Structure Theory in a Dynamic Data-Driven World
Applications in Process Mining and BPM
(extended abstract)

Wil M.P. van der Aalst

Eindhoven University of Technology, PO Box 513, Eindhoven, The Netherlands

Abstract. Until the turn of the century, most Petri nets were made by hand or generated from another model (e.g., though synthesis). Such Petri nets where mostly used to provide a specification or design of a system (as-is or to-be). Analysis of these nets aimed at detecting behavioral anomalies like deadlocks and livelocks (through verification) and understanding performance (through simulation or analytical techniques). *Structure theory* provided unique ways to facilitate such analysis by exploiting the structure of (subclasses of) Petri nets. However, over the last decade one could witness a dramatic change in the way we analyze the behavior of discrete processes and systems. Model-driven approaches are *replaced or complemented by data-driven approaches*. The abundance of event data makes it feasible to study processes and systems directly. *Process mining* techniques allow for the discovery of Petri nets from event data and can be used to check conformance. Through process mining we are able to connect Petri nets to mainstream developments related to Big data, data science, and machine learning. The direct confrontation between modeled and observed behavior is valuable, but also provides many challenges. For example, one needs to deal with huge event logs and processes that change over time and exhibit deviating behavior. *Can structure theory also play a key role in such data-driven analysis?* The answer is affirmative. Elements of structure theory are already widely used in process mining and Business Process Management (BPM). Moreover, further breakthroughs are possible by tailoring structure theory towards more data-driven problems.

1 Introduction

Traditionally, *Petri nets* are made by hand or generated from other models [23, 32, 34, 35]. Petri nets can be used to design or specify discrete dynamic systems. Most Petri nets described in literature where created *manually*. However, program code, lower-level models (e.g., transition systems), and higher-level models (e.g., BPMN or UML models) can be *transformed* into Petri nets. Most of these transformations are quite straightforward, although the devil is often in the details and abstractions are needed. For example, the de facto standard for business process modeling—BPMN (Business Process Model and Notation) [33]—uses token passing. Also UML activity diagrams use token-based semantics and a notation similar to Petri nets. Examples of transformations that are more involved include the Petri net synthesis techniques known under the name of "region theory" [25, 20, 10]. State-based region theory starts from a transition system and aims to produce a Petri net that has the same behavior while capturing concurrency. For example, in [20] it is shown that any transition system can be transformed into a bisimilar Petri net.

Given a Petri net, one can apply verification and performance analysis techniques. *Verification* is concerned with the correctness of a system or process. Verification techniques may be used to find deadlocks, livelocks, and other anomalies. It is also possible to define desirable properties in some temporal logic and then check whether the model has these properties. *Performance analysis* focuses on flow times, waiting times, utilization, and service levels. Typically, three dimensions of performance are identified: time, cost and quality. For each of these performance dimensions different Key Performance Indicators (KPIs) can be defined. Simulation, queueing models, or Markov models can be used to analyze systems with respect to such KPIs.

Mainstream analysis techniques do *not* exploit the structure of the model. For example, verification techniques may try to exhaustively traverse the state space and simulation approaches randomly sample behavior independent of the model's structure. One of the key advantages of using Petri nets is that knowledge about the structure can be exploited during analysis [16]. The *marking equation* can be used to rule out markings that cannot be reachable [37, 19]. *Siphons* and *traps* can be used to reason about deadlocks [21, 22]. Place and transition *invariants* are used to identify properties that are preserved because of the net's structure [27, 32, 31, 18]. *Reduction rules* can be used to make problems smaller while guaranteeing the same outcome [13, 14, 22, 46, 41]. *Free-choice nets* [15, 22, 39, 42], Petri nets without conflicting splits and joins [26], and *marked graphs* [32] are well-known subclasses of Petri nets. These subclasses can be identified based on their structure and often analysis becomes easier, e.g., one can decide whether a free-choice net is live and bounded in polynomial time [22]. Performance analysis may also benefit from structural theory [11, 17], e.g., one can compute performance bounds for marked graphs and free-choice nets.

Petri nets representing workflows or other types of business processes can also benefit from knowledge about the structure of the model. Consider for example the class of *workflow nets* (WF-nets) and the corresponding *soundness* notion [1]. A WF-net is a Petri net with a dedicated source place where the process starts and a dedicated sink place where the process ends. Moreover, all nodes are on a path from source to sink. A WF-net is sound if it is always possible to terminate and there are no dead parts in the model. Soundness can be checked in polynomial time for several subclasses, including free-choice WF-nets [7, 40].

The examples above show that structure theory allows for the identification of Petri nets whose structure strongly influences their behavior. Moreover, structure theory can also be used to compute bounds or shown the (im)possibility of particular behaviors.

Structure theory developed over the last fifty years with a strong focus on model-based analysis [16]. However, *the spectacular growth of data is rapidly changing the way we analyze behavior*. Rather than analyzing modeled behavior, we can now analyze the *actual* behavior of processes and systems!

Data are collected about anything, at any time, and at any place. It has become possible to record, derive, and analyze events at an unprecedented scale. Events may take place inside a machine (e.g., an X-ray machine or baggage handling system), inside an enterprise information system (e.g., an order placed by a customer or the submission of a tax declaration), inside a hospital (e.g., the analysis of a blood sample), inside a social network (e.g., exchanging e-mails or twitter messages), inside a transportation system (e.g., checking in, buying a ticket, or passing through a toll booth), etc. [5]. Events may be “life events”, “machine events”, or “organization events”. The term *Internet of Events* (IoE), coined in [4], includes (1) the Internet of Content (traditional web pages, articles, encyclopedia like Wikipedia, YouTube, e-books, newsfeeds, etc.), (2) the Internet of People (all data related to social interaction, including e-mail, Facebook, Twitter, forums, LinkedIn, etc.), (3) the Internet of Things (physical objects connected to the network), and (4) the Internet of Locations (data that have a geographical or geospatial dimension, e.g., generated by smartphones and cars). The IoE provides a new and extremely valuable source of information for analyzing processes and systems.

The abundance of event data triggers the question: *Do we need structure theory in this dynamic data-driven world?* We believe that, more than ever, there is a need to use and develop structure theory. This extended abstract only provides a few pointers in this direction. However, structure theory is already used in areas such as Business Process Management (BPM) and process mining. Moreover, in the era of Big data, there is a need to analyze processes efficiently. This can only be done by exploiting the structure of process models.

2 Process Mining and Business Process Management

Developments in Business Process Management (BPM) over the last two decades have resulted in a well-established set of principles, methods and tools that combine knowledge from information technology, management sciences and industrial engineering for the purpose of improving business processes [3, 24, 45]. Until recently, mainstream BPM approaches were predominantly model-driven without considering the “evidence” hidden in the data [3]. However, this changed dramatically with the uptake of *process mining*.

Process mining aims to *exploit event data in a meaningful way*, for example, to provide insights, identify bottlenecks, anticipate problems, record policy violations, recommend counter-measures, and streamline processes [5].

The interest in process mining is reflected by the growing number of commercial process mining tools available today. There are over 25 commercial products supporting process mining (Celonis, Disco, Minit, myInvenio, ProcessGold, QPR, etc.). All support process discovery and can be used to improve compliance and performance problems. For example, without any modeling, it is possible to learn process models clearly showing the main bottlenecks and deviating behaviors.

Starting point for any process mining effort is a collection of *events* commonly referred to as an *event log* (although events can also be stored in a database). Each event is characterized by:

– a *case* (also called *process instance*), e.g., an order number, a patient id, or a business trip,
– an *activity*, e.g., “submit form” or “make decision”,
– a *timestamp*, e.g., “2017-06-30T09:56:30+00:00”,

– additional (optional) *attributes* such as the *resource* executing the corresponding event, the *type* of event (e.g., start, complete, schedule, abort), the *location* of the event, or the *costs* of an event.

The lion's share of process mining research focused on the *discovery of process models from event data* [5]. The process model should be able to capture causalities, choices, concurrency, and loops. Process discovery is a notoriously difficult problem because event logs are often far from complete and there are at least four competing quality dimensions: (1) *fitness*, (2) *simplicity*, (3) *precision*, and (4) *generalization*. Most discovery algorithms described in the literature (e.g., the α-algorithm [8], the region-based approaches [12, 38, 44], and the inductive mining approaches [28, 29, 30]) produce formal models having clear semantics. All of these approaches use Petri nets as a representation or the results they return can easily be converted into Petri nets [5].

We strongly believe that the communities working on BPM and process mining can benefit more from structure theory. Moreover, we also believe that process mining provides novel and exciting challenges for people working on structure theory. Given the developments sketched before, it is important to use the abundantly available data. Purely model-driven analysis only makes sense when designing a completely new system of process.

In the remainder, we briefly sketch two examples where structure theory could play a more prominent role. In this extended abstract, we only highlight some of the opportunities and challenges without going into detail.

3 Process Discovery

The goal of process discovery is to learn a process model from event data. Typically, an *event log* $L \in \mathcal{B}(A^*)$ is used as input. L is a non-empty multiset of traces over some activity set A. A *process model* $Mod \subseteq A^*$ defines a set of traces over some activity set A. Different representations can be used to describe Mod. One can use for example a so-called *accepting labeled Petri net* described by the triplet $AN = (N, M_{init}, M_{final})$ where $N = (P, T, F, A, l)$ is a labeled Petri net, $M_{init} \in \mathcal{B}(P)$ is the initial marking, and $M_{final} \in \mathcal{B}(P)$ is the final marking. P is the set of places, T is the set of transitions, and F is the flow relation. Transitions can have a label as defined by labeling function $l \in T \not\rightarrow A$. Transition $t \in T$ has a label $l(t) \in A$ if $t \in dom(l)$. Otherwise, t is silent (i.e., its occurences are not recorded). Any firing sequence leading from M_{init} to M_{final} corresponds to an accepting trace $\sigma \in A^*$.[1] The set of all possible accepting traces defines the behavior of AN: $Mod_{AN} \subseteq A^*$.

A discovery algorithm can be described as a function $disc \in \mathcal{B}(A^*) \rightarrow \mathcal{P}(A^*)$. Note that $\mathcal{P}(A^*)$ denotes the powerset of traces over A, i.e., $disc(L) \subseteq A^*$. Ideally, the discovered model allows for all traces observed, i.e., $\{\sigma \in L\} \subseteq disc(L)$. However, it is easy to define degenerate solutions like $disc_{overfit}(L) = \{\sigma \in L\}$ and $disc_{underfit}(L) = A^*$ that do not provide any insights. $disc_{overfit}$ basically enumerates

[1] Note that one needs to apply the labeling function to each transition occurrence in the firing sequence. Transitions without a visible label are skipped.

the event log and is likely to severely overfit the data. $disc_{underfit}$ allows for any behavior involving activities A. Discovery function $disc$ should generalize over the input data that consists of examples only.[2] At the same time, we may want to abstract from infrequent behavior.

The *representation* of the discovered process model plays an important role in balancing between overfitting and underfitting. The so-called *representational bias* defines the class of model that can be returned by the discovery algorithm. Accepting labeled Petri nets form such a class. One can impose additional restrictions on the class of accepting labeled Petri nets. For example, one can limit the representational bias to free-choice nets, WF-nets, or sound WF-nets. Such constraints may aid the understandability of the resulting process models, e.g., free-choice nets separate choice and synchronization and WF-nets have a clear begin and end.

Discovery algorithms producing Petri nets may return a model that is not a WF-net or that is not sound. This makes the interpretation of the discovered process model very difficult. The α miner [8] and heuristic miner [43] aim to return a sound WF-net, but often do not. Parts of the model may be disconnected and cases may get stuck in the middle of the process. Discovered Petri nets having deadlocks and and livelocks are difficult to interpret: They should describe the observed behavior but confuse the analyst instead. The deadlocking or livelocking paths do not contribute to the set of accepting traces $Mod_{AN} \subseteq A^*$. Region-based approaches [12, 38, 44] provide more control over the result. However, without special provisions the set of accepting traces is ill-defined or hard to interpret. The family inductive mining approaches [28, 29, 30]) produce process trees which form a subclass of sound WF-nets. However, the output of these techniques is limited to process trees: a small and very particular subclass of process models.

We would like to discover process models with a configurable representational bias and therefore see *many opportunities for structure theory*. The representational bias, i.e., the class of models that can be discovered, should not be accidental. The class should be defined based on desirable (1) structural properties and (2) behavioral properties. Structural properties include possible constraints like:

- There is one source place and one sink place marking the start and completion of a case (i.e., a WF-net) [1, 7].
- There should be no mixtures of choice and synchronization (i.e., the net is free-choice) [22].
- Splits and joins should match (i.e., there are no PT- and PT-handles) [26].
- The sort-circuited Petri net should have an S-cover and/or a T-cover [22].
- Places cannot be a split and a join at the same time (for any $p \in P$: $|\bullet p| \leq 1$ or $|p\bullet| \leq 1$).
- Places have at most k inputs and outputs for any $p \in P$: $|\bullet p| + |p\bullet| \leq k$.
- Etc.

Behavioral properties include [7]:

[2] Loops can only be unfolded a finite number of times in the event log. Moreover, in case of concurrency, one cannot expect to see all interleavings in the log.

- Soundness: there are no dead parts and it is always possible to reach the final marking and when it is reached the rest of the net is empty.
- Generalized soundness: the same as soundness but with any number of tokens in the source place.
- Relaxed soundness: there is at least one execution that ends up in the final marking.
- Deadlock free: the only reachable dead marking is the final marking.
- Etc.

As shown in [2, 40, 7] there are interesting relations between structure and behavior. These are key to limit the search space to the desired class of models. It is not very effective to generate models first and subsequently check whether they match the desired representational bias. Therefore, structural techniques are needed to limit the search space *during* discovery.

4 Conformance Checking

After discussing the (possible) role of structure theory in control-flow discovery, we now look at the situation in which both a process model and an event log are given. The model may have been constructed by hand or may have been discovered. Moreover, the model may be normative or descriptive. *Conformance checking* relates events in the event log to activities in the process model and compares both. The goal is to find commonalities and discrepancies between the modeled behavior and the observed behavior.

For conformance checking an event log $L \in \mathcal{B}(A^*)$ and a process model $Mod \subseteq A^*$ are used as input. Here we assume that process model Mod was specified in terms of accepting labeled Petri net $AN = (N, M_{init}, M_{final})$ with $N = (P, T, F, A, l)$. The result of conformance checking is a diagnosis identifying and explaining discrepancies. Hence, a conformance checking algorithm can be described as a function $conf \in \mathcal{B}(A^*) \times \mathcal{P}(A^*) \rightarrow D$ where D is the set of possible diagnostics. For example, we may compute the fraction of cases in the log that fit the model perfectly. Formally: $conf(L, Mod) = \frac{|[\sigma \in L \mid \sigma \in Mod]|}{|L|}$ (note that L is a multiset and Mod is a set).

Simply counting the fraction of fitting cases is useful, but does not provide detailed diagnostics. Moreover, one cannot distinguish between cases that deviate just a bit and cases that are completely unrelated. Therefore, more advanced techniques have been developed. The *token-based conformance checking* approach proposed in [36] counts the number of missing and remaining tokens. State-of-the-art techniques in conformance checking are often based on the notion of *alignments* [6, 9]. Alignments relate events in the log to transition occurrences in the model. An alignment is a sequence of *moves*. There are three types of moves: synchronous moves (model and log agree), moves on model only (the model needs to make a move that is not matched by the event log), and moves on log only (an event in the log cannot be matched by the model). Here we cannot give the details. However, the construction of an optimal alignment can be formulated as a shortest path problem in the state space obtained by taking the synchronous products of both the model and log. This shortest path problem greatly benefits from the marking equation which can be used to (1) prune the state-space by removing paths that cannot lead to the final marking and (2) to compute underestimates for the remaining distance [6, 9]. This is a wonderful example of using structure theory in the context of process mining.

Apart from alignments there may be other opportunities for structure theory. If there is a clear relation between structure and behavior, then there are opportunities to speed-up conformance checking.

5 Outlook

In this extended abstract, we positioned structure theory in the context of more data-driven challenges. Structure theory has been applied to verification questions in Business Process Management (BPM). For example, the soundness notion for WF-nets can be related to a variety of "structural ingredients", e.g., by using properties specific for free-choice WF-nets or by applying the marking equation to get initial diagnostics. However, even more promising are the *applications of structure theory in process mining*. We provided two example questions (process discovery and conformance checking) where structure theory could play a prominent role. Process discovery is probably the most important and most visible intellectual challenge related to process mining. It is far from trivial to construct a process model based on event logs that are incomplete and noisy. New process mining approaches should reconsider the representational bias to be used. However, this is only feasible for real-life event logs if the structure can be related to behavior. Alignments are a powerful tool to relate modeled and observed behavior. However, computing optimal alignments requires solving large optimization problems for every trace in the event log. Fortunately, the marking equation can been used to prune the search space and guide the search algorithms.

We hope that this extended abstract will encourage people working on structure theory to consider the many interesting and challenging problems in process mining. There are great opportunities for original research and a need to better cope with the abundance of event data. Clearly, it does not make sense to consider only models when analyzing existing processes and systems. We should also take into account the data to remain relevant for the stakeholders.

References

1. W.M.P. van der Aalst. The Application of Petri Nets to Workflow Management. *The Journal of Circuits, Systems and Computers*, 8(1):21–66, 1998.
2. W.M.P. van der Aalst. Workflow Verification: Finding Control-Flow Errors using Petri-net-based Techniques. In W.M.P. van der Aalst, J. Desel, and A. Oberweis, editors, *Business Process Management: Models, Techniques, and Empirical Studies*, volume 1806 of *Lecture Notes in Computer Science*, pages 161–183. Springer-Verlag, Berlin, 2000.
3. W.M.P. van der Aalst. Business Process Management: A Comprehensive Survey. *ISRN Software Engineering*, pages 1–37, 2013. doi:10.1155/2013/507984.
4. W.M.P. van der Aalst. Data Scientist: The Engineer of the Future. In K. Mertins, F. Benaben, R. Poler, and J. Bourrieres, editors, *Proceedings of the I-ESA Conference*, volume 7 of *Enterprise Interoperability*, pages 13–28. Springer-Verlag, Berlin, 2014.

5. W.M.P. van der Aalst. *Process Mining: Data Science in Action*. Springer-Verlag, Berlin, 2016.
6. W.M.P. van der Aalst, A. Adriansyah, and B. van Dongen. Replaying History on Process Models for Conformance Checking and Performance Analysis. *WIREs Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.
7. 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.
8. W.M.P. van der Aalst, A.J.M.M. Weijters, and L. Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.
9. A. Adriansyah, B. van Dongen, and W.M.P. van der Aalst. Conformance Checking using Cost-Based Fitness Analysis. In C.H. Chi and P. Johnson, editors, *IEEE International Enterprise Computing Conference (EDOC 2011)*, pages 55–64. IEEE Computer Society, 2011.
10. E. Badouel, L. Bernardinello, and P. Darondeau. *Petri Net Synthesis*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag, Berlin, 2015.
11. G. Balbo and M. Silva, editors. *Performance Models for Discrete Event Systems with Synchronisations: Formalisms and Analysis Techniques*, Zaragoza, Sept 1998. Kronos.
12. R. Bergenthum, J. Desel, R. Lorenz, and S. Mauser. Process Mining Based on Regions of Languages. In G. Alonso, P. Dadam, and M. Rosemann, editors, *International Conference on Business Process Management (BPM 2007)*, volume 4714 of *Lecture Notes in Computer Science*, pages 375–383. Springer-Verlag, Berlin, 2007.
13. G. Berthelot. Checking Properties of Nets Using Transformations. In G. Rozenberg, editor, *Advances in Petri Nets 1985*, volume 222 of *Lecture Notes in Computer Science*, pages 19–40. Springer-Verlag, Berlin, 1986.
14. G. Berthelot. Transformations and Decompositions of Nets. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets 1986 Part I: Petri Nets, central models and their properties*, volume 254 of *Lecture Notes in Computer Science*, pages 360–376. Springer-Verlag, Berlin, 1987.
15. E. Best. Structure Theory of Petri Nets: the Free Choice Hiatus. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Advances in Petri Nets 1986 Part I: Petri Nets, central models and their properties*, volume 254 of *Lecture Notes in Computer Science*, pages 168–206. Springer-Verlag, Berlin, 1987.
16. 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.
17. J. Campos, G. Chiola, and M. Silva. Properties and performance bounds for closed free choice synchronized monoclass queueing networks. *IEEE Transactions on Automatic Control*, 36(12):1368–1381, 1991.
18. J.M. Colom and M. Silva. Convex geometry and semiflows in P/T nets, A comparative study of algorithms for computation of minimal P-semiflows. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 79–112. Springer-Verlag, Berlin, 1990.
19. J.M. Colom and M. Silva. Improving the Linearly Based Characterization of P/T Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 113–146. Springer-Verlag, Berlin, 1990.
20. J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Deriving Petri Nets from Finite Transition Systems. *IEEE Transactions on Computers*, 47(8):859–882, August 1998.

21. J. Desel. Basic Linear Algebraic Techniques of Place/Transition Nets. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 257–308. Springer-Verlag, Berlin, 1998.
22. J. Desel and J. Esparza. *Free Choice Petri Nets*, volume 40 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1995.
23. J. Desel, W. Reisig, and G. Rozenberg, editors. *Lectures on Concurrency and Petri Nets*, volume 3098 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2004.
24. M. Dumas, M. La Rosa, J. Mendling, and H. Reijers. *Fundamentals of Business Process Management*. Springer-Verlag, Berlin, 2013.
25. A. Ehrenfeucht and G. Rozenberg. Partial (Set) 2-Structures - Part 1 and Part 2. *Acta Informatica*, 27(4):315–368, 1989.
26. J. Esparza and M. Silva. Circuits, Handles, Bridges and Nets. In G. Rozenberg, editor, *Advances in Petri Nets 1990*, volume 483 of *Lecture Notes in Computer Science*, pages 210–242. Springer-Verlag, Berlin, 1990.
27. K. Jensen. Coloured Petri Nets and the invariant-method. *Theoretical Computer Science*, 14:317–336, 1981.
28. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Discovering Block-structured Process Models from Event Logs: A Constructive Approach. In J.M. Colom and J. Desel, editors, *Applications and Theory of Petri Nets 2013*, volume 7927 of *Lecture Notes in Computer Science*, pages 311–329. Springer-Verlag, Berlin, 2013.
29. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. In N. Lohmann, M. Song, and P. Wohed, editors, *Business Process Management Workshops, International Workshop on Business Process Intelligence (BPI 2013)*, volume 171 of *Lecture Notes in Business Information Processing*, pages 66–78. Springer-Verlag, Berlin, 2014.
30. S.J.J. Leemans, D. Fahland, and W.M.P. van der Aalst. Scalable Process Discovery and Conformance Checking. *Software and Systems Modeling*, pages ??–??, 2016.
31. J. Martinez and M. Silva. A simple and fast algorithm to obtain all invariants of a generalised Petri Net. In C. Girault and W. Reisig, editors, *Application and theory of Petri nets : selected papers from the first and the second European workshop*, volume 52 of *Informatik Fachberichte*, pages 301–310, Berlin, 1982. Springer-Verlag, Berlin.
32. T. Murata. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.
33. OMG. Business Process Model and Notation (BPMN). Object Management Group, formal/2011-01-03, 2011.
34. W. Reisig. *Petri Nets: Modeling Techniques, Analysis, Methods, Case Studies*. Springer-Verlag, Berlin, 2013.
35. 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.
36. A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33(1):64–95, 2008.
37. M. Silva, E. Teruel, and J.M. Colom. Linear Algebraic and Linear Programming Techniques for the Analysis of Place/Transition Net Systems. In W. Reisig and G. Rozenberg, editors, *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*, pages 309–373. Springer-Verlag, Berlin, 1998.
38. M. Sole and J. Carmona. Process Mining from a Basis of Regions. In J. Lilius and W. Penczek, editors, *Applications and Theory of Petri Nets 2010*, volume 6128 of *Lecture Notes in Computer Science*, pages 226–245. Springer-Verlag, Berlin, 2010.
39. E. Teruel and M. Silva. Liveness and home states in equal conflict systems. In M. Ajmone Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, pages 415–432. Springer-Verlag, Berlin, 1993.

40. H.M.W. Verbeek, T. Basten, and W.M.P. van der Aalst. Diagnosing Workflow Processes using Woflan. *The Computer Journal*, 44(4):246–279, 2001.
41. H.M.W. Verbeek, M.T. Wynn, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Reduction Rules for Reset/Inhibitor Nets. *Journal of Computer and System Sciences*, 76(2):125–143, 2010.
42. J. Wehler. Simplified Proof of the Blocking Theorem for Free-Choice Petri Nets. *Journal of Computer and System Science*, 76(7):532–537, 2010.
43. A.J.M.M. Weijters and W.M.P. van der Aalst. Rediscovering Workflow Models from Event-Based Data using Little Thumb. *Integrated Computer-Aided Engineering*, 10(2):151–162, 2003.
44. J.M.E.M. van der Werf, B.F. van Dongen, C.A.J. Hurkens, and A. Serebrenik. Process Discovery using Integer Linear Programming. *Fundamenta Informaticae*, 94:387–412, 2010.
45. M. Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag, Berlin, 2007.
46. M.T. Wynn, H.M.W. Verbeek, W.M.P. van der Aalst, A.H.M. ter Hofstede, and D. Edmond. Soundness-preserving Reduction Rules for Reset Workflow Nets. *Information Sciences*, 179(6):769–790, 2009.