Everything You Always Wanted To Know About Petri Nets, But Were Afraid To Ask

Wil M.P. van der Aalst[1,2] [0000−0002−0955−6940]

[1] Process and Data Science (PADS), RWTH Aachen University, Aachen, Germany
[2] Fraunhofer Institute for Applied Information Technology, Sankt Augustin, Germany
wvdaalst@pads.rwth-aachen.de

Abstract. Business Process Management (BPM), Process Mining (PM), Workflow Management (WFM), and other approaches aimed at improving processes depend on process models. Business Process Model and Notation (BPMN), Event-driven Process Chains (EPCs), and UML activity diagrams all build on Petri nets and have semantics involving 'playing the token game'. In addition, process analysis approaches ranging from verification and simulation to process discovery and compliance checking often depend on Petri net theory. For the casual user, there is no need to understand the underlying foundations. However, BPM/PM/WFM researchers and 'process experts' working in industry need to understand these foundational results. Unfortunately, the results of 50 years of Petri net research are not easy to digest. This tutorial paper provides, therefore, an entry point into the wonderful world of Petri nets.

Keywords: Petri Nets · Business Process Management · Process Mining.

1 Petri Nets and Business Process Management

Since their inception in 1962, Petri nets have been used in a wide variety of application domains. A more recent development is the foundational role of Petri nets in Business Process Management (BPM) [8] and related fields such as Process Mining (PM) [2] and Workflow Management (WFM) [3]. Many WFM systems are based on Petri nets. In fact, the first prototypes developed in the late 1970-ties (e.g., Officetalk and SCOOP) already used Petri nets [1]. In today's BPM/WFM systems, this is less visible. However, popular modeling languages such as Business Process Model and Notation (BPMN), Event-driven Process Chains (EPCs), and UML activity diagrams all borrow ideas from Petri nets (e.g., the 'token game' to describe semantics and to implement BPM/WFM engines and simulation tools) [5].

Petri nets also play a major role in the analysis of processes and event data. Many simulation tools are based on Petri nets [5]. Petri nets are also used for the verification of processes in WFM/BPM systems, e.g., to check soundness [4]. However, this possibility is not used much in practice. Conversely, Process Mining (PM) is much more widely used than simulation and verification. Petri nets are the most widely used representation in PM [2]. There are dozens of techniques that can discover a Petri net from event data. Moreover, almost all conformance checking techniques use Petri nets internally.

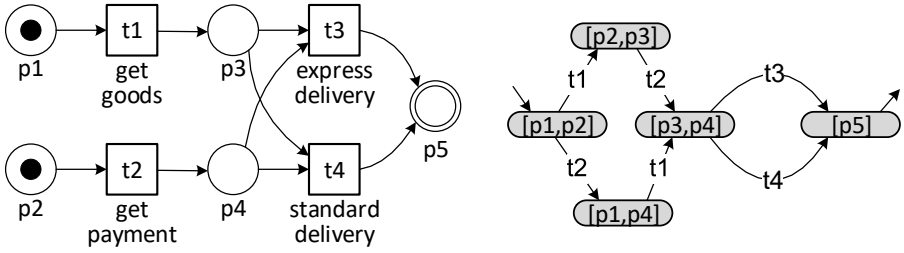

Fig. 1. An accepting Petri net N_1 (left) with transitions $T_1 = \{t1, t2, t3, t4\}$, places $P_1 = \{p1, p2, p3, p4, p5\}$, initial marking $[p1, p2]$, and final marking $[p5]$. N_1 allows for traces $traces(N_1) = \{\langle t1, t2, t3\rangle, \langle t2, t1, t3\rangle, \langle t1, t2, t4\rangle, \langle t2, t1, t4\rangle\}$. The reachability graph (right) shows the reachable markings $states(N_1) = \{[p1, p2], [p1, p4], [p2, p3], [p3, p4], [p5]\}$.

This short paper is based on a tutorial with the same name presented at the 17th International Conference on Business Process Management (BPM 2019) in Vienna in September 2019. Here, we can only show a few of the 'gems in Petri nets' relevant for BPM, PM, and WFM.

2 Accepting Petri Nets

Figures 1 and 2 show two so-called *accepting Petri nets*. These Petri nets have an initial state and a final state. States in Petri nets are called *markings* that mark certain *places* (represented by circles) with *tokens* (represented by black dots). The accepting Petri net N_1 in Figure 1 has five places. In the initial marking, $[p1, p2]$ two places are marked. Since a place may have multiple tokens, markings are represented by multisets. *Transitions* (represented by squares) are the active components able to move the Petri nets from one marking to another marking. N_1 has four transitions. A transition is called *enabled* if each of the input places has a token. An enabled transition may *fire* (i.e., *occur*) thereby consuming a token from each input place and producing a token for each output places. In the marking showing in Figure 1, both $t1$ and $t2$ are enabled. Firing $t1$ removes a token from $p1$ and adds a token to $p3$. Firing $t2$ removes a token from $p2$ and adds a token to $p4$. In the resulting marking $[p3, p4]$ both $t3$ and $t4$ are enabled. Note that both transitions require both input places to be marked. However, only one of them can fire. Firing $t3$ (or $t4$) removes a token from both $p3$ and $p4$ and adds one token to $p5$. Transitions may be labeled, e.g., transitions $t1$, $t2$, $t3$, and $t4$ represent the activities "get goods", "get payment", "express delivery", and "normal delivery" respectively. For simplicity, we ignore the transition labels and only use the short transition names.

The *behavior* of an accepting Petri net is described by all *traces* starting in the initial marking and ending in the final marking. $\langle t1, t2, t3\rangle$ is one of the four traces of accepting Petri net N_1. Figure 1 also shows the reachability graph of N_1. The reachability graph shows all reachable markings and their connections. N_1 has five reachable states.

Accepting Petri net N_2 depicted in Figure 2 also has five reachable states, but allows for infinitely many traces (due to the loop involving $t1$ and $t2$).

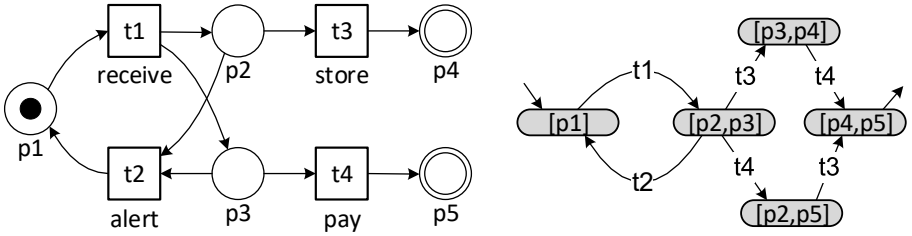

Fig. 2. An accepting Petri net N_2 (left) with initial marking $[p1]$ and final marking $[p4, p5]$. N_2 allows for infinitely many traces $traces(N_2) = \{\langle t1, t3, t4\rangle, \langle t1, t4, t3\rangle, \langle t1, t2, t1, t3, t4\rangle, \langle t1, t2, t1, t4, t3\rangle, \langle t1, t2, t1, t2, t1, t3, t4\rangle, \ldots\}$. The reachability graph (right) shows the reachable markings $states(N_2) = \{[p1], [p2, p3], [p2, p5], [p3, p4], [p4, p5]\}$.

3 Petri Nets Are More Declarative Than You Think

Petri nets are typically considered 'procedural' (like an imperative program) and not 'declarative'. However, an accepting Petri net *without* places allows for any trace involving the transitions in the net. Each place corresponds to a *constraint*. Consider for example the accepting Petri net N_3 in Figure 3. Place $p1$ models the constraint that $t1$ should occur precisely once. Place $p2$ models the constraint that $t2$ can only occur after $t1$ or $t3$. Each occurrence of $t2$ requires an earlier occurrence of $t1$ or $t3$ and, at the end, the number of occurrences of $t2$ is one less than the sum of $t1$ and $t3$. Place $p3$ models the constraint that each $t4$ occurrence should be preceded by a $t2$ occurrence and, at the end, the number of occurrences of both $t2$ and $t4$ need to be the same. Note that transition $t5$ is not constrained by one of the places and can occur at any point in time and an arbitrary number of times. Removing a place can only enable more traces, thus illustrating the declarative nature of Petri nets (anything is possible unless specified otherwise).

4 Structure Theory and the Marking Equation

Structure theory focuses on behavioral properties that can be derived from the structural properties of a Petri net [6, 7, 9]. It is impossible to go into details. Therefore, we restrict

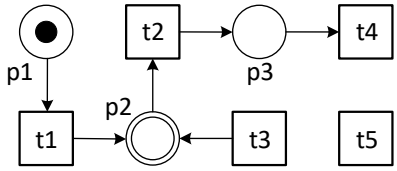

Fig. 3. An accepting Petri net N_3 with initial marking $[p1]$ and final marking $[p2]$ showing the declarative nature of Petri nets.

ourselves to the *marking equation* which nicely shows how linear algebra can be used to exploit the structure of a Petri net. To start, we represent the first two Petri nets as a matrix with a row for each place and a column for each transition. The so-called incidence matrix shows the 'net effect' of firing a transition (column) on each place (row).

$$\mathbf{N_1} = \begin{array}{c} \\ p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{array} \begin{array}{c} \begin{array}{cccc} t1 & t2 & t3 & t4 \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & -1 & -1 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \end{array} \qquad \mathbf{N_2} = \begin{array}{c} \\ p1 \\ p2 \\ p3 \\ p4 \\ p5 \end{array} \begin{array}{c} \begin{array}{cccc} t1 & t2 & t3 & t4 \end{array} \\ \begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & -1 & 0 \\ 1 & -1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array}$$

The incidence matrix imposes an order on places (rows) and transitions (columns). For N_1 and N_2, the order is $p1, p2, p3, p4, p5$ and $t1, t2, t3, t4$. $\mathbf{t} = (1, 1, 1, 0)^T$ is an example of a transition column vector assigning value 1 to $t1$, $t2$, and $t3$ and value 0 to $t4$. $\mathbf{p} = (1, 1, 0, 0, 0)^T$ is an example of a place column vector assigning value 1 to $p1$ and $p2$, and value 0 to $p3$, $p4$, and $p5$. Assume that $\mathbf{p}'$ and $\mathbf{p}''$ are two place column vectors representing the initial marking $\mathbf{p}'$ and a target marking $\mathbf{p}''$. If $\mathbf{p}''$ is reachable from $\mathbf{p}'$ in some Petri net having incidence matrix $\mathbf{N}$, then the so-called marking equation

$$\mathbf{p}' + \mathbf{N} \cdot \mathbf{t} = \mathbf{p}''$$

has a solution for some transition column vector $\mathbf{t}$ with non-negative values.

Consider N_1 in Figure 1. We are interested in the different ways to get from the initial marking $[p1, p2]$ to the final marking $[p5]$. Hence, $\mathbf{p}' = (1, 1, 0, 0, 0)^T$ and $\mathbf{p}'' = (0, 0, 0, 0, 1)^T$, resulting in the following marking equation:

$$\mathbf{p}' + \mathbf{N} \cdot \mathbf{t} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & -1 & -1 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} t1 \\ t2 \\ t3 \\ t4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \mathbf{p}''$$

Hence, we can infer from the marking equation that $t1 = 1$, $t2 = 1$, and $t3 + t4 = 1$. Since N_1 allows for trace $\langle t1, t2, t3 \rangle$, we know that $t1 = t2 = t3 = 1$ and $t4 = 0$ should be a solution. Suppose we would like to know whether N_1 allows for trace $\langle t1, t3, t4 \rangle$. Since $t1 = t3 = t4 = 1$ and $t2 = 0$ is not solution of the marking equation, we know that $\langle t1, t3, t4 \rangle$ is impossible *without* replaying the trace. For such a small example, this may seem insignificant. However, the marking equation provides a powerful 'algebraic overapproximation' of all possible traces. Note that the marking equation provides a necessary but not a sufficient condition. The algebraic overapproximation can be used to quickly prune search spaces in verification and conformance checking. For example, the marking equation can be used to guide the search for so-called optimal alignments in conformance checking [2].

The marking equation is related to place and transitions invariants. Any solution of the equation $\mathbf{p} \cdot \mathbf{N} = \mathbf{0}$ is a *place invariant*. For net N_1:

$$\mathbf{p} \cdot \mathbf{N} = (p1, p2, p3, p4, p5) \cdot \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & -1 & -1 \\ 0 & 1 & -1 & -1 \\ 0 & 0 & 1 & 1 \end{pmatrix} = (0, 0, 0, 0) = \mathbf{0}$$

For example, $\mathbf{p} = (p1, p2, p3, p4, p5) = (1, 0, 1, 0, 1)$ is the place invariant showing that the number of tokens in the places $p1$, $p3$, and $p5$ is constant. The so-called 'weighted token sum' is constant for any initial marking. Given the initial marking $[p1, p2]$, the weighted token sum is 1. If the initial marking is $[p1^2, p3, p4^3, p5]$, the weighted token sum is $(2 \times 1) + (0 \times 0) + (1 \times 1) + (3 \times 0) + (1 \times 1) = 4$ and will not change. $\mathbf{p} = (0, 1, 0, 1, 1)$, $\mathbf{p} = (1, 1, 1, 1, 2)$, and $\mathbf{p} = (1, -1, 1, -1, 0)$ are other place invariants since $\mathbf{p} \cdot \mathbf{N_1} = \mathbf{0}$.

Any solution of the equation $\mathbf{N} \cdot \mathbf{t} = \mathbf{0}^T$ is a *transition invariant*. For net N_2:

$$\mathbf{N} \cdot \mathbf{t} = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 1 & -1 & -1 & 0 \\ 1 & -1 & 0 & -1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} t1 \\ t2 \\ t3 \\ t4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \mathbf{0}^T$$

Any non-negative solution points to firing sequences returning to the same state. For example, $\mathbf{t} = (t1, t2, t3, t4)^T = (3, 3, 0, 0)^T$ is the transition invariant showing that if we are able to execute $t1$ and $t2$ three times, we return to the initial state. Again this property is independent of the initial marking. Trace invariants can again be seen as an 'algebraic overapproximation' of all possible traces returning to the same state.

5 A Beautiful Subclass: Free-Choice Petri Nets

The three models shown in figures 1, 2, and 3 are all *free-choice Petri nets*. These nets satisfy the constraint that any two transitions having the same place as input place should have identical sets of input places. Formally, for any two transitions $t1, t2 \in T$ such that $\bullet t1 \cap \bullet t2 \neq \emptyset$: $\bullet t1 = \bullet t2$. In Figure 1, $\bullet t3 \cap \bullet t4 \neq \emptyset$, but $\bullet t3 = \bullet t4 = \{p3, p4\}$. The free-choice requirement implies that choice and synchronization are 'separable', i.e., choices are 'free' and not controlled by places that are not shared by all transitions involved in the choice. Free-choice Petri nets are very relevant for BPM, PM, and WFM, because most modeling languages have constructs (e.g., gateways in BPMN, control nodes in UML activity diagrams, and connectors in EPCs) modeling AND/XOR-splits/joins. As a result, choice (XOR-split) and synchronization (AND-join) are separated.

To exploit the properties of free-choice Petri nets, we often 'short-circuit' the accepting Petri net, i.e., we add a transition consuming tokens from the places in the final marking and producing tokens for the places in the initial marking. This implies that

when reaching the final marking, it is possible to do a 'reset' and start again from the initial state.

We refer to [7] for the many results known for free-choice Petri nets, e.g., Commoner's Theorem, the two Coverability Theorems, the Rank Theorem, the Synthesis Theorem, the Home Marking Theorem, the two Confluence Theorems, the Shortest Sequence Theorem, and the Blocking Marking Theorem.

6 Conclusion

This short paper should be considered as a 'teaser' for researchers and experts working on BPM, PM, and WFM. Although often not directly visible, many techniques and tools depend on Petri nets. See [5–7, 9] to learn more about the Petri net theory. For the use of Petri nets in BPM, PM, and WFM, see [1–4].

Acknowledgments: We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

References

1. W.M.P. van der Aalst. Business Process Management: A Comprehensive Survey. *ISRN Software Engineering*, pages 1–37, 2013. doi:10.1155/2013/507984.
2. W.M.P. van der Aalst. *Process Mining: Data Science in Action*. Springer-Verlag, Berlin, 2016.
3. W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT Press, Cambridge, MA, 2004.
4. W.M.P. van der Aalst, K.M. van Hee, A.H.M. ter Hofstede, N. Sidorova, H.M.W. Verbeek, M. Voorhoeve, and M.T. Wynn. Soundness of Workflow Nets: Classification, Decidability, and Analysis. *Formal Aspects of Computing*, 23(3):333–363, 2011.
5. W.M.P. van der Aalst and C. Stahl. *Modeling Business Processes: A Petri Net Oriented Approach*. MIT Press, Cambridge, MA, 2011.
6. E. Best and H. Wimmel. Structure Theory of Petri Nets. In K. Jensen, W.M.P. van der Aalst, G. Balbo, M. Koutny, and K. Wolf, editors, *Transactions on Petri Nets and Other Models of Concurrency (ToPNoC VII)*, volume 7480 of *Lecture Notes in Computer Science*, pages 162–224. Springer-Verlag, Berlin, 2013.
7. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.
8. M. Dumas, M. La Rosa, J. Mendling, and H. Reijers. *Fundamentals of Business Process Management*. Springer-Verlag, Berlin, 2013.
9. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1998.