next cigarette, the supplier again arbitrarily selects two of the three ingredients and places them on the table. This cycle is repeated forever.

Figure-1 shows a P/T Petri Net model for the CSP, which will not work properly; this because, there is no way of preventing a smoker from grabbing the ingredient he already possess thus preventing the other needy one from taking it. For example, if the supplier puts tobacco and wrapping paper on the table, the smoker with tobacco (or the smoker with wrapping paper) may grab the two items creating a deadlock situation, as he blocks himself from smoking as well as the smoker with matches. As there is no mechanism in P/T Petri Net to prevent the danger of deadlock, P/T Petri Net cannot be used solve the CSP.

GPENSIM

GPenSIM is designed for increasing the decision making potentials from the Petri Net models, by implementing a variety of Petri Net extensions, supplemented by utility functions. GPenSIM is available from the website cited in the reference as [4]. Reference [9] gives an introduction to GPenSIM.

As GPenSIM is implemented on MATLAB platform, a Petri Net model created by GPenSIM consists of M-files. A typical Petri Net model consists of the following files:

- Petri Net Definition File (PDF): PDF defines the static Petri Net structure: this files declares the set of places, the set of transitions, and the set of arcs
- A main simulation file (MSF): MSF declares the initial dynamics of the model. E.g. the initial marking (tokens in different places), firing times of the non-primitive transitions, and system resources (explained later).
- COMMON_PRE: this file defines the enabling conditions (also known as guard conditions) of the transitions. Whenever a transition is enabled, the conditions defined in this file are checked to determine whether the enabled transition can start firing.
- COMMON_POST: this file defines the post-firing actions (if any) for transitions. For example, after firing, a transition may release some resources that were used during the firing; releasing the resource can be coded in the COMMON POST file.

SOLVING THE CIGARETTE SMOKERS' PROBLEM WITH GPENSIM

In this section the cigarette smokers' problem (CSP) is solved in four different ways, the first three solutions using different Petri Net extensions and the final one with GPenSIM convenience:

1. Inhibitor arc extension,
2. Color extension,
3. Priority extension, and
4. Using GPenSIM resources.

The following subsections present the solutions.

A. Petri Net model with inhibitor arc extension

Figure-2 shows the Petri Net model for the CSP that uses the inhibitor arc extension. On the supplier side, the inhibitor arcs make sure that the supplier selects two different ingredients, arbitrarily. The rest of the net make sure that the smoker without the two ingredients takes these two and start smoking.

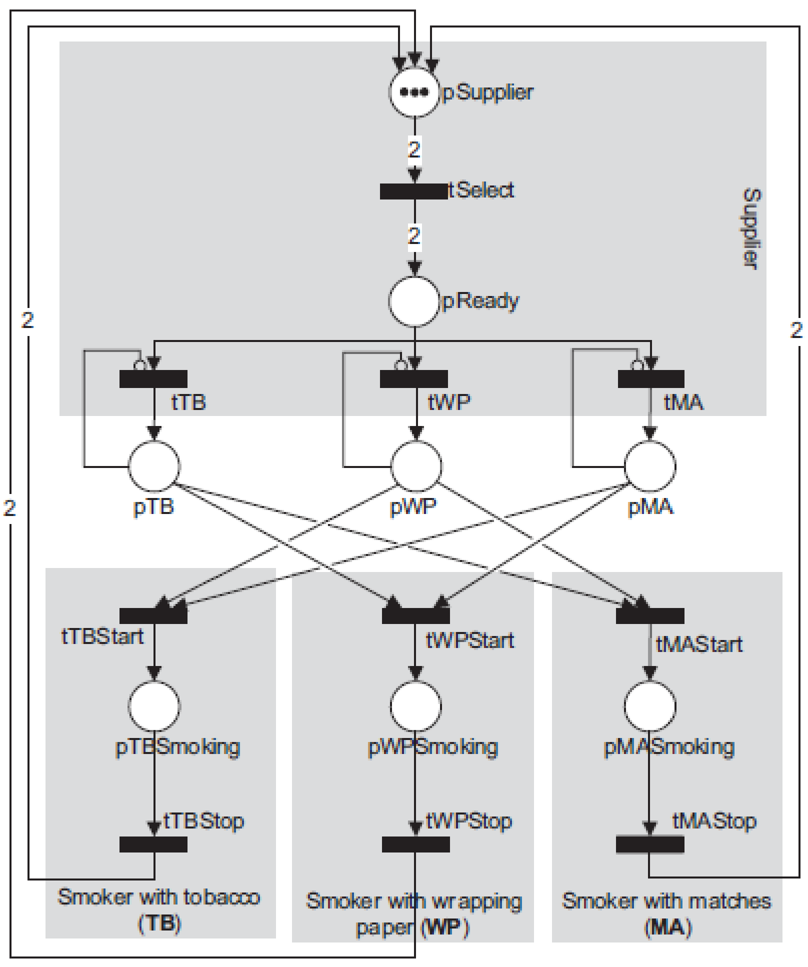


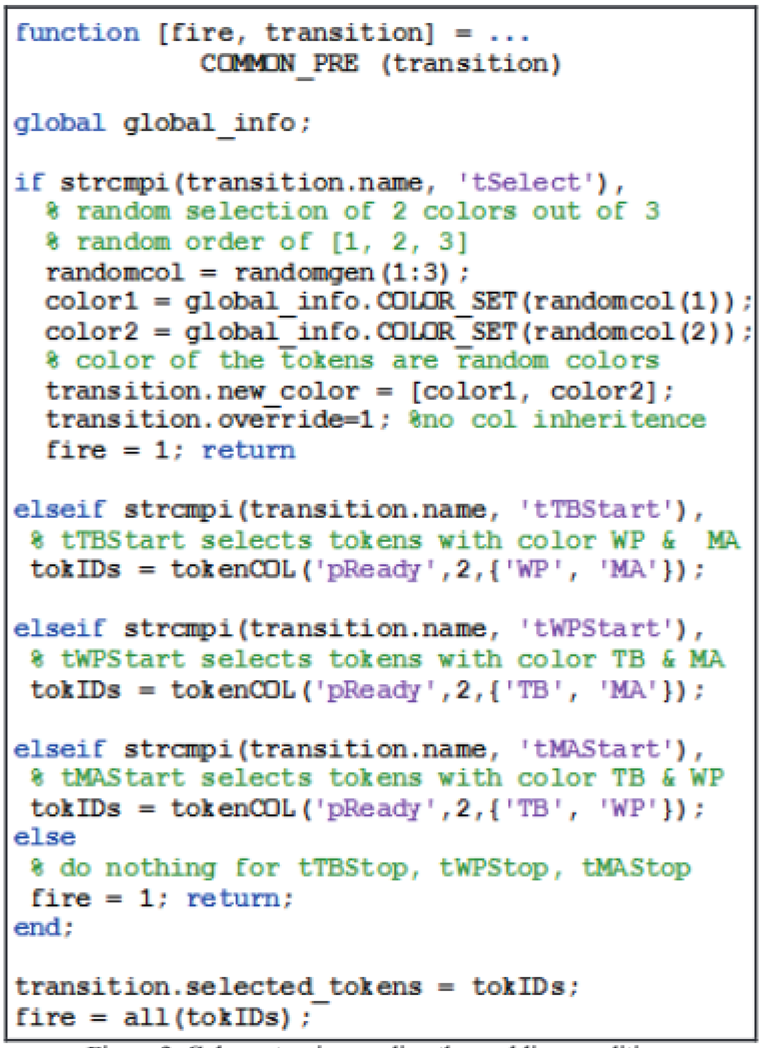
Figure 2: Petri Net model with inhibitor arcs

Figure-10 presents the reachability tree of the Petri Net model. Thus, using the inhibitor arc extension, analytical part of the model seems somewhat improved over its *P/T* model, even though it should remain the same as for bounded systems modeling capability of P/T Petri Net and the formalism with inhibitor arcs should have the same effect.

B. Petri Net model with Color extension

The Petri Net structure (static graph) of the model with color extension is exactly the same as the P/T Petri Net–the one that is shown in figure-1. The color extension is mainly to impose the following actions and enabling conditions on transitions:

1. When the transition 'tSelect' fires, it adds two colors (representing the two ingredients selected arbitrarily) to the tokens that are to be deposited into the output place 'pReady'.
2. Transition 'tTBStart' can only select tokens from the input place 'pReady', if the token has colors 'Wrapping Paper' and 'Match'. Similar enabling conditions are imposed on the transitions 'tWPStart' and 'tMAStart'.



```
function [fire, transition] = ...
              COMMON_PRE (transition)

global global_info;

if strcmpi(transition.name, 'tSelect'),
  % random selection of 2 colors out of 3
  % random order of [1, 2, 3]
  randomcol = randomgen(1:3);
  color1 = global_info.COLOR_SET(randomcol(1));
  color2 = global_info.COLOR_SET(randomcol(2));
  % color of the tokens are random colors
  transition.new_color = [color1, color2];
  transition.override=1; %no col inheritence
  fire = 1; return

elseif strcmpi(transition.name, 'tTBStart'),
 % tTBStart selects tokens with color WP &  MA
 tokIDs = tokenCOL('pReady',2,{'WP', 'MA'});

elseif strcmpi(transition.name, 'tWPStart'),
 % tWPStart selects tokens with color TB & MA
 tokIDs = tokenCOL('pReady',2,{'TB', 'MA'});

elseif strcmpi(transition.name, 'tMAStart'),
 % tMAStart selects tokens with color TB & WP
 tokIDs = tokenCOL('pReady',2,{'TB', 'WP'});
else
 % do nothing for tTBStop, tWPStop, tMAStop
 fire = 1; return;
end;

transition.selected_tokens = tokIDs;
fire = all(tokIDs);
```

Figure 3: Color extension: coding the enabling conditions and the actions in the COMMON_PRE file

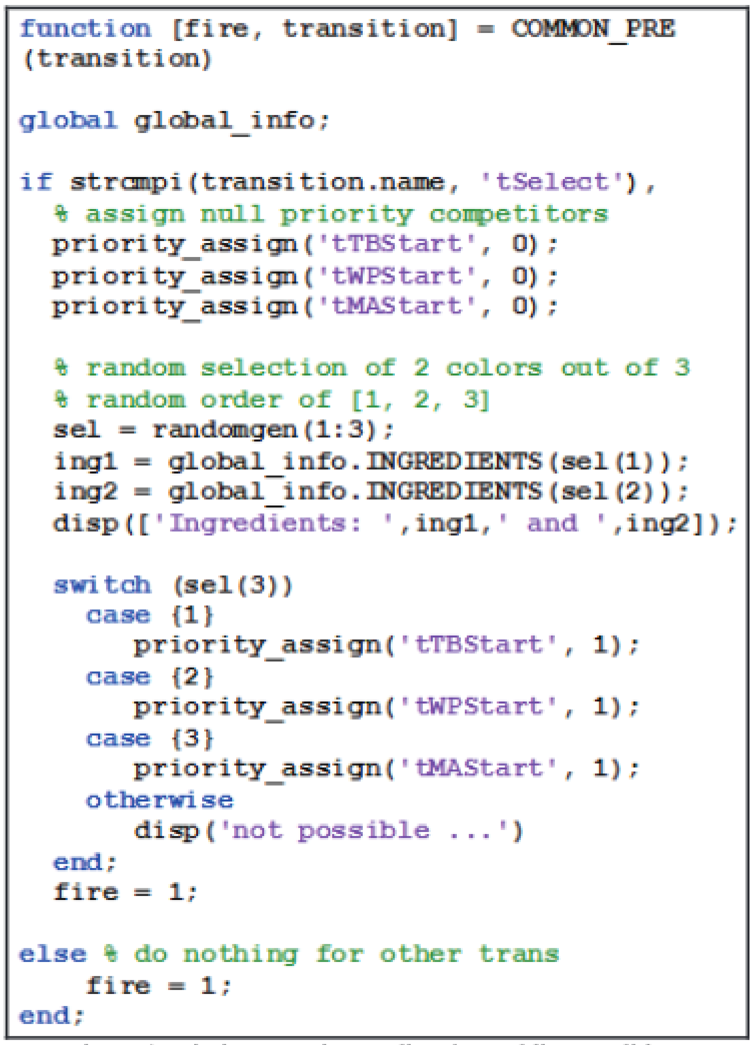
Figure-3 shows the COMMON_PRE file coding the enabling conditions and the actions imposed upon the transitions. The first part of the figure-3 shows the transition ‘tSelect’ adding two random colors to the tokens deposited into the place ‘pReady’ the colors represent the selected ingredients. The second part of the figure-3 shows the transitions ‘tTBStart’, ‘tWPStart’, and ‘tMAStart’, selecting the appropriate tokens from the place ‘pReady’.

The reachability tree obtained for the Petri Net model with color extension is exactly the same as the one obtained from the P/T Petri Net model. This seems correct as the modeling capability of P/T Petri Net and the one with color extension have the same modeling capability.

C. Petri Net model with Priorities

In this model too, the Petri Net structure (static graph) of the model with priorities is exactly the same as the one shown in figure-1. The manipulation of priorities of transitions when they are in conflict, is to make sure that the transition that is supposed to fire is given higher priority among the competing ones, so that it will fire before the other ones. The following actions and enabling conditions are imposed on the transitions:

1. When the transition ‘tSelect’ fires, it selects the two ingredients and increases the priority of the corresponding transition. E.g. if the ingredients ‘TB’ and ‘WP’ are selected, then the transition ‘tMAStart’ will be assigned highest priority over the other two transitions ‘tTBStart’ and ‘tWPStart’, so that ‘tMAStart’ will be able to grab the tokens from ‘pReady’.



```
function [fire, transition] = COMMON_PRE
(transition)

global global_info;

if strcmpi(transition.name, 'tSelect'),
  % assign null priority competitors
  priority_assign('tTBStart', 0);
  priority_assign('tWPStart', 0);
  priority_assign('tMAStart', 0);

  % random selection of 2 colors out of 3
  % random order of [1, 2, 3]
  sel = randomgen(1:3);
  ing1 = global_info.INGREDIENTS(sel(1));
  ing2 = global_info.INGREDIENTS(sel(2));
  disp(['Ingredients: ',ing1,' and ',ing2]);

  switch (sel(3))
    case {1}
       priority_assign('tTBStart', 1);
    case {2}
       priority_assign('tWPStart', 1);
    case {3}
       priority_assign('tMAStart', 1);
    otherwise
       disp('not possible ...')
  end;
  fire = 1;

else % do nothing for other trans
    fire = 1;
end;
```

Figure 4: Priority extension: coding the enabling conditions and the actions in the COMMON_PRE file

Figure-4 shows the COMMON_PRE file encoding the conditions and the actions imposed upon the transitions.

The first part of the figure-4 shows the transition ‘tSelect’ assigning null priority to all the three competing transitions. In the second part, two ingredients are randomly selected and the appropriate transition to receive the ingredients (tokens from the place ‘pReady’) is identified and its priority is increased.

The reachability tree obtained for the Petri Net model with priority extension is again the same as the one obtained from the P/T Petri Net model.

D. Using the GPenSIM Resources

As already stated in the beginning of this section, the first three models based on Petri Net extensions (inhibitor arcs, color extension, and priority extension) that are necessary to model a system that is otherwise not possible model with P/T Petri Net; thus, these extensions are out of necessity. This subsection presents an enhancement (convenience) to the modeler to create compact models.

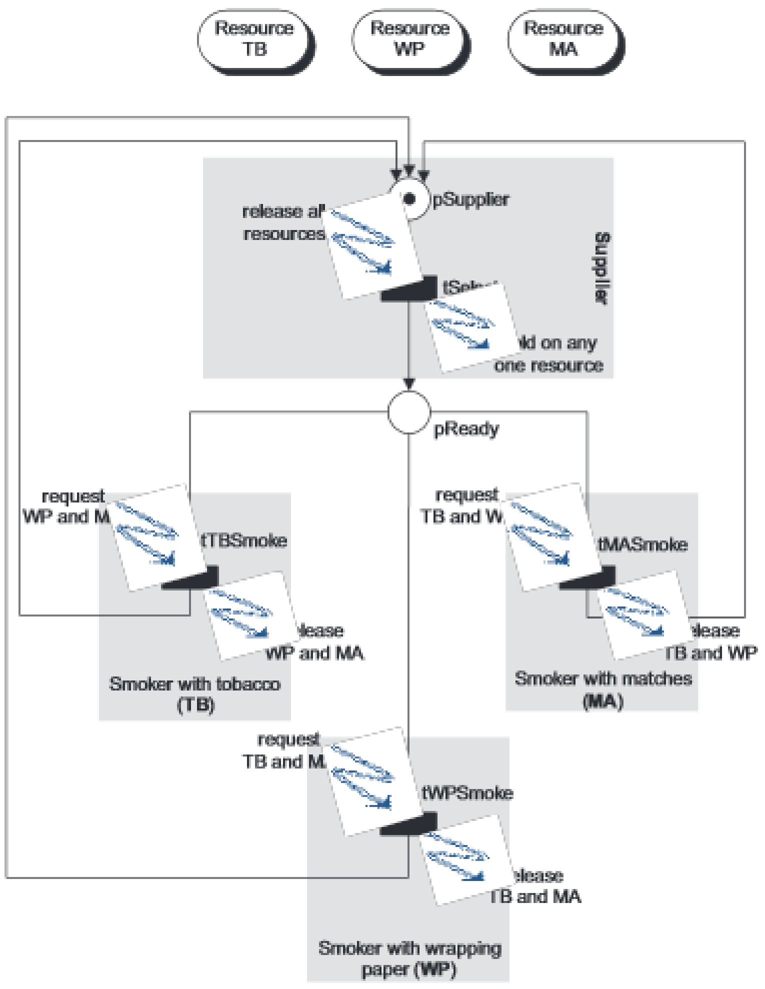


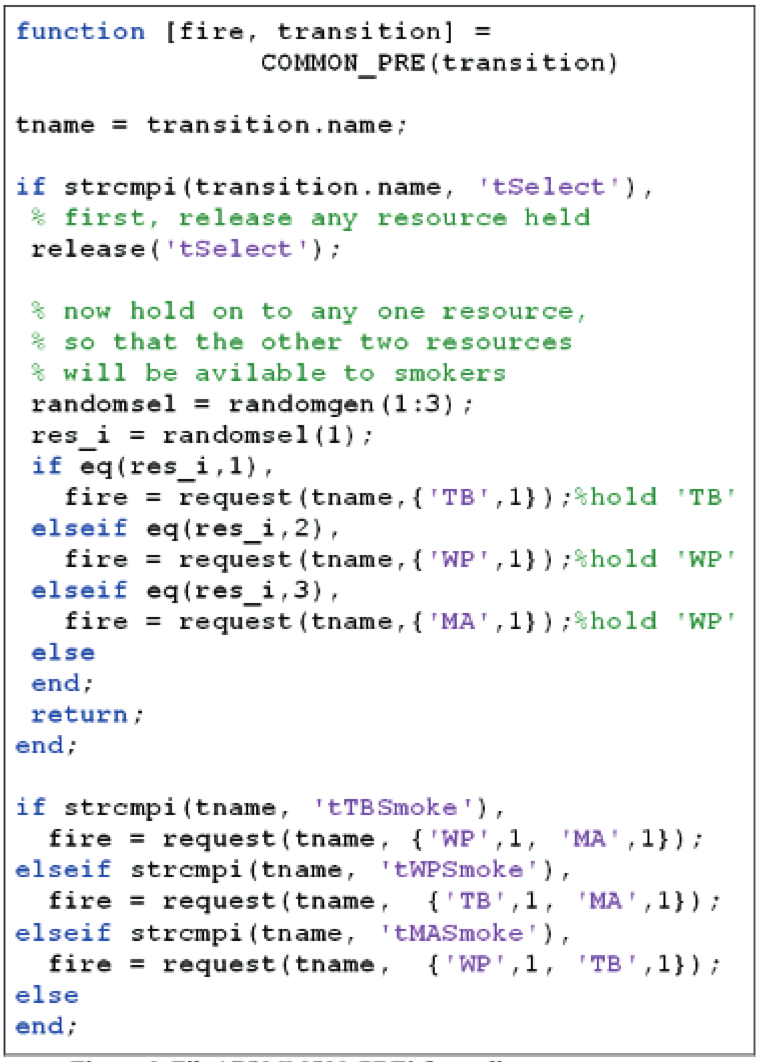
Figure 5: Petri Net model using Resources

Many studies report the huge size of Petri Net models of discrete event systems as one of the main problems associated with Petri Nets [10]; [11]. This problem is especially true when modeling resource scheduling. Even for a problem with few activities competing for a few resources, the resulting Petri Net model can be very large [12]; [13].

GPenSIM offers "*resources*" with which compact models of resource sharing and resource scheduling can be obtained. By using resources in GPenSIM, the size of Petri Net models are reduced as only the activities are shown in the Petri Net models and the resources are pushed into the background [13].

Figure-5 shows the Petri Net model that uses resources. In this model, the three ingredients ('TB', 'WP', and 'MA') are assumed to be resources. The transition 'tSelect' (representing the supplier) initially holds the three resources. When the firing completes, 'tSelect' releases only two of the three resources; 'tSelect' holds on to the third one, making that resource unavailable for any others.

Naturally, 'tTBSmoke' is the transition that represents the smoker with tobacco; if the transition 'tTBSmoke' is to fire, then it must request the resources 'WP' and 'MA'. Only if these resources are available ('tSelect' has already released these two resources), then 'tTBSmoke' can start firing. When 'tTBSmoke' completes firing, the used resources ('WP' and 'MA') are released. The other two transitions 'tWPSmoke' and 'tMASmoke' representing the other two smokers behave similarly.



```
function [fire, transition] =
               COMMON_PRE(transition)

tname = transition.name;

if strcmpi(transition.name, 'tSelect'),
 % first, release any resource held
 release('tSelect');

 % now hold on to any one resource,
 % so that the other two resources
 % will be avilable to smokers
 randomsel = randomgen(1:3);
 res_i = randomsel(1);
 if eq(res_i,1),
   fire = request(tname,{'TB',1});%hold 'TB'
 elseif eq(res_i,2),
   fire = request(tname,{'WP',1});%hold 'WP'
 elseif eq(res_i,3),
   fire = request(tname,{'MA',1});%hold 'WP'
 else
 end;
 return;
end;

if strcmpi(tname, 'tTBSmoke'),
  fire = request(tname, {'WP',1, 'MA',1});
elseif strcmpi(tname, 'tWPSmoke'),
  fire = request(tname,  {'TB',1, 'MA',1});
elseif strcmpi(tname, 'tMASmoke'),
  fire = request(tname,  {'WP',1, 'TB',1});
else
end;
```

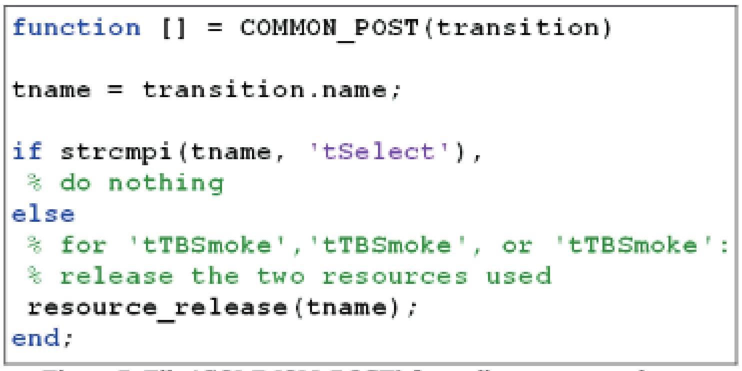
Figure 6: File 'COMMON_PRE' for coding resource usage

Transition 'tSelect' releasing two resources arbitrarily and holding on to the third one is coded in the COMMON_PRE file (figure-6). Also coded in the COMMON_PRE file is an enabled transition ('tTBSmoke', 'tWPSmoke', or 'tMASmoke') requesting specific resources before start of firing. Transitions releasing resources after firing, is coded in the COMMON_POST file. COMMON POST file is shown in figure-7.

The Petri Net that uses resources (figure-5) is intentionally compact and less detailed; the reachability tree for the Petri Net, shown in figure-8, proves the fact that the Petri Net model is trivial. Hence, it is obvious that decision power is taken away from the Petri Net model and moved to the underlying system (GPenSIM) that manages the resources.

The advantages of using resources in GPenSIM are two folded:

1. Petri Net models of manageable size can be obtained even for a system with many resources
2. GPenSIM provides a detailed report of resource usage. Figure-9 presents a report on resource usage when the Petri net model was executed.



```
function [] = COMMON_POST(transition)

tname = transition.name;

if strcmpi(tname, 'tSelect'),
 % do nothing
else
 % for 'tTBSmoke','tTBSmoke', or 'tTBSmoke':
 % release the two resources used
 resource_release(tname);
end;
```

Figure 7: File 'COMMON_POST' for coding resource releases

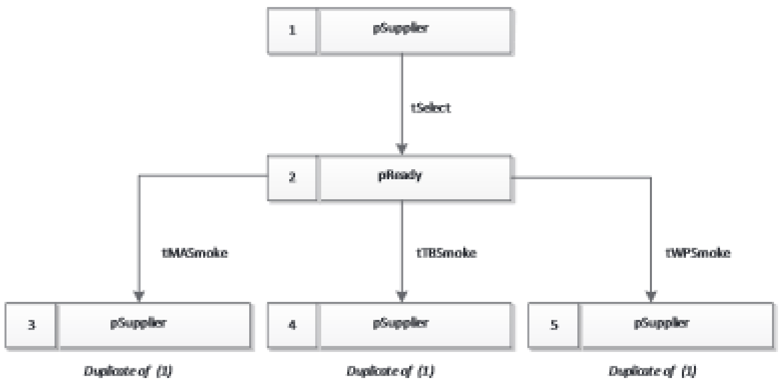


Figure 8: Reachability Tree of the Petr Net model using resources

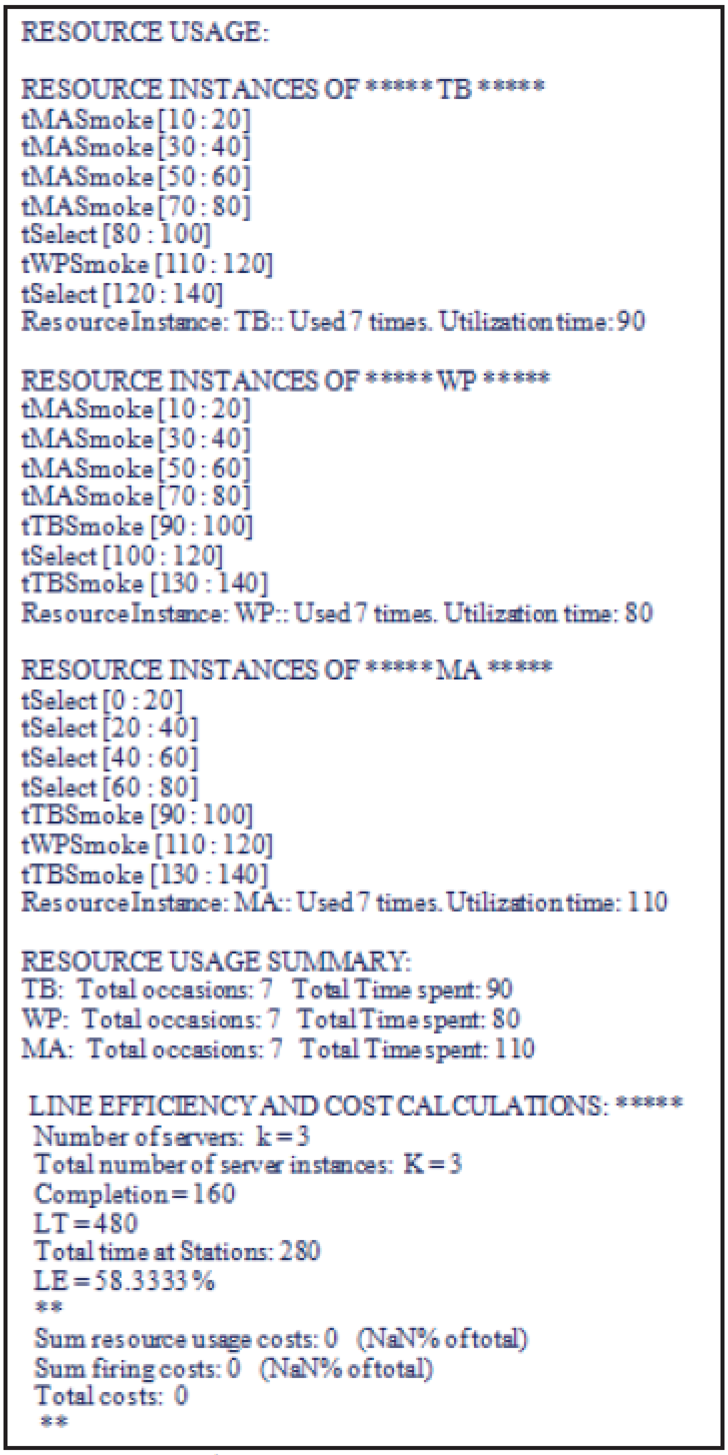
CONCLUSION

In this paper, we investigate a new Petri Net simulator called GPenSIM, and tested some of the extensions and facilities which have been implemented on this tool. The classical Cigarette Smokers' Problem (CSP) was taken as a case study to measure the ability of GPenSIM in using the different Petri Net extensions.

Petri Nets is a family of modeling formalisms (e.g. the ones discussed in this paper, P/T Petri Net, Colored Petri Net, Petri Net with priority, and Petri Net with inhibitor arcs) with different interpretations and abstraction levels.

Theoretically, the modeling capability of P/T Petri Net is exactly the same as of Colored Petri Net; Petri Net with inhibitor arcs should have the same modeling capability of Petri Net with priority, and these two formalisms should have much higher modeling power than P/T Petri Net as these two can model Turing machines, whereas P/T Petri Net cannot (this is not true for bounded systems); when modeling bounded systems, all the Petri Net formalisms should possess the same modeling power.

The modeling and simulations given in this paper show that the classical CSP can be modeled and analyzed by a variety of Petri Net extensions (such as inhibitor arc, color, and priority). GPenSIM's resources are a convenient way for minimizing the size of the Petri Net models, especially when there are many resources involved in the discrete event system. When using resources, GPenSIM provides a set of functions to make decisions from the compact Petri Net model.



```
RESOURCE USAGE:

RESOURCE INSTANCES OF ***** TB *****
tMASmoke [10 : 20]
tMASmoke [30 : 40]
tMASmoke [50 : 60]
tMASmoke [70 : 80]
tSelect [80 : 100]
tWPSmoke [110 : 120]
tSelect [120 : 140]
ResourceInstance: TB:: Used 7 times. Utilization time: 90

RESOURCE INSTANCES OF ***** WP *****
tMASmoke [10 : 20]
tMASmoke [30 : 40]
tMASmoke [50 : 60]
tMASmoke [70 : 80]
tTBSmoke [90 : 100]
tSelect [100 : 120]
tTBSmoke [130 : 140]
ResourceInstance: WP:: Used 7 times. Utilization time: 80

RESOURCE INSTANCES OF ***** MA *****
tSelect [0 : 20]
tSelect [20 : 40]
tSelect [40 : 60]
tSelect [60 : 80]
tTBSmoke [90 : 100]
tWPSmoke [110 : 120]
tTBSmoke [130 : 140]
ResourceInstance: MA:: Used 7 times. Utilization time: 110

RESOURCE USAGE SUMMARY:
TB:  Total occasions: 7   Total Time spent: 90
WP:  Total occasions: 7   Total Time spent: 80
MA:  Total occasions: 7   Total Time spent: 110

 LINE EFFICIENCY AND COST CALCULATIONS: *****
  Number of servers:  k = 3
  Total number of server instances:  K = 3
  Completion = 160
  LT = 480
  Total time at Stations: 280
  LE = 58.3333 %
  **
  Sum resource usage costs: 0   (NaN% of total)
  Sum firing costs: 0   (NaN% of total)
  Total costs:  0
   **
```

Figure 9: Report on resource usage

Acknowledgment

The author of this paper would like to thank the anonymous reviewer for carefully reading the manuscript and proposing major revisions to it. The anonymous reviewer's critical review has drastically changed this work to its current form.

References

W. Reisig, Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies, Springer, 2013

J. L. Peterson, Petri Net Theory and the Modeling of Systems. NJ, USA: Prentice-Hall, 1981

S. Patil, "Limitations and Capabilities of Dijkstra's Semaphore Primitives for Coordination Among Processes", Computation Structures Group Memo 57, Project MAC, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1971

R. Davidrajuh, "General Purpose Petri Net Simulator (GPenSIM) ". Available: http://www.davidrajuh.net/gpensim

D. Parnas, "On a Solution to the Cigarette Smokers' Problem (Without Conditional Statements)", Communications of the ACM, Volume 18, Number 3, (March 1975), pp. 181-183.

S. Kosaraju, "Limitations of Dijkstra's Semaphore Primitives and Petri Nets", Operating Systems Review, Volume 7, Number 4, (October 1973), pp. 122-126.

T. Agerwala, and M. Flynn, "Comments on Capabilities, Limitations and 'Correctness' of Petri Nets", Proceedings of the First Annual Symposium on Computer Architecture, New York: ACM, (1973), pp. 81-86.

R. Keller, "Vector Replacement Systems: A Formalism for Modeling Asynchronous Systems", Technical Report 117, Computer Science Laboratory, Princeton University, Princeton, New Jersey, (January 1974)

R. Davidrajuh, "Developing a Petri Nets based Real-Time Control Simulator", International Journal of Simulation, Systems, Science & Technology (IJSSST), vol. 12, issue. 3, pp. 28-36, 2012

H. Goto, Y. Hasegawa, and M. Tanaka, "Efficient Scheduling Focusing on the Duality of MPL Representatives", Proc. IEEE Symp. Computational Intelligence in Scheduling (SCIS 07), IEEE Press, Dec. 2007, pp. 57-64, doi:10.1109/SCIS.2007.357670. (Pubitemid 47431182)

S. Bardin, and L. Petrucci. From PNML to counter systems for accelerating Petri nets with FAST. In Proc. of the Workshop on Interchange Formats for Petri Nets (at ICATPN'2004), pages 26-40, Bologna, Italy, June 2004

N. Wu and M. Zhou, System modeling and control with resourceoriented Petri Nets. Boca Raton, FL: CRC Press, Taylor Francis Group, 2010

R. Davidrajuh, "Activity-Oriented Petri Net for Scheduling of Resources", Proc. IEEE International Conference on Systems, Man, and Cybernetics (IEEE SMC 2012), October 14-17, 2012, Seoul, Korea