GreatTeach: A Tool for Teaching (Stochastic) Petri Nets

Elvio Gilberto Amparore(✉) and Susanna Donatelli(✉)

Dipartimento di Informatica, Università di Torino, Turin, Italy
{amparore,susi}@di.unito.it

Abstract. GreatSPN is a collection of tools for modelling and analysis of systems using (stochastic) Petri nets. It features a modern Java-based graphical interface. But being technologically advanced does not mean that the tool learning curve is significantly simplified. In the tool the simple features that are needed for educational purpose are intermixed with more advanced features that require a deeper understanding of the formalisms and of the solvers. This paper presents GreatTeach, a streamlined and enriched version of GreatSPN meant for teaching resulting from our experience in teaching Petri nets to master students.

Keywords: GreatSPN · Teaching · Modelling · Verification

1 Introduction

Tools play a central role in teaching formal methods: most classes we know of use some form of tool support to teach students the basic concepts and to let them experiment the learned concepts on a (more or less realistic) examples with the help of a tool. Petri nets are very appealing when it comes to teach formal methods: the basics of the formalism are very simple (just the enabling and firing rule), and there is a widely accepted graphical representation. But Petri nets comes also in a large variety of definitions, with a rich set of analysis techniques, so typically Petri nets tools are rather rich in features. Moreover many tools also support the extension of Petri nets to consider time (non deterministic or stochastic), which increases the complexity. Once decided to teach Petri nets in a course, the teacher has a wide choice of tools that have rather diverse characteristics: from very simple educational tools that concentrate only on the token game, visualization of the incidence matrix and firing rule exemplification like in [15], to very powerful tools like CPN-tools [14] a tool for constructing, simulating, and performing analysis of Colored Petri Nets (CPN) models (timed and untimed). CPN-tools has a large community of users, a book [13] is available to learn the formalism, the tool and its industrial applications; the book also includes a chapter on teaching CPN as a full course. A feature to automatically grade student exercises (the Grade/CPN tool) is also available. The tool is quite tight to colored Petri nets and the graphical interface has some uncommon features, not obvious for newcomers. In between there are a large

© Springer International Publishing AG, part of Springer Nature 2018
V. Khomenko and O. H. Roux (Eds.): PETRI NETS 2018, LNCS 10877, pp. 416–425, 2018.
https://doi.org/10.1007/978-3-319-91268-4_24

number of tools stemming from research groups that are made available. Examples of tools for the definition and analysis of P/T nets, Stochastic Petri Nets (SPN) and various extensions are the suite made of the trio Snoopy, Charlie, and Marcie [11,12,16], GreatSPN [3], and TimeNet [20]. There are also tools that have been initially developed for teaching, in particular as a platform for letting students (and other researchers) add new analysis techniques, like, for example, PetriDotNet [19] and the Snoopy suite to a certain extent. A comprehensive survey can be found in [17], while [19] includes a comparison table for various tools.

The authors of [8] advocate that students should use industrial-size tools: indeed while it is true that a simple tool with few features it is likely to be more portable and easier to learn, it is also true that, having to change tool to access to more advanced features can significantly slow down the learning process. As far as we know there is no tool that has been explicitly designed with the objective of offering at the same time a rich set of analysis techniques and an easy-to-use tool for lab classes.

Motivations and Objectives. As a teacher of a course in verification of concurrent processes for master students in computer science at the University of Torino, one of the authors of this paper has taught for many years various formalisms and the associated analysis techniques. Students in the course learn and experiment (colored) Petri nets and CTL with the support of GreatSPN, guarded command languages and LTL/CTL with the support of nuSMV [10], and timed automata with Uppaal [6]. The use of GreatSPN in the course has been one of the main drivers behind the introduction of the new Java-based graphical interface (GUI) of GreatSPN [2] a few years ago. Although with the new GUI we observed a great increase in acceptance and appreciation from the students (GreatSPN surpassed Uppaal in the preferences of the students), the learning curve of the tool proved to be still quite steep. Observing the students during the lab classes and examining their questions and e-mails we realized that the students spent a significant amount of time in struggling among the numerous options and features offered by the tool. And this aspect was not mitigated neither by the precise indications given during lab classes nor by the availability of a number of video tutorials. From our "observation in the wild" we concluded that teaching requires the introduction of a "limitation of scope" in the tool.

We also observed, as many other teachers have done (see for example [8]) that the learning approach of the students has significantly changed over the years and that more and more the students tend to use the "learning by example" approach. The authors of [8] conclude that "formal methods are best taught by examples". In our course we do not totally follow this paradigm, since the course has also the learning objective of reading and writing formal statements and definitions, but over the years we have inverted the order of presentation of the material: we start with examples using the tools (we draw a net and we play the token game) and drive back to the formal definitions, possibly with a number of iterations. Of course to start with examples you need an adequate tool support: for each learning objective there should be adequate *graphical* tool

support. For example teaching decision diagrams (DD) for storing a reachability set (RS) is much easier if the tool also visualize the RS in DD form. In a similar manner, if I want to convey the information that DD size can heavily depend on the variable order (place order) I can use examples of Boolean functions from the literature or I can show a P/T net in which, by changing the place order in the DD, the DD size changes significantly, which requires a tool in which it is easy to change the variable order. In our experience this approach is effective only if the tool provides the required features in a straightforward manner.

Based on the above insights we have decided to produce a *streamlined* and *enhanced* version of GreatSPN [18], called GreatTeach. It is streamlined because it presents to the user an easy access to a subset of the GreatSPN resources (intended as Petri net classes, solution techniques, facilities) and it is enhanced because it has some new visualization possibilities that have been introduced as a support for teaching. At the same time all previous solvers are still available.

The paper develops as follows: Sect. 2 describes the architectural view of the new interplay between GUI and solvers, Sect. 3 describes how GreatTeach supports the learning objectives of a course on Petri nets, and Sect. 4 concludes with a discussion of the current status of GreatTeach and of future work.

2 GreatTeach, an Architectural View

GreatSPN includes facilities for drawing and solving many classes of nets, but for teaching we typically use only four of them: P/T nets and colored "well-formed" nets (WN) [9] in verification courses; (Generalized) stochastic Petri nets - (G)SPN [1] and colored extensions of Stochastic WN (SWN) [9] in performance evaluation courses.

The Status Quo. GreatSPN does not have a strict enforcement of the net type, which is instead computed "on-the-fly" depending on the included elements. For example a P/T net becomes a GSPN if a rate is added to a transition, and a P/T net becomes a WN if a color class is added to the net declarations. This choice is coherent with the idea that qualitative (untimed) and quantitative (stochastic) analysis should work in an integrated and synergistic manner.

Figure 1[A] shows the GreatSPN GUI for a P/T net. There are two menu bars: a command bar with icons to "manage" a net and a "project" (set of nets with associated measures) together with a few commands for quick analysis; a net editing bar which is the drawing bar for the net displayed in the main canvas. Tooltips are associated with the icons to make buttons self-explanatory. The net editing bar is context sensitive: if a WN is loaded in the canvas the editing bar shows additional elements, to include drawing elements for colors (color definition, color variables, etc.). The command bar instead *is not* context sensitive: so the command bar for P/T in Fig. 1[A] includes the unfolding command (useful only for colored nets). A similar situation is present for WN: the bar includes the icons for the computation of P- and T-semiflows, for which there is no solution algorithm in GreatSPN. If this feature is selected the tool notifies that this is

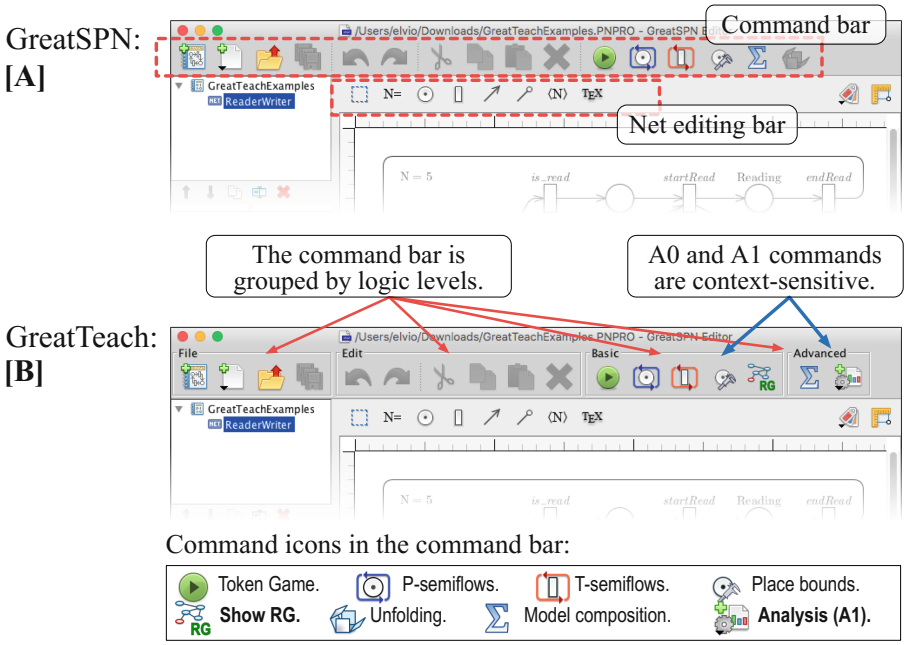

Fig. 1. The GUI for P/T nets: GreatSPN (left) and GreatTeach (right).

not possible for colored net. This is not a problem for an expert user, but it is annoying and quite distracting in lab classes.

Let us now consider how to an analysis when it is part, or when it is not part, of the command bar. P-semiflows *are* on the command bar: by pushing the button the P-semiflows are computed and graphically displayed in the main canvas over the net and textually on a lateral canvas, without any input from the user. A "close" button allows to come back to the normal net visualization. Reachability graph (RG) computation and visualization *is not* on the command bar. To run it the user has to: (1) go to the command bar and click the "add new page" button; (2) select "add a new list of measures" from the drop-down menu, which switches the GUI to the frame depicted in Fig. 2[A]; the figure depicts the remaining steps that are: (3) select the Petri net for which to perform the analysis; (4) select which target measure (in this case RG) to compute from the drop-down menu of the "Add measure" button; (5) select which solver to use; (6) set the additional parameters required by the solver (for example the type of algorithm and the maximum number of markings that we want to graphically display); (7) push the "compute" button, which opens a window with a log that allows, if necessary, to kill the solution; (8) once the computation has terminated push the "view result" button for the computed measure that, in this case, will open a pdf file with the RG visualization as a graph. Obviously 8 steps for an RG computation lead to a high cognitive load for a newcomer to the tool (imagine having to explain this in a lab class), while they could be acceptable for an expert, who may desire exactly this flexibility to try different solution algorithms (there are many ways of computing the RG in GreatSPN).

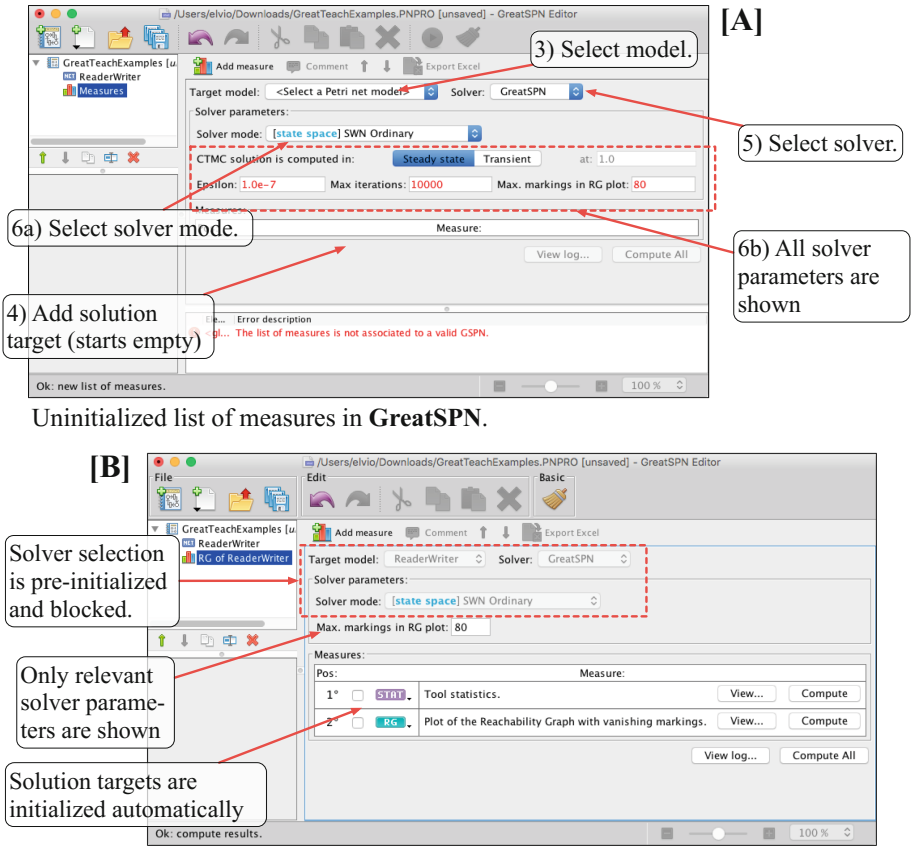

Fig. 2. Calling a solution in GreatSPN and in GreatTeach.

GreatTeach Architecture. The previous example clearly shows that GreatSPN and its GUI were designed for researchers, and that using a research tool to teach, although very desirable in our opinion, may not be effective with students. We have therefore re-structured the interplay between nets, solvers and target measures into three classes, called *architectural levels*:

A0. Commands accessible with a "push button". These are commands for which we have decided that there should be no additional interaction with the user, not even to kill the solver or to assign required parameters, that should be substituted by pre-defined, built-in, values.
A1. Commands that require an interaction with the user, but this interaction is in a simplified form; again, if a more complex interaction is needed by the solver, it should be simplified by making pre-defined choices.
A2. The full set of commands, solver and options currently available through the "list of measures" page.

The command bar has been re-organized into boxes: File, Edit, Basic, and Advanced, as in Fig. 1[B]. Commands designed according to the architectural level A0 are all placed in the "Basic" box. The ones designed at level A1 are put in the "Advanced" box, and they can be selected from a drop-down menu that is associated to the "Analysis" button (rightest button on the command bar). An example of the menu for P/T net is shown in Fig. 3[A]. Commands at level A3 are accessible through the standard "List of measure" page.

A good example for explaining the three levels is again the RG computation for which users of different expertise may require different types of computation. Indeed the first net drawn by a student has good chances to have an unbounded number of states, or too many states to make the picture of the RG meaningful, so the "push button" could be a good option but it should be carefully designed. A researcher working on very large nets may want instead to tailor all possible parameters. GreatTeach therefore has one RG computation for level. In A0 there is a "limited RG" solution, accessible through the button "showRG" (see Fig. 1[B]), that generates at most 10.000 states and displays the pdf of only the first 80 states and relative (possibly dangling) arcs. The pdf also includes standard statistics (number of states and of dead markings). This is a solver that is new to GreatTeach. At level A1, when RG computation is selected from the drop-down menu, a customized measure page is presented. This page is shown in Fig. 2[B]: the net name (target model) and the solver have been pre-selected and they are not modifiable, and the user can select a single parameter (the number of states to be drawn in the pdf of the RG). The target measures and parameters are pre-initialized. So the user can simply press the "Compute" button and, in two clicks, he/she gets with GreatTeach the same results that requires 8 steps in GreatSPN. Again, this feature has been developed specifically for GreatTeach. At level A2 the computation is as in GreatSPN: fully flexible but with 8 interactions.

Another important architectural change in GreatTeach is that also the command bar has been made context-sensitive to the type of the net currently displayed in the tool canvas. As can be observed by comparing the command bar portions of the GreatTeach GUI shown in Fig. 3 for P/T net (case A), for GSPN (case B) and for WN (case C) the available commands in the Basic and in the advanced box and drop down menu are now different. Note that the command bar implements the same full flexibility that GreatSPN has in changing the net editing bar: in GreatTeach if the canvas shows a P/T net and we add a color to a place, both the net editing bar and the command bar change accordingly to provide support for WN drawing and solution.

3 GreatTeach, a Teaching View

The choice of having three architectural levels was guided by a restructuring of our learning objectives into three levels. This classification is totally based on our experience since, as far as we know, there is no shared and accepted notion of basic or advanced knowledge for Petri nets. The idea is that the first two levels are for master courses, while the third level is for final projects. A generic description of the levels is:

L0: basic knowledge, sufficient for a student to produce and analyse simple properties on simple models, for all net classes
L1: intermediate knowledge, aimed at understanding an extended set of analysis techniques
L2: advanced knowledge, in particular to experiment and compare different solvers for the same target objectives.

In the remaining of the section we identify the learning objectives for each net class and we connect it with the organization into the three architectural levels of the solvers. For P/T we have, at a minimum:

L0: Understand and experiment with enabling and firing rule, RS and RG definition, bound of places, P- and T-invariants.
L1: Understand use of decision diagrams for RS computation, define and model-check CTL properties, build nets through net composition.
L2: Every other learning objective supported by GreatSPN, like to understand the differences between vanishing vs. tangible states, to generate plots based on parametric analysis, and to experiment and compare different solvers.

The matrix of learning objective levels vs. architectural levels of solvers should have a full diagonal: for an objective at level Li there should be *at least* one method at architectural level Ai, and this is indeed what we have now available in GreatTeach. The association of commands to a learning objective can be easily described with the support of the screenshots of the command bar included in Fig. 3. For P/T nets (Fig. 3[A]) the Basic box (level A0 commands) includes the icons for Token Game (to learn enabling and firing), place bounds computation, P- and T- semiflows computation and visualization and (limited) RS and RG

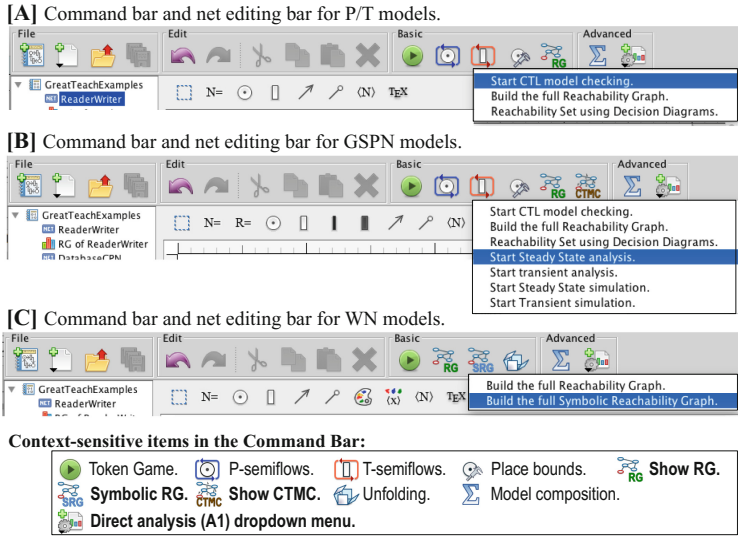

Fig. 3. Command bars in GreatTeach (Color figure online)

computation and visualization. In the legenda of the icons, we have used the boldface to emphasize the solver that have been *expressly developed or modified* for GreatTeach. The icon for the "Direct analysis (A1) drop-down menu" is in bold since *all the solvers in the drop-down menu* (level A1 solvers) have been explicitly developed or modified for GreatTeach. The Advanced box (level A1 commands) includes an icon for Model Composition (according to [7]) and a drop down menu with the three commands shown in the Figure, that meets the learning objectives in L1 and that can support a more complete analysis of the RS and RG computation (with respect to the level L0 commands present in the Basic box). Due to lack of space we cannot show an example, but the "Reachability set using Decision Diagram" builds the RS and produces a pdf of the decision diagram, a graphical feature that we have explicitly developed for GreatTeach, "Build the full RG" leads to the page shown in Fig. 2, and "Start CTL model checking" leads to a page in which the user can, with a single click, evaluate [4] a predefined CTL formula, or change it to fit their needs. The user can also specify a predefined variable order or a manual one. CTL can also be run at level A2, to experiment with different heuristics [5] for variable ordering. For GSPNs we have identified, at a minimum:

L0: as for P/T, plus the distinction between vanishing and tangible markings, the construction of the Continuous Time Markov Chain (CTMC) of the net with attention to the probabilities computed for conflict resolution.
L1: as for P/T, plus CTMC solution (transient and steady-state) based on RS or on simulation and computation and visualization of basic performance indices (token distribution in places and throughput of transitions).
L2: everything else, including model checking of the stochastic logic CSL^{TA}, definition of generic performance indices, advanced solution techniques.

Figure 3[B] is the command bar for GSPN. The solvers in the Basic box are the same as for P/T plus a CTMC computation and visualization at level A0. The visualized CTMC has the explicit indication of the rate computation, to learn the interplay between rates of exponential transitions and weights of immediate ones. The showRG icon builds a RG in which vanishing and tangible states are distinguished. The Advanced box has a drop down menu with four more options for transient/steady-state solution based on CTMC computation or simulation. These are level A1 solvers: they build a measure page that requires the minimum interaction from the user.

A similar identification of learning objectives has been done for WN and SWN. Without listing the full set of objectives, we can emphasize that the learning objectives for WN are basically the same as for P/T nets, plus the specific objectives connected to the definition and comprehension of the colors, of the colored and symbolic RG definition and of their differences (level L0). Figure 3[C] shows the command bar of WN. In the Basic box we have the Token game (colored), limited RG and limited SRG computation and the unfolding of a WN into an equivalent P/T, while the Advanced box includes Model Composition and the full construction of RG and SRG. For SWN the menu of the advanced box is enriched with the four stochastic solvers, as in the GSPN case, combined with the choice of using a CTMC built on the RG or on the SRG.

4 Conclusions and Future Work

Literature reports many studies on teaching formal methods in general and Petri nets in particular. In almost all cases, the authors agree that a course needs a mixture of theoretical concepts and practical experiments with software tools. In our teaching experience we could observe that a graphically nice interface helps, but it is not enough. The interface should be designed with teaching in mind and this is what we have done with GreatTeach. GreatTeach has some enhanced features for graphical visualization (DD and CTMC among others) and a streamlined interface: easy and/or basic learning objectives should require an easy interaction with the tool. The more the student learns, the deeper the interaction with the tool can be. There are many things we want to add as a support for teaching, among others: some RG analysis algorithms are currently not accessible, rewards for SPN should make the definition of measures significantly easier, and we need to include an easy way to introduce the students to our stochastic model-checker for CSL^{TA}. In introducing these new features we shall first define the level L_i of the learning objective, and then design the solver according to the chosen architectural level. This approach, we believe, is a contribution of the paper that goes beyond the specific tool considered.

The learning objectives of the previous section, and therefore the choice of the properties that are analyzable at level A0 and A1 are strongly dependent on the courses that we teach at the University of Torino. Different courses may need a different characterization. The newest release of GreatSPN is available at github.com/greatspn/SOURCES: having the source code available on Github should allow other research group to develop their own customization based on a specific choice of learning objective. Distribution to students of a frozen version is also important when working in lab classes, to ensure that all students work with exactly the same tool. A Virtualbox image of GreatTeach is available at http://www.di.unito.it/~greatspn/VBox/GreatTeach.ova.

The attentive reader may have noticed that the name of the editor on top of the GUI in Figs. 1 and 2 is GreatSPN on both sides. This is not an error: the architectural choices of GreatTeach led to a GUI that is more structured and has a number of additional features and solvers specifically developed for it. We believe that GreatTeach can work well also for expert researchers, therefore we have decided to use GreatTeach as the next release of GreatSPN, which also avoids the creation of a branch in the development process. In the paper we retain the two separate names for clarity.

References

1. Ajmone-Marsan, M., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modelling with Generalized Stochastic Petri Nets. Wiley, Hoboken (1995)
2. Amparore, E.G.: Reengineering the editor of the GreatSPN framework. In: PNSE@ Petri Nets, pp. 153–170 (2015)
3. Amparore, E.G., Balbo, G., Beccuti, M., Donatelli, S., Franceschinis, G.: 30 years of GreatSPN. In: Fiondella, L., Puliafito, A. (eds.) Principles of Performance and Reliability Modeling and Evaluation. SSRE, pp. 227–254. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-30599-8_9

4. Amparore, E.G., Beccuti, M., Donatelli, S.: (Stochastic) model checking in GreatSPN. In: Ciardo, G., Kindler, E. (eds.) PETRI NETS 2014. LNCS, vol. 8489, pp. 354–363. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07734-5_19
5. Amparore, E.G., Beccuti, M., Donatelli, S.: Gradient-based variable ordering of decision diagrams for systems with structural units. In: D'Souza, D., Narayan Kumar, K. (eds.) ATVA 2017. LNCS, vol. 10482, pp. 184–200. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68167-2_13
6. Behrmann, G., David, A., Larsen, K.G.: A tutorial on Uppaal. In: Bernardo, M., Corradini, F. (eds.) SFM-RT 2004. LNCS, vol. 3185, pp. 200–236. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30080-9_7
7. Bernardi, S., Donatelli, S., Horváth, A.: Implementing compositionality for stochastic Petri nets. Softw. Tools Technol. Transf. J. **3**(4), 417–430 (2001)
8. Cerone, A., Roggenbach, M., Schlingloff, H., Schneider, G., Shaikh, S.: Teaching formal methods for software engineering - ten principles. In: Informatica Didactica. University of Potsdam, Germany (2015)
9. Chiola, G., Dutheillet, C., Franceschinis, G., Haddad, S.: Stochastic well-formed colored nets and symmetric modeling applications. IEEE Trans. Comput. **42**(11), 1343–1360 (1993)
10. Cimatti, A., Clarke, E., Giunchiglia, F., Roveri, M.: NuSMV: a new symbolic model verifier. In: Halbwachs, N., Peled, D. (eds.) CAV 1999. LNCS, vol. 1633, pp. 495–499. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48683-6_44
11. Heiner, M., Herajy, M., Liu, F., Rohr, C., Schwarick, M.: Snoopy – a unifying Petri net tool. In: Haddad, S., Pomello, L. (eds.) PETRI NETS 2012. LNCS, vol. 7347, pp. 398–407. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-31131-4_22
12. Heiner, M., Schwarick, M., Wegener, J.-T.: Charlie – an extensible Petri net analysis tool. In: Devillers, R., Valmari, A. (eds.) PETRI NETS 2015. LNCS, vol. 9115, pp. 200–211. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19488-2_10
13. Jensen, K., Kristensen, L.M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer, Heidelberg (2009). https://doi.org/10.1007/b95112
14. Jensen, K., Kristensen, L.M., Wells, L.: Coloured Petri nets and CPN tools for modelling and validation of concurrent systems. Softw. Tools Technol. Trans. J. **9**(3), 213–254 (2007)
15. Mei, C., Zhang, X., Zhao, W., Periyasamy, K., Headington, M.: A tool for teaching Petri nets. J. Comput. Sci. Coll. **26**(5), 181–188 (2011)
16. Schwarick, M., Heiner, M., Rohr, C.: MARCIE - model checking and reachability analysis done efficiently. In: International Conference on Quantitative Evaluation of Systems, pp. 91–100 (2011)
17. Thong, W.J., Ameedeen, M.A.: A survey of Petri net tools. In: Sulaiman, H.A., Othman, M.A., Othman, M.F.I., Rahim, Y.A., Pee, N.C. (eds.) Advanced Computer and Communication Engineering Technology. LNEE, vol. 315, pp. 537–551. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-07674-4_51
18. University of Torino: the GreatSPN tool homepage. http://www.di.unito.it/~greatspn/index.html
19. Vörös, A., Darvas, D., Molnár, V., Klenik, A., Hajdu, Á., Jámbor, A., Bartha, T., Majzik, I.: PetriDotNet 1.5: extensible Petri net editor and analyser for education and research. In: Kordon, F., Moldt, D. (eds.) PETRI NETS 2016. LNCS, vol. 9698, pp. 123–132. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39086-4_9
20. Zimmermann, A.: Modelling and performance evaluation with TimeNET 4.4. In: Bertrand, N., Bortolussi, L. (eds.) QEST 2017. LNCS, vol. 10503, pp. 300–303. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66335-7_19