Budapest University of Technology and Economics
Faculty of Electrical Engineering and Informatics
Department of Measurement and Information Systems

Configurable Stochastic Analysis Framework for Asynchronous Systems

Scientific Students' Associations Report

Authors:

Attila Klenik
Kristóf Marussy

Supervisors:

dr. Miklós Telek
Vince Molnár
András Vörös

2015.

Contents

Összefoglaló A kritikus rendszerek – biztonságkritikus, elosztott és felhőalkalmazások – helyességének biztosításához szükséges a funkcionális és nemfunkcionális követelmények matematikai igényességű ellenőrzése. Számos, szolgáltatásbiztonsággal és teljesítményvizsgálattal kapcsolatos tipikus kérdés általában sztochasztikus analízis segítségével válaszolható meg.

A kritikus rendszerek elosztott és aszinkron tulajdonságai az *állapottér robbanás* jelenségéhez vezetnek. Emiatt méretük és komplexitásuk gyakran megakadályozza a sikeres sztochasztikus analízist, melynek számításigénye nagyban függ a lehetséges viselkedések számától. A modellek komponenseinek jellegzetes időbeli viselkedése a számításigény további jelentős növekedését okozhatja.

A szolgáltatásbiztonsági és teljesítményjellemzők kiszámítása markovi modellek állandósult állapotbeli és tranziens megoldását igényli. Számos eljárás ismert ezen problémák kezelésére, melyek eltérő reprezentációkat és numerikus algoritmusokat alkalmaznak; ám a modellek változatos tulajdonságai miatt nem választható ki olyan eljárás, mely minden esetben hatékony lenne.

A markovi analízishez szükséges a modell lehetséges viselkedéseinek, azaz állapotterének felderítése, illetve tárolása, mely szimbolikus módszerekkel hatékonyan végezhető el. Ezzen szemben a sztochasztikus algoritmusokban használt vektor- és indexműveletek szimbolikus megvalósítása nehézkes. Munkánk célja egy olyan, integrált keretrendszer fejlesztése, mely lehetővé teszi a komplex sztochasztikus rendszerek kezelését a szimbolikus módszerek, hatékony mátrix reprezentációk és numerikus algoritmusok előnyeinek ötvözésével.

Egy teljesen szimbolikus algoritmust javasolunk a sztochasztikus viselkedéseket leíró mátrix-dekompozíciók előállítására a szimbolikus formában adott állapottérből kiindulva. Ez az eljárás lehetővé teszi a temporális logikai kifejezéseken alapuló szimbolikus technikák használatát.

A keretrendszerben megvalósítottuk a konfigurálható sztochasztikus analízist: megközelítésünk lehetővé teszi a különböző mátrix reprezentációk és numerikus algoritmusok kombinált használatát. Az implementált algoritmusokkal állandósult állapotbeli költség- és érzékenység analízis, tranziens költséganalízis és első hiba várható bekövetkezési idő analízis végezhető el sztochasztikus Petri-háló (SPN) markovi költségmodelleken. Az elkészített eszközt integráltuk a PetriDotNet modellező szoftverrel. Módszerünk gyakorlati alkalmazhatóságát szintetikus és ipari modelleken végzett mérésekkel igazoljuk.

Abstract Ensuring the correctness of critical systems – such as safety-critical, distributed and cloud applications – requires the rigorous analysis of the functional and extra-functional properties of the system. A large class of typical quantitative questions regarding dependability and performability are usually addressed by stochastic analysis.

Recent critical systems are often distributed/asynchronous, leading to the well-known phenomenon of *state space explosion*. The size and complexity of such systems often prevents the success of the analysis due to the high sensitivity to the number of possible behaviors. In addition, temporal characteristics of the components can easily lead to huge computational overhead.

Calculation of dependability and performability measures can be reduced to steady-state and transient solutions of Markovian models. Various approaches are known in the literature for these problems differing in the representation of the stochastic behavior of the models or in the applied numerical algorithms. The efficiency of these approaches are influenced by various characteristics of the models, therefore no single best approach is known.

The prerequisite of Markovian analysis is the exploration of the state space, i.e. the possible behaviors of the system. Symbolic approaches provide an efficient state space exploration and storage technique, however their application to support the vector operations and index manipulations extensively used by stochastic algorithms is cumbersome. The goal of our work is to introduce a framework that facilitates the analysis of complex, stochastic systems by combining together the advantages of symbolic algorithms, compact matrix representations and various numerical algorithms.

We propose a fully symbolic method to explore and describe the stochastic behaviors. A new algorithm is introduced to transform the symbolic state space representation into a decomposed linear algebraic representation. This approach allows leveraging existing symbolic techniques, such as the specification of properties with *Computational Tree Logic* (CTL) expressions.

The framework provides configurable stochastic analysis: an approach is introduced to combine the different matrix representations with numerical solution algorithms. Various algorithms are implemented for steady-state reward and sensitivity analysis, transient reward analysis and mean-time-to-first-failure analysis of stochastic models in the *Stochastic Petri Net* (SPN) Markov reward model formalism. The analysis tool is integrated into the PetriDotNet modeling application. Benchmarks and industrial case studies are used to evaluate the applicability of our approach.

Chapter 1

Introduction

Árvíztűrő tükörfúrógép

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language. Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like “Huardest gefburn”? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Chapter 2

Background

2.1 Petri nets

Petri nets are a widely used graphical and mathematical modeling tool for systems which are concurrent, asynchronous, distributed, parallel or nondeterministic.

Definition 2.1 A *Petri net* is a 5-tuple $PN = (P, T, F, W, M_0)$, where

- $P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of places;
- $T = \{t_1, t_2, \ldots, t_m\}$ is a finite set of transitions;
- $F \subseteq (P \times T) \cup (P \times T)$ is a set of arcs, also called the flow relation;
- $W : F \to \mathbb{N}^+$ is an arc weight function;
- $M_0 : P \to \mathbb{N}$ is the initial marking;
- $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.

A *Petri net structure* without an initial marking is a 4-tuple $N = (P, T, F, W)$, while a Petri net with a given initial marking is denoted by (N, M_0) [2].

Arcs from P to T are called *input arcs*. The input places of a transition t are denoted by ${}^\bullet t = \{p : (p, t) \in F\}$. In contrast, arcs of the form (t, p) are called *output arcs* and the output places of t are denoted by $t^\bullet = \{p : (t, p) \in F\}$.

A *marking* $M : P \to \mathbb{N}$ assigns a number of *tokens* to each place. The transition t is *enabled* in the marking M_1 (written as $M_1[t\rangle$) when $M(p) \geq W(p, t)$ for all $p \in {}^\bullet t$.

Petri nets are graphically represented as edge weighted directed bipartite graphs. Places are drawn as circles, while transitions are drawn as rules or rectangles. Edge weights of 1 are ususally omitted from presentation. Dots on places correspond to tokens in the current marking.

If $M_1[t\rangle$ the transition t can be *fired* to get a new marking M_2 (written as $M_1[t\rangle M_2$) by decreasing the token counts for each place $p \in {}^\bullet t$ by $W(p, t)$ and increasing the token counts for each place $p \in t^\bullet$ by $W(t, p)$. Note that in general, ${}^\bullet t$ and $t^\bullet$ need not

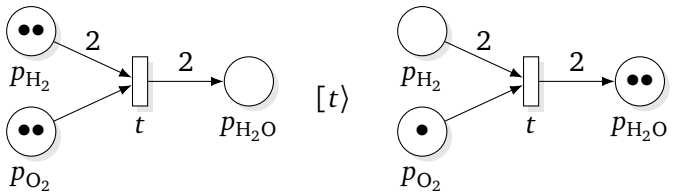

Figure 2.1: A Petri net model of the reaction of hydrogen and oxygen.

be disjoint. Thus, the firing rule can be written as

$$M_2(p) = M_1(p) - W(p,t) + W(t,p), \tag{2.1}$$

where we take $W(x,y) = 0$ if $(x,y) \notin F$ for brevity.

A marking M' is *reachable* from the marking M (written as $M \rightsquigarrow M'$) if there exists a sequence of markings and transitions for some finite k such that

$$M_1 [t_{i_1}\rangle M_2 [t_{i_2}\rangle M_3 [t_{i_3}\rangle \cdots [t_{i_{k-1}}\rangle M_{k-1} [t_{i_k}\rangle M_k,$$

where $M_1 = M$ and $M_k = M'$. A marking M is in the reachable *state space* of the net if $M_0 \rightsquigarrow M$.

Example 2.1 The Petri net in Figure 2.1 models the chemical reaction

$$2\,H_2 + O_2 \rightarrow 2\,H_2O.$$

In the initial marking (left) there are two hydrogen and two oxygen molecules, represented by token on the places p_{H_2} and p_{O_2}, therefore the transition t is enabled. Firing t yields the marking on the right where the two tokens on p_{H_2O} are the reaction products. Now t is no longer enabled.

Example 2.2 In Figure 2.2 we introduce the *SharedResource* model which will serve as a running example throughout this report.

The model consists of a single shared resource S and two consumers. Each consumer can be in one of the C_i (calculating locally), W_i (waiting for resource) and S_i (shared working) states. The transitions r_i (request resource), a_i (acquire resource) and d_i (done) correspond to behaviours of the consumers.

The Petri net model allows the verification of safety properties, e.g. we can show that there is mutual exclusion – $M(S_1) + M(S_2) \leq 1$ for all reachable markings – or that deadlocks cannot occur. In contrast, we cannot compute dependability or performability measures (e.g. the utilization of the shared resource or number of calculations completed per unit time) because the model does not describe the temporal behaviour of the system.

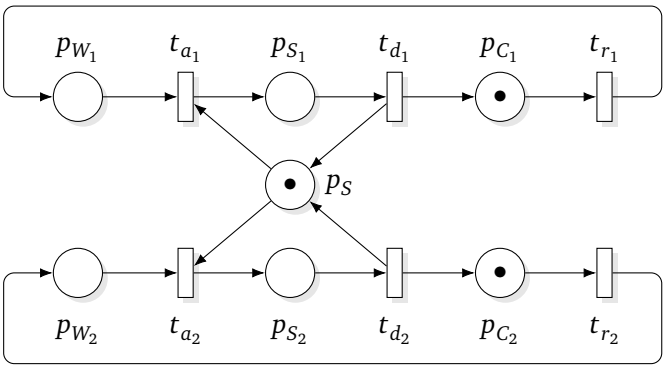

Figure 2.2: The *SharedResource* Petri net model.

2.1.1 Petri nets extended with inhibitor arcs

One of the most frequently used extensions of Petri nets is the addition of inhibitor arcs, which modifies the rule for transition enablement. This modification gives Petri nets expressive power equivalent to Turing machines [1].

Definition 2.2 A *Petri net with inhibitor arcs* is a 3-tuple $PN_I = (PN, I, W_I)$, where
- $PN = (P, T, F, W, M_0)$ is a Petri net;
- $I \subseteq P \times T$ is the set of inhibitor arcs;
- $W_I : I \to \mathbb{N}^+$ is the inhibitor arc weight function.

Let ${}^\circ t = \{p : (p,t) \in I\}$ denote the set of inhibitor places of the transition t. The enablement rule for Petri nets with inhibitor arcs can be formalized as

$$M[t\rangle \iff M(p) \geq W(p,t) \text{ for all } p \in {}^\bullet t \text{ and } M(p) < W_I(p,t) \text{ for all } p \in {}^\circ t.$$

The firing rule (2.1) remains unchanged.

2.2 Continous-time Markov chains

2.3 Stochastic Petri nets

Chapter 3

Stochastic analysis

3.1 Steady-state analysis

3.2 Transient analysis

3.2.1 Transient probability calculation

3.2.2 Accumulated probability calculation

3.3 Rewards and sensitivity

3.3.1 Stochastic reward nets

3.3.2 Sensitivity of rewards

Chapter 4

Efficient generation and storage of continous-time Markov chains

4.1 State-space exploration

4.1.1 Explicit state-space exploration

4.1.2 Symbolic methods

Multivalued decision diagrams

Edge-labeled decision diagrams

4.2 Storage of generator matrices

4.2.1 Explicit matrix storage

Dense matrices

Sparse matrices

Column major versus row major storage

Chapter 5

Algorithms for stochastic analysis

5.1 Steady-state analysis

5.1.1 Explicit solution by LU decomposition

5.1.2 Stationary iterative methods

Power iteration

Jacobi iteration and Jacobi over-relaxation

Gauss–Seidel iteration and successive over-relaxation

5.1.3 Krylov subspace methods

Biconjugate gradient stabilized (BiCGSTAB)

5.2 Transient analysis

5.2.1 Uniformization

Calculation of uniformization weights

- Weights for transient probability with *trimming*

- Weights for accumulated probability

Steady-state detection

5.3 Processing results

5.3.1 Calculation of rewards

Symbolic storage of reward functions

5.3.2 Calculation of sensitivity

Sensitivity of state probabilities

Sensitivity of rewards

Chapter 6

Configurable stochastic analysis

6.1 Matrix storage and algorithm selection in practice

6.2 Implementation of configurable workflows

Chapter 7

Evaluation

7.1 Benchmark models

7.1.1 Synthetic models

Resource sharing

Kanban

Dining philosophers

7.1.2 Case studies

Performability of clouds

7.2 Baselines

7.2.1 PRISM

7.2.2 SMART

7.3 Results

Chapter 8

Conclusion

8.1 Future work

References

[1] Piotr Chrzastowski-Wachtel. “Testing Undecidability of the Reachability in Petri Nets with the Help of 10th Hilbert Problem”. In: *Application and Theory of Petri Nets 1999, 20th International Conference, ICATPN '99, Williamsburg, Virginia, USA, June 21-25, 1999, Proceedings*. Vol. 1639. Lecture Notes in Computer Science. Springer, 1999, pp. 268–281. DOI: 10.1007/3-540-48745-X_16.

[2] Tadao Murata. “Petri nets: Properties, analysis and applications”. In: *Proceedings of the IEEE* 77.4 (1989), pp. 541–580.