PetriDotNet 1.5: Extensible Petri Net Editor and Analyser for Education and Research

András Vörös[1,2], Dániel Darvas[1], Vince Molnár[1,2], Attila Klenik[1], Ákos Hajdu[1,2], Attila Jámbor[1], Tamás Bartha[3], and István Majzik[1]

[1] Department of Measurement and Information Systems, Budapest University of Technology and Economics, Budapest, Hungary, {vori,darvas,molnarv,hajdua,majzik}@mit.bme.hu,
[2] MTA-BME Lendület Cyber-Physical Systems Research Group, Budapest, Hungary
[3] Institute for Computer Science and Control, Hungarian Academy of Sciences, Budapest, Hungary, tamas.bartha@sztaki.mta.hu

Abstract. PetriDotNet is an extensible Petri net editor and analysis tool originally developed to support the education of formal methods. The ease of use and simple extensibility fostered more and more algorithmic developments. Thanks to the continuous interest of developers (especially M.Sc. and Ph.D. students who choose PetriDotNet as the framework of their thesis project), by now PetriDotNet became an analysis platform, providing various cutting-edge model checking algorithms and stochastic analysis algorithms. As a result, industrial application of the tool also emerged in recent years. In this paper we overview the main features and the architecture of PetriDotNet, and compare it with other available tools.

Keywords: Petri nets · modelling · simulation · model checking · stochastic analysis · editor.

1 Introduction and Objectives

Ordinary and coloured Petri nets are simple, yet powerful formal modelling formalisms that are covered by most undergraduate Formal Methods courses. They are widely used to demonstrate concurrency, causality and other principles of systems design. In addition, thanks to their simple syntax and semantics, Petri nets help to introduce various generic concepts, such as state space, coverability graph, invariants, reachability, temporal logic and model checking. To support active learning, a demonstrator tool is needed to give the students insight into these concepts and let them do experiments on their own.

The goal of the Formal Methods courses is to teach students the foundations. Talented students, however, are interested in diving deeper into the theoretical

Authors' manuscript. Published in F. Kordon, D. Moldt (eds.): *Application and Theory of Petri Nets and Concurrency*, LNCS 9698, 2016. The final publication is available at Springer via http://dx.doi.org/10.1007/978-3-319-39086-4_9.

and algorithmic background. Petri nets have been studied for a long time, thus many analysis algorithms are described in the literature. The implementation and improvement of these algorithms can provide interesting challenge for the students, and it also gives an opportunity for the lecturers to assess the students' skills and scientific potential. In such cases a framework should be accessible for them to experiment with advanced algorithms and methods. Unfortunately, the freely available and actively maintained Petri net editor and analysis tools have different strengths and weaknesses (see more details in Section 5), thus it is hard to find a single tool that is able to satisfy the above mentioned requirements of the education as well as to provide analysis algorithms that can be effectively used and extended in research.

According to these needs, PetriDotNet, our Petri net editor and analysis tool offers the following main features:

- It provides a convenient graphical editor for creating (ordinary or coloured) Petri nets. Besides simulation, the basic step to understand the dynamic behaviour of nets, it is extended with analysis of structural and dynamic properties, as well as reachability and model checking capabilities in order to support the Formal Methods courses.
- Furthermore, PetriDotNet is a simple, generic and extensible platform for new, Petri net-based algorithms that provides wide access to the Petri net data structures and the graphical user interface.

The development of PetriDotNet started as an ordinary Petri net editor in 2007 [22] as a Master's thesis project (named as Petri.NET at that time). During the last nearly 10 years, the tool has been heavily modified, improved and extended, thanks to the contribution of many enthusiastic B.Sc., M.Sc. and Ph.D. students of the Fault Tolerant Systems Research Group (FTSRG) at the Budapest University of Technology and Economics.

The structure of this paper is the following. Section 2 describes the current functionality of PetriDotNet. The architecture of the tool is briefly described in Section 3. Next, we present some use cases in Section 4 and a comparison to other tools in Section 5. Finally, Section 6 discusses the installation and usage.

2 Functionality

This section overviews the main functionality of PetriDotNet and the plug-ins shipped with the tool.

Editor Features. First and foremost, PetriDotNet is an editor for Petri nets. It provides graphical editing capabilities (cf. Fig. 1) for both ordinary and well-formed coloured Petri nets (see [25] for the definition of the supported coloured Petri net variant). The tool supports Petri nets extended with inhibitor arcs, transition priorities, and places with limited token capacity. Moreover, the construction of hierarchical Petri nets is supported by allowing coarse transitions that can be refined by a subnet.

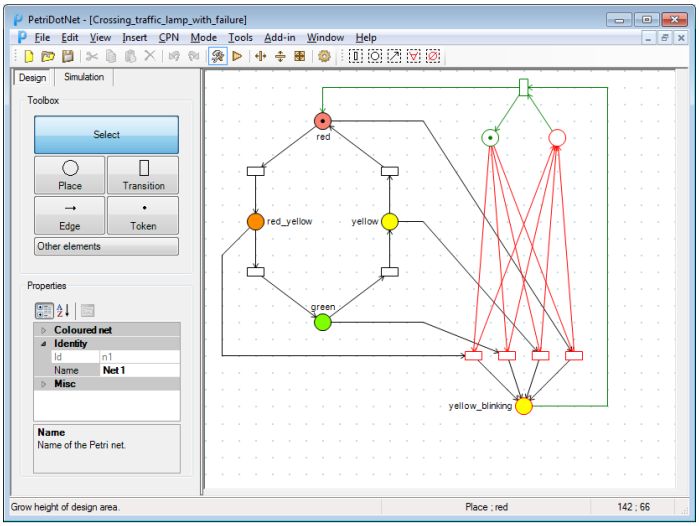

Fig. 1. The main window of PetriDotNet

The tool provides simulation functionality (*token game*) for Petri nets, where the simulation can be manually conducted or automatically executed. The tool is shipped with a plug-in that can perform large scale simulation, executing thousands or millions of non-deterministic firing, and then present the statistics.

To save and load the Petri nets, PetriDotNet supports two formalisms natively. The default format is PNML (Petri Net Markup Language) [15], a standard, XML-based Petri net description format. PNML is supported by various other tools, therefore this is an interface between these tools and PetriDotNet. A binary, custom file format is also supported that provides more efficient persistence for large models.

Plug-in Features. The functionality of the tool is extensible with plug-ins. Plug-ins can perform simulation tasks, provide analysis features (e.g. model checking) or export/import capabilities. Each plug-in can access the Petri net data models, use the graphical user interface, add new menu items, and call built-in PetriDotNet commands. The architecture of the tool is designed to keep the development of plug-ins simple, in order to help the users to focus on functionality instead of technology. See Section 3 for more details.

Export and Import Features. It is possible to export the constructed Petri nets into other Petri net formalisms, such as to the syntax of the GPenSIM[1] (General Purpose Petri Net Simulator) and the .pnt format of the INA[2] (Integrated Net Analyzer) tool. Also, the Petri net models can be translated into to

[1] `http://www.davidrajuh.net/gpensim/`
[2] `http://www2.informatik.hu-berlin.de/lehrstuehle/automaten/ina/`

the input format of SAL[3] (Symbolic Analysis Laboratory). Furthermore, import is also provided from the .net textual Petri net file format used by the INA/Tina[4] tools, among others. New import or export plug-ins can be developed easily as the internal model representations are simply accessible.

Formal Methods Course Plug-in. As one of the first motivations was to support the education, the framework has built in support for the following tasks:

- Calculating invariants, and displaying the results right on the Petri net,
- Generating the reachability/coverability graph, and exporting their graphical representation into image files,
- Computing various liveness properties [21].

The invariant analysis covers both P and T-invariants based on the well-known Martinez–Silva algorithm [17], and a different algorithm by Cayir and Ucer [2] that computes the bases of invariants.

Integrated Analysis Methods. In the last five years, in addition to the educational features, PetriDotNet became a Petri net analysis package providing plug-ins for a wide range of analysis methods. Among others, as detailed below, PetriDotNet supports advanced formal verification techniques based on decision diagrams and abstraction.

Saturation-Based Model Checking Algorithms. In PetriDotNet, various algorithms provide model checking based on the saturation algorithm [3–5]. The CTL model checking approaches are based on the work of Ciardo [28] and the bounded model checking approach is the extension of [27]. The LTL model checking algorithms are built on top of the ideas of [11, 14]. Our research resulted in significant extensions and improvements, this way PetriDotNet currently supports novel analysis algorithms as follows:

- CTL model checking of ordinary and coloured Petri nets based on traditional and extended versions of saturation [1, 25],
- Bounded CTL model checking based on a novel saturation-based algorithm, with various search strategies [10, 24],
- LTL model checking based on a novel synchronous product computation algorithm [20] and incremental SCC detection [19].

CEGAR-Based Reachability Algorithms. PetriDotNet includes reachability analysis algorithms based on Counterexample-Guided Abstraction Refinement (CEGAR) [6] for ordinary Petri nets. Petri net CEGAR-based algorithms overapproximate the set of reachable states using the state equation, which is a necessary criterion for reachability. The CEGAR algorithm for Petri nets introduced in [26] was the base of our work. Our implementation includes various search strategies, adapted to the characteristics of the different models [12, 13].

[3] `http://sal.csl.sri.com/`
[4] `http://projects.laas.fr/tina/`

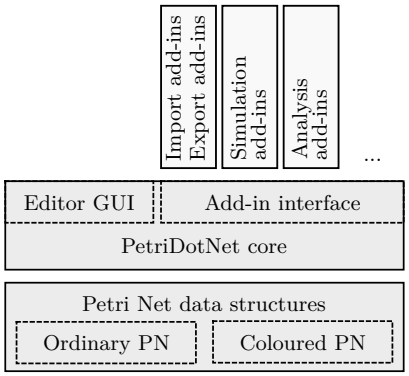

Fig. 2. High-level overview of the PetriDotNet architecture

Stochastic Analysis Algorithms. Recently the tool was extended to support the modelling and analysis of stochastic Petri net models. The goal was to provide a configurable stochastic analysis framework where various state space exploration, matrix representation and numerical analysis algorithms can be combined [16]. PetriDotNet provides the following stochastic analysis for ordinary stochastic Petri net models:

- Steady-state reward and sensitivity analysis,
- Transient reward analysis,
- Calculation of the mean time to reach a state partition, that is used to calculate mean-time-to-first-failure (MTFF) in dependability models.

3 Architecture

General Architectural Overview. The tool is written in C#, based on the Microsoft .NET framework. The architecture of PetriDotNet is kept as simple as reasonably possible. It is a modular tool: it provides some basic functionalities, and can be extended by various plug-ins.

The tool uses a base library defining the Petri net data structures, developed for PetriDotNet. This library contains object models for ordinary and coloured Petri nets. The PetriDotNet core contains the graphical user interface and the plug-in interface. The architecture of the tool is summarized in Fig. 2.

Plug-in Interface. To follow the previously presented educational goals, it is simple to extend PetriDotNet with a new plug-in. This allows a steep learning curve and low entry barrier, therefore the plug-in developers can focus on their algorithms, instead of the applied technologies. From the tool's point of view, a plug-in is just a .dll file in the `add-in` folder, in which at least one class implements the `IPDNPlugin` interface (see Fig. 3). Metadata about the plug-in (e.g. name, author, required PetriDotNet version) can be provided using annotations

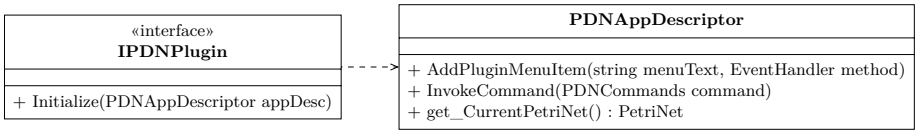

Fig. 3. PetriDotNet plug-ins

of this class (e.g. `[AddinAuthor("X. Y.")]`). When PetriDotNet starts, it loads all plug-ins and calls their `Initialize` method. In this method the plug-ins can make their menu contributions and store the application descriptor. This latter allows the plug-ins to call commands (e.g. save, load) and to access the currently active Petri net.

Being a .NET-based tool, PetriDotNet requires that the plug-ins are also implemented in one of the .NET languages. While having a graphical editor for Petri nets developed in .NET is a reasonable choice, implementing e.g. model checking algorithms seems to be uncommon, as managed languages are considered to have some overhead. However, *(i)* according to our experience the run time of the .NET-based implementations of various model checking algorithms proved to be competitive compared to their native version, and *(ii)* the development in .NET is easier and less error-prone than e.g. in C or C++ for computer engineering students, allowing them to make correct implementations in shorter time. Thus the choice of .NET can be regarded as sacrificing some run time performance in favour of development time, which is similarly important in our educational setting. If the performance needs cannot be satisfied using .NET, the plug-in can wrap or depend on a native implementation (.dll).

4 Use Cases

This section overviews the use cases where we applied PetriDotNet as an editor or an analysis tool. According to the original goals we start the overview with educational use cases, then we move on to industrial case studies.

Application in Education. PetriDotNet is used as an educational tool and a tool for the homework assignments in the Formal Methods course of the Budapest University of Technology and Economics since 2011. During this time, approximately 900 M.Sc. students attended the course. The stochastic analysis module of the tool is used for demonstration purposes in the Software and Systems Verification course to teach reliability modelling for the students.

Student Projects. To this day 23 B.Sc. and M.Sc. theses were written that applied or extended PetriDotNet. Besides, the various new formal verification algorithms resulted in 20 scientific papers presented in conferences or journals[5]. Several students who started to get familiar with research by extending and implementing

[5] See the complete list of related publications at `http://petridotnet.inf.mit.bme.hu/publications/`.

an algorithm in PetriDotNet are now Ph.D. students or planning to apply for post-graduate programmes.

Application in Industrial Cases. We are aware of various usages of our tool to model, simulate and analyse different real life systems.

- We have applied PetriDotNet to model and formally verify a safety logic of the Paks Nuclear Power Plant using saturation-based CTL model checking in [1, 25]. This work validated the coloured Petri net editing capabilities and proved the efficiency of our CTL model checking algorithms, as [1] presented the first successful formal verification of the complete safety logic.
- PetriDotNet was used to model and simulate sensor nets in [18] and in the FuturICT.hu project[6].
- PetriDotNet was applied to model and study railway interlocking systems [7].
- The R3-COP project[7] applied PetriDotNet to generate test input sequences for testing the robustness of communicating autonomous robots [9].
- Initial case studies were made to apply PetriDotNet to analyse control software used at the European Organization for Nuclear Research (CERN) [8].
- Stochastic analysis and MTFF computation were used in an industrial project at our department to evaluate safety (hazard rate) of an embedded control system. The mean time to reach undetected failures or shutdown was computed in a stochastic model of a two-channel architecture with separate diagnostic facility, comparison, and time-limited degraded (single-channel) functionality.

5 Comparison with Other Tools

During the last decades several Petri net based editor and analysis tools were implemented, some of them are surveyed in [23]. In our comparison (see Table 1) we focused on the freely available, actively maintained, and/or widely used tools. The table characterizes the supported nets, the tool features, and the analysis capabilities. It illustrates that PetriDotNet is a quite versatile tool, offering the features necessary for educational purposes, and also providing sophisticated analysis capabilities [19] that allow to use it as a modelling and analysis tool in research and industrial development projects.

As a highlight, we emphasize the extensibility (plug-ins). The only other tool providing a plug-in interface is Charlie, but Charlie is strictly an analysis tool, whereas PetriDotNet contains the editing, simulation, and analysis functions in a single integrated application. Moreover, the features of the plug-ins in Charlie are more limited, while PetriDotNet plug-ins can extend all of its main functions. Other important features of PetriDotNet are the standard PNML support, the efficient state space representation, the CTL and LTL model checking, and the reward-based stochastic analysis capabilities.

[6] `http://www.futurict.szte.hu/en/home/`
[7] `http://www.r3-cop.eu/`

Table 1. Comparison of Petri net editor and analysis tools

	Supported Petri nets						Features								Analysis tools						
Petri net tool	Place/Transition nets	Extended Petri nets	High-level Petri nets	Petri nets with time	Stochastic Petri nets	Continuous Petri nets	Graphical editor	Extensibility (plug-ins)	State spaces	Condensed state spaces	Token game	Fast Simulation	Net reduction	PNML support	Place/Trans. invariants	Structural analysis	Reachability graph an.	Simple performance an.	Advanced performance an.	Model checking	Free of charge
PetriDotNet	●	●	○	○	●		●	●	●		●	●		●	●	○	○	●	●	●	●
PEP	●	●	●	●			●		●	●	●		●		●	●					●
INA	●	●	●						●	●	●		●		●	●					●
Snoopy	●	●		●	●	●	●				●	●		○							●
Charlie	●	●						●	●						●	●	●			●	●
Marcie		●			●				●			●						●		●	●
PETRUCHIO	●	●	●	●	●		●		●		●	●			●			●			●
GreatSPN		●	●	●	●		●		●	●	●	●			●	●		●	●		●
TimeNET		●	●	●	●		●								●	○		●	●		○
CPN Tools			●	●			●		●	●	●	●			○		●	●		●	●

● – full support, ○ – partial support

In our view, having all the features summarized in Table 1 in a single, integrated, and extensible tool is a unique feature. Note that the specific features of our enhanced analysis algorithms (implemented as plug-ins) are detailed and compared to similar solutions in the respective papers (see the references in Sections 2 and 4).

6 Installation and Usage

The installation and usage of PetriDotNet is extremely simple. After downloading the tool from our website[8], the user has to extract the tool by executing the downloaded file. After this, the tool can be started by running `PetriDotNet.exe`.

During the development, we have paid special attention to keep the main features easy to use. Using the toolbox on the left side of the window it is straightforward to create or modify Petri nets. To use the more advanced features (e.g. analysis modules), we refer the reader to the user manual, accessible on our webpage.

Installing add-ins is similarly straightforward: the files of the add-in have to be placed under the `add-in` folder of the tool. From the next startup of PetriDotNet, the add-ins will be loaded and their menu items will show up in the `Add-in` menu.

PetriDotNet is shipped with a set of analysis and import/export plug-ins. They can be accessed from the `Add-in` menu. The invariant, reachability, CTL and LTL analysis algorithms are aggregated to a common interface that can be accessed by selecting the `Net analysis` item in the `Add-in` menu.

[8] `http://petridotnet.inf.mit.bme.hu/en/`

A set of example and benchmark models is distributed with the tool, located in the `models` folder.

Acknowledgement. The authors are grateful for all colleagues, former and present students and external users involved in the development, testing or the usage of the tool. Special thanks to Bertalan Szilvási for developing Petri.NET, the initial version of the presented tool.

This paper is partially supported by the MTA-BME Lendület 2015 Research Group on Cyber-Physical Systems and by the ARTEMIS JU and the Hungarian National Research, Development and Innovation Fund in the frame of the R5-COP and R3-COP projects.

References

1. Bartha, T., Vörös, A., Jámbor, A., Darvas, D.: Verification of an industrial safety function using coloured Petri nets and model checking. In: Proc. of the 14th Int. Conf. on Modern Information Technology in the Innovation Processes of the Industrial Enterprises. pp. 472–485. Hungarian Academy of Sciences (2012)
2. Cayir, S., Ucer, M.: An algorithm to compute a basis of Petri net invariants. In: 4th ELECO Int. Conf. on Electrical and Electronics Engineering. UCTEA, Bursa, Turkey (2005)
3. Ciardo, G., Marmorstein, R., Siminiceanu, R.: The saturation algorithm for symbolic state-space exploration. International Journal on Software Tools for Technology Transfer 8(1), 4–25 (2006)
4. Ciardo, G., Yu, A.J.: Saturation-based symbolic reachability analysis using conjunctive and disjunctive partitioning. In: Borrione, D., Paul, W. (eds.) CHARME 2005, LNCS, vol. 3725, pp. 146–161. Springer (2005)
5. Ciardo, G., Zhao, Y., Jin, X.: Ten years of saturation: A Petri net perspective. In: Transactions on Petri Nets and Other Models of Concurrency V, LNCS, vol. 6900, pp. 51–95. Springer (2012)
6. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000, LNCS, vol. 1855, pp. 154–169. Springer (2000)
7. Cseh, A., Tarnai, G., Sághi, B.: Petri net modelling of signalling systems [in Hungarian, original title: Biztosítóberendezések modellezése Petri-hálókkal]. Vezetékek Világa XIX(1), 14–17 (2014)
8. Darvas, D., Fernández Adiego, B., Blanco Viñuela, E.: Transforming PLC programs into formal models for verification purposes. Internal Note CERN-ACC-NOTE-2013-0040, CERN (2013)
9. Darvas, D., Vörös, A.: Saturation-based test input generation using coloured Petri nets [in Hungarian, original title: Szaturációalapú tesztbemenet-generálás színezett Petri-hálókkal]. In: Mesterpróba 2013. pp. 48–51 (2013)
10. Darvas, D., Vörös, A., Bartha, T.: Improving saturation-based bounded model checking. Acta Cybernetica (2014), accepted, in press. `http://petridotnet.inf.mit.bme.hu/publications/AC2014_DarvasEtAl.pdf`
11. Duret-Lutz, A., Klai, K., Poitrenaud, D., Thierry-Mieg, Y.: Self-loop aggregation product – A new hybrid approach to on-the-fly LTL model checking. In: Bultan, T., Hsiung, P. (eds.) ATVA 2011, LNCS, vol. 6996, pp. 336–350. Springer (2011)

12. Hajdu, Á., Vörös, A., Bartha, T.: New search strategies for the Petri net CEGAR approach. In: Devillers, R., Valmari, A. (eds.) Petri Nets 2015, LNCS, vol. 9115, pp. 309–328. Springer (2015)
13. Hajdu, Á., Vörös, A., Bartha, T., Mártonka, Z.: Extensions to the CEGAR approach on Petri nets. Acta Cybernetica 21(3), 401–417 (2014)
14. Heiner, M., Rohr, C., Schwarick, M.: MARCIE – model checking and reachability analysis done efficiently. In: Colom, J.M., Desel, J. (eds.) Petri Nets 2013, LNCS, vol. 7927, pp. 389–399. Springer (2013)
15. ISO/IEC 15909-2 Systems and software engineering – High-level Petri nets – Part 2: Transfer format (2011)
16. Klenik, A., Marussy, K.: Configurable stochastic analysis framework for asynchronous systems. Scientific students' associations report, Budapest University of Technology and Economics (2015), `http://petridotnet.inf.mit.bme.hu/publications/TDK2015_KlenikMarussy.pdf`
17. Martínez, J., Silva, M.: A simple and fast algorithm to obtain all invariants of a generalised Petri net. In: Girault, C., Reisig, W. (eds.) Application and Theory of Petri Nets, Informatik-Fachberichte, vol. 52, pp. 301–310. Springer (1982)
18. Milánkovich, A., Ill, G., Lendvai, K., Imre, S., Szabó, S.: Evaluation of energy efficiency of aggregation in WSNs using Petri nets. In: Proc. of the 3rd Int. Conf. on Sensor Networks. pp. 289–297. Science and Technology Publications (2014)
19. Molnár, V., Darvas, D., Vörös, A., Bartha, T.: Saturation-based incremental LTL model checking with inductive proofs. In: Baier, C., Tinelli, C. (eds.) TACAS 2015, LNCS, vol. 9035, pp. 643–657. Springer (2015)
20. Molnár, V., Vörös, A., Darvas, D., Bartha, T., Majzik, I.: Component-wise incremental LTL model checking. Formal Aspects of Computing (2016), in press, available on-line. DOI: 10.1007/s00165-015-0347-x
21. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580 (1989)
22. Szilvási, B.: Development of an education tool for the Formal methods course [in Hungarian, original title: Oktatási segédeszköz fejlesztése Formális módszerek tárgyhoz]. Master's thesis, Budapest University of Technology and Economics (2008)
23. Thong, W.J., Ameedeen, M.A.: A survey of Petri net tools. ARPN Journal of Engineering and Applied Sciences 9(8), 1209–1214 (2014)
24. Vörös, A., Darvas, D., Bartha, T.: Bounded saturation-based CTL model checking. Proceedings of the Estonian Academy of Sciences 62(1), 59–70 (2013)
25. Vörös, A., Darvas, D., Jámbor, A., Bartha, T.: Advanced saturation-based model checking of well-formed coloured Petri nets. Periodica Polytechnica, Electrical Engineering and Computer Science 58(1), 3–13 (2014)
26. Wimmel, H., Wolf, K.: Applying CEGAR to the Petri net state equation. In: Abdulla, P.A., Leino, K. (eds.) TACAS 2011, LNCS, vol. 6605, pp. 224–238. Springer (2011)
27. Yu, A.J., Ciardo, G., Lüttgen, G.: Decision-diagram-based techniques for bounded reachability checking of asynchronous systems. International Journal on Software Tools for Technology Transfer 11(2), 117–131 (2009)
28. Zhao, Y., Ciardo, G.: Symbolic CTL model checking of asynchronous systems using constrained saturation. In: Liu, Z., Ravn, A.P. (eds.) ATVA 2009, LNCS, vol. 5799, pp. 368–381. Springer (2009)