RESTART Simulation of Colored Stochastic Petri Nets

Armin Zimmermann
Technische Universität Ilmenau, Germany
Faculty of Computer Science and Automation
System and Software Engineering Group
armin.zimmermann@tu-ilmenau.de

Key words

RESTART/Splitting, colored stochastic Petri nets, software tool

Abstract

In this paper the combination of stochastic colored Petri nets and the RESTART method is proposed to evaluate performance measures of complex systems that depend on rare events. A prototype implementation is described, which is applied to a logistics example.

1 Introduction

Model-based evaluation of safety-critical embedded systems is an important aid in their design. This is especially true for non-functional properties; examples are real-time capabilities, fault-tolerance, and performance while taking into consideration failures as well as heavy load situations.

Their computer-based evaluation requires a model with performance measures and an evaluation technique implemented in some software tool. Simulation is the only tractable method to evaluate such models if they are subject to multiple non-Markovian activity delays (compare [7] for the Petri net context). If a performance result depends significantly on the probability of being in (or hitting a) certain rare state (set), the computational effort of a standard simulation becomes unacceptable, and rare-event simulation techniques must be applied [1].

However, despite their obvious advantages in areas that are vital for industrial applications, rare-event simulation techniques have not yet gained sufficient interest outside academia. The author believes that among the technical reasons for this are

- the amount of background knowledge necessary to apply rare-event simulation techniques. An example is the definition of importance function and

levels in a splitting setting. Automatic derivation of a good importance function for a given model is still an open research issue [6].

- limitations in the types of performance measures that can be estimated by rare-event simulation
- restrictions in the types of models for which rare-event simulation algorithms have been developed or adapted
- lack of software tools that are easy to use.

We consider the RESTART method [15, 16] in the following because of its robustness and wide range of applicability. Moreover, as it does not require as much information about the evaluated model as, e.g., importance sampling, we believe it will be easier to develop algorithms and tools that do not require a rare-event simulation expert to set up a meaningful simulation study.

The issue of allowing more general performance measures, namely by using freely definable rate and impulse rewards in the stationary evaluation of stochastic Petri nets, has been covered earlier [19]. It is based on associating weights with split paths as described in [12]. Reward variables are functions that return some value of interest from the stochastic process of a stochastic discrete-event model. Following the characterization given in [11], both *rate rewards* and *impulse rewards* (which are associated to *states* and *events* of the process, respectively) are considered. The probability of a rare event or set of rare states is a special case.

The goal of this paper is a contribution to the final two issues. It reports initial results of rare-event simulation for stochastic colored Petri nets (SCPN in the following). The results are being included in the software tool TimeNET [18, 20] which, until recently, included a restricted RESTART implementation for simple Petri nets only [9].

Complex stochastic discrete-event systems in areas such as manufacturing, telecommunication, or embedded safety-critical systems can often be modeled adequately with a colored Petri net [8]. Informally, its main difference to simple Petri nets is that tokens carry information, which influences the semantics of transition enabling and firing. A stochastic timed extension of colored Petri nets is, e.g., given in [20]. While the use of colored Petri nets usually results in more compact models for complex applications, it obviously does not simplify their performance evaluation[1]. Parallel simulation of SCPN is a technical approach to a simulation speedup, but it requires significant computing resources and adds to the complexity of the simulation algorithm [3]. There are a few modeling and simulation tools implementing performance evaluation algorithms for SCPN, but none of them supports (to the best of the authors' knowledge) their rare-event simulation.

[1] Except for the case when the model is inherently symmetrical, which can be described with a well-formed net [2] and exploited during simulation [4]

Section 2 briefly describes the SCPN model class using an example. Background and implementation of the RESTART algorithm in TimeNET for SCPN models are covered in Sections 3 and 4, respectively.

2 Colored Stochastic Petri Nets

This section informally describes stochastic colored Petri nets along an example that will later be evaluated with the algorithm and tool presented in this paper. A full description of the model class SCPN is outside the scope of this paper, but covered in [20].

Stochastic colored Petri nets are especially useful to describe complex stochastic discrete event systems. Objects which are created, changed and moved through a system are usually described by tokens in places. The application of classic Petri nets to examples in which these objects carry significant attributes leads to cluttered models in which places and transitions need to be unfolded to keep track of individual attributes.

Colored Petri nets [8] are one variant of Petri nets with individual tokens. Extensions of colored Petri nets by stochastic firing times have been introduced in [10, 17]. The kind of stochastic colored Petri nets used here are based on [8], but allow an easier specification of arc inscriptions. This makes it possible to generate efficient simulation algorithm code automatically. Moreover, a true stochastic timing semantics in accordance to the usual understanding of timed Petri nets is adopted.

The introduction of individual tokens requires an adapted syntax and semantics with respect to classic Petri nets. Attributes of tokens need to be structured and specified, resulting in *colors* (or *types*).

Figure 1 shows a colored Petri net model of a fictitious supply chain. There

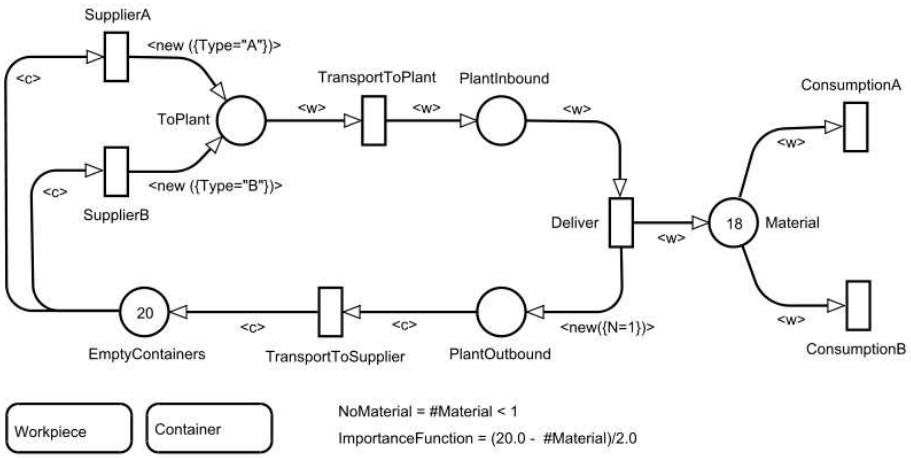

Figure 1: Stochastic colored Petri net model of a supply chain

are two types of tokens defined in the model, namely `Workpiece` for parts to be transported in a container, and `Container` for empty containers. Tokens of type `Container` have a number attribute `N`, while `Workpiece` tokens have a `Type` attribute which is a string.

Only tokens of one type are allowed in each place, thus a place has a corresponding type. In our model, places `EmptyContainers` and `PlantOutbound` may contain empty containers, and their corresponding type is therefore `Container`. All remaining places are of type `Workpiece`. Places may also have a capacity (maximum number of tokens allowed in them).

Numbers as arc information are no longer sufficient as in simple Petri nets. Transitions model activities, and their firings may depend on token attribute values and change them at firing time. A transition might have different modes of enabling and firing depending on its input tokens. Arc variables are used to describe these alternatives.

Transition `TransportToPlant` will be enabled for any kind of workpiece in place `ToPlant`, it just models the transport operation. There is one arc variable `w` on its input arc, which can be bound to a token in place `ToPlant`. Each possible association of tokens to (input) arc variables of a transition is called a *binding* [8].

Upon firing, output arc variables are evaluated, which in this case means that an identical token is put into place `PlantInbound`. Firing transition `SupplierA` models packing and sending a part of type `A` to the plant, and thus a container token is consumed and a workpiece token created with attribute `Type` set to `A`. This is done with the C++-like syntax shown. Tokens arriving in place `Material` model available parts in the plant, and are used up by some production activity modeled by transitions `ConsumptionA` and `ConsumptionB`. Each one of these transitions can only consume tokens of the correct type, because they have a guard function `w.Type == ''A''` (or `B`, respectively). This enables them only for arc variable settings with this property.

Just like in a stochastic Petri net, transitions have firing delays defined by probability distribution functions. In the example shown, all transitions have exponentially distributed firing delays. Transitions `TransportToPlant` and `TransportToSupplier` follow *inifinte server* semantics, i.e., they are concurrently enabled for each of their binding in every marking.

Performance measures have to be defined for a meaningful simulation. They may also depend on token attributes in a SCPN, but this is not necessary in our example model. We assume in the following that uninterrupted production is very important and that material available at the plant should thus never run out of stock. A corresponding time-averaged performance measure `NoMaterial` is thus defined to measure the probability of no tokens in place `Material`. The #-signs refers to the number of tokens in a place. If the system is properly set up, it should only rarely be in a state where `#Material` $<$ `1`.

3 RESTART Simulation of SCPNs

This section briefly recalls the approach presented in [19,20] for the computation of extended reward measures combined with RESTART. Descriptions of the RESTART technique in general can, e.g., be found in [13, 14].

Assume that the goal of a simulation is to estimate the probability $\mathrm{P}\{A\}$ of being in a set of state A in steady state, and that significant samples are generated only rarely because of the model structure. Let the set of all reachable states of a model be denoted by B_0, and the initial state of the system by σ_0. A is visited more frequently by concentrating on promising paths in the state set. If we can find a measure of "how far away" from A a state is, it becomes possible to decide which paths are more likely to succeed and should be followed more frequently.

Formally, define M subsets $B_1 \dots B_M$ of the overall state space B_0 such that $A = B_M$ and $B_M \subset B_{M-1} \subset \dots \subset B_1 \subset B_0$. The conditional probabilities $\mathrm{P}\{B_{i+1} \mid B_i\}$ of being in an enclosed set B_{i+1} under the precondition of being in B_i are much easier to estimate than $\mathrm{P}\{A\}$. The measure of interest can be obtained from the product of the conditionals.

Membership of a state visited during a simulation to one of the state sets B_i is defined by an *importance function* $f_I : B_0 \to \mathbb{R}$ together with a set of *thresholds* (denoted by $Thr_i \in \mathbb{R}, i = 1 \dots M$). A simulation is said to be in a *level* i if the current state σ belongs to $B_i \setminus B_{i+1}$.

RESTART measures the conditional probability of reaching a state out of set B_{i+1} after starting in B_i by a Bernoulli trial. If B_{i+1} is hit, the entering state is stored and the simulation trial is split into R_{i+1} trials. The simulation follows each of the trials to see whether B_{i+2} is hit and so on. A trial starting at B_i is canceled after leaving B_i if it did not hit B_{i+1}. Simulation of paths inside B_0 and $B_M = A$ is not changed.

Such an estimator for steady-state simulation implementations needs correction factors that take into account the RESTART splitting. We adopt the method of [12], where *weights* ω are maintained during the simulation run, which capture the relative importance of the current path elegantly.

The weights are computed as follows: A simulation run starting from the initial state $\sigma_0 \in (B_0 \setminus B_1)$ has an initial weight of 1, because it is similar to a "normal" simulation run without splitting. Whenever the simulation path currently in level i crosses the border to an upper level u, the path is split into R_u paths, which are simulated subsequently. The weight is divided by R_u upon splitting. Paths leading to a level $< i$ are discarded except for the last one of a split, which is followed further using the stated rules. The weight of the last path is multiplied by R_i when it leaves level i downwards. This procedure is repeated until the required result quality is achieved. This technique has the additional advantage of allowing "jumps of levels" over more than one threshold compared to the original method.

Reward variables for SCPN models are functions that return some value of interest from the stochastic process *Proc* of a SDES model. This process describes the state σ and possibly happening events *SE* at time t, *Proc* =

$\{(\sigma(t), SE(t)), t \in \mathbb{R}^{0+}\}$.

Reward variables describe combinations of a positive bonus or negative penalty associated to elements of the stochastic process. Two types of elements of such a reward variable have been identified in the literature [11]: *rate rewards* which are accumulated over time in a state, and *impulse rewards* which are gained instantaneously at the moment of an event. An instantaneous reward $R_{inst}(t)$ gained at a point in time can thus be defined.

$$R_{inst}(t) = \underbrace{rrate\big(\sigma(t)\big)}_{\text{rate rewards}} + \underbrace{\Delta \cdot \sum_{se \in SE(t)} rimp(se)}_{\text{impulse rewards}} \tag{1}$$

The value of the reward variable in steady-state can then be expressed as

$$rvar(Proc) = \lim_{x \to \infty} \frac{1}{x} \int_0^x R_{inst}(t)\, \mathrm{d}t, \tag{2}$$

which leads to a simulation estimator $\widehat{rvar}$

$$\widehat{rvar} = \frac{1}{T} \int_0^T \omega(t)\, R_{inst}(t)\ \mathrm{d}t \tag{3}$$

where T is the (sufficiently large) maximum simulation time spent in final paths of the RESTART algorithm, and $\omega(t)$ is the weight as managed by the algorithm.

Algorithm 1 implements a RESTART simulation of one path for a SCPN model including a reward variable *rvar*. Its parameters include RESTART-specific level and weight as well as the complete state of the simulation itself, namely state, event list, and simulation time. This is necessary to start simulation paths at splitting points. σ and EventList form together the complete state of the underlying generalized semi-Markov process (GSMP). RESTART-PATH is called initially with level zero[2], weight one, initial state, and time zero. Reward that is accumulated in a state and upon state change, and multiplied by the weight factor ω to compensate for the splitting. $Level(\sigma)$ returns the current level for a state, depending on importance function and thresholds as described before.

The function returns when the stop conditions are reached; an estimation of the performance measure is finally computed as $\frac{\text{Reward}_{rvar}}{\text{SimTime}}$. It should be noted that SimTime is not the sum of all simulated time spans as in a standard simulation, but the simulation time spent in all final paths.

Several variants of RESTART have been considered in the literature [5]. We follow the approach taken in [9,12], which can be characterized as *fixed splitting* and *global step* according to [5].

[2]We assume that $Level(\sigma_0) = 0$ for simplicity.

RESTARTPATH $(lvl, \omega, \sigma, \text{EventList}, t)$

Input: Level lvl, Weight ω, state σ, event list, time t
Output: Final state of the simulation: new $(\sigma, \text{EventList}, t)$

$(*$ main simulation loop $*)$
repeat
  $(*$ get new events $*)$
  UPDATEEVENTLIST$(\sigma, \text{EventList}, t)$
  $(*$ select executed event $*)$
  $(a, binding, t') :=$ SELECTEVENT(EventList)
  Event $:= (a, binding)$
  $(*$ update performance measure $rvar = (rrate, rimp, \cdot, \cdot)$ $*)$
  $\text{Reward}_{rvar} \mathrel{+}= \omega * \big((t' - t) * rrate(\sigma) + rimp(\text{Event})\big)$
  $(*$ execute state change $*)$
  $t := t'$; $\sigma := Exec(\text{Event}, \sigma)$
  $(*$ RESTART level control $*)$
  $lvl' := Level(\sigma)$
  if $lvl' > lvl$ **then** $(*$ split $*)$
    for $i = 1 \ldots R_{lvl'}$ **do** $(\sigma', \text{EventList}', t') :=$
      RESTARTPATH$(lvl', \frac{\omega}{R_{lvl'}}, \sigma, \text{EventList}, t)$
    $(*$ Continue the final path $*)$
    $\sigma := \sigma'$; EventList $:=$ EventList$'$; $t := t'$
until $(lvl' < lvl)$ **or** (stop condition reached, e.g. $t \geq$ MaxSimTime)
return $(\sigma, \text{EventList}, t)$

Algorithm 2: Main RESTART algorithm for steady-state simulation

4 Software Tool Implementation in TimeNET

Modeling and evaluation of complex discrete-event systems is only possibly in practice with the support of appropriate tools. TimeNET [18, 20] is a software tool for the modeling and performance evaluation of stochastic discrete-event models, mainly concentrating on Petri net variants. The tool is available free of charge for non-commercial use from `http://www.tu-ilmenau.de/sse/`.

Figure 2 shows a screen shot of an editing session using stochastic colored Petri nets. The tool contains a RESTART implementation for simple stochastic Petri nets [9], and is currently extended by a module implementing the algorithm described in this paper for SCPN models.

The RESTART algorithm described before is implemented as a variant of the standard SCPN simulation module. It reuses code that implements the semantics of SCPN models, which is possible because the RESTART algorithm is not model-type specific, and may thus be combined with any stochastic discrete-event formalism with an underlying GSMP. SCPN models are, in fact, translated into C++ code, which is then compiled and linked to a simulation kernel module.

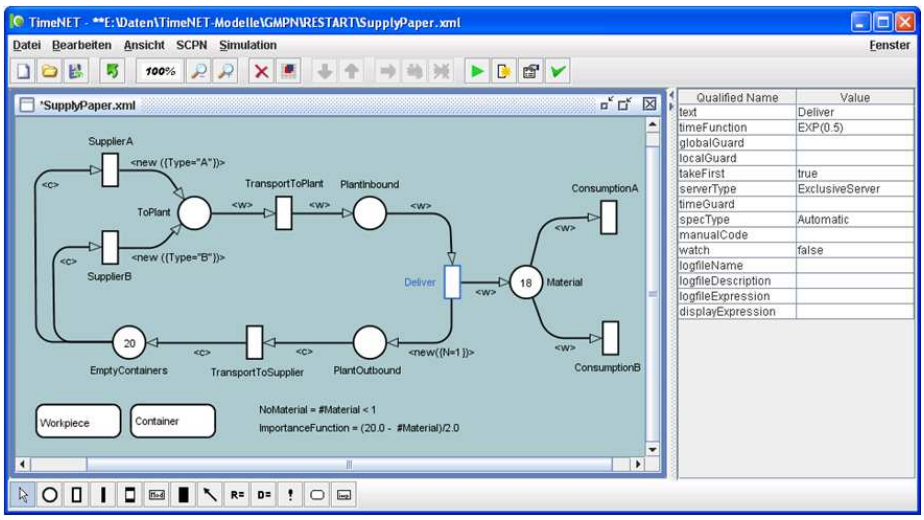

Figure 2: Sample screen shot of TimeNET

In difference to the standard simulation, intermediate states of the SCPN model have to be stored and reloaded at splitting points, which is a non-trivial task because of the inherently complex internal data representation. One reason for this is that types of places and tokens may be quite freely defined, and the actual code that manages user-defined types is generated and compiled during run-time in the initialization phase of a simulation.

A connection between SCPN model and RESTART algorithm is the importance function. $Level(\sigma)$ has to return an integer for any state in the main algorithm. In our first prototype implementation, a performance measure with a predefined name (`ImportanceFunction`) is used to specify the importance function for the tool. Thresholds (importance function values defining the levels) are assumed to be all natural numbers. In the model shown in Figure 1, the current RESTART level of a state is given by `(20-#Material)/2`. Place `Material` has a capacity of 20. The function thus returns values in the range 0..10, with higher numbers meaning a higher level and a state "closer" to one in which there is no material available. The splitting factor has to be set manually.

The prototype implementation has been successfully tested with a simple SCPN model of a M/M/1 queue to validate the results with analytical values. For example, the probability $9.536 * 10^{-7}$ of more than 20 customers in a queue with $\frac{\lambda}{\mu} = \frac{1}{2}$ has been evaluated both with standard simulation and the RESTART prototype. Within 50,000 samples the standard simulation did not hit the rare event, while the RESTART algorithm (with levels $2, 4, \ldots$ customers and splitting factor 4) already arrives at at remaining relative error of 6% within 10,000 generated samples.

The supply chain model shown in Figure 1 was evaluated as well. Performance measure `NoMaterial` converges with the RESTART algorithm within

50,000 samples to $2.6659 * 10^{-6}$, while the standard simulation is still far away. This RESTART run took 19 seconds on a 1.86 GHz Pentium laptop computer.

5 Conclusion

This paper introduced RESTART simulation for stochastic colored Petri nets. This combination allows to evaluate models of complex systems, in which rare events have a significant impact on performance measures. A prototype tool is presented, which is applied to a supply chain example. The proposed technique is, however, still far from being sufficient. In the future we plan to adapt and implement rare-event simulation techniques to models of complex technical systems, to make them accessible for non-expert users.

References

[1] J. Bucklew, *Introduction to Rare Event Simulation*, Springer Series in Statistics. Springer Verlag, 2004.

[2] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad, "Stochastic well-formed colored nets and symmetric modeling applications," *IEEE Transactions on Computers*, vol. 42, no. 11, pp. 1343–1360, 1993.

[3] A. Furfaro, L. Nigro, and F. Pupo, "Distributed simulation of timed coloured Petri nets," in *Proceedings. Sixth IEEE Int. Workshop on Distributed Simulation and Real-Time Applications (DS-RT'02)*. IEEE Computer Society, Oct. 2002, pp. 159–166.

[4] R. Gaeta, "Efficient discrete-event simulation of colored Petri nets," *IEEE Transactions on Software Engineering*, vol. 22, no. 9, pp. 629–639, 1996.

[5] M. J. Garvels and D. P. Kroese, "A comparison of RESTART implementations," in *Proc. 1998 Winter Simulation Conference*, 1998.

[6] M. J. Garvels, J.-K. C. Van Ommeren, and D. P. Kroese, "On the importance function in splitting simulation," *European Transactions on Telecommunications*, vol. 13, no. 4, pp. 363–371, 2002.

[7] R. German, *Performance Analysis of Communication Systems, Modeling with Non-Markovian Stochastic Petri Nets*. John Wiley and Sons, 2000.

[8] K. Jensen, *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*, EATCS Monographs on Theoretical Computer Science. Springer Verlag, 1992.

[9] C. Kelling, "A framework for rare event simulation of stochastic Petri nets using RESTART," in *Proc. of the Winter Simulation Conference*, 1996, pp. 317–324.

[10] C. Lin and D. C. Marinescu, "On stochastic high-level Petri nets," in *Proc. 2nd Int. Workshop on Petri Nets and Performance Models*, Madison, Wisconsin, 1987, pp. 34–43.

[11] W. H. Sanders and J. F. Meyer, "A unified approach for specifying measures of performance, dependability, and performability," in *Dependable Computing for Critical Applications*, Dependable Computing and Fault-Tolerant Systems, A. Avizienis and J. Laprie, Eds. Springer Verlag, 1991, vol. 4, pp. 215–237.

[12] B. Tuffin and K. S. Trivedi, "Implementation of importance splitting techniques in stochastic Petri net package," in *Computer Performance Evaluation, Modelling Techniques and Tools — 11th Int. Conf., TOOLS 2000*, Lecture Notes in Computer Science, vol. 1786. Schaumburg, IL, USA: Springer Verlag, 2000, pp. 216–pp.

[13] M. Villén-Altamirano and J. Villén-Altamirano, "Analysis of RESTART simulation: Theoretical basis and sensitivity study," *European Transactions on Telecommunications*, vol. 13, no. 4, pp. 373–385, 2002.

[14] ——, "Optimality and robustness of RESTART simulation," in *Proc. 4th Workshop on Rare Event Simulation and Related Combinatorial Optimisation Problems*, Madrid, Spain, Apr. 2002.

[15] ——, "On the efficiency of RESTART for multidimensional state systems," *ACM Transactions on Modeling and Computer Simulation*, vol. 16, no. 3, pp. 251–279, July 2006.

[16] M. Villén-Altamirano, J. Villén-Altamirano, J. Gamo, and F. Fernández-Cuesta, "Enhancement of the accelerated simulation method RESTART by considering multiple thresholds." in *Proc. 14th Int. Teletraffic Congress*. Elsevier Science Publishers B. V., 1994, pp. 797–810.

[17] A. Zenie, "Colored stochastic Petri nets," in *Proc. 1st Int. Workshop on Petri Nets and Performance Models*, 1985, pp. 262–271.

[18] A. Zimmermann, M. Knoke, A. Huck, and G. Hommel, "Towards version 4.0 of TimeNET," in *13th GI/ITG Conference on Measurement, Modeling, and Evaluation of Computer and Communication Systems (MMB 2006)*, March 2006, pp. 477–480.

[19] A. Zimmermann, "Applied restart estimation of general reward measures," in *Proc. 6th Int. Workshop on Rare Event Simulation (RESIM 2006)*, Bamberg, Germany, Oct. 2006, pp. 196–204.

[20] ——, *Stochastic Discrete Event Systems*. Springer, Berlin Heidelberg New York, 2007.