Visualizing Regions with a new Split-Screen View for the Online Tool *travis*

Benjamin Meis[1] Robin Bergenthum[2]

Abstract: The online tool *travis* can synthesize a k-bounded Petri net model from a reachability graph and unfold a k-bounded Petri net to its reachability graph. Synthesis is built on the theory of regions, but where *travis* and other existing tools only show the synthesized model, we want to present the synthesized model as well as the calculated regions after a user has performed the synthesis procedure. In this paper, we present a new split-screen module of *travis* which is able to visualize regions, calculated during the synthesis procedure, as well as markings of places related to single states of the reachability graph, calculated during the unfolding procedure. Both sets, i.e. regions and markings, help to really understand the relations between the behavioral and the synthesized models at hand, as well as the underlying fundamental concepts of state-based synthesis and reachability analysis. With these new features, *travis* is tailored to be used in a teaching environment to help students understand the concepts and notions of state-based synthesis.

Keywords: *travis*; synthesis; unfolding; Petri net; reachability graph; transition system; teaching; final state; local state; distributed state

1 Introduction

Complex systems, such as business processes, software systems, or protocols are often modeled by means of Petri nets [Aa13, Aa98, De01, De98, Pe81, Wo13]. Petri nets have a formal semantics and an intuitive graphical representation. However, constructing a Petri net model from a real world process is a costly and error-prone task [Aa13, Aa98, Ma07]. To support the modeling process, there is a big variety of different academic, as well as commercial tools to support the construction of valid process models. Most tools in that context feature some sort of synthesis or mining algorithm to automatically generate a process model from a behavioral specification. The most prominent tools in the area of synthesis are GENET [Ca09], Petrify [Co97], VipTool [Be08], and APT [Be15]. ProM [Aa09] is the most prominent framework in the area of process mining and supports a wide variety of process discovery techniques. The three most prominent commercial process

[1] FernUniversität in Hagen, Software Engineering and Theory of Programming, Universitätsstraße 1, 58097 Hagen, Germany benjamin.meis@fernuni-hagen.de
[2] FernUniversität in Hagen, Faculty of Mathematics and Computer Science, Universitätsstraße 1, 58097 Hagen, Germany robin.bergenthum@fernuni-hagen.de

mining tools are Celonis[3], Minit[4], and Disco [WGR12]. These tools have in common that they leave the relation between the resulting model and its underlying specification (e.g. an event log or a transition system) implicit, i.e. they only show the result of a synthesis procedure or a mining algorithm. In the present paper, we introduce a new split-screen module of *travis* to make the relation of two models explicit (e.g. a synthesized Petri net and its underlying behavioral specification or a Petri net and its calculated reachability graph), by also showing the relation between the input and the output by means of regions.

The focus of *travis* [Me17] is, on the one hand, to support the synthesis of systems with final states and, on the other hand, to introduce an easy to access online synthesis tool for the purpose of teaching theory related to modern synthesis techniques. Therefore, *travis* can synthesize a Petri net from a reachability graph and construct a reachability graph of a Petri net by an unfolding procedure. In all of its algorithms *travis* is able to handle and respect final states. *travis* was originally introduced in [Me17]. The goal of the new split-screen module is to apply *travis* in a teaching scenario where a split-screen view supports a learner during the modeling process by visualizing underlying concepts and intermediate results of the executed algorithms.

travis is developed in Java, using the Google Web Toolkit (GWT). GWT is an open source development kit for browser-based applications. The GWT compiler can generate HTML, CSS and JavaScript from Java code. Thus, *travis* is pure HTML, CSS and JavaScript. The *travis* homepage[5] provides a short introduction and the example files used in this paper.

In the current version, *travis* offers two different kinds of editors: a transition system editor including an event log loader, and a Petri net editor. A model browser of *travis* enables the user to switch between multiple models or to delete models. Figure 1 shows an overview of the main features of *travis* where each editor is depicted by a framed box.

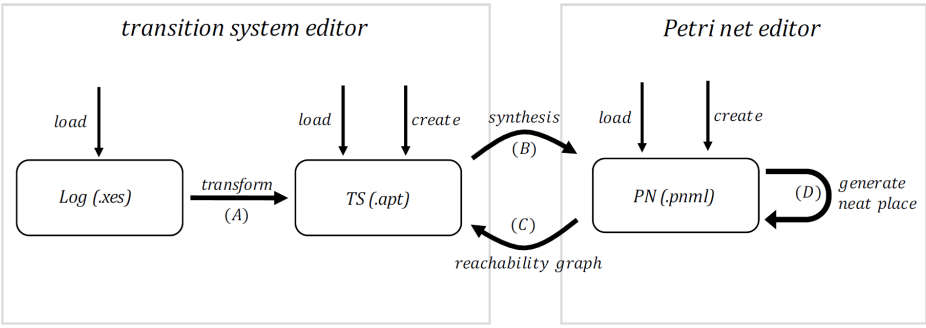


Fig. 1: Overview of the main features of *travis*

[3] Celonis: https://www.celonis.com/
[4] Minit: https://www.minit.io/
[5] Travis Homepage: http://www.fernuni-hagen.de/sttp/forschung/travis.shtml

The transition system editor of *travis* allows to load, create, and edit transition systems with or without final states. Within the editor, the user can toggle any set of states to be final. Of course, *travis* supports all standard editing features for states, labels, and transitions, and includes graph drawing algorithms like a simple spring layout and a tree layout. *travis* has a simple interface to import files. The event log loader of *travis* loads log-files in the standard XES file format [Ve11]. The user can drag-and-drop a stored file into the event log loader. To handle transition systems with final states, *travis* adapts the file format of the APT framework [Be15].

The Petri net editor of *travis* allows to load, create, edit, and simulate Petri nets with or without final markings. Again, *travis* supports all standard editing features and includes graph drawing algorithms. *travis* provides an options window where final markings can be created, edited, deleted, and highlighted in the Petri net at hand. *travis* uses the PNML file format [Ki06] to handle Petri nets and their sets of final markings. Using this format, *travis* is able to share Petri net files with analysis tools like VipTool, Charlie [He15] and WoPeD [Ec08].

In addition to the two main editors, Figure 1 depicts the core features and algorithms implemented in *travis* as bold arcs: (A) transformation of an event log into a transition system with final states, (B) synthesis of a k-bounded Petri net (see [Ca10]) extended by final states (see [Me16]) from a transition system with final states, (C) construction of the reachability graph – in terms of a transition system – with final states of a k-bounded Petri net with final states, and (D) adding a neat place (see [Me16]) to a k-bounded Petri net.

In the current version of *travis*, execution of either the synthesis (B) or the unfolding procedure (C) forces *travis* to switch to the editor related to the produced result i.e. either the transition system editor or the Petri net editor. Thus, the user loses information about the related construction process. For example, during the execution of the synthesis method, *travis* constructs the set of so-called minimal regions. This set characterizes the exact relation between the input (i.e. a transition system) and the calculated Petri net. Until now, *travis* was not able to visualize this set of regions although this is most valuable when trying to understand or analyze the processed synthesis procedure. Such a visualization is even more important when applying *travis* in a teaching scenario. The same holds for the unfolding procedure. Although, during the calculation of the reachability graph the set of all reachable states is computed, this information is lost when closing the Petri net editor and switching to the transition system editor.

In this paper, we present an obvious but elegant solution to the problem at hand. The actual version of *travis* features a new split-screen module which stores the set of minimal regions, as well as the set of reachable markings during the synthesis and unfolding procedures. *travis* is now able to visualize these sets using the new split-screen module by projecting regions to places of the Petri net and projecting markings to states of the transition system respectively the reachability graph.

In the next section we will recapitulate the concept of regions, transition systems, and reachability graphs, before we introduce the new split-screen module in section three.

2 Preliminaries

In this chapter, we will briefly recall the concepts of k-bounded Petri nets, their reachability graphs in terms of transition systems, and region-based synthesis. We refer the reader to [Ba15] for a detailed introduction.

Definition 2.1 (k-bounded Place/Transition Net) *A marked net is a tuple $N = (P, T, W, m_0)$, where P is a finite set of places, T is a finite set of transitions satisfying $P \cap T = \emptyset$, $W : (P \times T) \cup (T \times P) \to \mathbb{N}$ is a function defining the flow relation, and $m_0 : P \to \mathbb{N}$ is an initial marking.*

A transition $t \in T$ is enabled at a marking $m : P \to \mathbb{N}$, if for all $p \in P$: $m(p) \geq W(p,t)$ holds. If t is enabled at m, t can fire. Firing t transforms m to a marking m'. For every $p \in P$, m' is defined by the equation $m'(p) = m(p) + W(t,p) - W(p,t)$. We write $m \xrightarrow{t} m'$. We call $RS(N) = \{m \mid \exists t_1 \dots t_n \in T^*, m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} \dots \xrightarrow{t_3} m_{n-1} \xrightarrow{t_n} m\}$ the set of reachable markings. A marked net is k-bounded if $\forall m \in RS(N) \, \forall p \in P : m(p) \leq k$ holds.

A reachability graph is a transition system representing the behavior of a k-bounded place/transition net.

Definition 2.2 (Reachability Graph) *A transition system is a tuple $TS = (S, T, \Theta, s_0)$, where S is a finite set of states, T is a finite set of events, $\Theta \subseteq (S \times T \times S)$ is the set of labeled state transitions, and $s_0 \in S$ is the initial state of TS. Let $N = (P, T, W, m_0)$ be a marked place/transition net. The reachability graph of N is the transition system $RG(N) = (RS(N), T, \Delta, m_0)$, defined by $(m, t, m') \in \Delta$ if and only if $m \in RS(N)$ and $m \xrightarrow{t} m'$ holds.*

Assuming a transition system modeling the behavior of a system, the synthesis problem aims at generating a Petri net so that its reachability graph is isomorphic to the specified transition system, i.e. both graphs with labeled arcs are identical up to the names of states.

Definition 2.3 (Synthesis Problem) *Let TS be a transition system. The synthesis problem is to construct a k-bounded Petri net N such that its reachability graph is isomorphic to TS.*

The synthesis problem is tackled using the theory of regions. We briefly recapitulate the theory of regions for the synthesis of k-bounded nets, but refer the reader to [Co98, Ca08b, Ca08a, Ca10] for a more detailed introduction.

Definition 2.4 (Region) *Let $TS = (S, T, \Theta, s_0)$ be a transition system and r be a multiset of S. The gradient of a transition $(s, t, s') \in \Theta$ is defined as $\Delta_r(s, t, s') = r(s') - r(s)$. An event t has a constant gradient in r if $\forall (s, t, s'), (s'', t, s''') \in \Theta : r(s') - r(s) = r(s''') - r(s'')$ holds. r is a region of TS if all events $t \in T$ have a constant gradient in r.*

We synthesize k-bounded place/transition nets. Thus, we only consider k-bounded regions. The power of a multiset r is defined by $r^\diamond = \max_{s \in S} r(s)$. A region is k-bounded if the power of the related multiset is less or equal to k. Every minimal k-bounded region of a transition system defines a place in the resulting Petri net solving the synthesis problem.

3 Visualizing Regions

Like already stated in the introduction, *travis* has two built-in editors: a Petri net editor and a transition system editor. *travis* can synthesize a Petri net from a transition system (see (B) of Figure 1) and *travis* can calculate the reachability graph of a Petri net (see (C) of Figure 1) to kind of translate the reachability graph to a Petri net and vice versa. Until now, the user had access to only one of these editors at a time, i.e. *travis* either shows a single transition system or a single Petri net. Thus, it is not possible to see or visualize the direct relation between both models. In this chapter, we present a new split-screen module to show the Petri net and the related transition system.

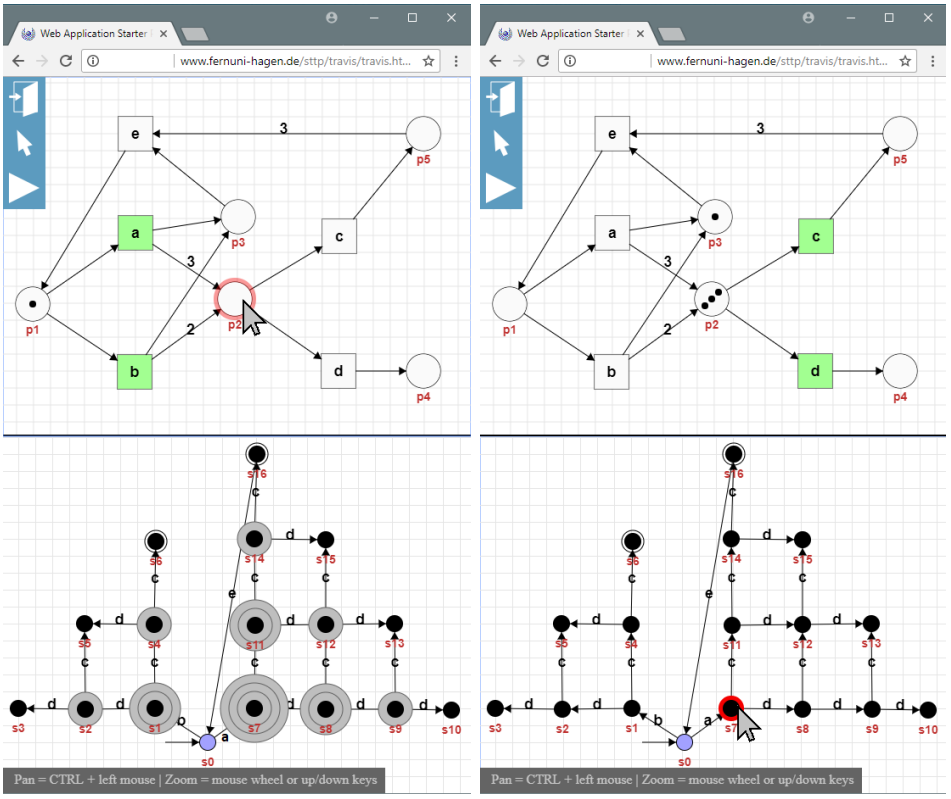
During the synthesis of a Petri net, *travis* calculates the set of regions of the transition system at hand. This set is the link between the behavioral model, i.e. the transition system, and the Petri net model. Every region is a multiset of states with constant gradients for equally labeled transitions. A region and the gradients directly define a visible one-place Petri net, which is a possible and valid sub-net of a solution of the synthesis problem. More precisely, if we merge the set of sub-nets related to the set of minimal regions, we get the final result of the synthesis procedure for a k-bounded net. Looking at this from the other direction, for every place of the synthesized Petri net there is exactly one minimal region responsible for the existence of this place. Thus, really understanding the set of minimal regions of a transition system helps to understand and grasp the synthesis result.

When calculating the reachability graph of a k-bounded Petri net, *travis* performs a breadth-first search on the set of consecutive enabled transitions of the Petri net and keeps track of the produced markings. Of course, every state of the reachability graph directly relates to a marking of the Petri net. But again, if we fix a place of the Petri net and project all calculated markings of all states to this place, we get a region of the reachability graph. So again, understanding this set of regions, as well as the set of produced markings of the reachability graph helps to clearly grasp the relation between the Petri net and its reachability graph.

Altogether, the set of regions and the set of reachable markings are two faces of the same coin. This set clearly pinpoints the relation between the transition system and the Petri net. We can calculate the set of regions by either performing the synthesis algorithm or

performing the unfolding algorithm of *travis*. We are able to visualize the set of regions by selecting one place and depicting the related region in the transition system and we are able to visualize the set of reachable markings by selecting a state and depicting the related marking in the Petri net.

To select places and states to visualize markings and regions, the new version of *travis* provides a split-screen module. The user simply clicks on the related button on the main toolbar of one of the editors to open this view. In this view, as depicted in Figure 2a and 2b, the Petri net editor is at the top and the transition system editor is at the bottom of the screen. At the upper left corner of each browser window, we see a small toolbar where the user can exit the split-screen view, can activate the default cursor to select and move elements, or can activate the token game mode to fire transitions of the Petri net.



(a) Place selected.

(b) State selected.

Fig. 2: The split-screen module of *travis*

As an example, we synthesize a 3-bounded Petri net from a transition system. Figure 2a shows both models. The transition system consists of 17 states and includes five different events, a, b, c, d, and e. When the user clicks on place p_2 of the synthesized Petri net, the

transition system editor shows the directly related multiset. The depicted multiset is a region, because all events have a constant gradient: the transition labeled by a has a gradient of $+3$, the transition labeled by b has a gradient of $+2$, transitions labeled by c have a gradient of -1, transitions labeled by d have a gradient of -1, and the transition labeled by e has a gradient of 0. The arc-weights of p_2 directly relate to the gradient of each event.

Whenever the user clicks on a place in the Petri net, the related minimal region of this place is depicted in the transition system. A region is a multiset of states of the transition system. Every state, which multiplicity is greater than zero is surrounded by a number of gray circles whereby the number of circles is equal to the multiplicity. For example, in Figure 2a, the state s_7 in the region related to place p_2 has three gray circles, i.e. in state s_7, there are three tokens on place p_2. The other way around, whenever the user clicks on a state of the transition system, *travis* shows the marking related to this state in the Petri net. Regarding regions, when a state of the transition system is selected, markings of the places in the Petri net are shown, whose regions have a multiplicity greater than zero in the selected state. For example, in Figure 2b, state s_7 is selected and only the regions of places p_2 and p_3 include state s_7 with a multiplicity greater than zero. Thus, there are tokens on places p_2 and p_3.

The actual version of *travis* supports a state-based synthesis, calculating regions of a transition system. Thus, the split-screen module is limited to the visualization of regions in transition systems. Of course, it is possible to transfer the concept of the split-screen module to other notions of regions and synthesis algorithms provided that such an algorithm produces intermediate results – like state-based regions – which can be depicted within the model editor.

As one possible use case, we used *travis* with its split-screen view at our chair. We conducted a seminar on the topic *Modeling Languages in Software Engineering* with 13 of our students where one of the students talks was about Petri net synthesis. After his talk, we attached a short practical phase where all students had the chance to use *travis* and the split-screen view. With *travis* they modeled an exemplary transition system and synthesized a Petri net model.

To gain more insight into the synthesis algorithm presented by the student and to better understand the relations between places and regions, the students used the split-screen view of *travis*, where they could click on a place to display the related region in the transition system. We got the feedback, that the split-screen view is quite intuitive to use and that the visualization of both models at the same time helps to analyze the relation between the input and the output model, especially because when switching the focus between the two models, there is no visual interruption, like there would be when only one editor could be shown at a time.

The new split-screen module of *travis* is suitable for many use cases. The use case mentioned above refers to a small group of students where each student starts a synthesis procedure for a given transition system and has to analyze the relation between places and regions

to gain more insights about how a Petri net is being synthesized. Another use case is the solving of a modeling task. Imagine a student who has to model a system by means of a Petri net. System modeling can be a faulty task. With *travis* and its split-screen view the student can calculate a reachability graph of the modeled Petri net and analyze the existing markings and regions by clicking on states and places. First, there is a direct visualization of local states in the transition system and distributed states in the Petri net, and second the visualization of regions helps the student to recognize modeling mistakes. Subsequently, the student can use an iterative approach, i.e. add, edit or delete places, transitions and arcs, recalculate the reachability graph and use the split-screen view to analyze and compare both models, until the Petri net model has the intended behavior.

As mentioned in the introduction, *travis* is an online tool which is executed in the browser and there is no need for any installation. Therefore, students and teachers within a learning environment have easy access to *travis* as a tool to support teaching scenarios, concerning the modeling of systems, especially with Petri nets.

4 Conclusion and Future Work

This paper presents the new split-screen module for the online tool *travis*. *travis* can be executed simply in a browser and is free from any registration. Within *travis*, one can synthesize a Petri net from a transition system or unfold a Petri net by calculating its reachability graph. In an educational context, the new split-screen module of *travis* can further support teaching scenarios by visualizing the relations between Petri nets and transition systems and reachability graphs respectively in terms of regions. For future work, we will extend the split-screen module such that the user gains more detailed insights into the synthesis and unfolding procedures and gets more information about intermediate results.

References

[Aa98] Aalst, W. M. P. van der: The Application of Petri Nets to Workflow Management. Journal of Circuits, Systems and Computers, 08(01):21–66, 1998.

[Aa09] Aalst, W. M. P. van der; Dongen, B. F. van; W. Günther, C.; Rozinat, A.; Verbeek, E.; Weijters, A.: , ProM: The Process Mining Toolkit, 2009.

[Aa13] Aalst, W. M. P. van der; Dongen, B. F. van: Discovering Petri Nets from Event Logs. In: ToPNoC VII, LNCS 7480, pp. 372–422. Springer, 2013.

[Ba15] Badouel, E.; Bernardinello, L.; Darondeau P.: Petri Net Synthesis. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2015.

[Be08] Bergenthum, R.; Desel, J.; Lorenz R.; Mauser S.: Synthesis of Petri Nets from Scenarios with VipTool. In: Applications and Theory of Petri nets, LNCS 5062, pp. 388–398. Springer, 2008.

[Be15] Best, E.; Schlachter, U.: Analysis of Petri Nets and Transition Systems. Electronic Proceedings in Theoretical Computer Science, 189:53–67, 2015.

[Ca08a] Carmona, J.; Cortadella, J.; Kishinevsky M.: A Region-Based Algorithm for Discovering Petri Nets from Event Logs. In: BPM 2008, LNCS 5240, pp. 358–373. Springer, 2008.

[Ca08b] Carmona, J.; Cortadella, J.; Kishinevsky M.; Kondratyev A.; Lavagno L.; Yakovlev A.: A Symbolic Algorithm for the Synthesis of Bounded Petri Nets. In: Applications and Theory of Petri nets, LNCS 5062, pp. 92–111. Springer, 2008.

[Ca09] Carmona, J.; Cortadella, J.; Kishinevsky M.: Genet: A Tool for the Synthesis and Mining of Petri Nets. In: ACSD '09. IEEE, pp. 181–185, 2009.

[Ca10] Carmona, J.; Cortadella, J.; Kishinevsky M.: New Region-Based Algorithms for Deriving Bounded Petri Nets. IEEE Transactions on Computers, 59(3):371–384, 2010.

[Co97] Cortadella, J.; Kishinevsky, M.; Kondratyev A.; Lavagno L.; Yakovlev A.: Petrify: A Tool for Manipulating Concurrent Specifications and Synthesis of Asynchronous Controllers. IEICE Transactions on Information and Systems, E80-D(3):315–325, 1997.

[Co98] Cortadella, J.; Kishinevsky, M.; Lavagno, L.; Yakovlev, A.: Deriving Petri Nets from Finite Transition Systems. IEEE Transactions on Computers, 47(8):859–882, 1998.

[De98] Desel, J.; Reisig, W.: Place/Transition Petri Nets. In: LNCS 1491, pp. 122–173. Springer, 1998.

[De01] Desel, J.; Juhás, G.: "What Is a Petri Net?" Informal Answers for the Informed Reader. In: Unifying Petri Nets, LNCS 2128, pp. 1–25. Springer, 2001.

[Ec08] Eckleder, A.; Freytag, T.: WoPeD - A Tool for Teaching, Analyzing and Visualizing Workflow Nets. 75:3–8, 2008.

[He15] Heiner, M.; Schwarick, M.; Wegener J.-T.: Charlie – An Extensible Petri Net Analysis Tool. In: LNCS 9115, pp. 200–211. Springer, 2015.

[Ki06] Kindler, E.: PNML: Concept, Status and Future Directions. 9:35–55, 2006.

[Ma07] Mayr, H.; Kop, C.; Esberger D.: Business Process Modeling and Requirements Modeling. In: First International Conference on the Digital Society (ICDS'07). IEEE, p. 8, 2007.

[Me16] Meis, B.; Bergenthum, R.; Desel J.: Synthesis of Elemenary Net Systems with Final Configurations, ATAED 2016. In: CEUR 1592, pp. 47–57. 2016.

[Me17] Meis, B.; Bergenthum, R.; Desel J.: travis - An Online Tool for the Synthesis and Analysis of Petri Nets with Final States. LNCS 10258, pp. 101–111. Springer, 2017.

[Pe81] Peterson, J. L.: Petri Net Theory and the Modeling of Systems. Prentice-Hall, Englewood Cliffs N.J., 1981.

[Ve11] Verbeek, H. M. W.; Buijs, J. C. A. M.; Dongen B. F. van; Aalst W. M. P. van der: XES, XESame, and ProM 6. In: LNBIP 72, pp. 60–75. 2011.

[WGR12] W. Günther, C.; Rozinat, A.: Disco: Discover Your Processes on Business Process Management (BPM 2012), Tallinn, Estonia, September 4, 2012. In: Proceedings of the Demonstration Track of BPM 2012. CEUR 940, pp. 40–44, 2012.

[Wo13] Wolfgang, R.: Understanding Petri Nets - Modeling Techniques, Analysis Methods, Case Studies. Springer, 2013.