



Wolfgang Reisig  
Grzegorz Rozenberg *Eds.*

# Carl Adam Petri: Ideas, Personality, Impact

 Springer

# Carl Adam Petri: Ideas, Personality, Impact

Wolfgang Reisig • Grzegorz Rozenberg  
Editors

# Carl Adam Petri: Ideas, Personality, Impact

 Springer

*Editors*

Wolfgang Reisig  
Institut für Informatik  
Humboldt-Universität zu Berlin  
Berlin, Germany

Grzegorz Rozenberg  
Leiden Institute of Advanced Computer  
Science  
Leiden University  
Leiden, The Netherlands

ISBN 978-3-319-96153-8      ISBN 978-3-319-96154-5 (eBook)  
<https://doi.org/10.1007/978-3-319-96154-5>

© Springer Nature Switzerland AG 2019

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG.  
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

# Preface

This book is a tribute to Carl Adam Petri (1926–2010), a highly original and visionary computer scientist. He was a pioneer of several key research directions in computer science and a towering figure in the field of distributed information processing systems. The ideas proposed by Petri were revolutionary and much ahead of their time; they constitute deep, broad, and lasting contributions to the field.

The multitude of models which are based on the conceptual framework proposed by Petri are collectively referred to as *Petri Nets*. Their strength is based on the combination of impressive theoretical foundations with very broad and impressive applications. In fact, *Petri Nets* has become a yardstick for other models of concurrent and distributed systems.

Petri was an interdisciplinary scientist, in particular he was deeply interested and knowledgeable in physics. As a matter of fact, he always stressed that his ideas originated in physics and his goal was to create foundations of informatics which are rooted in the laws of physics. His broader goal (dream) was to provide netheoretical foundations for physics.

He is perhaps the best known and most influential German computer scientist. However, in Germany Petri did not receive the recognition that he deserved. This may be partially explainable by the fact that he was totally uninterested in the social and political rituals that (much too) often play an important role in recognition by the academic world.

He was a very modest man with a remarkable personality, a “pure scientist”. Science was the essence of his life, everything else was a burden imposed upon him. Conversations with Petri were fascinating and enriching, as he was a patient listener while he also passionately and convincingly argued for his own deep and original ideas.

Carl Adam Petri became a source of inspiration to many. This book is a collection of reflections about Petri and his ideas by individual scientists. Some contributions are recollections of his remarkable personality by people who knew him, some are stories of how Petri’s ideas influenced various people’s scientific developments and careers, while others are more technical expositions (in a style accessible to a broader audience) of various ideas of Petri. Altogether, these individual stories

constitute a unique source of information about Carl Adam Petri and his ideas, and a touching tribute to a remarkable scientist.

Our invitations to contribute to this book were enthusiastically received. We are grateful to all authors for their contributions and fruitful interactions during the editing process. We are also thankful to Ronan Nugent from Springer-Verlag for pleasant and efficient cooperation in producing this book.

Berlin, Germany  
Leiden, The Netherlands  
Boulder, USA  
June 2018

Wolfgang Reisig  
Grzegorz Rozenberg

# Contents

## Part I Perspectives on Petri's Work

<b>Discovering Petri Nets: A Personal Journey</b> .....	3
Wil M. P. van der Aalst	
<b>Observations of a Lateral Entrant</b> .....	11
Einar Smith	
<b>Nets, Cats and Pigs: Carl Adam Petri and His Slides</b> .....	17
Giorgio De Michelis	
<b>Invention or Discovery?</b> .....	25
Kees M. van Hee	
<b>Petri's Understanding of Nets</b> .....	31
Dirk Fahland	
<b>On the Two Worlds of Carl Adam Petri's Nets</b> .....	37
Rüdiger Valk	
<b>Petri Nets: The Next 50 Years—An Invitation and Interpretative Translation</b> .....	45
Heinz W. Schmidt	
<b>Petri Nets Are (Not Only) Distributed Automata</b> .....	67
David de Frutos Escrig	
<b>Petri Nets: A Simple Language and Tool for Modeling Complex Ideas</b> ....	73
Ryszard Janicki	

## Part II Personal Recollections

<b>Carl Adam Petri: A Tribute from Aarhus</b> .....	81
Kurt Jensen and Mogens Nielsen	

<b>Some Interactions with Carl Adam Petri over Three Decades</b> .....	85
Manuel Silva	
<b>Petri Nets and Petri's Nets: A Personal Perspective</b> .....	93
Maciej Koutny	
<b>Coffee and Cigarettes</b> .....	97
Javier Esparza	
<b>Early Interactions with Carl Adam Petri</b> .....	105
Brian Randell	
<b>A Personal Journey in Petri Net Research</b> .....	111
Xudong He	
 <b>Part III Technical Themes</b>	
<b>Carl Adam Petri's Synchronic Distance</b> .....	119
Jörg Desel	
<b>How Carl Adam Petri Deeply Influenced My Understanding of Invariance and Parallelism</b> .....	133
Gerard Memmi	
<b>Toward Distributed Computability Theory</b> .....	141
Roberto Gorrieri	
<b>Petri Inheritance: The Foundation of Nondeterministic, Concurrent Systems</b> .....	147
Roberto Bruni and Ugo Montanari	
<b>Coordinating Behaviour</b> .....	155
Ekkart Kindler	
<b>Inductive Counting and the Reachability Problem for Petri Nets</b> .....	161
Peter Chini and Roland Meyer	
 <b>Part IV Connecting to Other Areas</b>	
<b>On Petri Nets in Performance and Reliability Evaluation of Discrete Event Dynamic Systems</b> .....	173
Gianfranco Balbo and Gianfranco Ciardo	
<b>Modelling Time Using Petri Nets</b> .....	187
Michel Diaz	
<b>All True Concurrency Models Start with Petri Nets: A Personal Tribute to Carl Adam Petri</b> .....	193
Wojciech Penczek	
<b>Petri Nets for BioModel Engineering: A Personal Perspective</b> .....	205
Monika Heiner	



**Petri Nets in Systems Biology: Transition Invariants, Maximal Common Transition Sets, Transition Clusters, Mauritius Maps, and MonaLisa** ..... 217  
Ina Koch

**From Nets to Circuits and from Circuits to Nets** ..... 227  
Jordi Cortadella

**Living Lattices** ..... 233  
Alex Yakovlev

**The Road from Concurrency to Quantum Logics** ..... 243  
Luca Bernardinello and Lucia Pomello

**Part I**  
**Perspectives on Petri's Work**

# Discovering Petri Nets: A Personal Journey



Wil M. P. van der Aalst

## 1 Introduction

Carl Adam Petri (12 July 1926–2 July 2010) pioneered the computer science discipline and is one of the founding fathers of the wonderful field of concurrency. He single-handedly started a new subfield of computer science focusing on *concurrency* [6]. As Robin Milner phrased it in the acceptance speech for his Turing Award in 1991: “Much of what I have been saying was already well understood in the sixties by Carl Adam Petri, who pioneered the scientific modeling of discrete concurrent systems. Petri’s work has a secure place at the root of concurrency theory!” Petri nets have become a standard tool for modeling and analyzing processes where concurrency plays a prominent role. The ideas have been embedded in many other process modeling notations. For example, the widely used BPMN (Business Process Model and Notation) models use token-based semantics [4]. After working with Petri nets for over 30 years, I remain surprised by the elegance of the formalism. Using a few basic concepts (places, transitions, arcs, and tokens) and the simple firing rule, one enters a new “world” where it is possible to model a wide range of behaviors and study non-trivial phenomena (conflict, concurrency, confusion, etc.).

According to [7], Petri invented Petri nets in 1939 at the age of 13 for the purpose of modeling chemical processes. However, many refer to his Ph.D. thesis, defended in 1962 [5], as the starting point for Petri nets. This is only partially true, since his Ph.D. thesis does not show the characteristic diagrams we know today. These emerged in the mid-1960s and subsequently conquered the world. Petri nets are used in a wide range of domains, directly benefiting from the theoretical foundations developed over the last 50 years.

---

W. M. P. van der Aalst (✉)

Lehrstuhl für Informatik 9 (Process and Data Science), RWTH Aachen University, Aachen, Germany

e-mail: [wvdaalst@pads.rwth-aachen.de](mailto:wvdaalst@pads.rwth-aachen.de)

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,

[https://doi.org/10.1007/978-3-319-96154-5\\_1](https://doi.org/10.1007/978-3-319-96154-5_1)

In the remainder, I will describe how I got submerged into Petri nets and how it has affected my research and scientific career. Moreover, I will link two of Petri's guiding principles to my current research area:

- **GP1:** *Concurrency should be a starting point for system design and analysis and not added as an afterthought (locality of actions).*
- **GP2:** *A formalism should be consistent with the laws of physics and not take any shortcuts at the foundational level.*

Anyone familiar with Petri's work and his lectures will recognize these guiding principles. I will relate these two principles to process mining and the broader field of Business Process Management (BPM). Process mining can be viewed as the missing link between model-based process analysis and data-oriented analysis techniques [12]. Many process mining techniques use Petri nets or related formalisms and directly benefit from the above two principles proposed by Carl Adam Petri. This paper concludes by discussing Petri's heritage.

## 2 Personal Journey into the Wonderful World of Petri Nets

I met Carl Adam Petri for the first time in 1989 in Bonn while attending the 10th International Conference on Applications and Theory of Petri Nets. I was doing my Ph.D. at the time and it was a very exciting and inspiring experience. This was my first Petri net conference, and I did not know that many Petri net conferences would follow. Over the last 30 years, I have served as a program committee chair (twice, in 2003 and 2017), organized the conference once, served as a steering committee member since 2003, and played many other roles (e.g., chairing different workshops and committees).

For my Master's project [9], I worked in the research group of Kees van Hee on a language and tool called *ExSpect*. In a way we "rediscovered" Petri nets, not knowing the seminal work done by Petri. Initially, we used triangles for places rather than circles and there were also other differences. However, the similarities between our work and (colored) Petri nets were striking. Therefore, we soon joined the Petri net community. I was responsible for adding time to our high-level *ExSpect* nets. *ExSpect* was developed concurrently with *Design/CPN* (which later evolved into *CPN Tools*, still hosted by our research group in Eindhoven). Both languages used colored tokens and provided hierarchy notions. During my Ph.D. I continued to work on *ExSpect*. My main focus was on the analysis of temporal behavior using both model checking and simulation [10]. The primary application domain for my work was the broader field of logistics. We analyzed supply chains, distribution centers, railway transport, container terminals, etc. using *ExSpect*.

After a few years I became the leader of a small research group in Eindhoven and my interests shifted to workflow management [13]. I noted the huge potential for applying Petri nets in this up and coming domain. We developed a framework for modeling and analyzing workflow processes based on Petri nets. This led to seminal notions such as WorkFlow nets (WF-nets) and soundness. Interestingly, we were able to use Petri net notions such as invariants, siphons, traps, reduction rules, etc. to verify the (in)correctness of workflow models stored in commercial systems such as *Staffware*.

After designing several workflow languages and developing several workflow management systems (YAWL, Declare, etc.), I got more and more interested in the *relationship between models and reality*. This was fueled by the repeated observation that simulation models (modeled in *CPN Tools* or *ExSpect*) rarely behave like real organizations, machines, and people. At the same time, workflow management research shifted from automation to *Business Process Management* (BPM). See [11] for a survey describing this transition. The scope of BPM extends far beyond workflow automation, including the understanding of why processes and organizations achieve particular performance levels (time, quality, costs, compliance, etc.). Process improvement requires a deep understanding of processes that cannot be obtained through modeling alone.

The desire to link models and reality naturally evolved into the new field of *process mining* around the turn of the century. We developed the first process mining algorithms around 1999. Initially, we used the term “workflow mining” rather than “process mining.” Process mining starts from event data and uses process models in various ways, e.g., process models are discovered from event data, serve as reference models, or are used to project bottlenecks onto. Many process mining techniques use Petri nets for obvious reasons. Later, I will elaborate on the role of Petri nets and Petri’s guiding principles in process mining.

Although I worked on very different topics (logistics, simulation, workflow verification, workflow automation, BPM, and process mining), all of my research was (and still is) related to Petri nets in some form.

Concurrently, I moved up the academic ranks and became assistant professor (1992), associate professor (1996), and full professor (2000). In 2003, I organized the Petri net conference in Eindhoven together with Kees van Hee. This conference was a big success and quite special because Carl Adam Petri gave a talk after not having attended the conference for many years (Fig. 1). Grzegorz Rozenberg (who can be considered to be the “godfather” of our wonderful Petri net community) encouraged me to organize a co-located event along side the Petri net conference. This resulted in the first Business Process Management conference (BPM 2003). Also within the BPM community Petri nets were adopted as a standard tool for modeling, analyzing, and enacting business processes. BPM is just one of many fields using Petri nets, illustrating the foundational nature of Petri’s ideas.



**Fig. 1** Carl Adam Petri (middle) and Grzegorz Rozenberg (left) during the Petri net conference we organized in 2003. The photo was taken just after Petri was honored with the prestigious title “Commander in the Order of the Netherlands Lion” by Her Majesty the Queen of the Netherlands

### 3 Concurrency is Key

Petri’s first guiding principle is “Concurrency should be a starting point for system design and analysis and not added as an afterthought (locality of actions)” (**GPI**). Many other modeling approaches start from a sequential view of the world and then add special operators to introduce concurrency and parallelism. Petri nets are inherently concurrent. Although Petri nets are often seen as a procedural language, they can be viewed as declarative. A Petri net without any places and a set of transitions  $T$  allows for any behavior involving the activities represented by  $T$ . Adding a place is like introducing a constraint. The idea that transitions (modeling activities or actions) are independent (i.e., concurrent) *unless specified otherwise* is foundational! This allows us to model things in a natural manner and also facilitates analysis. Actions are local and this allows us to understand things better while enabling “divide and conquer” approaches (e.g., decomposing analysis problems).

Mainstream notations for modeling processes use token-based semantics adopted from Petri nets. The de facto standard for business process modeling—BPMN (Business Process Model and Notation) [4]—uses token passing. Also UML activity diagrams use token-based semantics and a notation similar to Petri nets. Unfortunately, these languages provide a plethora of control-flow constructs, basically killing the elegance of the original proposition. However, in the back end of such languages and supporting systems, one can often find Petri nets. For example, BPMN models are often translated into classical Petri nets for verification.

## 4 Process Mining: Relating Observed and Modeled Behavior

Petri's second guiding principle is "A formalism should be consistent with the laws of physics and not take any shortcuts at the foundational level" (**GP2**). He often related concurrency theory to physics [2, 8]. However, the principle can also be applied to everyday discrete event processes (e.g., in manufacturing, healthcare logistics, luggage-handling systems, software analysis, smart maintenance, website analytics, and customer journey analysis). We seek models adequately describing these real-world phenomena. Interestingly, the digital universe and the physical universe are becoming more and more aligned, making it possible to study these discrete event processes much better. The spectacular growth of the digital universe, summarized by the overhyped term "Big Data," makes it possible to record, derive, and analyze *events*. Events may take place inside a machine (e.g., an X-ray machine, an ATM, or a 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. [12]. Events may be "life events," "machine events," or "organization events." I coined the term *Internet of Events* (IoE) to refer to all event data available [12].

The event data that are abundantly available allow us to relate real-life behavior to modeled behavior. More specifically, we can learn process models from such event data (process discovery) or replay event data on models to see discrepancies (conformance checking). This is exactly what process mining aims to do.

Process mining starts from *event logs*. An event log contains event data related to a particular process. Each event in an event log refers to one process instance, often called a case. Events related to a case are ordered. Events can have attributes. Examples of typical attribute names are activity, time, costs, and resource. *Process discovery* is one of the most challenging process mining tasks. Based on an event log, a process model is constructed thus capturing the behavior seen in the log. Dozens of process discovery algorithms are available and many produce Petri nets. Input for *conformance checking* is a process model with executable semantics and an event log. Discrepancies between the log and the model can be detected and quantified by replaying the events in the log. Simple conformance-checking approaches try to play the token game and count missing and remaining tokens. More sophisticated approaches solve optimization problems to find modeled behavior most related to the observed behavior. Some of the discrepancies found may expose undesirable deviations, e.g., conformance checking signals the need for better controls. Other discrepancies may reveal desirable deviations and can be exploited to improve process support.

The empirical nature of process mining immediately exposes formalisms that are not able to capture real-life behavior. Choosing the wrong "representational bias" results in discovered models that are poorly fitting (observed behavior is not allowed or the model is overfitted or underfitted) [12].

Petri nets are attractive for process mining given the abundance of analysis techniques. For example, conformance-checking techniques use the marking equation to dramatically reduce the search space when computing alignments. Moreover, the fact that “a Petri net without any places and a set of transitions  $T$ ” allows for any behavior involving the activities represented by  $T$ ” is a great starting point for process discovery. Obviously, such a Petri net is underfitted, but additional constraints can be introduced by adding places. This is related to the seminal idea of *regions* (both language-based regions and state-based regions) [1, 3]. The synthesis of Petri nets based on regions is one of the cornerstones of process discovery, very much in the spirit of Petri’s second guiding principle.

## 5 Petri’s Heritage

This short paper focused on two of Petri’s guiding principles: (1) concurrency should be a starting point for system design and analysis (and not added as an afterthought), and (2) a formalism should be consistent with the laws of physics and not take any shortcuts at the foundational level. I linked these two principles to my own research over the last 30 years and discussed how these principles relate to the emerging field of process mining. Obviously, concurrency of behavior and consistency with reality are key notions in process mining. However, above all, this paper described a personal journey reflecting on the influence of Petri’s work on my career and research aimed at discovering Petri nets from events.

Carl Adam Petri discovered his nets at a time when information processing was viewed as something sequential. Formal notations for concurrency and asynchronous distributed systems were uncovered by Petri’s seminal work. Petri nets are used in many domains and the strong theoretical foundation often helps to solve “wicked problems” and avoid reinventing the “wheels of concurrency.” For example, numerous workflow management, BPM, and process mining approaches directly build on Petri nets.

However, it remains crucial to invest in the foundations of non-sequential processes. Einar Smith’s book on Petri’s life and achievements [8] provides interesting insights into the “good old days” of scientific research at the Gesellschaft für Mathematik und Datenverarbeitung (GMD). At GMD in Schloss Birlinghoven there was still time to work on the theoretical foundations of computing. This is in stark contrast with today’s research practices driven by “quick wins” and application-oriented projects rather than long-term scientific goals. Today’s scientists simply do not have the time to take a step back and ask long-term questions in the way Carl Adam Petri did. *Would Petri have survived today’s research environment?* As part of his heritage we should ask ourselves this question repeatedly. This may help us to create better research environments for work on the true foundations of computing.



## References

1. E. Badouel, L. Bernardinello, P. Darondeau, *Petri Net Synthesis*. Texts in Theoretical Computer Science. An EATCS Series (Springer, Berlin, 2015)
2. W. Brauer, W. Reisig, Carl Adam Petri and Petri nets. *Informatik-Spektrum* **29**(5), 369–374 (1996)
3. A. Ehrenfeucht, G. Rozenberg, Partial (Set) 2-structures - Part 1 and Part 2. *Acta Inform.* **27**(4), 315–368 (1989)
4. OMG. Business Process Model and Notation (BPMN). Object Management Group, formal/2011-01-03 (2011)
5. C.A. Petri, Kommunikation mit Automaten. Ph.D. thesis, Fakultät für Mathematik und Physik, Technische Hochschule Darmstadt, Darmstadt, 1962
6. C.A. Petri, Introduction to general net theory, in *Net Theory and Applications: Proceedings of the Advanced Course on General Net Theory, Processes and Systems* (Hamburg, 1979), vol. 84, ed. by W. Brauer. Lecture Notes in Computer Science (Springer, Berlin, 1980), pp. 1–20
7. C.A. Petri, W. Reisig, Petri net. *Scholarpedia* **3**(4), 6477 (2008). [http://www.scholarpedia.org/article/Petri\\_net](http://www.scholarpedia.org/article/Petri_net)
8. E. Smith, *Carl Adam Petri: Life and Science* (Springer, Berlin, 2015)
9. W.M.P. van der Aalst, Specificatie en Simulatie met behulp van ExSpect (in Dutch). Master's thesis, Eindhoven University of Technology, Eindhoven, 1988
10. W.M.P. van der Aalst, Timed coloured Petri nets and their application to logistics. Ph.D. thesis, Eindhoven University of Technology, Eindhoven, 1992
11. W.M.P. van der Aalst, Business process management: a comprehensive survey. *ISRN Softw. Eng.* 1–37 (2013). <https://doi.org/10.1155/2013/507984>
12. W.M.P. van der Aalst, *Process Mining: Data Science in Action* (Springer, Berlin, 2016)
13. W.M.P. van der Aalst, K.M. van Hee, *Workflow Management: Models, Methods, and Systems* (MIT Press, Cambridge, 2002)

# Observations of a Lateral Entrant



Einar Smith

## 1 Meeting Petri

Helga Genrich put down the telephone receiver: “Carl Adam is in his office, he can receive you right away.” That is how in January 1984, I was introduced to Petri and his world. I had come to know Helga through a common friend, and at that time Helga worked on a research programme on Information and Law within the Petri-Institute in the GMD. In this context Petri was looking for somebody with a background in formal logic, and mathematical logic was in fact what I had specialised in during my university years.

Already my first encounter with Petri was fascinating. We discussed whether the Cartesian product in sets is associative (it is not), whether Keynesian economics was based on a sound mathematical model, and of course, what consequences Einstein’s views on simultaneity and causality had for a mathematical understanding of real-world processes.

When I had returned to Helga’s office, Petri phoned her and said: “He made an excellent impression.” So it came that I was offered a position as research assistant beginning on March 1st. However, since as a Norwegian citizen, I had to apply for a work permit in Germany through the German embassy in Oslo, I asked to postpone the entry date by, say, a month. Petri’s answer was laconic: “Your first duty will be to study the literature; and where you do that is up to you.”

---

E. Smith (✉)

Fraunhofer Institute for Algorithms and Scientific Computing (SCAI), Sankt Augustin, Germany

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,

[https://doi.org/10.1007/978-3-319-96154-5\\_2](https://doi.org/10.1007/978-3-319-96154-5_2)

## 2 The Society of Followers

What soon became clear for a lateral entrant was that Petri's words were often considered as eternal truths by his followers, almost approaching apotheosis within his lifetime. Many of the concepts Petri introduced were taken at face value, and studied for their mathematical properties rather than the deeper meaning Petri intended to express.

**Density and Completeness** For instance, books were filled with theorems on Petri's K-density and his first attempts at Dedekind completeness in discrete partial orders. Concerning K-dense orders Petri himself notes that the "requirement can only be violated if an infinite number of sequential processes have to wait for each other, i.e., in a situation of no practical interest." With K-density and Dedekind completeness, what Petri wanted was to consolidate the intuitive sensation of continuity on one hand and the discrete articulation of actual observations on the other. In this respect the first attempt to formulate Dedekind completeness did not turn out to be sufficiently sustainable.

In his second version, I was invited to take part as a co-author, an honour and privilege I was of course overwhelmed by [1]. Petri must have appreciated my (to be honest, very marginal) contribution, since also later I was once more invited to join him in a publication, see e.g. [2].

A detailed discussion of the concepts mentioned above, as well as many other aspects of Petri's work, can be found in [4].

**Contact** Other important examples where Petri's word was taken too literally concern the notions of contact, conflict and confusion. (Details can again be found in the biography mentioned above.)

On contacts, Petri often advocated the view that they are a consequence of badly designed models. As a consequence contacts were often excluded in the works of his followers.

However, on the other hand, Petri himself often also noted that contacts are important to detect unsafe situations: "An elementary particle can not occupy the space already occupied by another one, but a car unfortunately can, and so can a record of data."

**Conflict** As he expressed time and again, Petri believed that the world is causally deterministic, and conflicts only exist in partial systems; for instance: "If a transition set contains a conflict, then the system described by it possesses a non-empty environment."

**Confusion** This has an immediate and unfortunate consequence for the concept of confusion: Simply, that confusions cannot exist, because how can a border between system and environment exist such that (1) on one hand, the condition required for conflict resolution is located *outside* the system, but (2) on the other hand, there is possibly *no such condition at all*, simply because there is no conflict?

In Petri's own words: "When we encounter such a situation we may conclude that we have drawn the boundary between the system and its environment in an awkward manner, and that we should draw it somewhere else, in order to reduce confusion to mere conflict resolution. It follows that we really don't need to set up a theory of confusion - which would indeed be difficult to do. It has, in fact, been tried more than once and seems almost impossible."

In Petri's closer vicinity, the belief that confusion was to be—and could be—avoided, often gave reason to concentrate research activities on special limited net classes, where confusion was in fact excluded by construction.

However, later it turned out that—on the contrary—confusion is ubiquitous; it will inevitably appear, whenever the consequences of independent events have to be synchronised [3].

An often-overlooked implication of this theorem is that Petri's own construction of a conflict-free OR-gate in his dissertation cannot be correct.

In a later private conversation, Carl Adam commented with a sigh: "Before I used to claim that confusion can easily be avoided; now I claim the opposite."

**'Time' Is a Four-Letter Word** Closely related to confusion is the determination of order between independent events. The belief in pure causality among Petri's followers sometimes approached the surrealistic. The innocent remark that the winner of a 100-m dash is surely determined by the order between the independent events of the runners crossing the finish line could be met with a smile somewhere between sardonic and patronising: "Who knows if there is not some connecting causal parameter hidden in the background?"

In the most programmatic form, the reluctance towards confusion and timing was probably expressed by Anatol Holt: "Thus an extra causal factor, not represented explicitly in choices... has entered the scene, *namely the factor of relative timing*... This 'extra factor' is both technically and philosophically unacceptable. It is technically unacceptable because it destroys the possibility of tracing the outcomes of choices to the outcome of other choices... one of the key objectives, in my opinion, of an adequate system model."

Regarding the philosophical significance of the time factor, Holt remarks: "Communication and only communication establishes causal connections between choices. Concurrency was to express the relative freedoms that remain in the light of these relative causal constraints." According to Holt, something that may influence causality must be an element of communication, and that is not the case for the time factor.

In essence, Holts conclusion amounts to the view that what *should not* exist actually also *does not* exist. This is in complete concordance with a well-known observation by the German satirical poet Cristian Morgenstern in a poem "Die unmögliche Tatsache" (The impossible fact): "Weil, so schließt er messerscharf, nicht sein kann, was nicht sein darf." (For, he reasons pointedly, That which must not, can not be.)

### 3 The Competitors

In the modelling of distributed systems, Petri nets encountered existing competition, mainly from models based on sequential automata theory, where the concept of arbitrary interleaving became the standard method of formulating independence.

Petri nets appeared as late arrivals in this business. To be accommodated the net-representatives had to show they could do everything the others could, and even more so; as in the musical ‘Annie get your gun’: “Anything you can do, I can do better.”

But the rules of the game were set by the others. This meant in particular that concepts not expressible in interleaving approaches were not propagated offensively by net-followers. Unfortunately this concerned the main topics discussed above, such as contact and confusion.

In many scientific approaches other ideas are occasionally neglected with the remark “not invented here”. In net theory the opposite phenomenon “not invented by others” occasionally seems to prevail.

### 4 Consequences

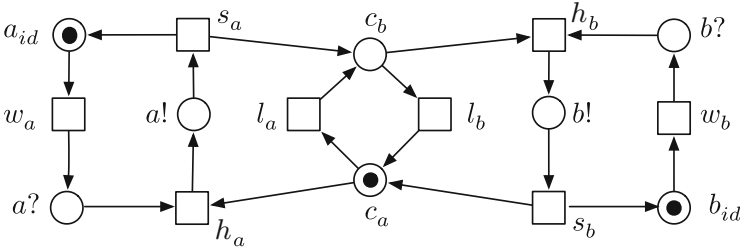
Perhaps it is time to review some of Petri’s fundamental and seminal ideas in the world of non-sequential processes and distributed systems. This includes the admission that even a genius like him was sometimes caught in a web of contradictory thoughts. Adapting Petri’s own words (actually originally referring to the German writer Johann Wolfgang von Goethe), we should “liberate ourselves from the prejudice that whatever is written by Petri must be great and good, simply because it is by Petri.”

In my opinion, the main obstacle Petri encountered was one he laid himself, namely to take philosophical determinism at face value. Without this hindrance, he had already developed all the tools necessary to deal with the ubiquitous timing issues in synchronisation.

In his approach to measurement theory, he noted that measurement is ultimately based on a notion of similarity, which is by nature non-transitive, a non-transitivity which however can be limited to a  $\pm 1$ -deviation. (For details again see [4].)

This approach can be carried over to the timing of independent events, and the design of corresponding system models.

**At the Bus Stop** As a simple illustration, consider a bus approaching a bus stop. If there is a passenger waiting, the driver has to stop and pick him up. If nobody is waiting, the driver may pass by without stopping. Now a client has complained to the bus company that he *was* waiting, but the bus nonetheless did not stop. The driver, on his side, contends that he had actually first noted the passenger in the rear mirror, because he definitely had arrived too late. It is one person’s word against another, and the case will probably not be resolved.



**Fig. 1** Controlled access to a printer shared by two agents *a* and *b*

But one thing *does* result: If there was a dispute about *this* bus, then the passenger is undoubtedly in time for the *next* one. Hence also in this case there is a typical  $\pm 1$ -problem lurking in the background. If the frequency of bus departures is sufficiently high, the negative consequences can be limited.

**Distributed Access** We present a somewhat larger example, in which conflict, confusion and the main timing ideas above are discussed in a model to control the access to a shared resource, say a printer, by two agents *a* and *b* (Fig. 1).

The system has to ensure that at any time *only one* user has access, and—on the other hand—that every user that requires the printer will eventually get it. Presently both agents are in states where they do not request access, represented by the tokens on the places  $a_{id}$  and  $b_{id}$  (the tag “id” is short for “idle”). In this situation *a*, say, can issue a request. In the model this is represented by the occurrence of  $w_a$ , after which the condition  $a?$  is satisfied (depicted graphically by moving the token).

The printer access control is based on a mechanism that alternately polls the demands of the agents. This is represented by the inner circle  $c_a-l_a-c_b-l_b$ . The token on  $c_a$  indicates that the printer is currently offered to *a* because now  $h_a$  is enabled. Occurrence of  $h_a$  withdraws the two input tokens and puts a token on  $a!$ , reserving the exclusive access for *a*. The transition  $s_a$  terminates the use; *a* returns to the state “idle”,  $c_b$ , is marked, and the printer is now ready for user *b*.

Assume that the control has been designed such that whenever agent *a* has issued a request (token on the condition  $a?$ ), and the control unit is ready to accept it (token on  $c_a$ ), then the printer will in fact be assigned to *a*. If additionally we can assume that signals will stabilise within one cycle, the worst case that can happen is that *a* is passed over *once*, when for example the request signal on  $a?$  has not yet stabilised.

As in the bus example above we again get a  $\pm 1$ -problem, with which we can live, but probably also have to live, since no decisively different solution seems viable: A system design without confusion appears impossible, and the control must rely on a priority management that decides on the temporal order of the concurrent independent events in the confusion. (However we stress the point that such temporal relationships are only evaluated in the *immediate vicinity of a potential conflict*. This does not imply any assumption whatsoever of a global time concept.)

*Challenge for adherents to the deterministic tradition outlined above:* Develop an equivalent model without confusion and local timing.

**Final Words** In this author's opinion the more promising alternative is to extend the traditional approach to include confusion and its consequences in the theory.<sup>1</sup> To formulate it with a famous quotation, attributed to various people, not least to the great Austrian composer Gustav Mahler: "Tradition is not the adoration of the ashes, but the passing of the flame."

## References

1. C.A. Petri, E. Smith, Concurrency and continuity, in *Lecture Notes in Computer Science Vol. 266: Advances in Petri Nets 1987*, ed. by G. Rozenberg (Springer, Berlin, 1987), pp. 273–292
2. C.A. Petri, E. Smith, The pragmatic dimension of net theory. *J. Integr. Des. Process Sci. Trans. SDPS* **2**(1), 1–7 (1998)
3. E. Smith, On the border of causality: contact and confusion. *Theor. Comput. Sci.* **153**(1–2), 245–270 (1996)
4. E. Smith, *Carl Adam Petri. Life and Science* (Springer, Berlin, 2016)

---

<sup>1</sup>A preliminary study on the semantics of nets with local priority constraints can be obtained on request.

# Nets, Cats and Pigs: Carl Adam Petri and His Slides



**Giorgio De Michelis**

Carl Adam Petri (1926–2010) never wrote a book, and his papers, amounting to a total of 13 during his scientific career, are generally not written in accordance with the standard format of scientific papers—they are so rich and dense that they constitute a serious challenge for the readers, even those who are already experts in the themes he treats. Moreover, except for the lemma dedicated to Petri Nets in Scholarpedia (2008) that he wrote together with Wolfgang Reisig [1], he did not write any paper after 1996. These simple facts are sufficient to show that he did not consider written texts to be the main way to state and record his ideas and to make them available to the public.

On the other hand, he dedicated great care to his slides. He prepared a new collection of slides for almost every talk he gave, at least in the last 35 years of his career, and, sometimes, he prepared them also for different occasions as personalized accounts of his scientific work. In a paper I wrote recently I discuss a slide that is quite important for understanding some ‘philosophical’ aspects of his work whose content never appeared explicitly in his writings [2]. It is therefore interesting to investigate in depth the role of slides in his scientific experience, also taking into account that technological innovation seriously influenced their preparation in the last 30 years.

The sets of slides he prepared for different occasions frequently contained some recurrent subsets of slides, dedicated to particular aspects of his theoretical work. The slides he used to support one of his talks, were, in some sense, a selection and/or a combination of slides taken from a virtual collection of all his slides, which was his ‘complete’ reference material. He continually updated this collection adding new slides to it, substituting and/or ‘modifying some of them.

---

G. De Michelis (✉)

Dipartimento di Informatica, Sistemistica e Comunicazione (DISCo), University of Milano – Bicocca, Milan, Italy

e-mail: [gdemich@disco.unimib.it](mailto:gdemich@disco.unimib.it)

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,

[https://doi.org/10.1007/978-3-319-96154-5\\_3](https://doi.org/10.1007/978-3-319-96154-5_3)



Thus sometimes these were exactly the same as the ones that he had used on previous occasions, sometimes they were new slides presenting his new findings and/or formulations, and sometimes these slides introduced some changes, showing that he was convinced that he had found a better way to express an idea or a fact. Thus, what I have called above the “complete collection” was continually and gradually changing together with his theories.

Einar Smith, in his biography of Petri, recalls a conversation they had about his slides: “Together with the notes for a talk, he once packed away several overhead slides, with the remark: «In case there are questions.» To the astonished, ‘How could he know what questions would be asked’, he had a simple reply: «I pretend to respond to the question, then show these slides and explain them»” [3, p. 106].

## 1 Handwritten Slides

The first slides of Carl Adam Petri that I saw—probably in the 1980s—were obviously handwritten. They were clear and elegant thanks to his calligraphy and good skills in designing them (an example is given in Fig. 1) and any set of them supporting a talk consisted of small subsets containing, in a concise way, all the material needed to treat a specific issue.

It is interesting to note that the slide of Fig. 1, originally prepared for a talk in Beijing in 1981, was reused for a talk he gave 8 years later, in 1989, in Milano with the title “Concurrency Theory and Combined Axiomatics for Concurrency, Causality and Possibility”. The slides presented in Milano contained subsets of slides from 1970, 1975, 1976, 1981, 1988, and 1989. This supports my claim that none of his talks was a repetition of a talk already given before but rather, each of his talks was a stage in a continual reworking of both his ideas and ways to present them.

## 2 Digitized Slides

With the rapid expansion of personal computers in the 1980s also the way Petri prepared his slides changed.

On the one hand, after some years, he began to prepare them with tools such as PowerPoint, which allowed him to improve their aesthetic and communicative quality. He used graphic editors to draw exact and precise diagrams, different character sets to differentiate the role of different captions, and a rich colour palette to make them more vivid and varied (Fig. 2).

Beijing '81

Petri

LAWS OF CONCURRENCY

A and B are called „concurrent“ iff:

- A and B occur at the same time
- A and B have parts a, b which occur at the same time.
- A and B can be interchanged in time, with the same total effect
- A and B are not in sequence
- A and B are causally independent of each other.

What are A and B if they can be called concurrent?

$$A, B \in X$$

$$co \subseteq X \times X$$

In practice, concurrency is used for describing/designing the coordination of activities serving a common purpose ...  
(Co-operation)

**Fig. 1** The slide introducing concurrency from a set of slides prepared for a Beijing talk (1981)

This was not achieved immediately. He was experimenting, moving from well-drafted diagrams, printed captions, and a limited use of colours, to blue or red backgrounds, rich choice of characters and sizes, and spatially distributed diagrams and captions. Slides were related more to what he was going to speak about in his talks, while their communication role concentrated the (mathematical) formalization of the main aspects of his theoretical work.

During the last years of his life, slides became ever more relevant for Petri, as he seemed to consider them a major way of transmitting his thoughts. It is therefore not accidental that in 2005 he copyrighted them (Fig. 3).

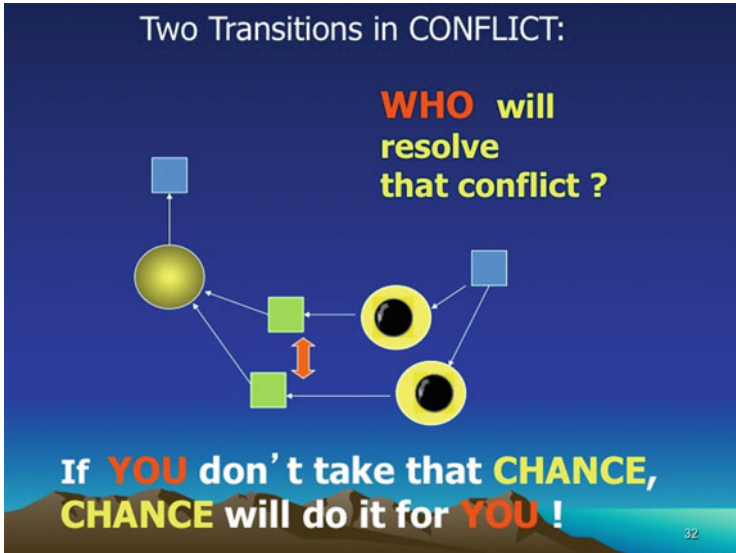



Fig. 2 Slide illustrating conflicts, from the “Definitions” set of 2004



**Notice**

**This collection of files, entitled**  
**“Systematics of Net Modelling”**  
**Sixth Edition, 2005**

**is Copyright © by C.A.Petri 2005**

Fig. 3 Prelude to a collection of slides

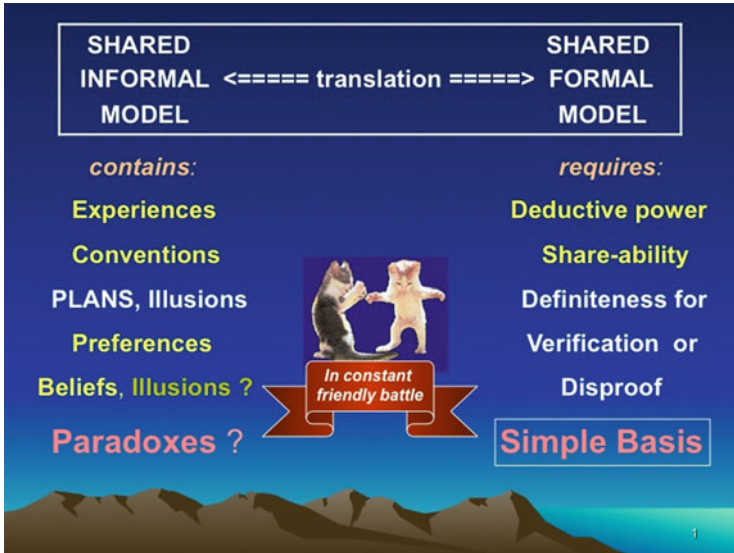


Fig. 4 One slide of the Petri-Zuse collection, 2007

### 3 Animals

With the prevalence of the communication role of his slides, Petri began to design them having in mind the audience looking at them: there were frequently phrases that explicitly addressed the audience, several captions had an ironic flavour, and, finally, cats and pigs began to appear in them.

Small cats were the most frequent hosts of Petri’s slides (see, e.g., Figs. 3 and 4): sometimes they played a role in explaining their meaning, sometimes they seemed not to be interested in what the slide presented, but they always interacted empathetically with the beholder.

Pigs were less frequent inhabitants of Petri’s slides, but their ironic message was stronger: e.g., in Fig. 5, a large group of pigs with, their backs towards the audience represents a ‘sceptical audience inspecting Universe’.

Sometimes, as in the confusion slide of Fig. 6, cats and pigs were both present, contributing to a sense of confusion emphasizing the content of the slide.

### 4 Gifts

In his final years slides were coloured, copyrighted, populated by animals, interfering with their content, and dialoguing ironically with the beholder. The personality of Petri clearly emerges from them. It seems that, through his slides, he was able

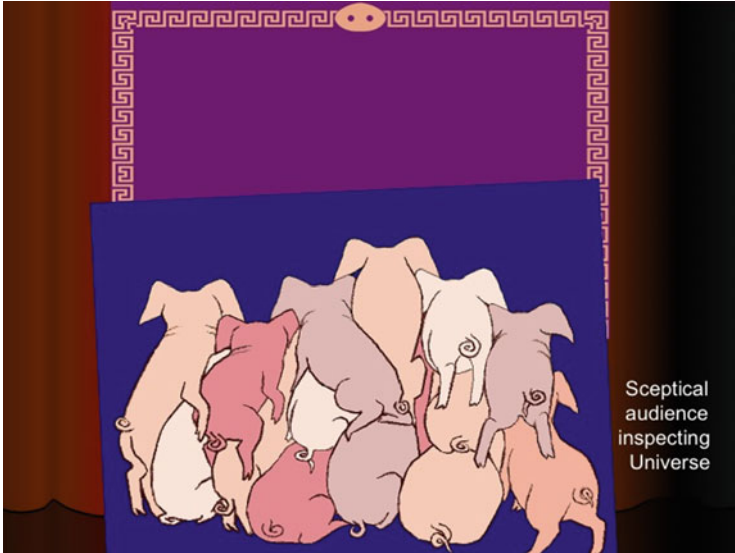


Fig. 5 Last slide of the Petri-Zuse collection, August 2007

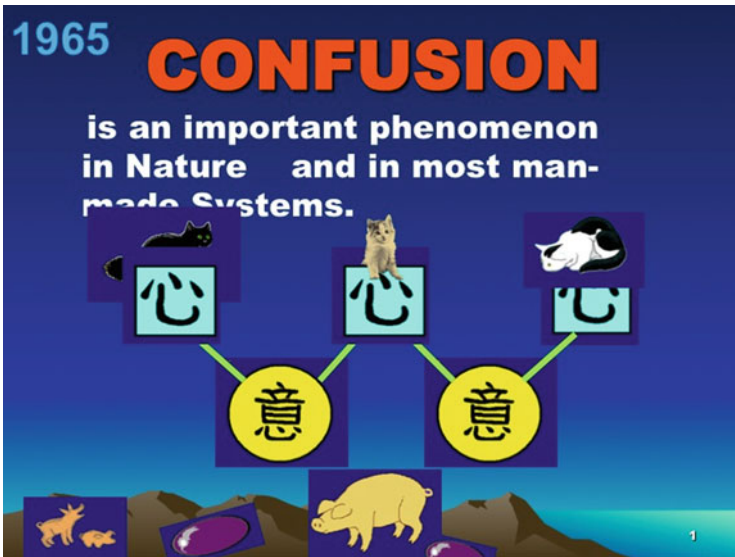


Fig. 6 Confusion slide in the file donated to Giorgio De Michelis, 2003



**Fig. 7** First slide of the collection donated to Giorgio De Michelis, 2003

to establish a warm relationship with his public, where his theories lost the typical aridity of scientific formal theories.

Consistently with this development, he began to create collections of slides as presents for friends and colleagues (see, e.g., Fig. 7): their contents were defined on the basis of the interests of the person to whom they were dedicated, and their presentation was always friendly and pleasant.

## References

1. C.A. Petri, W. Reisig, *Petri Nets* (2008), [http://www.scholarpedia.org/article/Petri\\_net](http://www.scholarpedia.org/article/Petri_net)
2. G. De Michelis, The contribution of Carl Adam Petri to our understanding of ‘computing’ (extended version), in *History and Philosophy of Computing*, ed. by F. Gadducci, M. Tavoanis, (Springer, Berlin, 2016), pp. 156–167
3. E. Smith, *Carl Adam Petri – Life and Science* (Springer, Berlin, 2015)

# Invention or Discovery?



Kees M. van Hee

Petri nets have had a great impact in science and engineering as the most successful formalism for modeling and analysis of processes with concurrency. In 1939, when Carl Adam Petri was only 13 years old, he already used the diagram technique for the description of chemical processes. So the first application was not in computer science. The formalism has been named after Petri and it is clear that he started a great scientific movement. Fifty-five years after Petri's seminal dissertation there are hundreds of researchers all over the world finding new properties of Petri nets and extending the formalism into new application domains. Thousands of engineers use Petri nets to design and analyze systems. There are dozens of software tools to support these engineering activities.

A purely philosophical question is: did Carl Adam *invent* Petri nets or did he *discover* them? The difference between these notions is clear: an invention is a creation by a person or a group of persons and a discovery is finding something that already existed. Instead of *creation* we also speak of *design*. First of all we have to define what we mean by 'Petri nets.' Automata were already known and so the concept of a system that, at discrete points in time, creates a state transition. The great new concept of Petri nets was the idea of *locality*: not one big state space where the whole state changes in one transition, but local states and local transitions, such that local transitions concurrently change only a part of the whole state. The locality of the state space is expressed by means of the notion of *places* and *tokens*. The locality of the transitions is expressed by the notion of the *transition*, which is in fact a local transition. So this concept is the basis of the formalism of Petri nets. Of course there is the notation or syntax as a bipartite graph and then there is the behavior of the system, e.g., expressed by the reachability graph, which can be seen

---

K. M. van Hee (✉)

Department of Mathematics and Computer Science, Eindhoven University of Technology,  
Eindhoven, The Netherlands

e-mail: [k.m.v.hee@tue.nl](mailto:k.m.v.hee@tue.nl)

as the semantics of the formalism. This is what Petri invented or discovered. He did not invent or discover the (infinite) set of all the *properties* of Petri nets! In fact his direct contribution to the large set of properties of Petri nets is rather modest, but he saw the big picture, inspired many followers, and asked them the right questions. It seems natural to consider the properties of Petri nets as part of the formalism. It is clear that the properties were not created but that they were discovered: they were there but we had to find them. So we have identified four elements in the Petri net formalism: (1) the concept, (2) the syntax, (3) the semantics, and (4) the properties. Which of these four elements were invented and which were discovered? The first three in any case were due to Petri himself.

If you let a group of people make a drawing of some object, you will never see two identical drawings. So creations seem to be unique. For discoveries this does not hold: when biologists are looking for a new specimen in the jungle they may find the same specimen in different places. The same holds also in physics: scientists discover elementary particles, they don't create them. We have already seen that the properties of Petri nets were discovered. The syntax or notation seems to be created. So the concept and the semantics are still to be classified as invention or discovery. It seems that they are closely entangled. So they are either both invented or both discovered.

In search of the answer I will sketch my own 'discovery' of Petri nets. As a mathematician skilled in operations research I worked for several years in industry and in 1984 I moved to computer science. My research topic was the development of methods for specification and modeling of business information systems. At that time database technology was coming up and an information system was considered a central database with lots of more or less independent applications around it. The starting point for the development of a new information system was the data model, which was in fact the specification of the database scheme. When the data model was established several application designers could start independently building their applications. Concurrency issues were not very important in that area. In industry, dataflow diagrams were the only tools to describe the information processing the business process and the information systems. They cannot express the order of processing and there was no natural way to integrate them with the data model. So I started a project to design a formalism that could model all aspects of an information system in an integrated way. A first attempt was a framework called SMARTIE [1]. Then my team developed another framework and a supporting software tool called exSpect [2]. (Wil van der Aalst was one of the team members.) When we showed it to our colleagues Willem Paul de Roever and Rob Gert in 1988 they said that it looked very much like the colored Petri nets developed by Kurt Jensen [3]. So we more or less *rediscovered* Petri nets. Later I remembered I had seen only one paper on Petri nets in the 1970s, but because at that time I was interested in modeling data-intensive systems I did not dig further into the field. So maybe this knowledge influenced the rediscovery in the subconscious.

But then we started studying the overwhelming literature on Petri nets and we joined the ITPN conference in 1989, where we met Carl Adam Petri in person for the first time in Bonn (Bad Godesberg). We completely adapted our work to the



Petri net formalism (e.g., in our diagrams we used circles for active components and rectangles for passive ones, so we switched them.) Since then we have applied our version of colored Petri nets to model all kinds of systems, not only information systems.

When the kick of being able to model whatever system we wanted was over in the 1990s, we became interested in the analysis of behavior. The possibilities for colored Petri nets are rather limited so we switched to classical Petri nets and their beautiful methods of analysis. Of course this provides only partial correctness, but it turned out to be very useful in practice.

In answering the question: was it an invention or a discovery?, it is interesting to look for similar developments. Conway's Game of Life is a formalism to describe self-reproducing systems (see [4]). It was designed in 1970 to answer the question of von Neumann whether such systems could exist. It seems widely accepted that Conway *created* the game. Also, most people say that, in 1936, Alan Turing *invented* the abstract machine that later was called after him.

In both cases the inventors were (probably) not aware of the tremendous number of interesting properties their invention had. This is a plea for considering it as a discovery, because one can discover a part of something, viz., only the part that is close to the surface while the deeper parts, such as the properties, still remain to be discovered. If you are still in favor of the invention option then you have to accept that the inventor has also created all the properties of the creation, but that they are still unknown and thus have to be discovered!

When Petri attended the Petri net conference in the Netherlands in 2003, he was honored with the prestigious title of Commander in the Order of the Netherlands Lion (Fig. 1). The Dutch governor who led the ceremony said in his speech that Petri's contribution to computer science was of the same order as the contribution of Crick, Watson, and Wilkins, who discovered DNA, to the life sciences. By the way, they received their Nobel Prize in 1962, in the same year Petri defended his Ph.D. Clearly Crick, Watson, and Wilkins did not create the structure of DNA, but they discovered it and made it visible and understandable.

After the Petri net conference of 2005 in Miami, Wil van der Aalst, Ron Lee, and I had dinner with Petri. And I asked him the question: "Did you invent the Petri nets or did you discover them?" It took him a while and then he answered: "I discovered them." This answer might have been expected because Petri was a modest man and putting himself in the role of a creator seemed to be too pretentious for him. But I am convinced that he meant what he said: the Petri nets were already there, they just appeared to him when he was thinking about concurrency. He did not see all the details and properties that were discovered by hundreds of researchers later on. But he had the big picture and I believe he saw things that still have to be discovered.

During his whole career he saw connections between Petri nets and physics, in particular quantum physics. And although I listened to several of his lectures (e.g., [5]) about these relationships I could not understand all of them. Two crucial points of Petri's are the conservation law of information and the idea that the elementary building blocks of physics are 'information flows' from which the (special) theory of relativity as well as quantum mechanics can be derived. Petri claims that Petri



**Fig. 1** From left to right: the author, Carl Adam Petri when he received the Dutch order of knighthood, and the governor of Noord Brabant, Frank Houben

net theory can be used to “Translate the main tenets of modern physics into their combinatorial form. In that form they are independent of scale, and relate to direct experience as well as to the sub-microscopic level of quantum foam” (see [5]).

Recently the Dutch physicist Erik Verlinde [6] discovered a new theory for the fundamentals of physics in which microscopic information units, the *qubits*, form the basis! (They are physically represented by strings and D-branes, with a size in the order of magnitude of Planck’s constant.) The conservation of information is the essence of his theory: so information cannot be created and can’t be destroyed. Entropy can be seen as the amount of information needed to describe the state of a system. Verlinde showed that the entropy of a black hole is the amount of information that has disappeared into it and is proportional to the surface of the black hole. A change in entropy will cause a force. Gravity may be considered to be caused by this phenomenon.

So it seems that Petri discovered more than we know yet! But the details still have to be discovered. If the conservation law of information claimed by Petri and Verlinde holds, then so-called human creations are in fact discoveries. This answers the question and is consistent with the answer of Petri.

## References

1. J.L.G. Dietz, G.J. Houben, K.M. van Hee, Modeling of discrete dynamic systems: framework and examples. *Inf. Syst.* **14**(4) (1989)
2. W. van der Aalst, P.J.N. de Crown, R. Goverde, K.M. van Hee, W. Hofman, H. Reijers, R. van der Toorn, ExSpect 6.4: an executable specification tool for hierarchical colored Petri nets, in *ICATPN 200*, ed. by M. Nielsen, D. Simpson. LNCS, vol. 1825 (Springer, Berlin, 2000), pp. 455–464, [www.exspect.com](http://www.exspect.com)
3. K. Jensen, Coloured Petri nets, in *Petri Nets Central Models and Their Properties*, ed. by W. Brauer, W. Reisig, G. Rozenberg. LNCS, vol. 254 (Springer, Berlin, 1987), pp. 248–299
4. M. Gardner, Mathematical Games – The fantastic combinations of John Conway’s new solitaire game “life”. *Sci. Am.* **223**, 120–123 (1970)
5. C.A. Petri, On the physical basics of information flow, results obtained in cooperation with Konrad Zuse, presented by Rüdiger Valk, in *ICATPN 2008*, ed. by K.M. van Hee, R. Valk. LNCS, vol. 5062 (Springer, Berlin, 2008), p. 12
6. E. Verlinde, Emergent gravity and dark universe, ArXiv:1611.02269v2 (2016)

# Petri's Understanding of Nets



Dirk Fahland

*My first personal exchange with Carl Adam Petri has had a long lasting impact on my work as a researcher and teacher. At that time I was working as a Ph.D. student in the group of Wolfgang Reisig at the Humboldt-Universität zu Berlin when Petri was visiting us. The group spent the afternoon listening to and discussing Petri's latest research. As we approached the evening and came to an end, there was still some time left before we would leave for dinner and Wolfgang Reisig suggested that I should ask Carl Adam Petri to explain the idea of Petri nets to me. This would be something he considered a truly fascinating angle best told by Petri himself. So, I sat down with Carl Adam Petri and asked him this question and what follows is the best attempt I can give to recollect his explanation.*

When we think about nets, we should not start thinking about places or transitions, but we should start by thinking about *tokens*, because tokens are the only elements in a net that have an interpretation in the physical world. Think of a token as any elementary entity of interest: at its lowest level, it could represent an elementary physical particle or atom. The entity carries its own *local properties*, which we may summarize as the entity being in some condition. Figure 1 (top left) abstractly represents an entity  $X$  in some condition  $A$  as a labeled token. I cannot remember whether we also discussed other interpretations of a token in our exchange, or at least these interpretations did not stick. Nevertheless, my personal opinion is that in the realm of information processing, Petri's fundamental interpretation of tokens goes as far as also including elementary units of information that are not necessarily bound to a particular physical representation.

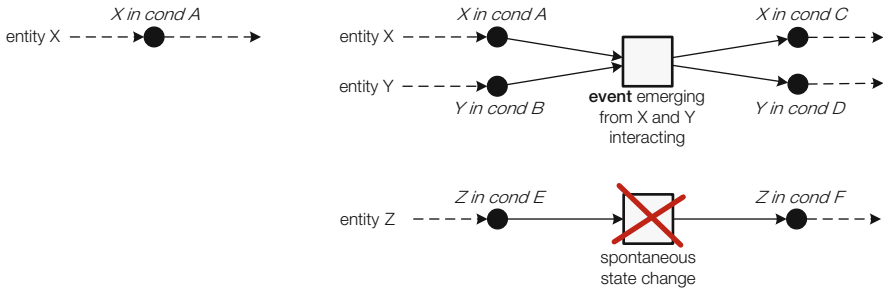
Now that we know what a token is, we can think of the universe as a vast collection of net tokens. Each entity with its own local properties can be described

---

D. Fahland (✉)

Department of Mathematics and Computer Science, Eindhoven University of Technology,  
Eindhoven, The Netherlands

e-mail: [d.fahland@tue.nl](mailto:d.fahland@tue.nl)

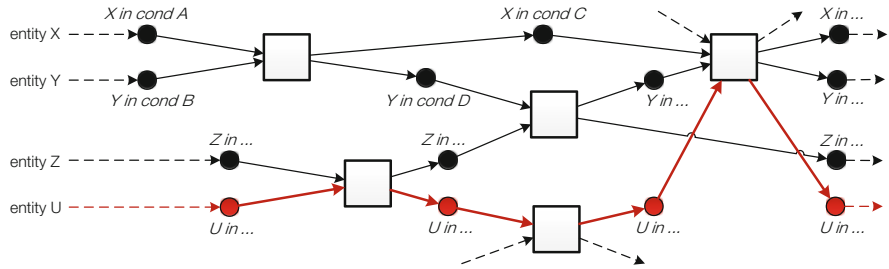


**Fig. 1** A token representing an entity in a particular condition (top left). An entity may change its condition by interaction with another entity in an event (top right). An entity cannot change its condition spontaneously without interaction with another entity (bottom right)

by a token in a particular condition. Of course an entity can change its local properties but—and this is the important point Petri raised to me—it can never change its local properties spontaneously on its own. An entity changes its properties only through an interaction with another entity. In net theory, we express a state change as a *local event*. A local event takes two (or more) tokens, each in their own pre-condition to the event, as input and results in two (or more) tokens, which are now in the post-conditions of the event. An event is purely local to the tokens that participated in the event, and we should interpret the event as something that emerged out of the interaction of the two tokens (or entities). There is no hidden or general mechanism behind this event, but the event occurred *because* these two entities interacted. Figure 1 (top right) illustrates an event involving entities *X* and *Y* where *X* changes from condition *A* to *C* and *Y* changes from condition *B* to *D*.

An event involving only a single entity as illustrated in Fig. 1 (bottom right) would describe a spontaneous change, which cannot occur. We discussed this last argument a bit longer. When taking seriously Petri’s interpretation of tokens as physical entities, then this model would not allow a description of the seemingly spontaneous events observed in particle physics. Petri’s opinion here was that a model that requires spontaneous state changes to describe a particular dynamics is incomplete as it lacks the causal explanation for the state change. He argued that we might not know the explanation yet, but the model ultimately has to be extended to be complete.

This idea of an event transforming properties of tokens entails that a token has a form of identity which it retains throughout the event even though its properties may change. After an event has occurred, each entity is now free to participate in another event with the same or another entity, leading to the occurrence of another event. In this way, each entity gets involved in a series of local events with other entities as it “*travels along its own path through the universe.*” The properties of an entity at any point in time can be explained in terms of the history of its interactions with other entities as it has been traveling through the universe. As each entity follows its own path and interacts with other entities, their paths get intertwined



**Fig. 2** A distributed run describing how entities *X*, *Y*, *Z*, and *U* change states in local events. Each entity follows its own path of local events (highlighted for *U*). The local events intertwine the paths of the entities involved, and the paths put the events into, a partial order

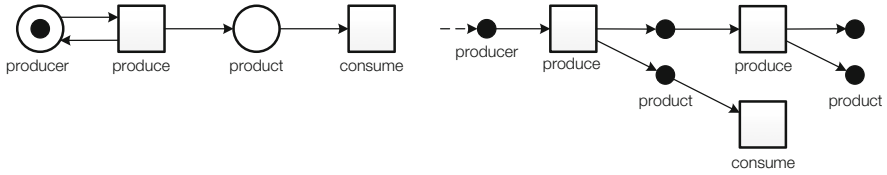
in their local interactions, creating a complex “fabric” of entity interactions. In net theory, we describe this behavior in terms of a *distributed run* [1] as illustrated in Fig. 2. Each entity *X*, *Y*, *Z*, and *U* follows its own path (highlighted for *U*), interacting at different moments with different entities in local events; events in the distributed run are *partially ordered* as, for instance, the first event involving *X* and *Y* is independent of the first event involving *Z* and *U*.

Put differently, according to net theory, the dynamics of the universe can be described as a large distributed run of its atoms. A (Petri) *net* is then merely an attempt to describe a small part of this fabric of entity interactions in a finite model. The tokens in a net are abstractions of the entities they describe, each *place* describes a condition an entity can be in (within the scope of the model), and each *transition* describes the kinds of events that the entities can be involved in (within the scope of the model). As we go from tokens, events, and runs to nets, we abstract away details—ideally to the level of detail we are interested in studying.

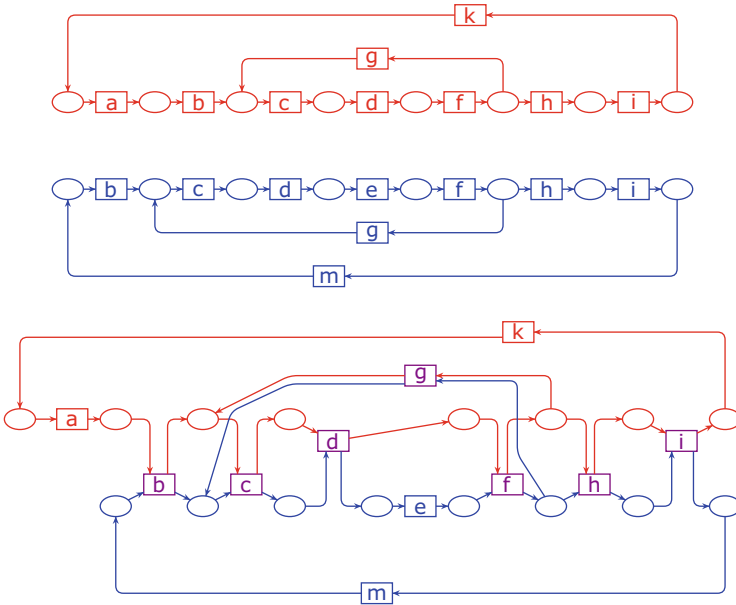
A non-deterministic choice between two different transitions in a net describes different evolutions of the same kind of entities. We did not discuss on this occasion how Petri interpreted non-determinism in a net: whether it was necessary (as the universe would be non-deterministic) or whether it indicated inherent incompleteness (as the universe would be deterministic).

There are a number of simple but important conclusions that can be derived from this view. No event can produce an entity out of nothing and no event can eliminate an entity into nothing. In terms of net theory, each event always has a token in its pre-condition and a token in its post-condition (each transition has a pre-place and a post-place). Further, no event involves just a single token. If we encounter such an event such as the one shown in Fig. 1 (bottom right), then our model is incomplete and must be extended to also consider other entities involved in the event to explain its occurrence.

Interestingly, classical elementary system nets allow the creation of models such as the one in Fig. 3 that are at odds with the above interpretation of tokens and events as changes of entities in a world governed by the constraints of physics. A net having a transition with a single pre-place, such as transitions *produce* and



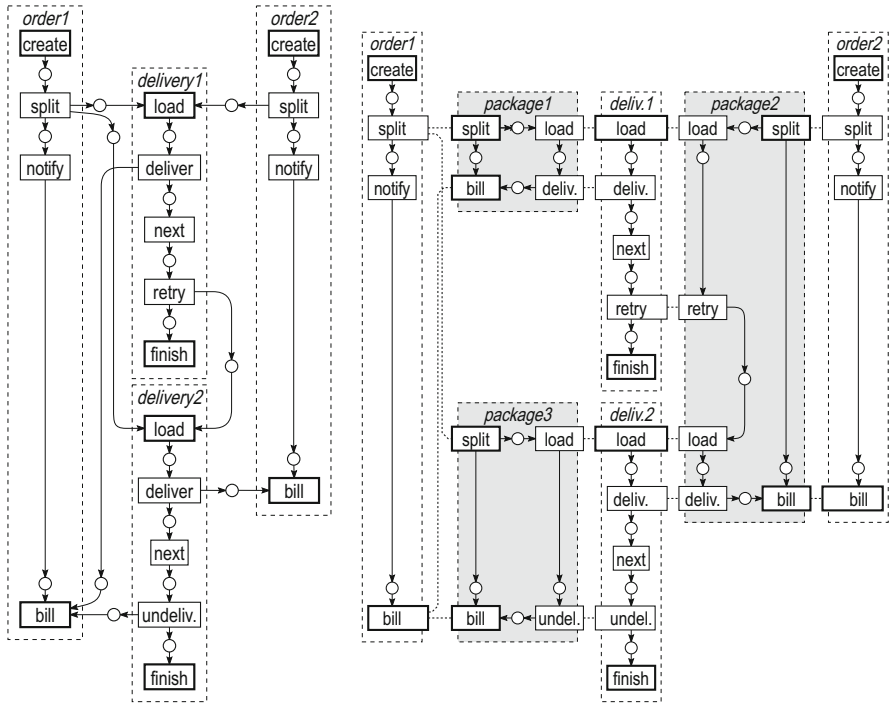
**Fig. 3** An elementary system net (left) and a prefix of its distributed run (right) that describe the creation of new entities (new tokens on *product*), and the spontaneous consumption of entities (through transition *consume*)



**Fig. 4** Example for introducing modeling concurrency with Petri nets to undergraduate students. The two separate nets (top and middle) describe the behavior of a guest and a waiter in a restaurant. Composing the nets on joint transitions (bottom) intertwines the interaction of the two entities in terms of their intertwined behavior

*consume*, is an incomplete model. Also transitions with a different number of pre- and post-places are incomplete. In the example, the occurrence of *produce* describes the spontaneous creation of a new *product* (a physical entity) without the necessary resources; the occurrence of *consume* describes the spontaneous consumption of a *product* into nothing. This behavior cannot occur in reality if conservation laws such as conservation of energy and matter hold. Such structures arise in our models typically to abstract from a larger, unbounded environment that we do not want to describe in detail, hence they are incomplete models on purpose.

Petri's interpretation has proven useful to me when teaching students to model with Petri nets to create models of complex dynamics. Figure 4 shows an example



**Fig. 5** Distributed run of the interaction of two instances of an *order* with two instances of a *delivery* (left) and the same behavior with the inclusion of a *package* as a third entity

of the composition of the behavior of two independent entities (a guest and a waiter at a restaurant) into a net that describes the joint behavior of the two entities, by merging joint transitions. This example helps students in an under-graduate course to relate the concept of a distributed marking to a real-world interpretation of a system involving multiple independent entities.

The distributed run in Fig. 5 describes rather complex dynamics that cannot be captured with established Petri net models [2], yet it adheres to the principles outlined above. The run describes an order-handling process at an online shop: two *orders* are handled by two *delivery* tours, where *order1* results in two packages (each handled by a different delivery tour), and the single package of *order2* gets handed over from *delivery1* to *delivery2*. The run on the left describes each event associated with a specific entity, but leaves the conditions between orders and deliveries without any entity. Restructuring the run by explicitly including three *package* entities allows events to be described as the interaction of two or more entities, is shown in Fig. 5 (right). This run arguably gives a more comprehensive account of the dynamics shown, and also reveals that net theory requires different composition concepts to describe this behavior. For instance, *order1* interacts with



*package1* and *package3* on *split* whereas *order2* interacts only with *package2* on *split*.

This last example is perhaps a useful illustration of how Petri's rather basic interpretation of principles still reveals in a structured way interesting aspects of distributed behavior that are easily overlooked in classical net theory.

## References

1. U. Goltz, W. Reisig, Processes of place/transition-nets, in *ICALP*, ed. by J. Diaz. Lecture Notes in Computer Science, vol. 154 (Springer, Berlin, 1983), pp. 264–277
2. D. Fahland, M. de Leoni, B.F. van Dongen, W.M.P. van der Aalst, Many-to-many: some observations on interactions in artifact choreographies, in *3rd Central European Workshop on Services and Their Composition, Services und ihre Komposition, ZEUS 2011* vol. 705 (2011), pp. 9–15. <http://ceur-ws.org/Vol-705/paper1.pdf>

# On the Two Worlds of Carl Adam Petri's Nets



Rüdiger Valk

With his work Carl Adam Petri initiated a scientific world that includes by now a huge number of publications. The area covers multiple aspects, but from a historical point of view there have been two main streams or worlds of thinking. This article describes how I experienced both of these worlds of working and thinking in over 40 years of my academic life.

During the last year of my academic studies toward a degree in mathematics in 1971 at the University of Bonn I was employed as a student assistant at a major research institution near Bonn, the *Gesellschaft für Mathematik und Datenverarbeitung (GMD)*, where Petri was the director of the *Institute for Information Systems*. As a student I saw Petri only from a distance, but I attended the Ph.D. Defense of Hartman Genrich, where he emphasized Petri nets as the description model of the future in contrast to the old-fashioned “steam engine” of automata theory (a field in which I was writing my diploma thesis). This experience motivated me to change my research interests to Petri nets after having finished my Ph.D. thesis on topological automata in 1974 at the University of Hamburg.

Following my education at the University of Bonn (and not at the GMD) my research on Petri nets started in the context of finite automata and formal languages, and my first major publication in this field was on “Regular Petri Net Languages” [22]. Such a publication would not have been possible within Petri’s research group. Petri always emphasized the point that modeling should be in direct accordance with physical laws and real-world observations. As a consequence, for example, multiple tokens were not used, and side conditions or token capacities for example were not considered. This was motivated by the requirement of “good modeling,”

---

R. Valk (✉)

Fakultät für Mathematik, Informatik und Naturwissenschaften, Universität Hamburg, Hamburg, Germany

e-mail: [valk@informatik.uni-hamburg.de](mailto:valk@informatik.uni-hamburg.de); <https://www.inf.uni-hamburg.de/inst/ab/art/people/valk.html>

but examples of such modeling had rarely been published at that time. The entire universe was conceived as a net, while we were working with small cutouts. Here I cannot describe this way of thinking in detail, but will subsume it under the heading of “General Net Theory (GNT).” At conferences, the GNT point of view has been advocated by the members of the Petri school in such a way that P. S. Thiagarajan was reported to call it the “Petri church.”

This was in contrast to the way Petri nets were treated outside the GMD group, where a model or a way of modeling was not a priori considered to be “good” or “bad,” instead, any formalism was worth studying. The value of such formalisms was determined by the way they fitted into the general framework of theoretical informatics or by their usefulness in applications. Here, I will call that world the “World of the Free Petri Net (FPN).”<sup>1</sup> These two worlds, GNT and FPN, were even distinguishable by the way nets were depicted: transitions were represented by simple bars in FPN, but by small boxes in GNT.

In 1976 I spent a research visit at the University Paris 6, invited by Claude Girault. There I met a young generation of Petri net researchers who became well known in the research community some years later (G. Roucairol, G. Memmi, G. Berthelot, G. Vidal-Naquet, and others). I remember the following scene describing the state of Petri net research at that time. As my first foreign language in high school had been French, I wanted to give my introductory talk on Petri nets in French. At that time no publication in French on the topic was known (at least to me). Therefore I had to learn the key notions (such as “firing a transition”) from my French colleagues just before the talk. Vice versa, papers on place invariants (e.g., by K. Lautenbach and H. Genrich) were available only in German. Therefore these French colleagues were strongly interested in translating them into the French language. During a later visit Claude Girault made the proposal to create a workshop organized by France and Germany. Following this proposal we organized the “First European Workshop on Application and Theory of Petri Nets” in 1980 [3] in Strasbourg. It became a series, later under the name of a conference.<sup>2</sup>

In 1978 I had a contribution at a conference in Zakopane, Poland, where I presented an extension of ordinary Petri nets that I called “Self-modifying Nets” [17]. But although (or because?) the model was inspired by a paper of H.E. Fuss from GMD, there was a critical comment by Petri himself saying that my definitions were not formally correct. At that time, I found this criticism rather strange since my argumentation was strictly formal, contrary to Petri’s papers, which did not follow the mathematical style with definitions, theorems, and proofs. Today, I think that Petri’s criticism came from his convictions with respect to modeling principles; e.g., in this case, graphical representations of Self-modifying Nets violated the principle of locality. My interest in using this formalism was to investigate the

---

<sup>1</sup>At that time GNT people called it “the token game approach.”

<sup>2</sup>International Conference on Applications and Theory of Petri Nets and Concurrency. It was the proposal of Claude Girault to mention “Applications” first, a tradition that still continues in the title of the conference. For the history of this conference see [15].

Turing completeness of different extensions of Petri nets, as Self-modifying Nets contain inhibitor nets, reset nets, etc. as subclasses.<sup>3</sup>

In the year 1979 Wilfried Brauer in cooperation with Carl Adam Petri and Brian Randell organized the Advanced Course "Net Theory and Applications" at the University of Hamburg [1]. At that time, due to external constraints, the two worlds of GNT and FPN were obliged not only to meet, but to coordinate their concepts and definitions. This had already happened at a meeting at the GMD some time before the Advanced Course started. The representatives of both worlds were (symbolically) sitting face to face on the two sides of a large table and had intense discussions about the topics to be presented, about the lectures to be held and the definitions to be used. For example, an agreement was reached to call a net with multiple tokens on places and arc weights a *Place Transition Net*, whereas the favorite model of the GNT world was called a *Condition Event Net*. The conference itself was a highlight in the history of Petri net development, and I had the pleasure to be the chairman of Petri's lecture. Researchers and students from all over the world attended,<sup>4</sup> and when walking downtown in a guided tour, participants were amused to see a physical (St.) Petri church in red-brick Gothic style.

The power of modeling with Petri nets came to my attention more strongly than ever before when, with Eike Jessen as co-author, I developed a new course bringing together topics from classical books on operating systems, concurrency, and queueing theory. Here I was able to design convincing net models for many classical examples found in those books. The course was later published by Springer [6]. Much of the material was also used in the five chapters I contributed to a book that was the outcome of a European project [4]. In the context of the first of these two books a certain modeling problem appeared, namely that tasks or programs are moving and changing objects in a (queueing) system. This had been the motivation for the more elaborate Petri net formalism of Object Systems first published in 1991 [18] and described in more detail in [19]. Later on, in a personal letter Petri appreciated this development. An important result of this work was the development of the Petri net tool RENEW by Kummer [9], Daniel Moldt, and others, which is able to model place/transition nets as well as coloured nets, object nets, and object-oriented nets, all addressed by the reference nets formalism. The work with this tool is still important and has produced complex multi-agent systems and support for collaborative software engineering [14].

In 1988 Carl Adam Petri became a professor at the University of Hamburg. The proposal for this appointment was made by Hermann Flessner, who had been in contact with Konrad Zuse and Petri some years before at Hannover. As the responsible professor I had to initiate the necessary decisions at the Institute of Informatics, the University and the Government [2]. As a consequence, for many years Petri regularly gave lectures and guided seminars and research groups on

---

<sup>3</sup>Note that Pr/T-nets or Colored Nets were not known at that time. In later years Self-modifying Nets were intensively studied by Philippe Darondeau and his colleagues.

<sup>4</sup>Among whom was G. Rozenberg, who wanted to be introduced to the field.



**Fig. 1** Hold up high the torch of political correctness

topics of his choice. This way we learned his way of thinking and obtained a better understanding of the results of his research, many of which were (and still are) not available in the literature. Notorious were the long sessions until late at night with discussions on the topics of the day's lecture given by Carl Adam, ignoring lunch and dinner breaks in a room full of smoke (see the picture Fig. 1, designed by Carl Adam himself). He loved these meetings with students very much as he didn't have that kind of environment for discussions at the GMD institute.

Among the student participants the core group included Uwe Fenske, Stefan Haar, Peter Langner, Hartmut Müller, and Mark-Oliver Stehr<sup>5</sup> who were regularly joined by several student and research staff members of the university and, in particular, by our theory group. While attempting to absorb as much as possible of Petri's ideas, they especially dealt with concurrency theory and cycloids. On the occasion of this article, from their many notes Uwe Fenske gave me the following two of Petri's statements (originally in German): May 25, 1990: "Electron orbitals are nothing else than four dimensional cycloids" and May 30, 1990: "Parallel computing via informatics achieves more than sequential computing, which is still denied by theoretical computer scientists today."

Petri's approach encouraged them to apply his ideas to other disciplines, "such as linguistics. Peter Langner developed so-called "ChronoNarratio-Graphs," which are

<sup>5</sup>As they were always hanging on Petri's every word (but not knowing the bon mot of Thiagarajan's "Petri church") they were called Petri's Disciples ("Petri-Jünger").

Petri nets expressing the causality and independence of courses of action in literary texts, as for instance in Tolstoy's novel "Anna Karenina" [11].

Inspired by long meetings and discussions with C. A. Petri, Mark-Oliver Stehr developed and studied a more general theory of partial cyclic orders that includes cycloids as a special case [16]. He is now a senior scientist at SRI International, Menlo Park, California, and he recently sent me a very enthusiastic letter from Petri, dated March 23, 1997, on Kummer and Stehr's paper on Petri's axioms for concurrency [10], which was also a result of Petri's lectures in Hamburg and Stehr's investigation of concurrency theory under the guidance of Jozef Gruska. In this letter Petri emphasized the importance of an axiomatic system based on causality and on concurrency relations, the general value of measurement methods for "Rough Sets," and the notion of coherence in general sciences. When reading the letter now, Stehr said he was surprised that Petri explicitly mentioned Biology, as he himself is working today with biologists at SRI on modeling biological systems. Stehr expects that Petri's approach to causality, already very closely related to his own current work, might become even more relevant in view of the exponentially increasing flood of experimental/observational data enabled by the latest technologies.

With my colleagues Daniel Moldt and Rolf v. Lüde, a sociologist, I directed some projects to combine the methods from informatics and sociology within a new discipline, called *socionics* [8]. The goal of these projects was to model theories of the famous sociological schools of Norbert Elias, Pierre Bourdieu, Heinrich Popitz, Niklas Luhmann, and others in such a way that they can be used for the design of computer systems involved in human or quasi-human societies or multi-agent systems. In these modelings, Petri nets have been successfully used. An important idea was to structure nets according to the modeled system. In the context of sociology especially the agent-oriented approach *MULAN* proved to be very suitable [7]. For one of the books on *socionics* [23] that documented these results, C. A. Petri wrote a preface, dated December 22, 2002. In this preface he argued that the considerably increased potentials of communication are also influencing sociology. But these effects should not be forced into the "corset" of traditional mathematics or informatics. It is not the precision known from mathematics or physics that is appropriate in such a model, but rather the simplicity and refutation-definiteness to describe the margins (he wrote "Spielräume") of actors. These margins were limited by causal dependencies or independencies that were more adequate for description than totally ordered time scales of events. Therefore, Petri welcomed the given approach to model parts of sociologic theories by formal theories using net theory.

In 2007 I was asked to chair the Petri net conference 2008 in Xian, China together with Kees van Hee. In a letter dated May 20, 2007 Carl Adam wrote: "I would enjoy very much to fly to China, in particular with you as chairman. However, my age as well as my health do not allow it. But I still think about producing a PPT presentation which you can present." After some discussions, Kurt Jensen informed me that in order to have a different and shorter contribution than Petri's presentation at the Miami conference, the steering committee had agreed that I should make such a presentation on the basis of Petri's proposals as an invited lecture. Following this agreement I decided to make my own presentation, partly using Petri's slides,

and planned to only present material I could clearly understand. This decision was accepted by Carl Adam, but it had the consequence that for a whole year nearly every day I had e-mail exchanges with him to discuss things that were not clear to me. Carl Adam appreciated this procedure since it gave him the opportunity to discuss his ideas, to remove errors, and to find new representations.

The starting point of Petri's slides was his 3-year collaboration with Konrad Zuse (1910–1995) on the idea of a Computing Universe [12]. They agreed that some of the main tenets of newer physics would have to be expressed, at least those of quantum mechanics and of special relativity. Discussing which tenets should be tackled, Zuse proposed “Those which can be understood by an engineer.” But many years passed before the deterministic approach of the physicist Gerard 't Hooft (Nobel Prize in 1999) made possible a complete elaboration of the originally conceived ideas.<sup>6</sup>

In the slides Petri follows the principles of combinatorial modeling, which is a proper synthesis of continuous and discrete modeling. Petri gives a reformulation of fundamental physical laws that allows a combinatorial representation. Further important notions are those of slowness, measurement, uncertainty of counting, determinism, and cycloids. Of particular importance for Petri was his discovery that fundamental gates of Boolean circuits, such as XOR-transfer, majority-transfer, or Quine transfer, are topologically equivalent to some of his cycloids. In a recent mail, Mark-Oliver Stehr told me that Petri's Quine transfer, also known as the Fredkin gate, has now been experimentally realized in quantum computing for the first time by researchers of the Griffith University and the University of Queensland [13].<sup>7</sup> In the abovementioned personal communication Petri wrote to me: “It is one of my most important concerns to confront the stochastic way of thinking with the combinatorial one. It seems to me that combinatorial results are irrefutable whereas the stochastic view will finally lead to the well-known problems in quantum mechanics.” When my slides<sup>8</sup> were finished, I asked him in what ways the current presentation was different from the one at the Miami conference. Carl Adam answered: “In Florida, with six hours I had plenty of time. Furthermore, the successful presentation there was a festive experience for me.”

Looking back, the abovementioned contacts with Petri show only a very small part of his entire work. Nevertheless, they document his universal mind, ranging from very detailed physical knowledge through mathematical skills and to formal reasoning about communication disciplines. Coming back to my introduction, two

---

<sup>6</sup>Published in a 2002 paper entitled “Determinism beneath Quantum Mechanics”.

<sup>7</sup>We cite from [5, p. 51]: “The CN gate, the Toffoli gate and the Fredkin gate were first presented by C. A. Petri in 1965, but their publication in 1967, in German and in a not too widespread proceedings, went apparently unnoticed by most of those working on reversible computing. However, in view of the above fact, it would perhaps be historically more proper to talk about Petri-Toffoli and Petri-Fredkin gates. Petri has also shown the universality of these two gates for classical reversible computing.”

<sup>8</sup>My slides can be downloaded from [21] in their original keynote-format, but also as ppt- or pdf-documents.

worlds in the field that he established are highlighted in this article: the ordinary one, where most Petri net researchers are working today, modeling systems and proving properties such as liveness, invariants, subclasses, and so on, which is a successor of both the GNT and FPN worlds; and the second world in which Petri seemingly was most interested, namely the foundations of systems compliant with the physical laws of nature, which undoubtedly is GNT.

And there's me, having started in the FPN world in 1974 and finding myself today partly in the second one, as a wanderer between the worlds, still doing research on Petri's cycloids [20].

## References

1. W. Brauer (ed.), *Net Theory and Applications*. Lecture Notes in Computer Science, vol. 84 (Springer, Berlin, 1980)
2. Budelmann. Document granting the title of professor (1988). <http://www.informatik.uni-hamburg.de/TGI/mitarbeiter/profs/petri/Petri-Prof.pdf>
3. C. Girault, W. Reisig (eds.), *Application and Theory of Petri Nets, Selected Papers from the First and the Second European Workshops 1980 and 1981*. Lecture Notes in Computer Science, vol. 52 (Springer, Berlin, 1982)
4. C. Girault, R. Valk (eds.), *Petri Nets for System Engineering – A Guide to Modeling, Verification, and Applications* (Springer, Berlin, 2003)
5. J. Gruska, *Quantum Computing*. Advanced Topics in Computer Science Series (McGraw-Hill, New York, 1999)
6. E. Jessen, R. Valk, *Rechensysteme, Grundlagen der Modellbildung* (Springer, Berlin, 1987)
7. M. Köhler, D. Moldt, H. Rölke, Modelling the structure and behaviour of Petri net agents, in *Proceedings of the 22nd Conference on Application and Theory of Petri Nets 2001*, ed. by J.M. Colomand, M. Koutny. Lecture Notes in Computer Science, vol. 2075 (Springer, Berlin, 2001), pp. 224–241
8. M. Köhler, R. Langer, R. von Lüde, D. Moldt, H. Rölke, R. Valk, Socionic multi-agent systems based on reflexive Petri nets and theories of social self-organisation. *J. Artif. Soc. Soc. Simul.* **10**(1), 1–3 (2007)
9. O. Kummer, *Referenznetze* (Logos Verlag, Berlin, 2002)
10. O. Kummer, M.-O. Stehr, Petri's axioms of concurrency – a selection of recent results, in *Proceedings of the 18th International Conference on Application and Theory of Petri Nets, Toulouse*. Lecture Notes in Computer Science, vol. 1248 (Springer, Berlin, 1997), pp. 195–214
11. P. Langner, D. Marszk, ChronoNarratio-Graphen: Ein Modell chronologischer Beziehungen in Erzähltexten. *Linguistische Berichte* **147**, 409–435 (1993)
12. C.A. Petri, On the physical basics of information flow – results obtained in cooperation with Konrad Zuse, in *Applications and Theory of Petri Nets*, ed. by K.M. van Hee, R. Valk. Lecture Notes in Computer Science, vol. 5062 (Springer, Berlin, 2008), p. 12
13. Phys-Org. Physicists demonstrate a quantum Fredkin gate. <https://phys.org/news/2016-03-physicists-quantum-fredkin-gate.html>
14. D. Schmitz, D. Moldt, L. Cabac, D. Mosteller, M. Haustermann, Utilizing Petri nets for teaching in practical courses on collaborative software engineering, in *16th International Conference on Application of Concurrency to System Design, ACS D, Torun* (IEEE Computer Society, Los Alamitos, 2016), pp. 74–83
15. M. Silva, Half a century after Carl Adam Petri's Ph.D. thesis: a perspective on the field. *Annu. Rev. Control* **37**(2), 191–219 (2013)



16. M.-O. Stehr, Thinking in cycles, in *Application and Theory of Petri Nets 1998, 19th International Conference, Lisbon, Proceedings*, ed. by J. Desel, M. Silva. Lecture Notes in Computer Science, vol. 1420 (Springer, Berlin, 1998), pp. 205–225
17. R. Valk, On the computational power of extended Petri nets, in *Mathematical Foundations of Computer Science 1978*, ed. by G. Goos, J. Hartmanis. Lecture Notes in Computer Science, vol. 64 (Springer, Berlin, 1978), pp. 526–535
18. R. Valk, Modelling concurrency by Task/Flow-EN-systems, in *Proceedings 3rd Workshop on Concurrency and Compositionality*. GMD-Studien, vol. 191 (Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, Bonn, 1991)
19. R. Valk, Petri nets as token objects-an introduction to elementary object nets, in *Application and Theory of Petri Nets 1998, 19th International Conference, Lisbon, Proceedings*, ed. by J. Desel, M. Silva. Lecture Notes in Computer Science, vol. 1420 (Springer, Berlin, 1998), pp. 1–25
20. R. Valk, On the structure of cycloids introduced by Carl Adam Petri, in *Application and Theory of Petri Nets and Concurrency*. Lecture Notes in Computer Science, vol. 10877 (Springer, Berlin, 2018), pp. 294–314
21. R. Valk, C.A. Petri, On the physical basics of information flow - results obtained in cooperation with Konrad Zuse. <http://www.informatik.uni-hamburg.de/TGI/mitarbeiter/profs/petri.html>
22. R. Valk, G. Vidal-Naquet, Petri nets and regular languages. *J. Comput. Syst. Sci.* **23**(3), 299–325 (1981)
23. R. von Lüde, D. Moldt, R. Valk, *Sozionik. Modellierung soziologischer Theorie*. Wirtschaft – Arbeit – Technik series (LIT-Verlag, Münster, 2003)

# Petri Nets: The Next 50 Years—An Invitation and Interpretative Translation



Heinz W. Schmidt

## 1 Introduction

Carl Adam Petri wrote his famous thesis over 50 years ago. In another 50 years a future generation of scientists and practitioners will possibly look back on a hundred years of Petri nets. Or will they? What will they consider to be the historic landmarks in their writing on Petri nets and the pioneer Petri himself? Will these landmarks be the same the community has read and written about for the last 50 years? Will net theory have merged with, and be subsumed under, the other mathematical theories on which it is based? Or will it continue to be a bridge between theory and applications for engineered systems and models of cyberphysical reality—models in which energy and information are treated on a par, and time and space are intertwined in networks of fundamentally asynchronous but coordinated and parallel change of vast numbers of parts.

This chapter is an invitation to join this endeavour, and contribute to this 100-year history. For a broad discipline like Petri nets, this will be the work of many. We re-emphasise selected elements of Petri's roadmap as expressed repeatedly in the writing of the computer pioneer, or as recollected from his seminar presentations and communications.

The focus is on providing an interpretation that is accessible to other sciences, both natural and social, and to practitioners. Along the way we provide some scattered cultural and personal context of influences at the time, moving back and forth through Petri's life, times and work, and also attempting to capture the influence Petri, his institute and Bonn had on the author and a group of people around him.

---

H. W. Schmidt (✉)

School of Science (Computer Science & Software Engineering), RMIT University, Melbourne, VIC, Australia

e-mail: [heinz.schmidt@rmit.edu.au](mailto:heinz.schmidt@rmit.edu.au)

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,  
[https://doi.org/10.1007/978-3-319-96154-5\\_7](https://doi.org/10.1007/978-3-319-96154-5_7)

45

However, first we look back at the time of Petri's thesis project and the major principles and research streams over the past 50 years.

## 1.1 Past

Petri nets were introduced by the applied mathematician and later computer scientist Carl Adam Petri over 50 years ago. Petri obtained his Ph.D. degree in 1962 at the Technical University of Darmstadt, in the Faculty of Mathematics and Physics, where his first advisor, Prof. Walther, lectured. At that time, Petri himself worked as the assistant of Prof. Unger, the chair of the Institute for Applied Mathematics at the University of Bonn. Petri had already been the assistant of Unger, when the latter still chaired the Institute for Practical Mathematics and Projective Geometry at the University of Hannover. Prof Unger was also Petri's second advisor and de facto his main advisor (see Fig. 3).

In his thesis [10],<sup>1</sup> Petri introduced nets to develop a physically realistic notion of Turing machines with a number of classical physics assumptions dropped, both computationally and in the physical realisation proposed. Petri argued from the start that under his assumptions, synchronisation based on a global clock (or dense real time) led a model with truly concurrent evolution of physically distributed objects to inconsistency with relativity and quantum mechanical theory, which real, parallel communication devices and protocols ultimately need to build on.

Petri's thesis introduced different forms of nets (such as 'k-form nets') that were already much more general and non-classical than the place-transition nets that have become synonymous with Petri nets in worldwide use today. Still today, large numbers of future scientists, engineers and practitioners leave our universities having learned only fleetingly, if at all, about general net theory, modern physics, Gödel's proof (of incompleteness) and any number of related philosophical, mathematical and information science issues that defined the lay of the land for Petri, and which still plague scientists and practitioners designing and building real physical and socio-technical systems of any scale with true parallelism. Computer science has remained largely classical. While the modern paradigms have been widely accepted,<sup>2</sup> the ideas in themselves do not form new theory, nor shake and break the skins of the old ones. Those old paradigms stick, like the proverbial old wineskins that we fill with our new wines.

Petri's visionary ideas were viewed as eccentric initially. As documented in his thesis, Petri's interest has always been the pragmatic aspect, including the guidance provided by experimental physics or the needs of practicing engineers dealing with highly parallel systems at the hardware, software or socio-technical level.

---

<sup>1</sup>Nearly 4000 citations (Google Scholar).

<sup>2</sup>Albeit sometimes only grudgingly: see the experimental results on teleportation, only a couple of years ago, and reactions in the literature confirming the misnamed 'spooky action at a distance'.

## 1.2 Present

Today Petri nets have a large number of branches. They are used in many engineering and technology disciplines due to their appealing conceptual simplicity, diagram representation and support for automated analysis tools. There are numerous special theories and interpretations of nets usually un/foldable (mappable) into one of the common Petri net models. Nets have been used to define the semantics of programming constructs or languages in a mathematically precise way. Net theory has also found its way into the intersection of computer science with other disciplines. The multidisciplinary and transdisciplinary application and theory of Petri nets is a testament to the deep connection with physics, physical chemistry and science generally. I only mention a few examples and perhaps the lesser-known ones outside of computer science, electronics and computer systems engineering:

- Music descriptions and processing [5] for composition and analysis, including ordering, concurrency, repetition, timing, synchrony and non-determinism;
- Reachability graphs of Predicate-Transition nets to significantly reduce the state space of quantum computing models [9] for purposes of quantum model checking;
- Stochastic and fluid Petri nets in biochemistry [2] as models for both stable (or invariant) metabolic states and transitions between such states.

Petri invented the graphical net notation at age 13 in 1939 to model chemical processes and used this notation in his professional life before beginning his Ph.D. work. Due to their graphical presentation, Petri nets have spread into many practical applications and are defined in industry standards:

- ISO/IEC15909-1:2004 Software and Systems Engineering—High-level Petri nets for systems and software engineering—used in the design and analysis of discrete event systems, including parallel and distributed systems;
- ISO/TR 12489:2013 Reliability modelling and calculation of safety systems in the petrochemical and gas industries;
- ISO/IEC15909-2:2011 Software and Systems Engineering—High-level Petri nets interchange formats for Petri nets, High-level Petri nets and others.

These and further standards are under continuous development at national and international levels. Elements of Petri nets are also used explicitly in numerous de facto industry standards such as activity diagrams in the Unified Modelling Language for software engineering and various systems and software design methods widely used in the software industry or software-intensive industries such as the automotive industry, robotics and manufacturing. Their graphical notation offers intuitive practical use, focus on critical situations in process modelling, and error analysis. Their execution semantics, in contrast to many other visual modelling languages, has an exact mathematical definition with a well-developed mathematical theory for process analysis.

While much of the theory of Petri nets is still held and created in Europe, the US has played a key role in the popularity of Petri nets in theory, practice and

teaching—not least through a strong MIT interest in the late 1960s, when Anatol Holt and others turned to net theory and contributed to the theory of the dominant Petri net class of place-transition nets, and its direct or indirect use in dynamic software models.<sup>3</sup>

In a nutshell, today Petri nets are alive, blossoming and thriving with no end in sight.

### 1.3 Future

A recent Google Trends check<sup>4</sup> reveals that Google searches for Petri nets are dominated by searches in correlation with *software* and the *person Petri* roughly equal at the top, followed by *simulation* and *system* at 3/4 of the top frequency, and then *workflow*. While Germany tops the list followed by Russia Italy and France, the United States has a significant presence on the list with a slightly permuted set of correlated terms. For example here *deadlock* shoots up past *workflow* and the *Unified Modeling Language* appears. In the US some search spikes seem to coincide with times of tests and exams. We should not underestimate the role of US universities, standards and practices in this powerhouse of software engineering in driving the development of Petri nets over the next decades, beside the European strengths in software intensive embedded and connected infrastructures in Industry 4.0 domains. I dare to include the emergent and future disciplines of quantum software engineering and biomaterials programming, which are already gaining momentum at the modelling and simulation level and are poised to make significant commercial inroads. These systems will be truly parallel, software controlled and hybrid (discrete-continuous). History has shown that new programming paradigms do not remain in the hands of mathematicians and engineering-scientists. As they become mainstream, diagram design notations suitable for practitioners, designers and testers, combined with expressive and easy-to-use scripting, emerge for rapid program development, driving down cost-benefit ratios and accelerating time to market.

## 2 Petri's Zeit

### 2.1 Time

Beside removing continuous real time (the reals  $\mathbb{R}$ ) and a single global clock from a priori assumptions ruling synchronisation, Petri's nets take a number of other discrete or finitary axiomatisation steps, including causality, local time and

---

<sup>3</sup>Anatol Holt's role includes many contributions. He also coined the term 'Petri nets', according to a letter Petri sent to Holt, as reported in a memorial lecture for Petri.

<sup>4</sup>May 10, 2017.

limited spatial neighbourhoods. They aim to represent topological and geometrical properties by the static structure or architecture of the nets including elements of change. To bring the concurrent effects of these dynamics to life, only the redistribution of tokens of information needs to be animated in these graphs. Where this is too unwieldy, special algebraic and graphical notations are used to show any critical evolution of the system under study or test, or solutions to problems formulated. The seeds of the modern axioms were already sown in Petri's thesis. The static representation of change allows teachers of Petri nets to present complex dynamics as a game, where student players enact processes, and the net lays out the spatio-temporal structures and defines the rules of change including constraints on independent moves. Tokens of various shapes and information content represent the parts or particles, and are moved around on the net so players can record mutual agreements, document the current states of parts and enact the transitions, possibly with choices, dependent on the rules of the specific net interpretation. Moreover Petri's *k*-forms (a specific kind of nets), which he used to build a Turing machine, allowed for a physical system with a growing number of component nets. This was to avoid requiring an infinite tape but equally to avoid the crippling limitations of a bounded finite-state machine for his general asynchronous computational machines. Relativistic time and the problem of synchronising large numbers of microscopically small parts (such as in computer chips) has often been a starting point in Petri's talks. In his thesis, however, quantum-level uncertainty and relativistic time (bounded signal speed) are put on the same footing. Regardless of whether clock ticks are counted, errors in timing or location, or mass concentrations in space, Petri kept insisting to those listening, these measurements are ultimately realised using devices that have to follow the laws of modern physics. These impose relativistic bounds and Heisenberg uncertainty. Petri chooses explicitly not to present discrete mathematical models of parallel systems as an approximation of an analogue world (with continuous time and space given a priori). Rather, he accepts a finitary continuity of nets and mappings to continuous reals as an idealisation of an ultimately quantised world. In 1962 he wrote

We will speak as if the discrete objects of the theory are embedded in a continuous space-time world in order to facilitate understanding. It should be noted, however, that this is by no means necessary; the apparent vivid clarity of the continuum ultimately rests on the erroneous assumption that the axiom of density has an operational meaning. This axiom is rather a linguistic instrument for hypothesis formation in the sense of inductive logic.

As continuous real time is a priori absent in Petri nets, it must be reintroduced appropriately in special net interpretations or models, when needed,<sup>5</sup> based on net structures and partial or cyclic partial orders only; and the occurrence elements of

---

<sup>5</sup>Often with a warning by the computer pioneer to make sure temporal assumptions are consistent with concurrency axioms: "This is only for convenience and ease of understanding in classical terms"; "Continuous real-time is based on the erroneous assumption of density, which contradicts our axioms in terms of measurement"; "Two integer clocks can be compared to arbitrary rational precision and ultimately violate either spatial density bounds for information or speed bounds for synchronisation when the machine keeps getting extended".

computations to which a net gives rise, have to be mapped consistently to space-time. However for a consistent modern physics notion of time, for example for the use of nets in chemistry, metabolic biological processes or quantum computing, different variations of Petri nets have been built in the absence of a compatible discrete mathematical and axiomatic foundation of quantum physics and/or the absence of a suitable quantum Petri net theory that places Petri nets into the emerging theory of quantum automata and formal languages.

Common to all these nets is that the elementary states (usually denoted by  $S$ ) are definite in location or space (as indicated by the term *place*<sup>6</sup> in some net interpretations) but have no definite single time point in (local or global) time. Instead they are thought and taught to be associated with a temporal region or duration when mapped into physical global space and time by practitioners. They are contingent. Moreover, looking at concurrent states, their simultaneity, while explicitly possible, is objectively transient and hence potentially not observable. Fundamentally, state elements give rise to spatial characteristics of the net. Dually, the elementary transitions ( $T$ ) have a definite local time (point) and indefinite location. An elementary transition  $t$  represents an interaction causally dependent on state elements in its pre-set  $\bullet t$ . State elements in the post-set  $t\bullet$  depend on  $t$ . As an interaction between two or more parts, the location of the transition is indefinite, related to all its interacting parts, in a region defined by the parts' possibly multiple locations. A high-level transition or state may refine or unfold into lower-level transition-bound and state-bound nets, respectively. Thereby transitions may gain stability and temporal duration from underlying states, and states instability and local dynamics from underlying transitions. These refinements or foldings (in the other direction) preserve net topology [12]. The causal relationship or flow ( $F$ ) from pre-set states to transitions or from transitions to post-set states may be further quantified and qualified, depending on the special net interpretation. For example, quantities may specify the number, colour or value of tokens. Qualities in some nets include triggering (exhibitory) or cancelling (inhibitory) of transition firings depending on such quantities, or on predicates in terms of information carried by such tokens. In this sense, causality is the primary concept ordering transition occurrences, not time. The resulting causal ordering of firing sequences may be regarded as a local and discrete time. Thus a transition firing defines a discrete time-like point measurable/noticeable in terms of net structure, while the marking of a state defines a space-like point. The waves of discrete firings and markings across the net define a partially ordered global time in Petri nets in terms of so-called (Dedekind) cuts<sup>7</sup> of maximal sets of mutually independent net elements.

---

<sup>6</sup>The original German terms 'Stelle' and 'Stelligkeit' also mean 'position' and 'arity' for token distribution variables in transitions or the entire net as functions on such distribution vectors.

<sup>7</sup>Dedekind cuts at the event *occurrence net* level, i.e. in terms of elementary firings and marking.

An overriding guiding principle of Petri's axiomatisation of concurrency and synchronisation is that

**(I) The system model must be consistent with the laws of physics.**

Other luminaries acknowledge this principle:

Another important point, which may originally have seemed merely eccentric, but now looks rather ahead of its time, is the extent to which Petri's thinking was explicitly influenced by physics ([...] As one example, note that K-density comes from one of Carnap's axiomatizations of relativity). To a large extent, and by design, Net Theory can be seen as a kind of discrete physics: lines are time-like causal flows, cuts are space-like regions, process unfoldings of a marked net are like the solution trajectories of a differential equation. This acquires new significance today, when the consequences of the idea that Information is physical are being explored in the rapidly developing field of quantum informatics. (Abramsky [1])

Much of what I have been saying was already well understood in the sixties by Carl Adam Petri, who pioneered the scientific modeling of discrete concurrent systems. Petri's work has a secure place at the root of concurrency theory. He declared the aim that his theory of nets should—at its lowest levels—serve impartially as a model of the physical world and as a model of computation. [...] Already, for him, a memory register and a program are modeled by the same kind of object—namely a net—and this breaks down the active/passive dichotomy. (Robin Milner in his Turing Award Lecture 1991 [8])

Both Abramsky and Petri complained at times that too many concurrency models (including special new types of nets) are cooked up quickly as variations of existing mathematical models because of the taste of computer scientists. In Petri's view, the ultimate proof was not in the convenience of notation, but in the realisation of parallel systems and the adequacy of the nets in describing them. This principle of an experimental information theory of concurrency, not unlike the applied and experimental physics environment that Petri grew up in, is the other side of the coin of Principle (I) above:

**(I') Experiment and computation shall prove the utility of the model and the theory.**

Petri was aware of the grand fusion of geometry, abstract algebra, arithmetic and topology occurring (and still unfinished) and driven by mathematical physics and experimental physics combined. In his later lectures, he occasionally mentioned Hilbert and Minkowski (cf. Fig. 3). The latter had lectured at Bonn and given his famous speech in 1906, at a historic Cologne Workshop: "The views of space and time which I wish to lay before you have sprung from physics. [...] Space by itself and time by itself will totally fade into shadows." Petri saw his nets making a contribution by linking to physics through a deep rooting in relativistic and quantal measurement. Towards the end of his career and into his retirement he kept working with collaborators including Konrad Zuse and connecting the lowest-level meshes of his nets with quantum foam in discrete complex spaces. The most recent lectures of the late Petri bring home his view that in order to make space for parallelism present in our natural world from the subatomic level to everyday life, methods must be shaped that are rooted in the mathematics of quantum mechanics. This necessitated methods allowing for, and reasoning with, uncertain judgement of temporal and spatial distance at any scale. These would then give sufficient wiggle space to



parallelism and acausal relationships, even at the level of socio-technical systems where evidence is judged on the basis of de facto and principally *incomplete and uncertain* measurement, by people and organisations.

## 2.2 *Zeitgeist and Place*

Let us briefly look at the time and context into which Petri's initial ideas were born. Petri was an applied mathematician by training with applications in physics before starting his Ph.D. at the Faculty of Natural Sciences at the Rheinische-Friedrich-Wilhelms University Bonn.<sup>8</sup> Petri conducted his Ph.D. project in an environment of applied mathematicians applying themselves to physics and solving practical problems in geometry and PDEs, with a particular focus on the novel computational approaches possible with computers. Even in my undergraduate days at Bonn, in the first half of the 1970s mathematics and computer science students had to take a common foundation in physics, and started largely with an applied mathematics background, with practice in Fortran and Lisp<sup>9</sup> (plus other language options such as APL, ALGOL and PL/I) with numerous other computer science courses. From 1958, Heinz Unger was the chair of the Institute for Applied Mathematics at Bonn—appointed by Prof. Ernst Peschl, his former Ph.D. advisor. Unger's institute maintained a focus on numerical methods applied to physics. Other new chairs were subsequently appointed and covered mathematical physics and statistical methods. Computer science was not an established discipline yet. Prof. Böhling (cf. also Fig. 3), another former Ph.D. student of Unger around Petri's Ph.D. time, was the first Computer Science Chair at Bonn in the late 1960s.

This was the immediate environment in which Petri grew to become a scientist and leader.

The *Zeitgeist* around the time of Petri's thesis project was affected by global winds of change, which had started well before. While Kurt Gödel had written his proof 30 years earlier, his crucial works were still being digested at universities including Bonn. Science was grappling with the related conundrum, that on the one hand certain aspects of physical reality are fundamentally beyond the realm of precise measurement, yet on the other, probabilities are measurable as physical reality and therefore they exist. But had experimental science not held *that which is not measurable is not science* and *that which is measured in scientific experiments*

---

<sup>8</sup>Well before I studied there and met him while working as a student and later researcher at the GMD Schloß Birlinghoven, St. Augustin, near Bonn, where he was the foundation director of the Institute for Information Systems Theory (1968) later renamed to Methods Foundations, while I was conducting research in the neighbouring Institute for Systems Technology working on compiler construction from 1976 and later on software theory, the Segras specification language and a toolkit bridging algebraic-categorical specification and Petri nets.

<sup>9</sup>Course given by Klaus Berkling who later built the GMD reduction machine for his variant of the  $\lambda$ -calculus.

*needs a logical interpretation?* Different interpretations and philosophical schools were vying for the ultimate language and logic underpinning of quantum physics and relativity theory. The Vienna Circle (including Gödel) had disintegrated due to Nazi persecution and emigration before WWII, but their school of thought now, after WWII, began influencing the thinking once again.<sup>10</sup> Von Neumann had just passed away a few years earlier (1957) leaving deep works on axiomatisation of quantum physics, logic, probability and computing (including brain computing) and a legacy of collected works that would only successively appear, including on cellular and parallel automata.<sup>11</sup> The British Prime Minister Harold Macmillan had decided to give independence to several colonies and gave his famous *wind of change* speech in early 1960. US President John F. Kennedy had announced on May 25, 1961, the commitment to *landing a man on the moon and returning him safely*, and asked for an extra nine billion dollars in science, technology and defence funding. Petri submitted his thesis 2 months later on July 27, 1961 in this time of daring, courage and challenge,<sup>12</sup> which was also the height of the cold war. Kennedy visited Germany in June 1963. JFK gave a speech in Berlin with the famous quote “Ich bin ein Berliner” and in Cologne (near Bonn) at the site of the old city hall and of the relics of the medieval synagogue and Jewish quarter<sup>13</sup> (now the site of the planned Jewish Museum celebrating 1700 years of Jewish history in the Cologne-Bonn area), just a short walk from the cathedral. JFK famously and comedically ended his speech to the locals at this historic place and time with “Koelle Alaaf” but not without mentioning Albertus Magnus (Albert of Cologne) who taught Thomas of Aquinas. (See photo in Fig. 1a from a recent visit to Cologne and this site.)

Classical physics continued to be challenged: Feynman was publishing and lecturing on Quantum electro-dynamics at the beginning of the 1960s, which would earn him a share of the 1965 Nobel Prize in Physics.

It stands to reason that Petri would have been acutely aware of many of these streams of global thinking and ideas. Antiauthoritarianism and anti-establishment sentiment was growing well before student revolts of the late 1960s and early 1970s, and locally—especially in the Cologne-Bonn region, where Petri lived and worked, and Bonn the capital, the seat of federal government, was hence the target of choice for protest marches. At the time of Petri’s thesis project, Cologne-Bonn had also become a hub of philosophical, political and cultural movements. It was a centre and meeting point for German and international avant-garde with the now world-famous and late composer Stockhausen drawn into the scene by the artist Mary Bauermeister, and then writing many of his revolutionary scores for ‘statistical’ and ‘spatial’ music, which was played on unconventional devices and deliberately

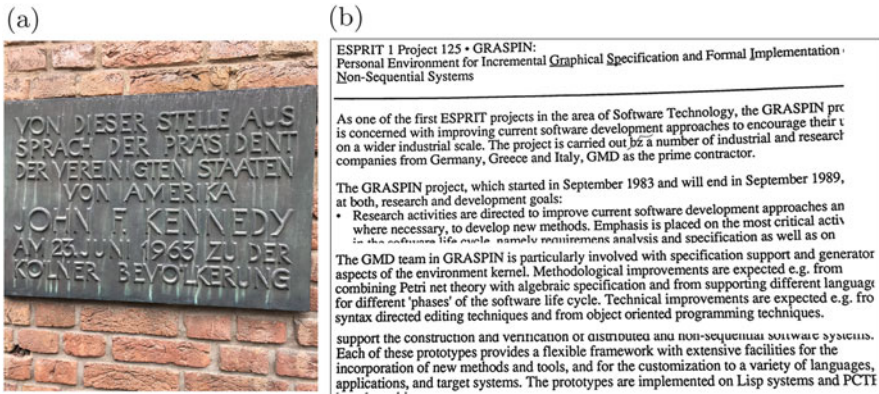
---

<sup>10</sup>Logical positivism: Only statements that are empirically verifiable are empirically meaningful. But how can we even set a test and verify without prior theory and meaning?

<sup>11</sup>Taught in Bonn, where, to my delight, his entire collected works was held in multiple copies.

<sup>12</sup>Only to be admitted to his defence on June 20, 1962, almost a year later.

<sup>13</sup>Fifty years before Germany would become the country with the fastest-growing Jewish population outside Israel.



**Fig. 1** (a) Left: (1963) John F. Kennedy visited Cologne, in times of global change; (b) Right: (1983) Draft European Project GRASPIN press release; Petri nets for industrial software engineering

questioned the experience of time, with temporal dynamics with or without states and transitions—inspired by quantum physics and its modern philosophical notions. Stockhausen, who had studied at Cologne, Paris and Bonn, moved to Kürten in 1961 and began one of his most productive and revolutionary phases not too far from the GMD, and held professorial positions in Cologne and at a couple of US universities. In those years, elements of the later Fluxus movement were born, and then grew in the sixties when Mary got drawn to New York (1962–1970) by Pop Culture. Fluxus was later described as “the most radical and experimental art movement of the sixties”. Key figures of the later Fluxus movement and of the avant-garde, poets, musicians, mathematicians, composers of experimental music, and artists congregated at Cologne happenings from 1959–1960, in Mary Bauermeister’s studio in the Lintgasse in Cologne: David Tudor, John Cage, George Brecht, La Monte. Young, Nam June Paik, Mary Bauermeister, Karlheinz Stockhausen and others. At the beginning of the 1970s, Mary moved to a neighbouring suburb to mine in the broader Cologne area, and later became a good friend.<sup>14</sup>

Zeitgeist is not confined to singular happenings; it weaves together diverse perspectives and crosses generations, and cultures; its ebbs and tides hit shores in successive waves in culture, the sciences, technology and the wider society. A young democratic West Germany with an old history and science grew up only slowly in the shadows of the Holocaust [3], with the curses and blessings of European allies, the help of a buoyant USA and admiration for popular figures like JFK and Bob Dylan, apparently one of Petri’s favourite musicians.

<sup>14</sup>Through art, shared Yoga interest and hobbies such as bee keeping and, not least, Math tutoring of her children and endless fascinating table conversations and occasional short courses on Eastern philosophy and Hermetic literature.

In 1968, from this historically strong Bonn University, Peschl, Unger and others were also instrumental in founding the Gesellschaft für Mathematik und Datenverarbeitung (GMD—German National Research Centre for Mathematics and Computer Science), in which Petri soon would lead his institute, and in which Prof. Krückeberg, another former Ph.D. student of Unger, later became the technical director, after Peschl's, and Unger's directorships.

In parallel to the creation of the GMD, the term 'software crisis' was coined in 1968 at a NATO conference held in Munich. This led many universities in Germany and across the world to create computer science departments and eventually full CS degrees. While offering CS subjects and research to mathematics students for some time, Bonn had just introduced CS before I enrolled in winter semester 1971, in Mathematics, Computer Science and Asian Philology, with some overloading in courses and practicals. Many of the professors listed in Fig. 3 (bottom) for Bonn (yellow) and GMD+Bonn (green) were lecturing at Bonn during the 1970s, when I took courses offered by Unger, Peschl, Böhling, Indermark, Olivier, Hasenjäger<sup>15</sup> and others (not in the graph). It is interesting to note the central role that Peschl and Unger played in applied mathematics, and in the creation of computer science at Bonn, beside the foundation of the GMD, and how this is reflected in the Ph.D. ancestry graph, too.

Already in 1974, I began work at the GMD as a research assistant, before graduating. This also offered the opportunity to source a Master's thesis within an applied context. 1974 was also the year I married. Hence, this appointment and the option of a thesis were important also to set me on a path of completion and independence, and I gave up the idea to complete—beside Computer Science—a full degree in Mathematics and at least undergraduate in Asian Philology. I assume the first CS students in Bonn graduated in 1975, as my GMD colleague Cornelius Hopmann mentioned later that he had been the fourth graduate from the new CS degree, in 1975. According to my low student ID number, I would have been among the first one or two cohorts to enrol in it. In mid-1976, immediately after graduating—then from a ten-semester degree<sup>16</sup> including my thesis project—I was hired full-time by the GMD as a researcher. I had already met Petri once or twice during this time. My Ph.D. project on Petri nets and algebra would only begin in the early 1980s, and was entirely conducted at the GMD Bonn (Schloß Birlinghoven, St. Augustin). I travelled for meetings with my main advisor, Prof. Hans-Jörg Kreowski, at Bremen, or he visited GMD. Locally, I had access to my second advisor, Prof. Kurt Lautenbach, who also joined me at my defence in Bremen, and who was one of Petri's close collaborators (see also Fig. 3).

Through the GMD and the foundation of with generous state and national funding, Bonn had well-advanced computing platforms already since the foundation

---

<sup>15</sup>Who goes back to Hegel, not shown in the graph.

<sup>16</sup>Many students started a professional career during their last year of study or earlier. The average number of semesters was therefore 12 or more in this unregulated environment, due to part-time study in advanced years.

of the GMD, still at my time, and I believe now that GMD is merged into the Fraunhofer. These included IBM 360/50 systems with, at the time, significantly large storage and horsepower, but later also MIT Lambda Lisp machines (beta tested at GMD, we were told they were the first outside the US)<sup>17</sup> and they were soon replaced by (more reliable) Symbolics Lisp machines. Subsequently, Symbolics 3600 series machines were used in our projects for visually editing Petri net and algebraic specifications, rewriting, unfolding and implementing them. The close interaction and long-lasting friendship between Bernd Krämer, Dimitrios Christodoulakis and the author (cf. Fig. 3) was forged in those years, at GMD Bonn, and continued into our professorial careers. This early work on Petri nets, specification and implementation found space to expand in a successful European ESPRIT grant, GRASPIN (Fig. 2) from 1983 [6, 7, 16, 17], and then involved translating some of the methods into an industrial context at Siemens in Munich, Olivetti in Pisa and other participating companies. Later our tools were ported to (co-branded) Xerox Interlisp AI machines (“West Coast Lisp”) by Siemens and we ported them to Unix running on Sun Microsystems.

### 3 Petri’s Vision

Among the many visionary themes that still run through the theory and application of Petri nets, how can we identify the, say, two or three principles that are the most outstanding? We approach this question by calling on Petri’s guidance, looking at his thesis and comparing candidate themes with his (and others’) later publications, and a recollection of his inspiring talks and conversations. To excel over the next 50 years, such key characteristics must attract the next generation of brightest minds by

1. contributing and being seen to *contribute computer science understanding and methods to other collaborating sciences*, notably physics and related sciences (given Petri’s scientific origin there);
2. *growing* as an active field in its own right, with deep knowledge and beautiful mathematics solving or helping to *solve grand challenges* and delivering *break-through methods*;
3. *offering* rich opportunities for *impactful translation into education and practice* of systems and software engineering, natural sciences and artificial intelligence, including global and virtual collaboration platforms.

---

<sup>17</sup>From memory, 1979, due to Petri’s close links with MIT; I could make it crash occasionally by fast mouse movements; it was usable, and entirely written in Flavors, the object-oriented East Coast Lisp, which I would use later for both compiler construction and algebraic Petri net specification, and as a target for pilot implementations.

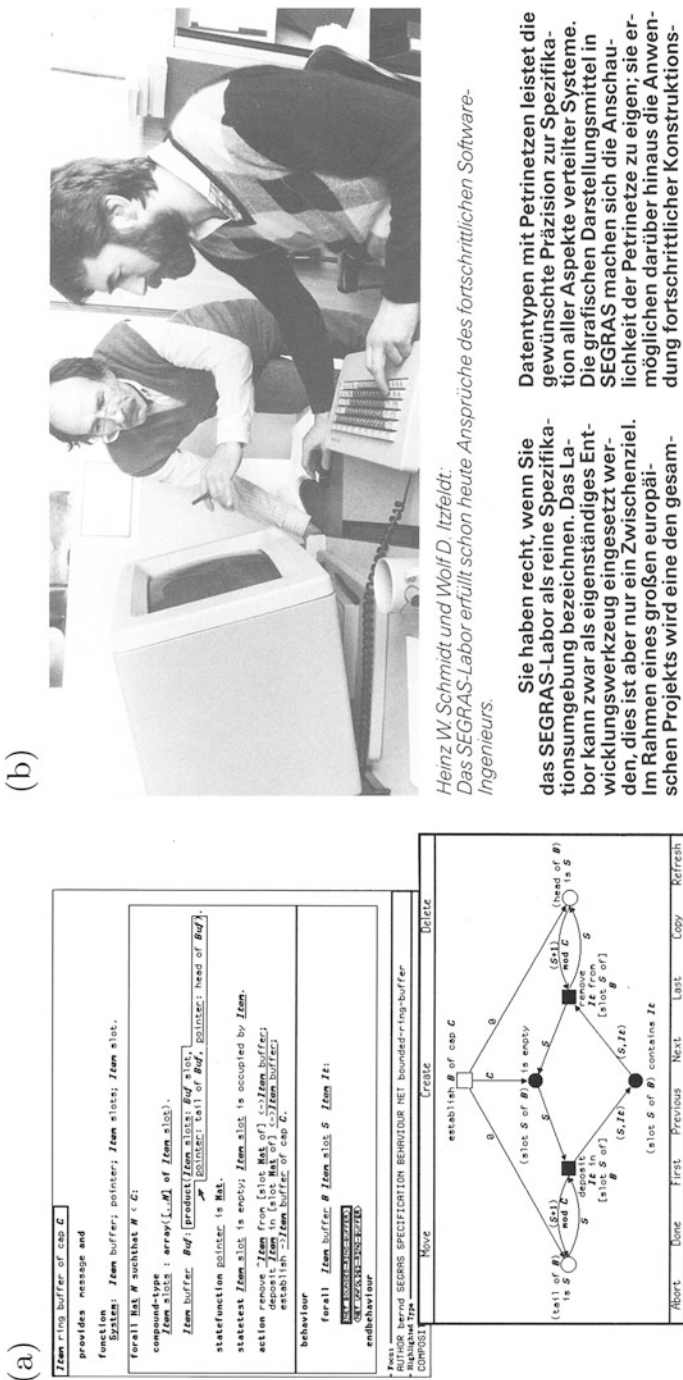


Fig. 2 (1986) Trial demos and draft material for CeBIT'86 demonstration of SEGRAS software development kit. (a) Segras personal SDK screenshots; (b) Photo: front, Heinz W. Schmidt (then senior researcher GMD-IST); back, Wolf Itzfeld (GRASPIN contract manager, GMD-IST)

In principle (I/I') we have already identified the importance of modelling realism through experiment. What are the other two?

### 3.1 *Make Space for Parallelism*

True parallelism requires wiggle space for the asynchronous participants in interactions and therefore relaxation of metrics, especially continuous metrics. A foremost guiding principle is therefore:

**(II) Make true concurrency a starting point rather than an afterthought.**

This applies to engineering designs as much as to the axiomatic design of net theory. The principle respects partial independence and prioritises causality over temporal order.

Petri repeatedly stressed the point that (global) metric properties of building blocks in models and machinery *must be relaxed* to faithfully achieve parallelism and asynchronous operation, i.e. local control of actions. This is necessitated by de facto *unbounded extensibility in terms of dynamic allocation of additional physical machinery* such as (in today's terms) memory banks, cloud servers, physical resources, say in disaster emergency services, recruitment of brain regions for highly demanding tasks, or ensemble formation and entanglement of particles in quantum electrodynamics. Relaxation, so Petri claimed in his thesis already, *can be achieved* in exchange for structural characteristics that enable proper synchronisation and communication consistent with the physical realism of the computing devices, their operation, and their communication under the principles of modern physics. These structures occur now in a *topology* of Petri nets, in which it is still possible to resurrect a discrete and (in the limit case) continuous geometry by counting alone.<sup>18</sup> In his talks, Petri occasionally quoted Kronecker, who famously said “God made the integers, all else is the work of man.”

Clearly Petri nets thus take an architectural approach to systems from their very beginning—in the sense of the IEEE standard definition of systems architecture: as the characterisation of components, their interrelation and interaction with the environment of the system. This approach, too, was visionary and predates, for example, the maturing of software architecture in the course of the 1990s.<sup>19</sup>

In his *Ph.D. thesis*, Petri showed how this relaxation of metrics could be achieved for a realisation of the Turing machine in terms of his nets, including the formalisation of the Turing tape, which is usually not formalised but simply taken for granted in mathematical introductions to the Turing machine model.

---

<sup>18</sup>This is out of the scope of this work. We refer to work by Valk, Smith, and Petri himself.

<sup>19</sup>As celebrated for instance by Mary Shaw's and David Garlan's articles, or speeches on software architecture practice and research at the International Conference on Software Engineering (2000 and beyond), the flagship conference of the software engineering discipline.

For Petri, a key axiomatic consequence of this was to accept

- upper bounds for (temporal) signal speed and (spatial) information density;
- limitations of any finite-size machine to iterative classes of input sequences;
- unbounded on-demand extensions of finite-size machines for processing recursively defined input sequences.

Consequently, classical automata theory devices violate at least one of the above requirements after a finite number of extension steps.

*In his later writing*, already from 1980 [11, 12], Petri and his inner circle of researchers return to continuity, topology and the connection to metrics, introducing both N-density and K-density of certain nets. Net topology is based on the relation of proximity ( $P$ ) of a local state with a transition ( $sPt$ , read  $s$  at  $t$ ) if and only if there is a flow relation between the two. Thus transitions relate to the inseparable annihilation and creation of local state, at the heart of causal connectedness and change, and represented by movement of tokens, in the space-time structures of nets. Einstein is famously quoted as saying: “Nothing happens, until something moves.” The relation of Unitary axiomatically defined net structures to metric spaces (including the continuous coordinate systems and time central to engineering mathematics) remains a topic in Petri’s later work and that of others, is does the continuity of net mappings in terms of net topology. Petri wrote [13] that “net theory was originally presented as the combinatorial topology of causality”.

Many information theory researchers dealing with concurrency are still grappling with its fundamental concepts. There are multiple proofs that true concurrency includes but is different from interleaving concurrency. If Feynman’s living things (in his own words) can all be understood in terms of jiggling and wiggling (of atoms), then Petri’s nets provide the topological wiggling space to do so in very large numbers, asynchronously and scalably, and in hybrid (discrete and continuous) terms. Petri’s universal gate, studied in his thesis in 1962 for reversible computing, was rediscovered after joint work as the Fredkin and Toffoli gate in the late 1970s and acknowledged by them in 1982 [4]. Some authors now call it the Petri-Toffoli-Fredkin gate. Moreover, the arbitrary ordering of truly concurrent events in interleaving semantics is not a decision or conflict in Petri nets. Leslie Lamport writes in connection with a 2003 paper on arbiter-free synchronisation<sup>20</sup>:

In Petri nets, arbitration appears explicitly as conflict. A class of Petri nets called marked graphs, which were studied in the early 70s by Anatol Holt and Fred Commoner, are the largest class of Petri nets that are syntactically conflict-free. Marked-graph synchronization is a natural generalization of producer/consumer synchronization. It was clear to me that marked-graph synchronization can be implemented without an arbiter, though I never bothered writing down the precise algorithm. I assumed that marked graphs describe precisely the class of synchronization problems that could be solved without an arbiter.

That marked-graph synchronization can be implemented without an arbiter is undoubtedly obvious to people like Anatol Holt and Chuck Seitz, who are familiar with multiprocess

---

<sup>20</sup><https://www.microsoft.com/en-us/research/publication/arbiter-free-synchronization/> accessed mid-May 2017.



synchronization, Petri nets, and the arbiter problem. However, such people are a dying breed. So, I thought I should write up this result before it was lost. [...] it occurred to me that it would be fitting to contribute some unpublished 25-year-old work.

### 3.2 *Combinatorial Nets, Logic and Algebra*

High-level Nets, Coloured Petri Nets, and variations of Algebraic Petri Nets [6, 14, 17] have become a corner-stone of Petri nets connecting them with abstract data types and high-level programming languages, analysis methods and tools. They permit abstraction, templates, genericity, foldings, reductions, parameterisation, well-founded compositionality and a finitary representation without the a priori assumption of finiteness or even boundedness. Hopcroft, Motwani and Ullman wrote in their well-known book on automata theory:

In fact one could argue that a computer with 128 megabytes of main memory and 30 gigabyte disk, has “only”  $256^{30,128,000,000}$  states, and is thus a finite automaton. However, treating computers as finite automata (or treating brains as finite automata, which is where the finite automata idea originated), is unproductive. The number of states involved is so large, and the limits are so unclear, that you don’t draw any useful conclusions. In fact, there is every reason to believe that, if we wanted to, we could expand the set of states of a computer arbitrarily.

Most importantly, through their connection to algebra and logic, Petri nets make it possible to abstract from the specifics of a given parallel behaviour or communication problem, and characterise the common structures and invariants in the dynamic variance and change, especially when it is fraught with the combinatorial complexity of parallel processes, for which the unfriendly term ‘state space explosion’ has been coined. Specific method worth mentioning and connected in fundamental ways with Petri nets must include the linear algebraic and numerical methods used in the special case of place-transition nets and their polynomial variant of algebraic nets, especially the methods of S-invariants and the lesser-known but equally important dual T-invariants.

In general net theory [12], different special net interpretations become specialisations of higher-level ones through net mappings (morphisms). The algebraic and logical characterisations may also help bridge from nets to algebraic models used in other fields of science to contribute or borrow specific, net methods. In his thesis Petri foreshadows this link and emphasises, well beyond his logic gates:

The transition to formal logic, algebra, and topology is immediately possible because of the axiomatic method by which we introduce the concepts, and the theorems present there can be applied directly to nets.

Petri returns to this point repeatedly in several of his papers, including his papers on concurrency [11], for example connections to algebra and logic through so-called

facts and violations,<sup>21</sup> a paper on general net theory [12] illustrating a special role for algebra, and papers and talks around 2000 on culture and mathematics of nets [13]. While striving for simplicity where possible, Petri also stressed in his thesis

We do not ask which concepts are, in truth, the simplest. Only the successful application of the theory, to be built upon the mathematical model proposed, will be claimed as a justification of the conceptualizations.

Hence a further guiding principle might be formulated as follows:

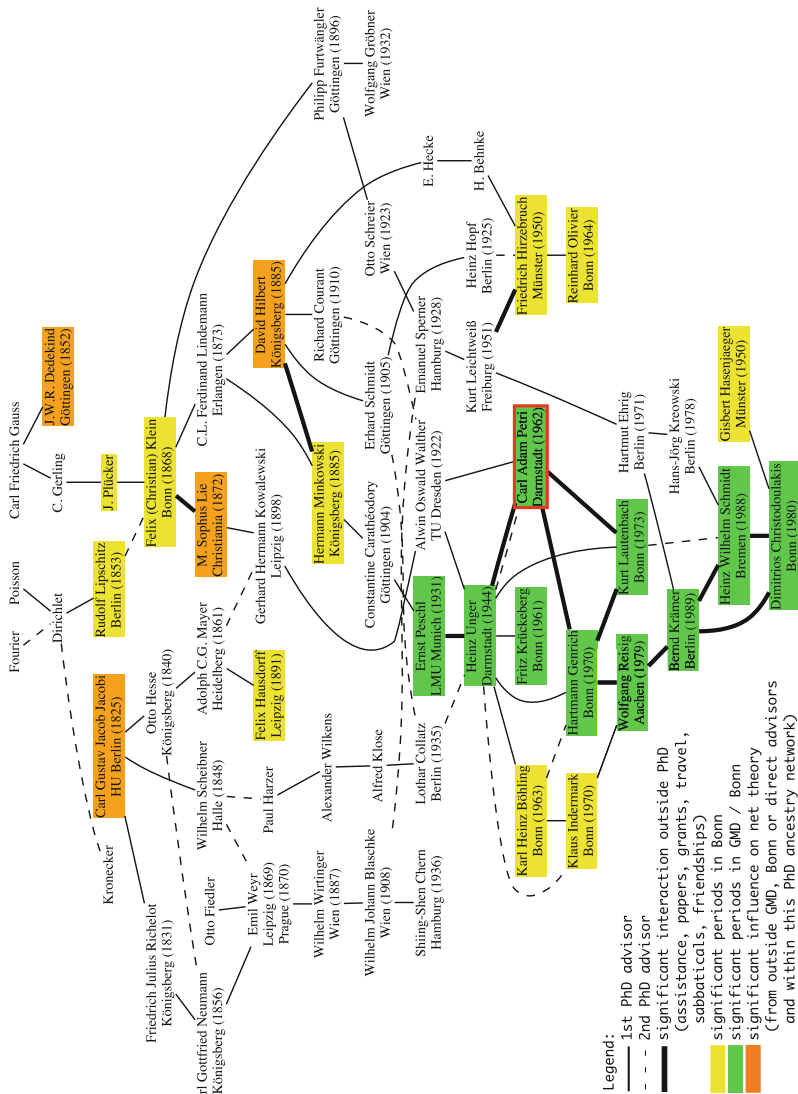
**(III) Focus on methods and tools for analysis before conceptual simplicity.**

The algebraic, linear algebraic and polynomial representation of Petri nets has received much attention. In practice, the ‘additive’ net structure is often pleasantly more compact than the ‘multiplicative’ structure of its reachability graph. Especially, the net may be finite or even small, while, through the folding structure and dynamics in the markings, its reachability space can be very large or infinite. But not all problems can be solved at the net level. For a specific interpretation such as stochastic nets or quantum automata with their probability amplitude stochasticity in complex numbers instead of real-valued probabilities, even the discrete reachability structure implied by the net can be a further massive reduction compared to that of the original stochastic or quantum automaton. This connection has been exploited, for example by Leo Ojala’s group [9], when representing the control of quantum cellular machines as Petri nets to dramatically reduce the quantum model-checking problem. For some types of systems, Petri nets pick a balance between capturing the partly finitary structure of change and permitting the sometimes efficient analysis of that part, factoring it out and leaving the ‘nastier’ parts in the space of the reachability structures. Typical Petri net matrix calculations are not only economic; they also work in largely reduced representations of reachable markings, in some sense allowing the conversion of space into time or vice versa, ultimately through algebraic transformations of the nets themselves, using group theory and net foldings [17]. Working on the graph in Fig. 3, I also came to realise again that Felix Klein had spent significant time at Bonn University and that the core of his new group theory was shared with me by my high-school mathematics teacher, Dr. Bruno Bosbach, who later became Professor at Kassel University. While he held the position at the Albertus-Magnus Gymnasium (High School) in Bensberg, he also lectured and published on group theory at Wuppertal University, and offered group theory to gifted students at high school.<sup>22</sup>

---

<sup>21</sup>Distinguishing evidence, affordance and allowance of harmful or unsafe behaviour.

<sup>22</sup>Several of us went on to study mathematics, not least due to Dr. Bosbach and his group theory workshops.



**Fig. 3** A subgraph of the Bonn Ph.D. genealogy, reachable from C.A. Petri, the author, and their immediate collaborators during the author’s time at GMD. The data were mined from the open data Mathematics/CS genealogy as of mid-May 2017 (see [www.genealogy.math.ndsu.nodak.edu](http://www.genealogy.math.ndsu.nodak.edu))—a service of North Dakota State University in association with the American Mathematical Society). This data set may not be complete. Evidence for periods at Bonn (yellow), or GMD and Bonn (green), and for significant interaction (thick lines), outside of Ph.D. advice already in this graph, was collected from open data, biographies (on Wikipedia and Springer Link at [link.springer.com](http://link.springer.com) [15, 18]), and via personal observation and communication

## 4 Petri the Person

Petri nets have accompanied me throughout my working life from my early work on algebraic specification and Petri nets with one of the first working models of algebraic Petri nets in two 1984 GMD publications using Goguen-style algebraic specification and semantics. Petri has thus been a lasting and deep influence on me and directly or indirectly on my collaborators and Ph.D. students.

In his talks and the personal communications I had with him, I always experienced Petri as humble, somewhat shy.<sup>23</sup> Yet he presented his bold ideas with exceptional clarity and occasionally carefully chosen subtlety in nuances and always a deep understanding of neighbouring fields, especially physics.

Petri took everyone with him. For example he would, metaphorically, put the audience into transitions and let them play the token game to illustrate parallel independence of action and share some aspects of this quickly. Petri had a knack for drawings, not unlike a cartoonist. His talent allowed him to condense his message visually and playfully to achieve a surprising insight by suddenly transforming the drawing into a similar specific net pattern abstracting the principle, regardless of the limitations of the medium: an old-tech foil, a blackboard, the back of an envelope at a workshop or hand-drawn figures in a journal article. His 1996 paper on space and time is a case in point for mastery in both words and pictures.

Petri was also a master of the art of questioning his philosophy and answering questions. For example, in the same paper, he answers the question “Is not net theory listed among discrete models? Are you playing with words?” by following “By no means. Consider, e.g., that every open covering of an articulation (ONet) is overlap-connected. The set of possible measurement results for a continuous variable is therefore overlap-connected; that is, between any two results, there is a finite chain of intermediate possible results such that neighbours in the chain are compatible (do not contradict each other). This does not hold for discrete models.”

I felt very privileged when joining the GMD and beginning my research work in this context. Petri’s reputation had already grown; he was surrounded by world-class researchers and visitors that came and went. Petri and others of reputation, leading neighbouring institutes, had bold long-term plans to create the first this or that and had track records demonstrating feasibility. Especially for students and young researchers connected directly or indirectly with Petri, there was a mix of the experiences of walking on holy ground and partaking in an exciting transformation, not unlike the Facebook, Google or Apple (still) phenomenon these days, and consistent with the focus on fostering a busy international research exchange in other high-profile centres such as Prof. Hirzebruch’s at Bonn [15].

In the space of quantum information theory and quantum computing, and generally across the sciences, Petri’s foresight lent his net theory and community an eerily prescient focus on relevant topics arising from modern physics foundations

---

<sup>23</sup>Except for smoking without hesitation and without interruption, which was not uncommon in those days.

in combinatorial geometry and topology. Net theory also has unexplored depths and untranslated results. Watch this space or join the community and journey!

## 5 Conclusion

In this work we aimed to shine a spotlight on some key principles of Petri nets worth dissemination, exploration and experimentation over the next 50 years. Such progress cannot be the work of a single individual or a small group but requires a large and thriving community of practitioners, academic researchers and teachers. Petri wrote [13]: “we stand only at the beginning of a Long March, having just taken the first sure steps, while the end is not in view”.

We have drawn attention to aspects of Petri nets designed to contribute to multidisciplinary sciences that increasingly demand ab initio combinatorial modelling, simulation and experimentation. Especially in the not too distant future and likely well within half a century, massively parallel nanorobotics, brain information processing and quantum computing will require programming and software engineering methods and development workflows that are agile, intelligibly and easy to use by practitioners and at the same time trustworthy, with solid mathematically founded analysis tools, to build machine-age systems that build reliable, safe and secure systems—the hallmarks of Petri net theory and practice, today.

This presentation also aimed to be authentic and relevant in providing a backdrop and historic context of the times when Petri developed his nets. In the scope and length of this essay, it was possible only to sketch the times, the issues and the people, in particular Petri himself. However, there is a rich and worthwhile literature for the interested reader. Moreover, we took an evidence-based and scientific data analysis approach to the task. Petri’s thesis is written in German but it should not be too hard to locate or provide a proper translation that is better than the interpretation of the thinking that I have attempted here. We have made an effort not to use much mathematical notation, of which the Petri net literature is full, and to keep mathematical understanding at a general level for scientists in domains other than Petri nets or computer science. It is hoped that the essay provides broader access and some measure of attractiveness to young interested researchers, to enter this special field, or try and apply its results to their chosen field in science.

**Acknowledgements** Writing this essay and chapter allowed me to go through a small collection of ‘gem’<sup>24</sup> papers and notes and really question hard what the longer-lasting issues were.<sup>25</sup> I used data analytics methods and results that would not have been possible without open science repositories such as the Mathematics/Ph.D. Genealogy project cited, the Research Data Alliance project for

---

<sup>24</sup>I had carried them to the USA and Australia in long-distance moves and finally replaced most with their digital version, creating wiggle room in my study for other parallel interests.

<sup>25</sup>Including why I was holding on to them.

global research transparency,<sup>26</sup> Wikipedia science resources, and tools in eResearch at RMIT, where one of my roles is that of eResearch (eScience) director. This is gratefully acknowledged. This writing also allowed me to revive long-forgotten memories that otherwise would not have come to the fore—some shared here. More importantly and publicly, I owe deep thanks to the people I mentioned here or cited, notably Petri and his inner circle, and to many more there was no space or focus to mention, including close family and friends, whose journey was interwoven with the work and the times I write about. Their influence, especially in the early years of my study and career, should not be underestimated. What we are or become includes a fair measure of those around us, willing to share kindly and give forward, and willingness to accept, give back or pass on. I would also like to thank Marie-Luise Christ-Neumann and Jürgen Christoffel for keeping and sharing mementos of our former project at the GMD Bonn, and Alice Schopp, who reconnected me with Bruno Bosbach.

## References

1. S. Abramsky, What are the fundamental structures of concurrency? We still don't know! *Electron. Notes Theor. Comput. Sci.* **162**, 37–41 (2006)
2. P. Baldan, N. Coceo, A. Marin, M. Simeoni, Petri nets for modelling metabolic pathways: a survey. *Nat. Comput.* **9**(4), 955–989 (2010)
3. M. Bremer, *In the Shadow of the Holocaust – The Changing Image of German Jewry After 1945* (United States Holocaust Memorial Museum, Washington, 2008)
4. E. Fredkin, T. Toffoli, Conservative logic. *Int. J. Theor. Phys.* **21**(3), 219–253 (1982)
5. G. Haus, A. Rodriguez, Music description and processing by Petri Nets, in *Advances in Petri Nets*, ed. by G. Rozenberg. *Lecture Notes in Computer Science*, vol. 340 (Springer, Berlin, 1988), pp. 175–199
6. B. Krämer, SEGRAS—a formal and semigraphical language combining Petri nets and abstract data types for the specification of distributed systems, in *Proceedings of the 9th International Conference on Software Engineering, ICSE '87* (IEEE Computer Society Press, Los Alamitos, 1987), pp. 116–125
7. B. Krämer, H.W. Schmidt, Types and modules for net specifications, in *Concurrency and Nets*, ed. by K. Voss, H.J. Genrich, G. Rozenberg (Springer, Berlin, 1987), pp. 269–286
8. R. Milner, Elements of interaction: Turing Award Lecture. *Commun. ACM* **36**(1), 78–89 (1993)
9. L. Ojala, O.-M. Penttinen, E. Parviainen, Modeling and analysis of Margolus quantum cellular automata using net-theoretical methods, in *Applications and Theory of Petri Nets*, ed. by J. Cortadella, W. Reisig. *Lecture Notes in Computer Science*, vol. 3099 (Springer, Berlin, 2004), pp. 331–350
10. C.A. Petri, Kommunikation mit Automaten, Ph.D. thesis, University Bonn, 1962
11. C.A. Petri, Concurrency, in *Net Theory and Applications*, ed. by W. Brauer. *Lecture Notes in Computer Science*, vol. 84 (Springer, Berlin, 1980), pp. 251–260
12. C.A. Petri, Introduction to general net theory, in *Net Theory and Applications*, ed. by W. Brauer. *Lecture Notes in Computer Science*, vol. 84 (Springer, Berlin, 1980), pp. 1–19
13. C.A. Petri, Cultural aspects of net theory. *Soft Comput.* **5**(2), 141–145 (2001)
14. W. Reisig, Petri nets and algebraic specifications. *Theor. Comput. Sci.* **80**(1), 1–34 (1991)
15. W. Scharlau, *Das Glück Mathematiker zu sein: Friedrich Hirzebruch und seine Zeit* (Springer, Berlin, 2017)

---

<sup>26</sup><https://www.rd-alliance.org/>, of which I am a member.

16. H.W. Schmidt, *Specification and Correct Implementation of Non-sequential Systems Combining Abstract Data Types and Petri Nets* (Oldenbourg, Munich, 1989)
17. H.W. Schmidt, Prototyping and analysis of non-sequential systems using predicate-event nets. *J. Syst. Softw.* **15**(1), 43–62 (1991)
18. E. Smith, *Carl Adam Petri: Eine Biographie* (Springer, Berlin, 2014)

# Petri Nets Are (Not Only) Distributed Automata



David de Frutos Escrig

## 1 From Automata to Petri Nets: Simple but Productive

Let me start by expressing my gratitude to *Carl Adam Petri* for inventing *Petri Nets*, which have been the topic of an important part of my research and therefore of my enjoyment while playing the game of *Science*. Petri Nets have, for sure, one of the most important features of a useful and interesting theory, viz., *simplicity*. Two bright scientists who have worked on the formalisation of *Programming* and *Concurrency* emphasised this motto throughout their career, with titles and quotations that will remind us of this feature forever: *Beauty is our Business* [8], a compendium of papers that honours *Edsger W. Dijkstra*, and the famous sentence by *C.A.R. Hoare* from his *Turing Award Lecture* [9]: “There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies and the other way is to make it so complicated that there are no obvious deficiencies.”

Like these two geniuses, and many others, Petri contributed to the development of Science and Engineering by proposing some *simple* but *revolutionary* ideas that paved the way for new approaches that have proved to be extremely successful in many applications. Certainly, to make them applicable, it is also customary that we need to partially abandon that simplicity, because *real systems* are unfortunately contaminated by various practical aspects that make them complicated to develop and study. Fortunately, most of these visionaries got their recognition sooner or later during their careers, and this was also the case for Petri. But at the same time, it is not uncommon that the simplicity of their proposals is used by some people to undervalue their goals, with the argument that anybody would have

---

D. de Frutos Escrig (✉)

Departamento de Sistemas Informáticos y Computación - Facultad CC. Matemáticas, Universidad Complutense de Madrid, Madrid, Spain

e-mail: [defrutos@sip.ucm.es](mailto:defrutos@sip.ucm.es)



eventually proposed such “obvious” (because of their simplicity) theories. I have the impression that Petri was, for sure unfairly, sometimes treated that way, possibly as a consequence of the fact that he preferred to concentrate on the conceptually essential aspects of his developments and thus remained intentionally outside (or at least far away from) the developments that later made his proposals applicable, and thus popular. His personal attitude towards life, and in particular the main interests that guided his scientific career, are wonderfully described in his recent biography by Smith [17].

In fact, Petri nets were initially conceived as a way of describing chemical reactions. Therefore, we might say that there was a bit of *serendipity* in the fact that they were later recognised (by Petri himself!) to be a great mechanism to describe the essentials of *Concurrent Computation*. But again, many important discoveries were connected to this serendipitous path, mainly because very often it is difficult to distinguish *casuality* from *causality*.

In his Ph.D. Thesis *Kommunikation mit Automaten*, Petri presents the essential features of *Distributed-Concurrent Computation* and demonstrates how they are (simply!) represented in his net models: A *synchronous* execution of independent actions, which at the same time are (either alone or when all in a certain group have been executed) the *cause* that makes possible some new executions; also possibly producing *choices*, when several possible futures are in *conflict*, together with the idea of *transformation* (data are transformed, and therefore consumed, when producing the corresponding result).

When we nowadays contemplate the definition of Petri Nets, it is not unusual to see them as the *natural* distributed version of classical *Automata*. In fact, Petri uses the term “automata” in the title of his thesis, although that title was so short (and thus simple!) that many quite different interpretations of his intention in choosing that title have been considered. But let us discuss in more detail that parallelism. A classical automaton has an (atomic) state, which can be changed when it reads a letter of the input word. Then the word is accepted whenever the state reached after the whole word has been read is in a certain category of accepting states. Now we distribute both the state and its changes. But the most important change is that we totally change the nature of the gadget: instead of viewing it as a *language recogniser* we look at it as a *behaviour producer*. Thus, we are interested in the sequences of transition steps that can be produced from a certain initial state. We talk about steps here because we also consider the *parallel* execution of some transitions at the same time (which produces a single step). The choices discussed above will possibly produce a *non-deterministic* behaviour. Certainly, classical automata can also be non-deterministic, but in this case this is just a technical convenience, since it is easy to turn any non-deterministic automation into an equivalent deterministic automation. Instead, non-determinism is an essential feature of net behaviours that results as a consequence of (resolution of) conflicts.

Therefore, a simple generalisation produces revolutionary changes whenever we (read “originally Petri” here) notice that the *convenient* way of using the generalised gadget is simply the *opposite* to the way that the original item was used. There are other consequences of the change of our point of view: now the input must

be reflected at the initial state from which we start to compute; this input is *transformed* by the execution of the induced steps of transitions, which become the expected output; the classical *input-output* finite computations to compute mathematical functions are turned into *natural* infinite computations that simply produce behaviours constituted by a sequence of steps. And all these changes just “come for free”, as a consequence of the adequate reading of the provided definition.

A very interesting fact that also appears in most of the important developments related to the *Theory of Computation* is that Petri understood that in order to develop a “good” theory, some clearly unimplementable premises, such as unboundedness of machines, must be assumed. At the same time, Petri nets become interesting in applications when they are defined as finite objects. However, unboundedness must remain somewhere if we want to claim that they are *strong enough* as computation mechanisms. An alternative way of getting this unboundedness was represented by *Place-Transition Nets*, where places can support a set of tokens, instead of just Boolean information, as was the case for the simpler *Case Event Nets*, originally defined and studied by Petri. Of course, when the former were introduced, Petri was aware of their importance and applicability, but as stated already, he remained loyal to his original model, mainly because once the key features to express concurrency were already present, they could be studied without being disturbed by technical obstacles caused by any *unnecessary* extension.

It was a great honour and quite a privilege for me to have had an opportunity to meet Petri several times during my career. It is true that my own research has been devoted to Place-Transition Nets and several of their variants, but it would definitely have not been possible without Petri’s pioneering contributions. Next, I will give a short review of them, stating the new features that my colleagues and I have studied, thus modestly contributing to the development of Petri net theory. Its enormous size nowadays proves the huge importance of the *simple*, but extremely clever, seed that Petri planted in the forest of Science.

## 2 Decidability of Properties of Some Variants of P/T Nets

Whenever decidability and/or complexity of properties of any kind of system are considered, it is essential that those properties are defined over an infinite set, for instance the whole set of natural numbers  $\mathbb{N}$  or any of its product sets  $\mathbb{N}^k$ . This is clearly the case when P/T nets with arbitrary initial markings are considered, *Monotonicity* is an essential property of plain P/T nets, which makes impossible the correct encoding of any complete *conditional* test. As a consequence, P/T nets cannot be *Turing-complete*, because *Reachability*, which somehow includes the *Termination Problem* for Petri nets, turns out to be decidable. Instead, it becomes undecidable whenever *inhibitor arcs* are added to the basic model. Once we know that plain P/T nets are monotonic, a natural property that somehow replaces reachability in many situations is *Coverability*, where we are interested in the reachability of any marking *covering* (thus being larger than) some given

marking. This is also decidable for ordinary nets, and in fact much easier to decide, using any variant of the *coverability tree* [11].

Our first work in this field was devoted to the study of the decidability of another related property: *Home States*. A home state of a system is a marking that can always be reached from any reachable marking of the system. I proved that this property is also decidable for any P/T system [4] and later we proved in [6] that this is still the case when we turn single states into their coverings, thus getting the notion of *Home Space*.

When considering extensions of this basic model, we have found several situations where the thin borderline between decidable and undecidable models is reached, for instance, in the case of *timed-arc* Petri nets, where the tokens have an associated *age* and the precondition arcs that control the firing of transitions can take into account the age of the tokens to be consumed by their firing. We found such a borderline situation because *timed reachability* was proved undecidable [16], while instead *timed coverability* properties remain decidable [7]. It is really curious how the precise way in which time is added to nets is responsible for reaching that borderline. When time is just associated with the duration of transitions, timed reachability remains decidable [15].

This shows that whenever we make an extension to the basic model, we need to carefully study the resulting expressive power. For example, when the expressive power provided by inhibitor arcs is reached, the (full) obtained model becomes unmanageable. In such cases, we should look for adequate constraints by means of which the effect of added features is somehow “tamed”, so that the systems resulting from those limitations remain manageable. In a similar way, when the precise border between decidability and undecidability is reached, we can still analyse the constructed systems as long as the properties to be analysed remain on the safe side. If that were not the case, we would need again to look for limitations on the use of the extended model that keep it manageable. Finally, when there is no increase of the expressive power, we can safely use the obtained model, even though this will not represent any “added value”. However, the fact that the corresponding feature has been explicitly introduced will make the obtainable systems more readable, even if the use of the available procedures to check their properties might require a costly elimination of the syntactic sugar represented by the new incorporated features.

More recently, we studied other, more sophisticated, extensions where *security* features were considered. In particular, we studied nets with *replicable* components, which somehow correspond to the idea posed by Petri of having nets with unbounded structure, and also nets including tokens with individual *identity* (*pure names*, using the technical terminology), so that these identities can be (only once) produced, copied under strict control, and compared (by means of an *equality* operator). Again, the borderline between decidability and undecidability is reached by both extensions. As a matter of fact, they were proved to be absolutely equivalent (i.e. each one can be reduced to the other) [13]. The surprising result came when we considered both extensions at the same time. Then, even coverability becomes undecidable, so that we need some limitations on the use of the added features, if we want to get a manageable model [14].

I conclude the presentation of the research on the extension of P/T nets that we have done by referring to *Ambient nets*. They constitute a combination of the *Ambient Calculus* [3], developed by Luca Cardelli and Andrew Gordon, and Petri nets, so that transitions can be used either to *move or to open* the represented ambients. In this way we obtained a structured model where *mobility* of components is a primitive feature [5]. This work was a continuation of previous work, where we developed our *TPBC* [1], a timed extension of the *Petri Box Calculus* [2], developed by Eike Best, Raymond Devillers, and Maciej Koutny. Their original proposal looked for an algebraic treatment of nets that allows a modular/composable development of nets in a structured way. For sure, the lack of a primitive to organise big nets into (composed) components in a systematic way is the main drawback of Petri nets. As stated in the first section of this paper, Petri nets can be seen as a distributed version of automata, and therefore are (initially) conceived as state machines, whose behaviour is globally defined, even if this tries to capture a *local* essence (each transition is only connected to a small number of places in its vicinity). But the whole behaviour is defined as the unstructured aggregation of those local firing rules, so that no systematic modular design is originally supported.

The original PBC included a Petri net *Algebra*, for which an *operational semantics* was defined by a set of SOS-rules. It also supports a *denotational semantics* that defines the net that captures the intended behaviour of each term from the algebra, including the interpretation of recursively defined terms by means of an adequate *fixpoint* operator. Moreover, a modular approach was also possible by means of the so-called *operator boxes*. We extended all these elements to the timed case, thus obtaining our TPBC [1].

As succinctly presented above, our work mainly concentrates on the study of the expressiveness of several extensions of the basic model of P/T nets. This research illustrates the potential flexibility of the basic model of Petri nets, which can be extended, more or less naturally, by introducing different mechanisms to capture the corresponding added features in a direct way. By the way, there are many other characteristics that I have not cited here, such as probabilistic information (where we also made a short incursion [12]), values as tokens, and in general *Coloured* Petri nets [10], which enabled the development of practical tools such as *CPNs*, by Kurt Jensen, which offered to the users the possibility to develop and analyse quite useful systems.

But there are still many other conceptual and practical issues that underscore the attractiveness of Petri nets. I am sure that most of them are discussed somewhere else in this volume, but just to mention some of them, we have the *Structural theory*, that studies the behavioural properties that can be inferred from some particular properties of the *form* of a net; the *Algebraic properties*, that can be obtained from the *Matrix* representation of the firing rule; the *Language theoretical* issues related to the use of nets as language generators; and many others.

## References

1. O. Marroquín-Alonso, D. de Frutos-Escrig, Extending the Petri box calculus with time, in *Application and Theory of Petri Nets 2001, 22nd International Conference, ICATPN 2001*, ed. by J.M. Colom, M. Koutng. Lecture Notes in Computer Science, vol. 2075 (Springer, Berlin, 2001), pp. 303–322
2. E. Best, R. Devillers, M. Koutny, *Petri Net Algebra* (Springer, Berlin, 2001)
3. L. Cardelli, Abstraction for mobile computation, in *Secure Internet Programming, Security Issues for Mobile and Distributed Objects*, ed. by J. Vitek, C.D. Jensen. Lecture Notes in Computer Science, vol. 1603 (Springer, Berlin, 1999), pp. 51–94
4. D. de Frutos-Escrig, Decidability of home states in place transition systems. Technical report, Dpto. Informática y Automática. Univ. Complutense de Madrid (1986)
5. D. de Frutos-Escrig, O. Marroquín-Alonso, Ambient Petri nets. *Electron. Notes Theor. Comput. Sci.* **85**(1), 39 (2003)
6. D. de Frutos-Escrig, C. Johnen, Decidability of home space property. Technical report, Laboratoire de Recherche en Informatique. Univ. de Paris-Sud, Centre d’Orsay, Report LRI-503 (1989)
7. D. de Frutos-Escrig, V. Valero-Ruiz, O. Marroquín-Alonso, Decidability of properties of timed-arc Petri nets, in *Application and Theory of Petri Nets 2000, 21st International Conference, ICATPN 2000*, ed. by M. Nielsen, D. Simpson. Lecture Notes in Computer Science, vol. 1825 (Springer, Berlin, 2000), pp. 187–206
8. W.H.J. Feijen, A.J.M. van Gasteren, D. Gries, J. Misra, *Beauty Is Our Business: A Birthday Salute to Edsger W. Dijkstra* (Springer, Berlin, 2011)
9. C.A.R. Hoare, The emperor’s old clothes. *Commun. ACM* **24**(2), 75–83 (1981)
10. K. Jensen, L.M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems* (Springer, Berlin, 2009)
11. R.M. Karp, R.E. Miller, Parallel program schemata. *J. Comput. Syst. Sci.* **3**(2), 147–195 (1969)
12. H. Macià, V. Valero-Ruiz, F. Cuartero, D. de Frutos-Escrig, A congruence relation for sPBC. *Formal Methods Syst. Des.* **32**(2), 85–128 (2008)
13. F. Rosa-Velardo, D. de Frutos-Escrig, Name creation vs. replication in Petri net systems. *Fundam. Inform.* **88**(3), 329–356 (2008)
14. F. Rosa-Velardo, D. de Frutos-Escrig, Decidability problems in Petri nets with names and replication. *Fundam. Inform.* **105**(3), 291–317 (2010)
15. V. Valero-Ruiz, D. de Frutos-Escrig, F. Cuartero, Simulation of timed Petri nets by ordinary Petri nets and applications to decidability of the timed reachability problem and other related problems, in *Proceedings of 4th International Workshop on Petri Nets and Performance Models, PNPM 1991* (IEEE, Los Alamitos, 1991), pp. 154–163
16. V. Valero-Ruiz, D. de Frutos-Escrig, F. Cuartero, On non-decidability of reachability for timed-arc Petri nets, in *Proceedings of 8th International Workshop on Petri Nets and Performance Models, PNPM 1999* (IEEE Computer Society, Los Alamitos, 1999), pp. 188–196
17. E. Smith, *Carl Adam Petri: Life and Science* (Springer, Berlin, 2015)

# Petri Nets: A Simple Language and Tool for Modeling Complex Ideas



Ryszard Janicki

I first heard the name ‘Petri nets’ in the Fall of 1976. I was finishing my Ph.D. (Theory of Coroutines) in the Computing Center (now the Institute of Computer Science) of the Polish Academy of Sciences in Warsaw under the supervision of Antoni Mazurkiewicz. He and Józef Winkowski were looking for a convenient tool to deal with concurrency problems and organized a series of seminar talks on Petri nets. They were aware of Petri’s ideas from his talk and paper presented at the 2nd International Conference on Mathematical Foundations of Computer Science (MFCS’73) in Štrbské Pleso, Czechoslovakia in 1973 [10]. Very soon Petri nets became the tool of choice for many people doing research in concurrency and loosely connected to the Mazurkiewicz and Winkowski groups.

My first paper involving Petri nets was published in 1978 [3] and in the same year I had the pleasure of meeting Carl Adam Petri in person. This happened in Zakopane, Poland, during the MFCS’78 conference. We discussed—well I was mainly listening—the relationship between nets, time, space, and composing concurrent systems from sequential components. Professor Petri did not publish too often, but all of his papers had a huge impact. The topics we discussed in Zakopane in 1978 were eventually published by Professor Petri in 1996 [11]. This was my only meeting with Carl Adam Petri; in those times I had more personal contacts with his disciples and colleagues from GMD, Bonn, such as Hartman Genrich, Kurt Lautenbach, and P. S. Thiagarajan. Yet, I believe this short encounter convinced me that I should use Petri nets to model or at least to test and to illustrate my models of concurrent operators and systems. And I did. Since 1978 concurrency theory has been one of my major research topics and I use the Petri nets approach and philosophy very often [4–7].

---

R. Janicki (✉)

Department of Computing and Software, McMaster University, Hamilton, ON, Canada

e-mail: [janicki@mcmaster.ca](mailto:janicki@mcmaster.ca)

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,

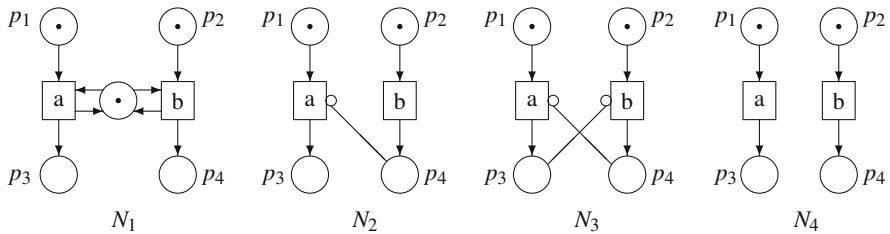
[https://doi.org/10.1007/978-3-319-96154-5\\_9](https://doi.org/10.1007/978-3-319-96154-5_9)

One of the fundamental assumptions underlying the *initial* Petri nets theory was that independent events (occurrences of actions) may be observed in any order. Sequences (or step sequences) that differ only w.r.t. their ordering of independent events are identified as belonging to the same concurrent run of the system under consideration. This assumption was adopted by Antoni Mazurkiewicz among others when he introduced the idea *traces* as equivalence classes of sequences comprising all (sequential) observations of a single concurrent run [9], and by Maciej Koutny and myself in our work on maximal concurrency and the reduction of reachability graphs [4].

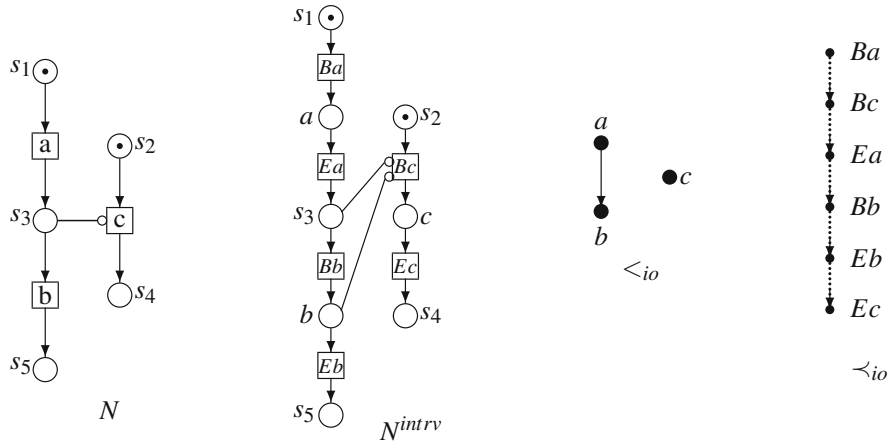
For me, and many others, the main attractions of Petri nets are their simplicity, flexibility, and ability to model complex structures involving concurrency in a relatively simple and convincing manner. There are very many different extensions of the original Petri nets. One of the first simple but very powerful extensions was the addition of *inhibitor arcs* by T. Agervala and M. Flynn in 1973 [1].

Inhibitor arcs allow a transition to check for the absence of a token. For example, the arcs between transition *a* and place *p*<sub>4</sub> in the nets *N*<sub>2</sub> and *N*<sub>3</sub> and between transition *b* and place *p*<sub>3</sub> in the net *N*<sub>3</sub> from Fig. 1 are inhibitor arcs. They were introduced in [1] to solve a synchronization problem not expressible in classical Petri nets. In principle they allow ‘test for zero’, an operator the standard Petri nets do not have. Elementary Petri nets with inhibitor arcs are very simple. They are just classical one-safe place-transition nets without self-loops extended with inhibitor arcs (cf. [5]). Nevertheless they can easily express complex behaviors involving ‘not later than’ cases [5, 7, 8], priorities, various versions of simultaneities, etc.

They also allow a very simple representation of more refined versions of (generalized) independency relations. Figure 1, which is a motivating example in [5], illustrates four basic cases of different behaviors that might be generated by two, generally independent, events *a* and *b*. This and similar examples modeled with Petri nets were crucial in the development of the theory of concurrency, where causality is represented by two distinct relations instead of just partial orders (first proposed in [5]), and in various generalizations of traces [6, 7]. The situation represented by



**Fig. 1** Simple inhibitor nets illustrating four distinct behaviors generated by two events *a* and *b* that can be interpreted as independent (w.r.t. generalized independency). The net *N*<sub>1</sub> allows sequences *ab* and *ba* but not a step sequence  $\{a, b\}$ , *N*<sub>2</sub> allows a sequence *ab* and step sequence  $\{a, b\}$  but not the sequence *ba*, *N*<sub>3</sub> only allows a step sequence  $\{a, b\}$ , and *N*<sub>4</sub>, which is a standard elementary Petri net, allows sequences *ab*, *ba* and the step sequence  $\{a, b\}$



**Fig. 2** Illustration of ‘holding a token’ semantics for inhibitor nets. The interval order  $<_{io}$  is an intuitively possible run in the net  $N$  (by ‘holding a token’ in  $c$  when  $a$  ends and  $b$  starts). The interval order  $<_{ia}$ , is represented among others by the total order  $<_{io}$  which is itself represented by a sequence  $s = BaBcEaBbEbEc$ . The sequence  $s$  is a firing sequence of the net  $N^{intrv}$

the net  $N_1$  of Fig. 1 led to the introduction of *mutex arcs* by Kleijn and Koutny [8], which made modeling of such cases simpler, more convincing, and more intuitive [7].

The models proposed and discussed in [5, 7] are not restricted to Petri nets, but Petri nets played a crucial role in their development and provided the basic intuitions and motivations.

It was argued by Norbert Wiener in 1914 [14] (and later more formally in [5]) that any execution that can be observed by a single observer must be an interval order. This implies that the most precise observational semantics is defined in terms of interval orders. However, generating interval orders directly is problematic for most models of concurrency, including Petri nets. This is because the only feasible sequence representation of interval order is by using the Fishburn Theorem<sup>1</sup> [2] and the sequences of *beginnings* and *endings* of events involved (usually denoted by  $Bx$  and  $Ex$  for an event  $x$ ) [2, 5]. This makes modeling concurrent behaviors involving system runs represented by interval orders increasingly difficult when compared with runs represented by event sequences or step sequences. Here is the point where the flexibility and ease of constructing intuitive but formally correct extensions of Petri nets becomes very helpful. Figure 2 illustrates a transformation of an ordinary

<sup>1</sup>A partial order  $<$  (of events) on  $X$  is an *interval order* if and only if there are mappings  $B: X \rightarrow Y$ ,  $E: X \rightarrow Y$ , some  $Y$  and a total order  $<$  (of event beginning and endings) on  $Y$  such that  $B(x) < E(x)$  and  $x < y \Leftrightarrow E(x) < B(y)$ , for all  $x, y \in X$  (cf. [2, 5]). Often  $Bx$  and  $Ex$  are written instead of  $B(x)$  and  $E(x)$ . In Fig. 2, the total order  $<_{io}$  is a possible total order representation of the interval order  $<_{io}$ .



inhibitor net  $N$  into its interval representation  $N^{intrv}$  i.e., an inhibitor net where transitions are instantaneous (zero time) event beginning and ends. The nets  $N$  and  $N^{intrv}$  are equivalent in the sense that, when ‘holding a token’ operational semantics is applied to  $N$ , any interval order representing a run in  $N$  can be represented by the appropriate firing sequence of beginnings and ends in and vice versa all appropriate firing sequences of  $N^{intrv}$  have their interval order counterpart run in  $N$ . This example provided motivation and intuition to a recently published paper on modeling concurrency with interval orders [6] (which can be seen as a descendant of [5]).

If inhibitor arcs are not involved, transforming  $N$  into  $N^{intrv}$  is rather simple and intuitive: we just replace each transition  $t$  by the subnet  $Bt \rightarrow t \rightarrow Et$ , as often happens in timed Petri nets [15]. However for inhibitor arcs the situation is a little bit more complicated. Intuitively, assuming events  $a$ ,  $b$ , and  $c$  take finite time to execute and we can ‘hold a token’ when  $c$  is executed so it can overlap with both  $a$  and  $b$ , the interval order  $<_{io}$  from Fig. 2 is a legitimate run (from the initial marking  $\{s_1, s_2\}$  to the marking  $\{s_3, s_4\}$ ) in the net  $N$  from the same figure. However the step sequence  $\{a\}\{b, c\}$  is not, as a token in  $s_3$  after firing  $a$  prevents  $c$  from being fired, both individually and together with  $b$ . Hence  $BaEaBbBcEbEc$ , etc., must not be a firing sequence of  $N^{intrv}$  i.e., we must have the inhibitor arc connecting  $Bc$  with  $b$  in  $N^{intrv}$ . ‘Holding a token’ semantics was also used in other influential results involving interval orders such as [12, 13], albeit with different firing rules for nets with or without inhibitor arcs.

While Petri nets are not the only formalism that is able to represent and implement the model of concurrency based on interval orders proposed in [6], without the intuitions and motivations Petri nets provided, the model of [6] (and its ancestor [5]) would never have come to the existence.

To sum up, for almost 40 years Petri nets and also many of Petri’s ideas [10, 11] provided fruitful motivation and a convenient tool for most of my research related to concurrency and their traits can be found in most of my papers in this area of research.

## References

1. T. Agerwala, M. Flynn, Comments on capabilities, limitations and “correctness” of Petri nets. *Comput. Architect. News* **4**(2), 81–86 (1973)
2. P.C. Fishburn, Intransitive indifference with unequal indifference intervals. *J. Math. Psychol.* **7**, 144–149 (1970)
3. R. Janicki, Synthesis of concurrent schemes, in *Proceedings of MFCS’78*, ed. by J. Winkowski. Lecture Notes in Computer Science, vol. 64 (Springer, Berlin, 1978), pp. 298–307
4. R. Janicki, M. Koutny, Using optimal simulations to reduce reachability graphs, in *Proceedings of CAV’90*, ed. by E.M. Clarke, R.P. Kurshan. Lecture Notes in Computer Science, vol. 531 (Springer, Berlin, 1991), pp. 166–175
5. R. Janicki, M. Koutny, Structure of concurrency. *Theor. Comput. Sci.* **112**(1), 5–52 (1993)

6. R. Janicki, X. Yin, Modeling concurrency with interval traces. *Inf. Comput.* **25**(1), 78–108 (2017)
7. R. Janicki, J. Kleijn, M. Koutny, Ł. Mikulski, Step traces. *Acta Inf.* **53**, 35–65 (2016)
8. J. Kleijn, M. Koutny, Mutex causality in processes and traces of general elementary nets. *Fundam. Inf.* **122**(1,2), 119–146 (2013)
9. A. Mazurkiewicz, Concurrent program schemes and their interpretation. TR DAIMIPB-78, Computer Science Department, Aarhus University (1977)
10. C.A. Petri, Concepts of net theory, in *Proceedings of MFCS'73*, Štrbské Pleso (1973), pp. 137–146
11. C.A. Petri, Nets, time and space. *Theor. Comput. Sci.* **153**(1–2), 3–48 (1996)
12. R.J. Van Glabbeek, F. Vaandrager, Petri net models for algebraic theories of concurrency, in *PARLE Vol. II*, ed. by J.W. de Bakker et al. Lecture Notes in Computer Science, vol. 259 (Springer, Berlin, 1987), pp. 224–242
13. W. Vogler, Partial order semantics and read arcs. *Theor. Comput. Sci.* **286**(1), 3363 (2002)
14. N. Wiener, A contribution to the theory of relative position. *Proc. Camb. Philos. Soc.* **17**(5), 441–449 (1914)
15. W.M. Zuberek, Timed Petri nets and preliminary performance evaluation, in *Proceedings of the 7th Annual Symposium on Computer Architecture* (ACM, New York, 1980), pp. 89–96

**Part II**  
**Personal Recollections**

# Carl Adam Petri: A Tribute from Aarhus



Kurt Jensen and Mogens Nielsen

At Aarhus University we are delighted to be given the opportunity to pay a special tribute to Carl Adam Petri. We were both fortunate to meet Carl Adam on several occasions, but, more importantly, his visions and his research played a significant role in the early development of the scientific basis of our Department of Computer Science. As you will see, our concrete introduction to and our work with Petri Nets was not based on strong collaboration with Carl Adam himself, but rather with a number of his “disciples”.

Looking back in time, the foresight of Carl Adam Petri was truly astonishing, when he introduced the notion of Petri Nets in his Ph.D. dissertation in 1962, “Kommunikationen mit Automaten”. Remember that this was at a time when computing was conceived purely as a sequential concept with no notions of, e.g. parallelism and communication, and yet Petri had the vision to introduce his formalism of nets capturing fundamental concepts for computing as we know it today—notably the foundational concepts of concurrency, conflict and causal dependency. It took several years before we—as well as most of the rest of the world—discovered the significance of his ideas and visions.

Our Department of Computer Science at Aarhus University was formally established in 1971 as part of the Institute for Mathematics, with a curriculum focusing on the theoretical aspects of computing and with no elements of concurrency. This situation changed dramatically in 1977. At the time, Antoni Mazurkiewicz had just joined our department, and in the years to follow he played a significant role in our early development of concurrency, as witnessed, e.g. by his influential early paper on Trace Languages, written during his time in Aarhus. Importantly, one of Antoni’s first actions in 1977 was to invite two of Petri’s colleagues from Bonn, Hartmann Genrich and Kurt Lautenbach, to our department, where they gave an inspiring 1-

---

K. Jensen (✉) · M. Nielsen  
Department of Computer Science, Aarhus University, Aarhus, Denmark  
e-mail: [kjensen@cs.au.dk](mailto:kjensen@cs.au.dk); [mn@cs.au.dk](mailto:mn@cs.au.dk)

week course on the intriguing world of Petri Nets. We both attended the course with several of our colleagues, and the course would turn out to have a profound influence on our careers—and on the development of our department as a whole.

We were both fascinated by Petri's foundational ideas, and we both immediately took up the challenge of understanding and utilizing the insight provided to us—with different approaches and in different scientific directions.

Kurt's early approach was to investigate the expressive power of Petri Nets in defining the semantics of DELTA, a systems description language developed by Kristen Nygaard, also visiting our department at the time [3], and this led to a lifelong development of so-called High-Level Petri Nets, summarized in the development and applications of the formalism Colored Petri Nets and a range of associated tools [2]. It is worth acknowledging that the first notion of High-Level Petri Nets originated from Petri's group in Bonn, and also that the first Petri Net tool was developed for Petri's group in Bonn by Robert Shapiro. During a visit to GMD Kurt worked with a prototype of the tool. A few years later Robert Shapiro moved the tool to the Macintosh platform, and it became the basis for the development of the Design/CPN tool at Meta Software, Cambridge, Massachusetts. The main architects behind Design/CPN were Robert Shapiro, Kurt Jensen and Peter Huber, heading a large, highly international group of developers, including Hartmann Genrich from GMD.

Starting from the year 2000 a group of people at Aarhus University developed CPN Tools [1], which was heavily based on experiences with the practical use of Design/CPN. The main architects behind CPN Tools were Kurt Jensen, Søren Christensen, Lars M. Kristensen and Michael Westergaard. In 2010 CPN Tools was transferred from Aarhus University to the Technical University of Eindhoven; it then had more than 10,000 licenses. CPN Tools is still the leading tool for editing, simulation and analysis of High-Level Petri nets. The tool features incremental syntax checking and code generation, which take place while a net is being constructed. A fast simulator efficiently handles untimed and timed nets. Full and partial state spaces can be generated and analyzed, and a standard state space report contains information such as boundedness properties and liveness properties.

Mogens went to Edinburgh University for his postdoc in 1977, starting investigations on the theoretical foundation of Petri Nets with colleagues working on models for concurrency, notably Glynn Winskel and Gordon Plotkin, leading to many years working on foundational models for concurrency. At the time, formal calculi for concurrency were developed, in Edinburgh notably the calculus CCS by Robin Milner, and the predominant models for these calculi were so-called interleaving models, expressing concurrency in terms of non-determinism. Our original idea was to develop alternative models capturing Petri's fundamental concepts of, e.g. concurrency, conflict and causal dependency. Our first published result [5] provided a tight relationship between the world of Petri Nets and the world of domains developed in denotational semantics by Dana Scott, introducing the notion of event structures.

During the following years, many such alternative models were developed by researchers all over the world, at the time often referred to as “true concurrency” models, and after Mogens' return to Aarhus University, we were fortunate to be

able to establish a group of distinguished researchers contributing during the 1980s and 1990s to the foundation of models for concurrency.

Glynn Winskel joined our department in 1980, and in 1982 he introduced his groundbreaking categorical approach to models for concurrency [6]. Glynn's framework provided the mathematical foundation for understanding fundamental concepts like unfolding of models such as Petri Nets as well as formal relationships between models, as summarized in our survey of models for concurrency from 1995 [7]. It is worth noticing that many of the underlying concepts of models treated in [7] can be traced back to the original concepts and insights of Petri.

P.S. Thiagarajan arrived in 1983 from Petri's group in Bonn, and this led to many years of collaboration, starting with a paper on degrees of non-determinism and concurrency in Petri Nets from 1984 [4]. Vladimiro Sassone arrived at our department in 1992 from Ugo Montanari's group in Pisa, also leading to many years of collaboration on models for concurrency, and in more recent years also on other lines of research.

As indicated, the visit from Bonn to our department in 1977 was a true game-changer for both of us, but also for our department as a whole, where concurrency soon became a part of our curriculum—and this is still the case today.

We are grateful to have been a small part of the truly impressive development of science based on Carl Adam Petri's ideas over the many years since 1977, and we are delighted to have witnessed the development of a strong Petri Net community worldwide. In this connection, we must acknowledge the efforts of Grzegorz Rozenberg, who played an essential role in propagating Petri's ideas and in building the Petri Net community.

As mentioned above, we did not have a strong working relationship with Carl Adam himself. However, we both met and talked to Carl Adam on several occasions, and we were always impressed by his friendly personality, his open-mindedness to science, and his scientific generosity in conversations. As an example, we experienced with admiration his insight and genuine interest in practical applications and tool development—despite his own inclination for theoretical foundational work. His visions and insights had enormous impact not only on us and our department, but on computer science as a whole.

## References

1. CPN Tools homepage, [www.cpntools.org](http://www.cpntools.org)
2. K. Jensen, Coloured Petri nets. Basic concepts, analysis methods and practical use, in *Mono-graphs in Theoretical Computer Science*, vols. 1–3 (Springer, Berlin, 1992–1997)
3. K. Jensen, M. Kyng, O.-L. Madsen, A Petri net definition of a system description language, in *Semantics of Concurrent Computation*, ed. by G. Kahn. Lecture Notes in Computer Science, vol. 70 (Springer, Berlin, 1979), pp. 348–368
4. M. Nielsen, P.S. Thiagarajan, Degrees of non-determinism and concurrency: a Petri net view, in *FSTTCS'84*, ed. by M. Joseph, R. Shyamasundar. Lecture Notes in Computer Science, vol. 181 (Springer, Berlin, 1984), pp. 89–117

5. M. Nielsen, G. Plotkin, G. Winskel, Petri nets, event structures and domains, Part 1. *Theor. Comput. Sci.* **13**(1), 85–108 (1981)
6. G. Winskel, Event structure semantics for CCS and related languages, in *ICALP'82*, ed. by M. Nielsen, E.M. Schmidt. *Lecture Notes in Computer Science*, vol. 140 (Springer, Berlin, 1979), pp. 561–576
7. G. Winskel, M. Nielsen, Models for concurrency, in *Handbook of Logic in Computer Science*, ed. by S. Abramsky, D. Gabbay, T.S.E. Maibaum, vol. 4 (Oxford University Press, Oxford, 1995), pp. 1–148

# Some Interactions with Carl Adam Petri over Three Decades



Manuel Silva

Science and Technology are social constructions. Nevertheless, in their development some people contribute in outstanding ways. As recognized by Isaac Newton in a letter to Robert Hooke (1676): “If I have seen further, it is by standing on the shoulders of giants.”

In relatively very few cases, the name of a researcher is given to a theory for an entire subfield. This is the case with the so-called *Petri Nets* (PNs), a Systems Theory initially inspired by Carl Adam Petri (1926–2010). The first stone of this construction (in which Petri nets are not defined!) is his Ph.D. dissertation [6].<sup>1</sup> We can say that Petri was a mathematician—more precisely a systems theorist—in a computer science environment. Nevertheless, he was not a classical mathematician. Interested in the description of some real-life situations, he essentially worked at the conceptual level, providing foundations for new ways of representing systems. This may be viewed as a profile less frequently exercised with success than that of a “theorem prover.” In other words, he was much more conceptual than technical.

In view of the long *life cycle* of many systems (conception and modeling; analysis and synthesis from different perspectives; implementation and operation) and of the diversity of application domains, it seems desirable to have a *family* of formalisms rather than a collection of “unrelated” or weakly related formalisms. *Coherence* among models usable at different phases (untimed preliminary models, different kinds of timed models for performance evaluation and control, etc.), *economy* in the

---

<sup>1</sup>For its translation into English see: C. A. Petri, Communication with automata. Rome Air Development Center TR-65-377, New York, 1966.

---

M. Silva (✉)

Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, Zaragoza, Spain

e-mail: [silva@unizar.es](mailto:silva@unizar.es)



transformations, and *synergy* in the development of models and theories are among the expected advantages.

The conceptual seeds proposed by Carl Adam Petri do not just flourish as a single formalism, initially the so-called *Condition/Event nets*, where the states are expressed in terms of Boolean variables. They blossom as a set of formalisms that constitute the foundations of a *modeling paradigm*, a conceptual framework that allows us to obtain “derived” formalisms from some common concepts and principles. In any case, it is important to remark that proceeding within a given modeling paradigm does not mean at all that the work is not really multidisciplinary! In our view, the PN modeling paradigm derives from the “cross-product” of the different levels of *abstraction* in PNs (Condition/Event; Place/Transition; Colored. . .) and different *interpreted extensions* (for example, considering external events or conditions and actions, adding (quantitative) time in one of the many possible ways, etc.) [14].

I met Carl Adam Petri for the first time in the “capital of the French Alps,” in 1976. Hosted by Gabrièle Saucier and Joseph Sifakis, he and Anatole Holt visited the École Nationale Supérieure d’Informatique et de Mathématique Appliquées de Grenoble (ENSIMAG). At that time I was working at the Laboratoire d’Automatique de Grenoble (LAG) as an “allocataire de recherche CNRS.”

To meet him was an important goal for me. In fact, some time before, under the supervision of René David, I had started my Ph.D. My initial topic was based on interconnected finite-state automata (as the modeling formalism) and modular hardware implementation with CUSAs.<sup>2</sup> However at that time, *Programmable Logic Controllers* PLCs and microprocessors became promising means for implementing logic controllers. On the other hand, Petri Nets and their modeling and analysis advantages (and limitations) started to become better known. So I proposed to René David, my advisor and a long-time friend, to radically move the topic of my Ph.D. work towards PNs and programmed implementations. A drastic change!

I attended the talks of Petri and Holt at the ENSIMAG. In particular, I managed to have discussions with Carl Adam. I explained to him part of my difficulties with coding in sequence instructions that represent “parallel” evolutions in a logic controller, for example. The basic idea was that the underlying “serialization” allows the introduction of some wrong behaviors, hazards due to the “sequential programming” [12]. In fact, it quickly became obvious to me that the programmed implementation of net models was not the kind of problem that he was interested in. In fact, he was involved in questions of a quite different nature, among others, on *Synchronic Distance* (SD), a *metric* in the firing of transitions, a fundamental concept introduced in [7] and later generalized on several occasions (see, e.g., [15]). This clearly meant that our interests were almost “disjoint” at that time. However, this did change later on!

---

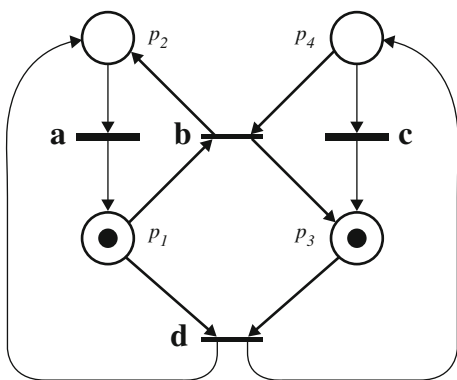
<sup>2</sup>*Cellule Universelle pour Séquences Asynchrones*, a CUSA is an interesting kind of memory cell that keep “hazards under control.” It was proposed by René David in his Thèse d’État in 1969. For a later but easy to reach reference, see [1].

Let me restrict this note to some points of interaction concerning two different topics, which we discussed several times. The first one that I briefly recall here is closely related to the already mentioned questions that were of interest to him in 1976. It led to the so-called *Synchrony Theory*. The second is *Fluidization* of discrete event models of net systems. He showed strong interest in both topics; even some potential cooperation was considered, although it was never realized.

The next interaction I evoke here took place in Milano in May 1986, at the International Seminar on Applicability of Petri Nets to Operations Research. It was organized by Anastasia Pagnoni at Bocconi University. This interaction continued in June, at the 7th European Workshop on Applications and Theory of Petri Nets (EWATPN, Oxford, June). At that time I was interested in dependencies between transition firings in (weighted) net models. I presented to Carl Adam some weaknesses of synchronic distance, a concept that does not take into account some important transition-firing dependences. My starting consideration was that SD-relations (they hold if there exists a weighting for the firing of transitions such that the SD-measure is finite) were useful to deal with dependences “like” stoichiometric ratios in chemistry, as they are concerned with some kind of linear (and fixed) dependencies in the firing count of very long firing sequences (i.e., some “rigid” relationships between transition firings in long runs). Nevertheless, many firing dependences were not captured. Therefore, together with Tadao Murata (University of Chicago), we were working on a generalization: Bounded-Fairness (BF) relations. Two transitions in a PN system are said to be in a BF-relation if there exists a positive integer  $k$  such that neither of them can fire more than  $k$  times without the other transition firing, in any firing sequence starting at any reachable marking. Of course, if two transitions are in a SD-relation, they are in a BF-relation, but the reverse is not true. As a matter of fact, Carl Adam always appreciated the net system given in Fig. 1, the “smallest” live, bounded, and reversible ordinary net system having two transitions in a BF-relation (transitions  $a$  and  $d$ ), but not in an SD-relation.

The following two interactions I want to briefly recall here happened at the University of Zaragoza. In 1987 we organized the 8th European Workshop on

**Fig. 1** Transitions  $a$  and  $d$  are in a BF-relation, but there does not exist a finite synchronic distance for them (see, e.g., [9, 11, 13])



Application and Theory of Petri Nets (EWATPN) and Carl Adam came to Zaragoza. With José Manuel Colom, we considered linear-programming techniques (LP) in order to compute certain *synchronic bounds* and to characterize the corresponding *synchronic relations*. Weak and strong *duality* theorems, *unboundedness* theorems, and *alternatives* theorems were jointly considered for the first time in a systematic, way in the context of analyzing Place/Transition models (for a revised version of the paper presented at the workshop see [11]), which led to several polynomial-time computations.<sup>3</sup> Even though the merging of PNs and LP was partly a computational topic, we still discussed it. Moreover, it would be easy to observe that this systematic use of LP consists of a *relaxation* of solutions into the non-negative reals, which can be “interpreted” as dealing with autonomous *continuous* PNs. In fact, at the same meeting, continuous PNs were defined at the net level, rather than at the level of the state or fundamental equation, by René David and Hassan Alla, our colleagues from Grenoble [2] (for broader perspectives on this kind of net systems, see [3, 10]). Nevertheless, continuous PNs were received with some reluctance by the community, just as timed PNs had been some years before. The suspicious reception at that time derives from the evaluation of their relevance to the series of EWATPN meetings, because “truly discrete” models were not being considered.

A special volume of “Advances in Petri Nets” with the title *Concurrency and Nets* [16] was prepared as a tribute to Carl Adam Petri on the occasion of his 60th birthday and it was delivered to him at the banquet of the EWATPN meeting in Zaragoza (Fig. 2). It was followed by “jotas aragonesas,” songs and dances from Aragón that Carl Adam really liked. Our paper “Towards a Synchrony Theory for P/T Nets” was published in that special volume [16].

Founded in 1974 as Escuela Técnica Superior de Ingenieros Industriales and renamed Centro Politécnico Superior (CPS) since 1989, our “young” school of engineering celebrated its 25th anniversary in April 1999. We decided to award the first three doctorates *Honoris Causa* in Engineering by our university, an institution founded by King Carlos I (Emperor Charles V) in 1542.<sup>4</sup> For that purpose, in accordance with the three classical basic pillars of technology, three relevant scientists were selected: one from *materials* (Steve Tsai, Stanford University), the second one from *energy* (Amable Liñán, Universidad Politécnica de Madrid), and the third one from *information*: Carl Adam Petri, as representative of Systems Theory and Computer Science (GMD, Bonn).

To honor Carl Adam, the day before the main ceremony we organized an international seminar with talks given by Gianfranco Balbo (“On Petri Nets

---

<sup>3</sup>Among the precedent works, Genrich and Lautenbach [4] use LPP for computing in *marked graphs* (a restricted subclass of ordinary net models for which the incidence matrix is fortunately *unimodular*). Additionally, Memmi and Roucairol [5] and Sifakis [8] use the so-called Minkowski-Farkas lemma or alternatives theorems in isolation.

<sup>4</sup>Therefore, in 2017 we celebrated its 475th anniversary. The foundation was made at the request of the syndics of Zaragoza. The emperor signed the privilege that elevated the medieval Estudio General de Artes de Zaragoza to the rank of university of all sciences in the Cortes de Monzón. This new academic rank of the old medieval institution was confirmed by pontifical authority in 1554, when Julius III issued a papal bull, confirmed by Paul IV in 1555.



**Fig. 2** Spanish researchers on concurrent systems with Carl Adam Petri at the 8th European Workshop on Applications and Theory of Petri Nets (Zaragoza, 1987). Some are engineers from companies such as Standard Eléctrica/ALCATEL or Siemens España, others are computer theorists and systems engineers from Universidad Complutense and Universidad Politécnica de Madrid, Universidad Politécnica de Valencia, and Universidad de Zaragoza, the host institution

and Performance Evaluation: The GSPN case,” Universitá di Torino), Jonathan Billington (“On the ISO/IEC Petri Net standard (15909),” University of South Australia), David de Frutos (“Decidable properties in Timed Petri Nets,” Universidad Complutense de Madrid), Maciej Koutny (“Combining Petri Nets and Process Algebras,” Newcastle University), and Manuel Silva (“On continuous Petri Nets,” Universidad de Zaragoza). Carl Adam found intriguing the “smooth” passage from discrete event systems to hybrid and continuous “relaxed” models. In particular, he was surprised by the fact that continuous timed net models under infinite server semantics may have discontinuous performance measures in steady-state (Fig. 3).

The last time I met Carl Adam Petri was in June 2005, in Miami, at the 26th International Conference on Applications and Theory of Petri Nets (ICATPN). I had the honor then to give a keynote on “Continuization of Timed Petri Nets: From Performance Evaluation to Observation and Control.” Therefore, we discussed the fluid “views” of discrete models, a topic that he openly declared as very interesting. Once again, he told me about the importance of the *duality* of reactants and reactions, which in time suggested to him the idea of the separation of places from transitions. In an enjoyable conversation I claimed that my game was “in moles, not in molecules, so I was able to consider fractions of moles.” Affectionately, he stated that my interest in fluid net systems was “not surprising,” because of my degree in Industrial-Chemical Engineering from the Universidad de Sevilla.

Carl Adam Petri was always a warm-hearted person.



**Fig. 3** At the Universidad de Zaragoza (15 April 1999): (1) Manuel Silva and José Manuel Colom Carl Adam Petri the *Honoris Causa* Doctorate; (2) At the main stairs of the Paraninfo building, with representatives of Australia, Canada, France, Italy, Spain, and the United Kingdom

## References

1. R. David, Modular design of asynchronous circuits defined by graphs. *IEEE Trans. Comput.* **26**(8), 727–737 (1977)
2. R. David, H. Alla, Continuous Petri nets, in *Proceedings of the 8th European Workshop on Application and Theory of Petri Nets*, Zaragoza (1987), pp. 275–294
3. R. David, H. Alla, *Discrete, Continuous and Hybrid Petri Nets*, 1st ed. (Springer, Berlin, 2010) (2004)

4. H.J. Genrich, K. Lautenbach, Synchronisationsgraphen. *Acta Inf.* **2**, 143–161 (1973)
5. G. Memmi, G. Roucairol, Linear algebra in net theory, in *Net Theory and Applications*, ed. by W. Brauer. Lecture Notes in Computer Science, vol. 84 (Springer, Berlin, 1979), pp. 213–223
6. C.A. Petri, Kommunikation mit Automaten. Ph.D. thesis, Technischen Hochschule Darmstadt, 1962
7. C.A. Petri, Interpretations of net theory. Technical Report 75/07, GMD, St. Augustin (1975)
8. J. Sifakis, Structural properties of Petri nets, in *Mathematical Foundations of Computer Science 1978*, ed. by J. Winkowski. Lecture Notes in Computer Science, vol. 64 (Springer, Berlin, 1978), pp. 474–483
9. M. Silva, Towards a synchrony theory for P/T nets, in *Concurrency and Nets*, ed. by K. Voss et al. (Springer, Berlin, 1987), pp. 435–460
10. M. Silva, Individuals, populations and fluid approximations: a Petri net based perspective. *Nonlinear Anal. Hybrid Syst.* **22**, 72–97 (2016)
11. M. Silva, J.M. Colom, On the computation of structural synchronic invariants in P/T nets, in *Advances in Petri Nets 1988*, ed. by G. Rozenberg. Lecture Notes in Computer Science, vol. 340 (Springer, Berlin, 1988), pp. 387–417
12. M. Silva-Suárez, R. David, On the programming of asynchronous sequential systems by logic equations, in *IFAC International Symposium on Discrete Systems* (Dresden, GDR), *IFAC Proceedings*, vol. 10(2), Part 2 (1977), pp. 120–129
13. M. Silva, T. Murata, B-fairness and structural B-fairness in Petri net models of concurrent systems. *J. Comput. Syst. Sci.* **44**(3), 447–477 (1992)
14. M. Silva, E. Teruel, A systems theory perspective of discrete event dynamic systems: the Petri net paradigm, in *Symposium on Discrete Events and Manufacturing Systems. CESA '96 IMACS Multiconference*, Lille, July 1996, pp. 1–12
15. I. Suzuki, T. Kasami, Three measures for synchronic dependence in Petri nets. *Acta Inf.* **19**(4), 325–338 (1983)
16. K. Voss, H. Genrich, G. Rozenberg (eds.), *Concurrency and Nets* (Springer, Berlin, 1987)

# Petri Nets and Petri's Nets: A Personal Perspective



**Maciej Koutny**

In the early 1960s, Carl Adam Petri made a fundamental contribution to Computer Science by introducing a model of non-sequential systems and non-sequential computation based on the explicit notion of concurrency, causality, and conflict. After that, his *Petri's nets* were the subject of extensive investigations, analyses, and practical implementations. In particular, the original model developed by Carl Adam Petri has been extended in several directions motivated by both theoretical and practical concerns, yielding high-level, stochastic, and hybrid Petri nets, to name but a few. Despite significant differences between some of these models, they still share essential characteristics of the original design, such as the notion of distributed state, and are therefore instances of the general *Petri nets*. Petri nets have been intertwined with my academic life right from the beginning. I was also lucky to discover a roadmap through Petri's nets thanks to personal, highly intense meetings with Carl Adam Petri, who exposed me to his particular views and results concerning the concurrency model that he had invented.

My first encounter with Petri nets happened in the academic year 1980/1981 during a course given at the Warsaw University of Technology by Ryszard Janicki. This optional course was at first treated with some caution by my fellow students, as formal models of concurrent systems were not covered by the standard curriculum at that time. However, the course and the topic of Petri nets itself turned out to be intriguing and inspiring, generating a lot of interest and indeed enthusiasm on our part. In particular, both I and my future wife Marta Pietkiewicz decided to continue the theme and in 1982 completed our M.Sc. dissertations on Petri nets under the supervision of the course lecturer who encouraged us to delve deeper into this fascinating subject. I feel that the adjective 'fascinating' is fully justified in this case as Petri nets introduce you into an immensely complicated world of

---

M. Koutny (✉)

School of Computing Science, Newcastle University, Newcastle upon Tyne, UK

e-mail: [maciej.koutny@ncl.ac.uk](mailto:maciej.koutny@ncl.ac.uk)

concurrent systems with a remarkable and unique ease and power, making sure that before long you are absorbed into their world hook, line, and sinker. Hence, not surprisingly, both Marta and I have subsequently completed our Ph.D. theses on problems belonging to the field of Petri nets and concurrent systems. Moreover, for several years we have been conducting collaborative research on topics concerned with the synthesis of Petri nets from their behavioural specifications.

Since joining Newcastle University in 1985, my research interests have been strongly influenced and linked with Petri nets, including both their theory and practical applications. Shortly after the move, on the occasion of the Petri Nets conference held in Oxford in 1986, it was possible for me to engage with the international community working on Petri nets. Attending this meeting was a significant step in my research career and it has since become a recurring item on my travel agenda (now also due to chairing its Steering Committee). In particular, attending the conference in 1986 led to closer involvement with the Petri nets community, resulting in participation in two EU Basic Research Action projects (Demon and Caliban) led by Eike Best, both concerned with further development of Petri nets and their applications. Working on these projects also provided an opportunity to meet in person with Carl Adam Petri. During two visits to GMD Bonn, around 1990, I was invited to personal meetings during which he explained several subtle points concerning Petri's nets, presenting in a crisp and methodical manner both challenging problems and his own preferred solutions, and discussing several tantalising ideas, both old and new. In this way, I became better acquainted with the fundamental ideas that influenced the development of Petri's nets as well as with Carl Adam Petri's personal view on the possible ways of developing them further. I still recall these meetings as something special, as they exposed me to new ways of thinking about the foundations of Petri nets. Without a doubt, they influenced my later work on extending the causal semantics to Petri nets with inhibitor arcs, including their process semantics.

Throughout my academic career I was mainly concerned with the theoretical aspects of Petri nets, collaborating, among others, with Eike Best, Raymond Devillers, Ryszard Janicki, Victor Khomenko, Hanna Kludel, Jetty Kleijn, Marta Pietkiewicz-Koutny, Brian Randell, Grzegorz Rozenberg, and Alex Yakovlev. However, I always felt that they are also a powerful model to be applied when solving important practical problems in various areas related to concurrent system design and analysis. A notable example of such an application is an Impact Case Study submitted by Newcastle University to the recent UK research assessment, REF 2014. The case study involved Petri net synthesis based on the theory of regions of transition systems as well as causality-based verification algorithms and tools, rooted in the theory of processes initiated by Carl Adam Petri. It was awarded the highest 4\* rating, demonstrating that Petri nets, providing a deep understanding and handling of state and action information, are a suitable model to deal with complex concurrent systems and their behaviours.

During the past 35 years I have been influenced in many ways by Carl Adam Petri's research, and I have derived true satisfaction from working on the model that he invented. I have also greatly enjoyed meeting Carl Adam Petri in person, both in



GMD and at other encounters, such as the award of the degree of Doctor Honoris Causa by the University of Zaragoza in 1999. In particular, I have found him to be very kind towards younger members of the academic community, and always keen to go to great lengths to open their eyes to the beauty of science and the power of logical thinking.

# Coffee and Cigarettes



Javier Esparza

“Coffee and Cigarettes” is a movie by Jim Jarmusch consisting of 11 independent episodes in which characters discuss quite extravagant topics—such as caffeine popsicles—while drinking coffee and smoking cigarettes (see Fig. 1). My only one-to-one encounter with Carl Adam Petri, in March 1989, could well have been the 12th episode of this movie. I didn’t drink any coffee at the time, and I have never smoked, but Petri drank and smoked for three. Since only Tom Waits is smoking in the figure, he’d be Petri, and I’d be Iggy Pop.

I was a young and inexperienced Ph.D. student. I had graduated in Physics less than 2 years before, and I had taken the impulsive and undocumented but, in retrospect, excellent decision of starting a Ph.D. in Petri net theory at the University of Saragossa under the supervision of Manuel Silva. My very first paper had just been accepted for presentation at the Petri Net Conference, and Eike Best had invited me to visit the Gesellschaft für Mathematik und Datenverarbeitung (GMD), where he worked as a researcher in Petri’s group, to give a talk.

I gave my talk in the morning. Petri didn’t attend, but somebody told him about it, because that afternoon Eike informed me that Petri wanted to talk to me the next day.

I had never met Petri before. When I entered his office, he was sitting at a table, big like a bear, with a mug of coffee in front of him and a menthol cigarette between his fingers—according to Eike Best, most likely the brand shown in Fig. 2.

During the next two hours Petri refilled his mug again and again, smoked one cigarette after another, and explained his current project to me. In the last years, he and his group had axiomatized causality and concurrency—more precisely, he had communicated his ideas in a sort of Socratic dialogue to the researchers of his

---

J. Esparza (✉)

Faculty of Computer Science, Technische Universität München, Garching bei München, Germany  
e-mail: [esparza@in.tum.de](mailto:esparza@in.tum.de)



Fig. 1 Iggy Pop and Tom Waits in "Coffee and Cigarettes" (2003)



Fig. 2 Petri's menthol cigarettes

group, who had developed them further and published them.<sup>1</sup> I knew that Petri was interested in Physics, and that he understood the theory of nonsequential processes as a sort of information theory compatible with the principles of special relativity. He told me that his new goal was to axiomatize not only causality and concurrency, but also what he called alternative events (today we would probably call them events in conflict). He explained that his notion of alternative was inspired by Quantum Mechanics. I'm not sure he explicitly mentioned the path integral formulation of Quantum Mechanics, where one cannot say which is the trajectory followed by a particle, only assign a probability to each trajectory, but that's what he seemed to have in mind.

Petri used two photocopies sketching the current version of his theory to explain his ideas, making corrections and additions with a red pen. After the meeting I kept the photocopies, which are lying on my desk as I type this note (Figs. 3 and 4). They contain a tentative definition of Complete Orders of Signals (COS). A COS is a tuple  $(H; al, bi, co)$ , where  $H$  is a set of events and  $al, bi, co \subseteq H \times H$  are relations on  $H$  satisfying 12 axioms (Fig. 3). The last three axioms,  $\Sigma_{10}$  to  $\Sigma_{12}$ , involve sets  $\mathbb{A}, \mathbb{B}, \dots, \mathbb{F}$ . On the second photocopy (Fig. 4) Petri wrote a note explaining their intended meaning: three sets of Alternatives,  $\mathbb{B}$ i-lines, and Cases, two sets called  $\mathbb{D}$ omain and  $\mathbb{E}$ ffect, and a set of  $\mathbb{F}$ ronds.<sup>2</sup>

Petri spoke slowly, making long pauses. I listened to him in awe, understood almost nothing, and struggled to understand the connection to Quantum Mechanics. At some point he mentioned Heisenberg's uncertainty principle, and I told him, trying to show off, that I didn't think Heisenberg had correctly interpreted his own principle. (I was just repeating what I had read in a paper ...) I could see that Petri was slightly upset, this mere pup questioning Heisenberg, such a great man ... Very gently, he told me how as a teenager he would cycle to Heisenberg's home just to see him pass by, too shy to talk to him. And I remember being moved by the tone of his voice and the respect for science pouring out of him.

After two hours Petri was not done yet, but we were interrupted by Eike, who opened the door, said he and others were going to lunch, and asked if I wanted to join. Petri mumbled something like "Oh yes, lunch ... You should go, young people have to eat. I never have lunch myself." I left with the two photocopies under my arm. I later learnt that Eike had come to rescue me, because one never knew how long a meeting with Petri could take. Jörg Desel, who also worked at GMD for several years,<sup>3</sup> told me that in one occasion Petri kept talking until dark, never bothering to switch the lights on.

---

<sup>1</sup>The similarity of the working processes of Petri and Socrates was pointed to me by Manuel Silva.

<sup>2</sup>I also own a second piece of Petri memorabilia: a Petri-Puzzle, signed by Petri, that was distributed at the ceremony in which he was handed the Siemens Ring by the German president Roman Herzog (Fig. 5). After the ceremony Petri sat at a table and we queued for him to sign our puzzles with his careful calligraphy.

<sup>3</sup>He even had to pass a job interview with Petri, which must have been an extraordinary experience.

COS( $\Sigma$ ) 1 10.3.89 Petri

Complete Orders of Signals

" $\Sigma$  is a system representing a complete order of signals iff axioms  $\Sigma 1 - \Sigma 12$  hold". Formally:

$COS(\Sigma) \Leftrightarrow \Sigma = (H; al, bi, co)$  with  $al, bi, co \subseteq H \times H$  and:

- $\Sigma 1: al \cup bi \cup co \neq H \times H$
- $\Sigma 2: al \cap bi = bi \cap co = co \cap al = id$  co  $\neq$  co  $\cup$  id
- $\Sigma 3: al^{-1} = al, bi^{-1} = bi, co^{-1} = co$   $\subseteq := \subset \cup id$
- $\Sigma 4: \tilde{al} = \tilde{bi} = \tilde{co} \quad \therefore = id$
- $\Sigma 5: al^* = bi^* = co^* \quad \therefore = H \times H$
- $\Sigma 6: P^2 = Q^2 = W^2 = \emptyset$   $\notin \emptyset$
- $\Sigma 7: \text{Within each } b \in B : (P \cup Q)^* = b \times b$
- $\Sigma 8: \text{within each } s \in S : al = id = co^2 \subseteq co = bi^2$
- $\Sigma 9: \text{Within each } \tau \in T : co = id = al^2 \subseteq al = bi^2$
- $\Sigma 10: A \otimes E, B \otimes F, C \otimes D$
- $\Sigma 11: D \times F \rightarrow A, E \times D \rightarrow B, F \times E \rightarrow C$
- $\Sigma 12: A \# B, B \# C, C \# A; A \# A, B = B, C \# C.$

Defs:

$P := \tilde{al} - (bi \cap co \cup al \cap bi)$	$x \tilde{al} y \Leftrightarrow \{z   z \tilde{al} x\} = \{z   z \tilde{al} y\}$
$WQ := \tilde{al} - (bi \cap al \cup co \cap bi)$	$al^* = \bigcup al^n : n \in \mathbb{N}, al^0 := id$

"crosses"  $A \otimes E \Leftrightarrow \forall a \in A, e \in E : a \cap e \neq \emptyset$

"separates"  $A \# B \Leftrightarrow \forall a \in A \vee b \in B : a \cap b = \emptyset$   
 $\wedge \forall b \in B \vee a \in A : a \cap b = \emptyset$

"intersect in"  $D \times F \rightarrow A \Leftrightarrow \forall d \in D, f \in F \forall a \in A : d \cap f = a$   
 $\wedge \forall a \in A \forall d \in D, f \in F : d \cap f = a$

Note:  $D \times F \rightarrow A \Leftrightarrow F \times D \rightarrow A$

Fig. 3 Petri's axioms for COS

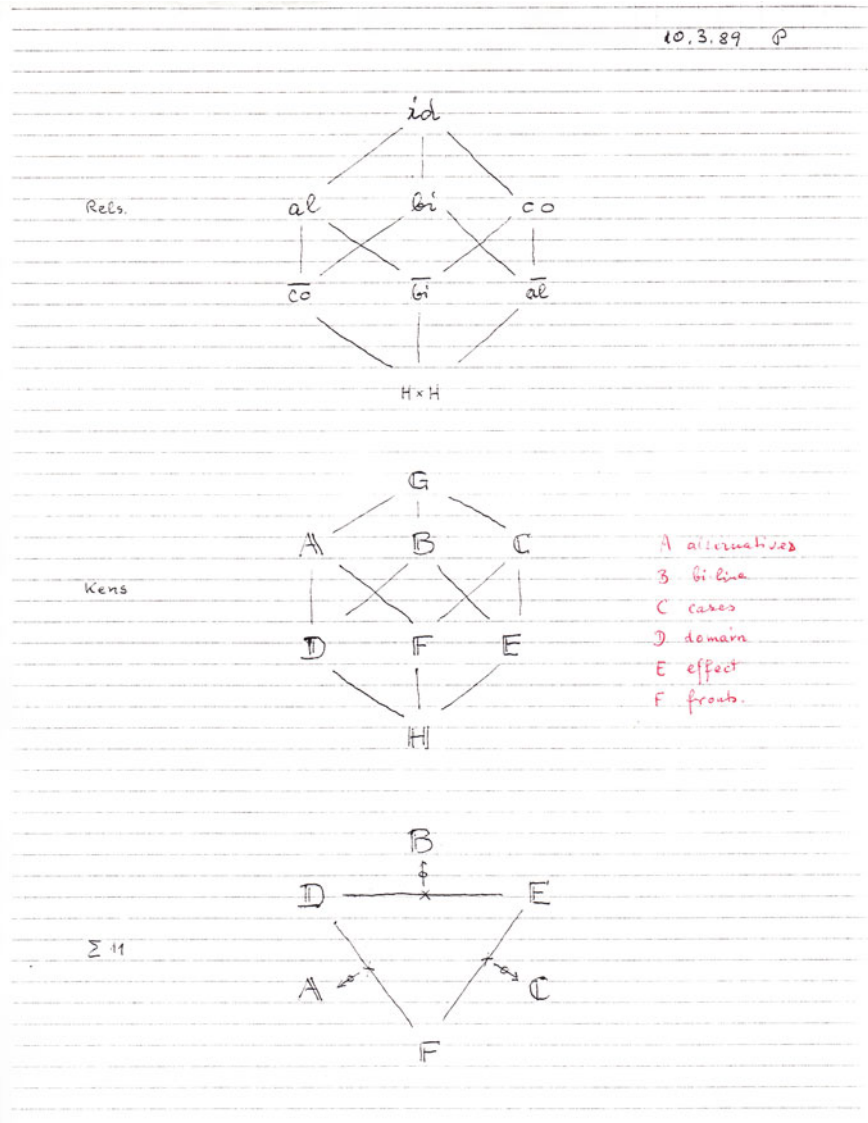
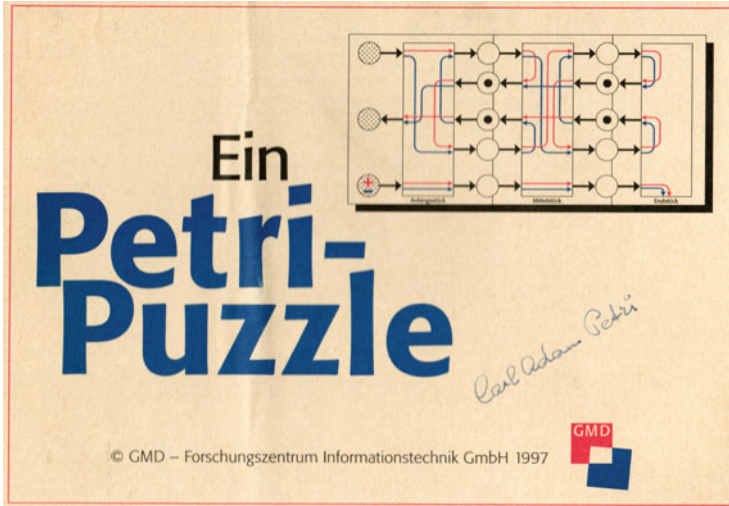


Fig. 4 Petri's sets for COS



**Fig. 5** A Petri puzzle, signed by Petri

End of story? Not quite... Some months after my meeting with Petri, Eike Best got a professorship at the University of Hildesheim, and offered me an assistant position. Inspired by Eike's work with César Fernandez [2], I started to think of applying the theory of nonsequential processes to model checking. In 1991 Eike and I wrote a paper [1], but I wasn't happy because our techniques only worked for nets with one single nonsequential process. Then I read a paper by McMillan [4] showing how to unfold a Petri net into an *occurrence net*—an extension of nonsequential processes with conflict due to Nielsen, Plotkin, and Winskel [5]—and how to use this unfolding for verification. McMillan's paper immediately reminded me of my encounter with Petri. Occurrence nets, one of the models Petri was struggling to capture axiomatically, were the model I had been looking for! I spent a large part of the next years developing an unfolding approach to model checking, which Keijo Heljanko and I finally published in a book [3].

So, even though that morning of March 1989 I didn't understand much and found Petri's ideas obscure, I ended up working on them for about 15 years. And the quest goes on: the best paper award of CONCUR 2015 went to a contribution by Rodriguez, Sousa, Sharma, and Kroening on combining unfoldings and partial order reduction [6].

I think this little story is typical of Petri's legacy. His ideas were often strange and sometimes extravagant, but always deeply original and powerful. So much so that, sooner or later, they have always ended up spreading throughout the world.

## References

1. E. Best, J. Esparza, Model checking of persistent Petri nets, in *Computer Science Logic, 5th Workshop, CSL '91, Berne, October 7–11, 1991, Proceedings*, ed. by E. Börger, G. Jäger, H. Kleine Büning, M. M. Richter. Lecture Notes in Computer Science, vol. 626 (Springer, Berlin, 1991), pp. 35–52
2. E. Best, C. Fernández, *Nonsequential Processes - A Petri Net View*. EATCS Monographs on Theoretical Computer Science, vol. 13 (Springer, Berlin, 1988)
3. J. Esparza, K. Heljanko, *Unfoldings - A Partial-Order Approach to Model Checking*. Monographs in Theoretical Computer Science. An EATCS Series (Springer, Berlin, 2008)
4. K.L. McMillan, Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits, in *Computer Aided Verification, Fourth International Workshop, CAV '92, Montreal, Canada, June 29–July 1, 1992, Proceedings*, ed. by G. von Bochmann, D.K. Probst. Lecture Notes in Computer Science, vol. 663 (Springer, Berlin, 1992), pp. 164–177
5. M. Nielsen, G.D. Plotkin, G. Winskel, Petri nets, event structures and domains, part I. *Theor. Comput. Sci.* **13**(1), 85–108 (1981)
6. C. Rodríguez, M. Sousa, S. Sharma, D. Kroening, Unfolding-based partial order reduction, in *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, September 1–4, 2015*, ed. by L. Aceto, D. de Frutos Escrig. Leibniz International Proceedings in Informatics, vol. 42 (Dagstuhl Publishing, Dagstuhl, 2015), pp. 456–469



# Early Interactions with Carl Adam Petri



**Brian Randell**

It is a great pleasure to provide a few words about my own personal memories of Carl Adam Petri, whose work has strongly influenced my research into a number of aspects of system dependability over the years.

I think I first learned of Carl Adam Petri and his work on net theory in the early 1970s from my colleague Peter Lauer. Peter had joined me at Newcastle University in 1972 from the IBM Vienna Laboratory—I had reached Newcastle from the IBM Research Laboratory in Yorktown Heights some 3 years earlier. At Newcastle, starting in about 1975, Peter used Petri Nets to provide a formal treatment [1] of the Path Expression concurrent-programming notation. (This notation had been invented by Roy Campbell, then one of our Ph.D. Students, and Nico Habermann, of Carnegie-Mellon University [2].) With this work, Peter initiated a still-continuing and indeed flourishing line of research at Newcastle on concurrency theory and applications, now led for many years by my colleagues Maciej Koutny and Alex Yakovlev.

Though my role in this concurrency research was mainly that of an admiring bystander, I was intrigued and attracted by what I learnt of net theory from Peter and his colleagues. I assume that this is why, in 1976, I had no hesitation in inviting Carl Adam Petri to Newcastle for the first, and I fear only, time. This was to take part in our International Seminar on Teaching Computer Science.

This series of annual seminars, to an audience mainly of senior computer science professors from across Europe, commenced in 1968 and continued for 32 years. It was sponsored for many years by IBM, thus enabling us to invite leading speakers from across the world, and to provide a very hospitable environment in which all the participants could interact intensively.

---

B. Randell (✉)

School of Computing Science, Newcastle University, Newcastle upon Tyne, UK

e-mail: [Brian.Randell@ncl.ac.uk](mailto:Brian.Randell@ncl.ac.uk)

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,

[https://doi.org/10.1007/978-3-319-96154-5\\_14](https://doi.org/10.1007/978-3-319-96154-5_14)

The 1976 Seminar was on Distributed Computing System Design. Carl Adam Petri was one of six speakers. (The others were Joel Aron, Michael Jackson, Bill Lynch, Doug Ross and Wlad Turski.) Carl Adam gave three excellent lectures. The texts of these lectures—two on the subject of “General Net Theory” [3], and one on “Communication Discipline” [4]—were included in the Report of this Seminar, which was widely distributed. They are, I was surprised to find, the fourth- and fifth-earliest original English-language publications by Carl Adam listed in the Petri Net Bibliography [5].

On rereading these lectures I can readily understand why I was so impressed when I heard Carl Adam present them. I cannot resist quoting his first lecture’s opening words:

Those of you who attended this conference last year may remember Anatol Holt’s lecture ‘Formal Methods in System Analysis’. My intention then, had been, this year to supplement his lecture by three hours of concentrated blackboard mathematics, because I felt that nothing needed adding to Holt’s lectures in terms of words and figures. But I now think I ought to keep the mathematics to a minimum, both in order to give a general idea of the content of the theory, and to raise the entertainment value from negative to zero.

In fact his lectures provided an excellent account of the basic concepts and the generality of the aims of net theory, illustrated by wonderfully simple yet subtle graphical examples—so a few years ago I took great pleasure in arranging for the texts of the lectures to be made available online.

To the best of my recollection the next occasion on which I had the pleasure of meeting, and listening to lectures by, Carl Adam Petri was at the 1979 Advanced Course on General Net Theory of Processes and Systems [6]. To my pleased surprise I had been invited by the Course Director, Wilfried Brauer (Univ. Hamburg), to be one of the Course Co-Directors, together with Carl Adam. Newcastle’s more substantive contribution to the Course was however provided by Eike Best, then one of Peter Lauer’s Ph.D. students, who gave two of the lectures.

I was pleased to find that the correspondence I had with Wilfried Brauer concerning this Course has been retained in my files. This indicates that the course was held at the University of Hamburg, 8–18 October 1979, “under the auspices of the Commission of the European Communities [and] financed by the Ministry for Research and Technology of the Federal Republic of Germany”. The (preliminary) course timetable indicates that the other lecturers, besides Carl Adam himself and Eike Best, were to be Hartmann J. Genrich (GMD, St. Augustin), Claude Girault (Univ. Paris VI), Matthias Jantzen (Univ. Hamburg), Kurt Lautenbach (GMD, St. Augustin), Jerry D. Noe (Univ. Washington), Horst Oberquelle (Univ. Hamburg), Suhas S. Patil (Univ. Utah), Gérard Roucairol (Univ. Paris VI), Robert M. Shapiro (Meta Software Corp.), Joseph Sifakis (IMAG Grenoble), P.S. Thiagarajan (GMD, St. Augustin), Rüdiger Valk (Univ. Hamburg) and Konrad Zuse (Zuse KG). Unfortunately, I was able to attend only part of the Course myself—but I was able to be present at the ceremony at which the University of Hamburg conferred an Honorary Doctorate on Konrad Zuse.

However, the main set of interactions I had with Carl Adam Petri, and the main focus of these reminiscences, was in connection with my membership, 2 years later, of the Progress Evaluation Committee that the Gesellschaft für Mathematik und Datenverarbeitung (GMD) established in 1981 to review the work of the ISF Institute that Carl Adam Petri directed at GMD St Augustin, Bonn. As a result I made a number of visits to Bonn in late 1981 and early 1982 to attend meetings of the Committee. (I vividly recall that several of these visits involved travel difficulties brought on by the wintry conditions.)

I believe these were my first visits to GMD. I was amused to find that, as with a number of other large governmental or industrial research laboratories with which I was familiar, GMD had taken over a very impressive mansion (Schloss Birlinghoven) and its grounds, in which a set of functional modern buildings had been constructed for the researchers, leaving the senior administrators to enjoy the architectural splendours of the original mansion. (These we eventually got to see when invited to a meeting in what I seem to recall was the Director's Office.)

The other members of the Committee were Wilfried Brauer, Herbert Fiedler (Univ. Bonn), Kristen Nygaard (Univ. Oslo) and Rainer Prinoth (GMD, Darmstadt), just two of whom I knew already (Brauer and Nygaard). I recall my surprise at learning, probably at the very first meeting, that the Committee had in fact been set up by the GMD Central Administration in the expectation of, perhaps the hope for, a negative evaluation. Indeed, it appeared they were seeking justification for closing down ISF! This was the background against which we had a number of very interesting discussions with Carl Adam and various other members of ISF, and other sections of GMD, during which we soon established that we were all favourably impressed by the work of ISF, and inclined to regard it as one of GMD's most impressive research activities. Thus rather than support the GMD Administration, we found ourselves becoming quite critical of it.

Unfortunately my files do not contain any documents about the work of the commission, and I do not have a copy of our final report. Indeed, it is perhaps unsurprising, given the embarrassment it must have caused, that no copies of the report can now be found. However Eike Best had retained, and has provided me with a copy of, the report's Summary of Recommendations. I find that these, though guardedly phrased, fit well with my recollection that our criticisms were mainly of GMD itself rather than Carl Adam and his Institute.

The first few recommendations were constructive suggestions concerning the Petri Institute's publication policy, and future work:

- a) With respect to net theory and its applications ISF should—concentrate on consolidating net theory—restrict application examples to a few important applications within computer science, and worked out in detail—publish its important original contributions in internationally established refereed journals—produce research monographs and textbooks—engage more intensively in teaching (university courses, summer schools, GMD seminars, etc.)

Our Report went on to make supportive recommendations concerning NETLAB, which was at the time a GMD Project, though later it seems that Aarhus University is where this project continued:

- b) The NETLAB project may be the starting point for the development of a “system development environment” building on the “world view” associated with net theory. We strongly recommend ISF and GMD to pursue the NETLAB project so as to make it useful both within and outside GMD.
- c) NETLAB should be a cornerstone in the much wider effort, integrating also a wide range of other facilities structurally compatible with the net theory approach. IST should immediately assign researchers to full-time involvement in the project, and at a later stage assume main responsibility for the NETLAB-related system development environment effort.
- d) ISF must, however, accept an ongoing commitment to the creation of further net theory-based tools which contribute to this environment.
- e) ISF, in addition to its basic research activities, should have personnel able to assist in practical problems related to applications of the theory.

There followed various other recommendations concerning some of the other groups in GMD whose work related to that of ISF, and the report concluded with some recommendations that were more general in nature, and implicitly quite critical of various aspects of the GMD Administration itself:

- k) GMD should reassess the suitability of its current methods of planning and reviewing the work of institutes, particularly as regard work largely of a fundamental and/or theoretical character.
- l) Consideration should be given to introduction of some scheme for enabling creative researchers to be rewarded and recognised appropriately without having to carry inappropriate administrative burdens.
- m) If the present system of projects is to be continued, means should be found for providing project leaders with authority commensurate with their responsibilities.
- n) The various administrative agencies within GMD should act simply as services on which institute directors and project directors can call, as of right, for purposes that are in line with their currently agreed project and institute plans.

In fact, GMD is not the only institution I am familiar with for which such recommendations are all too appropriate, so I am pleased to take this opportunity to quote them here, and plan to reuse them if and when appropriate circumstances arise again.

As I’ve already indicated, our Report was not well received by the GMD Administration, and my understanding is that Carl Adam and ISF did not gain as much benefit from the Committee’s praise and support as we hoped it would. Nevertheless, I recall my participation in the work of this Committee with some pride, given the high regard I had, and have, for Carl Adam’s work, and the extent to which my own and my colleagues’ research has benefitted, and indeed continue to benefit, from it. So I am delighted to have had this opportunity to shine a spotlight on this perhaps now little-known early assessment of the early work of his ISF Institute.

## References

1. P.E. Lauer, R.H. Campbell, A description of path expressions by Petri nets, in *Proceeding of 2nd ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages* (ACM, New York, 1975), pp. 95–105
2. R. Campbell, A.N. Habermann, The specification of process synchronization by path expressions, in *Proceeding of Symposium on Operating Systems*, ed. by E. Gelenbe, C. Kaiser. Lecture Notes in Computer Science, vol. 16 (Springer, Berlin, 1974), pp. 89–102.
3. C.A. Petri, General net theory, in *Proceeding of 1976 Joint IBM/University of Newcastle upon Tyne Seminar on Computing System Design* (Computing Laboratory, Newcastle upon Tyne, 1977), pp. 131–169, <http://homepages.cs.ncl.ac.uk/brian.randell/GeneralNetTheory.pdf>
4. C.A. Petri, Communication disciplines, in *Proceeding of 1976 Joint IBM/University of Newcastle upon Tyne Seminar on Computing System Design* (Computing Laboratory, Newcastle upon Tyne, 1977), pp. 171–183, <http://homepages.cs.ncl.ac.uk/brian.randell/CommunicationDisciplines.pdf>
5. *The Petri Nets Bibliography*. <http://www.informatik.uni-hamburg.de/TGI/pnbib/>
6. W. Brauer (ed.), *Net Theory and Applications*. Proceedings of the Advanced Course on General Net Theory of Processes and Systems. Lecture Notes in Computer Science, vol. 84 (Springer, Berlin, 1980)

# A Personal Journey in Petri Net Research



Xudong He

## 1 Learning Petri Nets in Graduate Schools

The first time I learned the concepts of Petri nets was during my MS degree study at Nanjing University, China in 1982. I was working on my MS thesis using formal method VDM for defining the semantics of the system programming language XCY developed by my advisor, Prof. Jiafu Xu (a computer science pioneer in China). Unsatisfied with the capability of VDM to deal with concurrency, I was looking for other formalisms for specifying concurrent systems. After talking to Prof. Xu about my ideas, he assigned me to review two journal papers on using formal methods for concurrent systems, one on Petri nets and the other on temporal logic.

After finishing my MS degree at Nanjing University in 1984, I went in 1985 to Virginia Tech in USA to pursue my Ph.D. In the first quarter at Virginia Tech, I took a computer architecture course that required a project. One of the project topics was to use a Petri net tool to model and simulate an architecture, which I selected happily and got a perfect score on the project. The experience of this project and the associated research, especially reading Prof. Reisig's paper on using Petri nets in software engineering [1], greatly influenced my dissertation research on integrating formal methods including Petri nets in software development. I finished my Ph.D. dissertation in 1989 and produced several journal publications on using predicate transition nets [2] in modeling systems [3] and integrating predicate transition nets with temporal logic [4].

---

X. He (✉)

School of Computing and Information Sciences, Florida International University, Miami, FL, USA

e-mail: [hex@cs.fiu.edu](mailto:hex@cs.fiu.edu)

## 2 Researching and Teaching Petri Nets at Universities

In 1989, I joined North Dakota State University (NDSU) as an assistant professor after I obtained my Ph.D. degree from Virginia Tech. There I continued my research on Petri nets. My initial focus was integrating predicate transition nets with other formalisms including algebraic specifications [5, 6] and Z [7] to combine the dynamic semantics and concurrency control structures of Petri nets with the rich data definition and abstraction capabilities of other methods. My subsequent research aimed at making Petri net models more modular by introducing hierarchies into predicate transition nets [8]. My last major research effort at NDSU was using hierarchical predicate transition nets to model object-oriented systems and various UML notations [9, 10].

I joined Florida International University (FIU) in 2000. My research focused on developing a software architecture modeling method based on predicate transition nets and temporal logic [11, 12]. At the same time, I worked on various analysis techniques for Petri nets including testing [13], scheduling techniques in dealing with time [14], and model checking through net translation [15]. I also applied Petri nets to model agent-oriented systems [16–19] and to support aspect-oriented modeling approaches [20]. Currently, my research work involves the development of a predicate transition net modeling and analysis environment PIPE+ [21–23] by leveraging several third-party analysis tools including model checker SPIN [24], term rewriting system Maude [25], and SMT solver Z3 [26].

During my 30-year professional career at NDSU and FIU, I graduated 15 Ph.D. students and 36 MS students. All my Ph.D. graduates' dissertations covered some aspects of Petri nets and many of my MS students also used Petri nets in their work. Petri nets have also been covered in several of my graduate-level courses including advanced software engineering, software specification, and software verification.

## 3 Organizing the ATPN Conference in Miami and Meeting Carl Adam Petri in Person

In 2005, I had the great honor and privilege to serve as the organizing chair of the 26th International Conference on Application and Theory of Petri Nets, in Miami. I had the great pleasure to invite Carl Adam Petri to give a keynote speech and subsequent talks at FIU for 3 weeks. During Carl Adam's 3 week stay in Miami, I had many interactions and discussed many interesting things with him. The wonderful memories are shown in Fig. 1 ((1) Carl Adam in my lab with my graduate students, (2) Carl Adam and me on FIU campus, (3) Carl Adam on Miami Beach, (4) Carl Adam at Everglades National Park). Carl Adam's brief interactions with my two sons might also have an impact on them. My older son, Grant Ho, who played violin during the conference banquet, graduated from Stanford University with distinction in computer science in 2014, and is now fifth-



**Fig. 1** Memories in Miami, July 2005

year Ph.D. student pursuing system security research at University of California at Berkeley ((5) Carl Adam with Grant). My younger son, Albert Ho, graduated from Princeton University with higher honor in computer science in 2018, and is now a software engineer at Facebook. ((6) Carl Adam with Albert). Carl Adam and I had personal communications for several years until 2007. Carl Adam not only exchanged holiday greetings with me, but also sent me birthday greetings. I will always cherish great memories of meeting and communicating with him and remember this computer science giant's influence on my own career as well as on my family. Carl Adam's wisdom and inspiration will continue to impact my professional and personal life.



## References

1. W. Reisig, Petri nets in software engineering, in *Petri Nets: Applications and Relationships to Other Models of Concurrency*, ed. by W. Brauer, W. Reisig, G. Rozenberg. LNCS, vol. 255 (Springer, Berlin, 1986), pp. 63–96
2. H.J. Genrich, K. Lautenbach, System modeling with high level Petri nets. *Theor. Comput. Sci.* **13**(1), 109–136 (1981)
3. X. He, I.A.N. Lee, A methodology for constructing predicate transition net specifications. *Softw. Pract. Exper.* **21**(8), 845–875 (1991)
4. X. He, J.A.N. Lee, Integrating predicate transition nets with first order temporal logic in the specification and verification of concurrent systems. *Form. Asp. Comput.* **2**(3), 226–246 (1990)
5. C. Kan, X. He, High level algebraic Petri nets. *Inf. Softw. Technol.* **37**(1), 23–30 (1995)
6. C. Kan, X. He, A method for constructing algebraic Petri nets. *J. Syst. Softw.* **35**(1), 12–27 (1996)
7. X. He, PZ nets – A formal method integrating Petri nets with Z. *Inf. Softw. Technol.* **43**(1), 1–18 (2001)
8. X. He, *A Formal Definition of Hierarchical Predicate Transition Nets*. Proceeding of the 17th International Conference on Application and Theory of Petri Nets (ICATPN'96). Lecture Notes in Computer Science, vol. 1091 (Springer, Berlin, 1996), pp. 212–229
9. X. He, Formalizing use case diagrams in hierarchical predicate transition nets, in *Proceeding of the IFIP 16th World Computer Congress*, Beijing, China, August (2000), pp. 484–491
10. X. He, Y. Ding, Object orientation in hierarchical predicate transition nets, in *Concurrent, Object-Oriented Programming and Petri Nets*, ed. by G.A. Agha, F. De Cindio, G. Rozenberg. Lecture Notes in Computer Science, vol. 2001 (Springer, Berlin, 2001), pp. 196–215
11. J. Wang, X. He, Y. Deng, Introducing architectural specification and analysis in SAM through an example. *Inf. Softw. Technol.* **41**, 451–467 (1999)
12. X. He, Y. Deng, A framework for developing and analyzing software architecture specifications in SAM. *Comput. J.* **45**(1), 111–128 (2002)
13. I.T. Zhu, X. He, A methodology of testing high-level Petri nets. *Inf. Softw. Technol.* **44**, 473–489 (2002)
14. D. Xu, X. He, Y. Deng, Compositional schedulability analysis of real-time systems using time Petri nets. *IEEE Trans. Softw. Eng.* **28**(10), 984–996 (2002)
15. X. He, H. Yu, T. Shi, J. Ding, Y. Deng, Formally analyzing software architectural specifications using SAM. *J. Syst. Softw.* **71**(1–2), 11–29 (2004)
16. J. Ding, P. Clarke, D. Xu, X. He, Y. Deng, A formal model-based approach for developing an interoperable mobile agent system. *Multi-Agent Grid Syst. Int. J.* **2**(4), 401–412 (2006)
17. L. Lian, S. Shatz, X. He, Flexible coordinator design for modeling resource sharing in multi-agent systems. *J. Syst. Softw.* **82**(10), 1709–1729 (2009)
18. L. Chang, S. Shatz, X. He, A methodology for modeling multi-agent systems using nested Petri nets. *Int. J. Softw. Eng. Knowl. Eng. IJSEKE* **22**(7), 891–926 (2012)
19. L. Chang, X. He, A methodology to analyze multi-agent systems modeled in high level Petri nets. *Int. J. Softw. Eng. Knowl. Eng. IJSEKE* **25**(7), 1199–1235 (2015)
20. X. He, A comprehensive survey of Petri net modeling in software engineering. *Int. J. Softw. Eng. Knowl. Eng. IJSEKE* **23**(5), 589–626 (2013)
21. S. Liu, R. Zeng, X. He, PIPE+ – A modeling tool for high level Petri nets, in *Proceeding of International Conference on Software Engineering and Knowledge Engineering (SEKE11)*, Miami (2011), pp. 115–121

22. S. Liu, R. Zeng, Z. Sun, X. He, Bounded model checking high level Petri nets in PIPE+ Verifier, in *Proceeding of International Conference on Formal Engineering Methods (ICFEM 14)*, ed. by S. Merz, J. Pang. LNCS, vol. 8829 (Springer, Berlin, 2014), pp. 348–363
23. X. He, R. Zeng, S. Liu, Z. Sun, K. Bae, A term rewriting approach to analyze high level Petri nets, in *Proceeding of the 10th Theoretical Aspects of Software Engineering Conference (TASE 16)*, IEEE, Los Alamitos (2016)
24. G. Holzmann, *The SPIN Model Checker* (Addison-Wesley, Boston, 2004)
25. M. Clavel, F. Duran, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, C. Talcott, *All about Maude – A high-performance logical framework: how to specify, program and verify systems in rewriting logic* (Springer, Berlin, 2007)
26. L. de Moura, N. Bjørner, Z3: an efficient SMT solver, in *Proceeding of 15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, ed. by R. Ramakrishnan, J. Rehof. Lecture Notes in Computer Science, vol. 4963 (Springer, Berlin, 2008), pp. 337–340

**Part III**  
**Technical Themes**

# Carl Adam Petri's Synchronic Distance



Jörg Desel

## 1 Introduction

Synchronic distance between activities or—in terms of Petri net theory—between transitions was a major topic in early papers and books on net theory; see, e.g., [3, 10]. It provides a measure of the degree of dependency between the respective transition occurrences.

Carl Adam Petri has developed this concept over years, starting with his dissertation thesis. Several refinements of definitions were based on the cooperation of Petri with members of his research group at the former GMD (Gesellschaft für Mathematik und Datenverarbeitung) in Sankt Augustin, Germany. I was a student researcher at this institute during the early 1980s, and I witnessed extensive discussions between Petri and his close colleagues about the definition of synchronic distance. This experience is worth recapitulating not only because synchronic distance was an important ingredient of Petri net theory at that time, but also because it illustrates both the openness of Carl Adam Petri and the mutual respect and appreciation between scientists in Petri's group, a fertile atmosphere created and stimulated by Petri. Of course, Petri was considered a great authority by everybody when it came to fundamental definitions. So the discussions were somehow asymmetrical. Essentially, Petri's colleagues provided small examples and asked Petri about his interpretation of the presented situations, trying to create a sound theory incorporating all the answers of Petri. On the other hand, Petri learned from the questions and seeming contradictions, and so he sometimes changed his mind. More often, however, he used the opportunity to widen the scope of his theory, creating new aspects relevant to the theory, so that often his answer was “it

---

J. Desel (✉)

Lehrgebiet Softwaretechnik und Theorie der Programmierung, FernUniversität in Hagen, Hagen, Germany

e-mail: [joerg.desel@fernuni-hagen.de](mailto:joerg.desel@fernuni-hagen.de)

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,  
[https://doi.org/10.1007/978-3-319-96154-5\\_16](https://doi.org/10.1007/978-3-319-96154-5_16)

119

depends”. For me, being at the beginning of my scientific career, it was a fascinating opportunity to learn more about the deep thoughts of Petri and the foundations of his theory, which goes far beyond the popular token game of Petri nets. I was surprised that little definition problems rather than huge and complex examples led to all these considerations.

The concept of synchronic distance applies in particular to pairs of transitions of a Petri net that can each occur arbitrarily often. If, in any occurrence sequence of the net, the difference between the number of occurrences of the first transition and the number of occurrences of the second transition stays within a finite interval, then these two transitions are said to be *synchronized*, and the size of the smallest interval with this property of the interval is their *synchronic distance*. More specifically, if, in each occurrence sequence, the number of occurrences of a transition  $a$  exceeds the number of occurrences of a transition  $b$  by at most  $n$ , and likewise the number of occurrences of  $b$  exceeds the number of occurrences of  $a$  by at most  $m$ , then the synchronic distance between  $a$  and  $b$  is  $n + m$ ; in a formula,  $\sigma(a, b) = n + m$ . If no such numbers  $n$  and  $m$  can be found, then there is no finite synchronic distance between  $a$  and  $b$  and we write  $\sigma(a, b) = \infty$ .

The definition of synchronic distance might seem to be rather obvious and straightforward. However, for systems—and their Petri net models—that have only finite runs, it turned out to be tricky to find a sound and intuitive definition. Obviously, it cannot be argued that  $\sigma(a, b) = \infty$  if neither transition  $a$  nor transition  $b$  can occur arbitrarily often; so for each such pair an intuitive notion of synchronic distance had to be found (since it is not trivial to see whether the number of possible occurrences of a transition is finite or infinite, a definition relating only to transitions that might occur infinitely often would be odd). Specifically, the question of how to define the synchronic distance between two transitions that can only occur once was not obvious. Exactly this question was a matter of long discussions in Petri’s research group.

This contribution aims at three goals: After introducing synchronic distance informally in Sect. 2, I will recapitulate in Sect. 3 the history of synchronic distance, rummaging in old papers of Petri starting with his famous doctoral thesis. Section. 4 compares variants of the usual definition. Building on this basis, Sect. 5 recalls the discussions mentioned above about the definition of synchronic distance in fundamental situations. Synchronic distance and the related notion of synchronic structure do not play any important role anymore in current work on Petri net theory or its applications. However, in Sect. 6 three examples will demonstrate the reappearance of the concepts in more recent work, although the name “synchronic distance” has apparently been forgotten by many researchers.

## 2 The Concept of Synchronic Distance

For an illustration of the concept “synchronic distance”, consider two equally sized chain wheels as in Fig. 1. Assume that every turn of the first chain wheel causes an

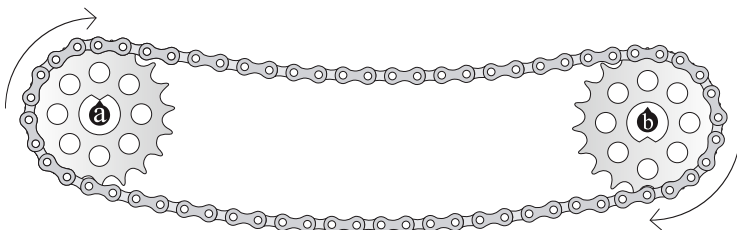


Fig. 1 A tight chain

event  $a$  and that every turn of the second causes  $b$ . Since the two wheels are mutually connected by a circular chain, the number of turns of the two wheels will be almost the same, no matter how long they proceed to turn. If the chain is relatively tight, as in the above figure, we expect for any future situation that the numbers of  $a$ -events and  $b$ -events that have occurred so far is almost equal. More precisely, if the first event is  $a$ , i.e., if  $a$  occurs first before  $b$  occurs first, then, for any finite run, the number of  $a$ s can exceed the number of  $b$ s by at most 1, whereas we will never see more  $b$ s than  $a$ s. If the first event is  $b$ , then the converse holds true. In both cases, the synchronic distance between  $a$  and  $b$  is 1, i.e.,  $\sigma(a, b) = 1$ .

Now we consider a longer chain between the two chain wheels, which still have the same size and the same distance, as in Fig. 2. Here the two wheels are more loosely coupled, and so are the events  $a$  and  $b$ . Still, the numbers of occurred  $a$ -events and  $b$ -events are related at any time, but the possible mutual deviation grows with the length of the chain. For example, if we can observe at most two more  $a$ -events than  $b$ -events and at most one more  $b$ -event than  $a$ -events, then  $\sigma(a, b) = 2 + 1 = 3$ .

Notice that this chain wheel metaphor does not work when we assume two wheels connected by a belt. Even if the diameters of the wheels are almost the same, any tiny difference will eventually lead to an arbitrarily large difference between the numbers of occurred  $a$ -events and  $b$ -events, assuming that we can run the little machine as long as we like. Notice also that a synchronic distance of 0 between  $a$  and  $b$  is impossible, because it would imply that  $a$  and  $b$  always occur

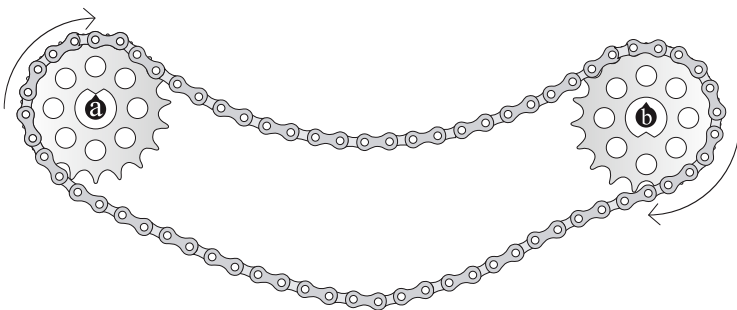


Fig. 2 A loose chain

coincidentally. First, this would be an unsolvable technical challenge. Second, it would be impossible to verify the coincidence of the events, assuming that the wheels are some way apart from each other.

The notion of synchronic distance discussed so far was generalized in two directions: a weighted synchronic distance corresponds to connected chain wheels of different sizes; if one wheel has twice as many sprockets as the other one, it will turn at half the speed, whence its corresponding event will appear less often. Assuming the weight 2 for this event, we can again characterize the coupling degree by means of synchronic distance, where the weighted event is counted twice. Another generalization considers the synchronic distance between sets of events rather than between events. If, for example, our (equally sized) chain wheels cause two events in each turn, say  $a_1$  and  $a_2$  for the first wheel and  $b_1$  and  $b_2$  for the second, then the synchronic distance between the sets  $\{a_1, a_2\}$  and  $\{b_1, b_2\}$  is finite, and it again depends on the length of the chain. If a chain wheel causes an event  $a_1$  or an event  $a_2$  with each turn (depending on some condition not explicated in the example), whereas the other one still causes always  $b$ , then only the synchronic distance between  $\{a_1, a_2\}$  and  $\{b\}$  is able to express the mutual synchronization.

### 3 Petri's Work on Synchronic Distance

One might argue that the core idea of synchronic distance can already be found in Petri's doctoral thesis [5, p. 42]. There Petri postulated

Even speed of clocks can only be established by communication.

and used the argument that any tiny difference of clock speeds will eventually lead to an arbitrary difference between their mutual number of ticks. He then discussed that even speed has no obvious definition, but can only be defined by means of a closed signal chain containing both clocks. Since Petri did not assume the existence of time as an objective and available entity, no clock speed can be called accurate (with respect to what?). Hence, according to Petri, we can compare the speed of a clock only with the speed of another clock. If we want both clocks to run with a limited error w.r.t. the other clock, we must provide communication means between the two clocks, in both directions. In other words: Two clocks have a limited divergence only if there are mutual signals between the clocks. One obvious consequence of these considerations is that synchronicity between two distinct clocks with its traditional interpretation "ticks at the same time" is meaningless.

We will consider in the sequel two machines *tick* and *tack*, which can be considered as clocks, but might have any interpretation. The clock *tick* repeatedly produces "tick" and after each "tick" sends a message *I said "tick"* to *tack*. After receiving this message, *tack* is allowed to "tack", and then sends back a message *I said "tack"* to *tick*, which can only then produce the next "tick"; see Fig. 3.

In [7], Petri considered Petri nets (which he always called just nets) and defined a binary relation  $D$  between sets of transitions. Two such sets  $A$  and  $B$  are in the relation  $D$  if, in any process representing a run, occurrences of transitions from  $A$

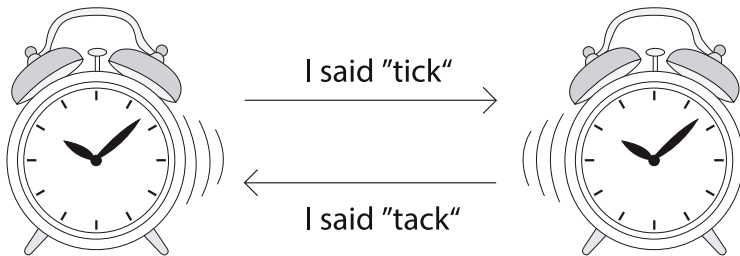


Fig. 3 Two communicating clocks

strictly alternate with occurrences of transitions from  $B$ . Roughly speaking, in our terminology,  $A$  and  $B$  are in the relation  $D$  if and only if  $\sigma(A, B) = 1$  (we ignore the subtle difference between Petri's notion of process and other definitions of causal behavior). The clocks  $tick$  and  $tack$  mentioned above are in the relation  $D$ .

Now let us consider a third clock  $tock$ , which is connected to  $tack$  in the same way as  $tack$  is connected to  $tick$ , i.e.  $tack$  and  $tock$  exchange messages and are thus in the relation  $D$ ; see Fig. 4. The clocks  $tick$  and  $tock$  are not as closely related as the other pairs of clocks, because the sequence of events

tick tack tick tock...

is possible, where we can see two "tick"s before the first "tock".

Since  $tick D tack$  (i.e.,  $tick$  and  $tack$  are in the relation  $D$ ) and  $tack D tock$ , we can write  $tick D^2 tock$ . In [7], Petri defined the smallest number  $m$  such that  $A D^m B$  as the synchrony of two transition sets  $A$  and  $B$  and also used the notion *synchronous with distance  $m$* . It is worth mentioning that he defined *synchronous transitions* neither by synchronic distance 0, which only applies to identical sets of transitions, nor by synchronic distance 1, which represents strict alternation, but by synchronic distance 2! This becomes meaningful when we consider our clock  $tack$  from above as a controller of the clocks  $tick$  and  $tock$ . Repeatedly,  $tack$  sends to both other clocks its message, then both clocks act concurrently, and so on. This type of concurrent behavior is apparently closer to the intuition of synchrony than identity or alternation.

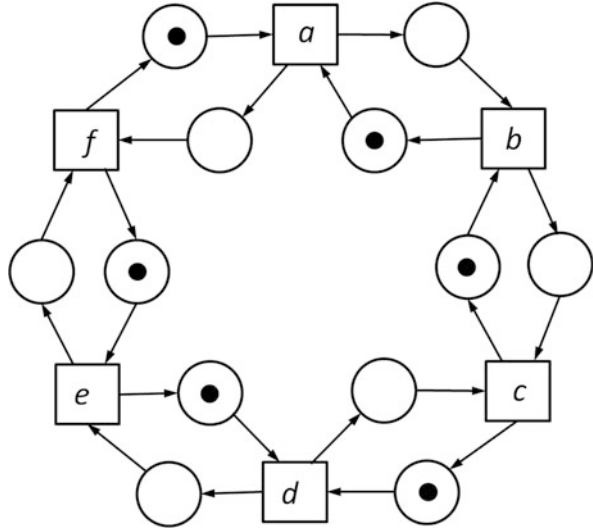
In the later paper [8], Petri defined synchronic distance as the maximal deviation between occurrences of two transitions in a run, where here a run is a feasible



Fig. 4 Three communicating clocks



Fig. 5 A counterexample



sequence of transition occurrences:

$$\sigma(a, b) = \max_{\tau \text{ is a run}} |(\text{number of } a\text{s in } \tau) - (\text{number of } b\text{s in } \tau)|$$

In the same paper, he referred to the previous definition based on the alternation relation  $D$ . Clearly, the definition based on alternation provides an upper limit for the newly defined synchronic distance  $\sigma$ . However, the two definitions do not coincide, as Petri showed by means of a counterexample; see Fig. 5.

From the structure of this net, we obtain

$$a D b D c D d D e D f D a$$

and thus  $a D^3 d$ , but not  $a D^2 d$ . However, since the outer cycle of this net contains only two tokens, we have  $\sigma(a, b) = 2$ .

It is important to notice that a run in the sense of Petri can start with any reachable marking, not necessarily with a given initial one. Consider, for example, a net with two transitions  $a$  and  $b$  such that

$$\dots a \dots b \dots b \dots a \dots a \dots b \dots b \dots$$

is a possible run. In none of its prefixes, also constituting runs, does the absolute difference between the number of  $a$ -occurrences and  $b$ -occurrences exceed 1. However, starting with the marking reached after the first  $a$ , we observe a subsequence with two  $b$ s and no  $a$ , which is a run proving  $\sigma(a, b) \geq 2$ .

An earlier, informal paper of Petri [6] already defined a notion of synchronic distance between clocks. In this work, Petri also referred very implicitly to the

relation  $D$  (actually, he referred to the incidence matrix of the considered net; two sets of transitions are in the relation  $D$  if they are the pre-set and the post-set of a place). In this early paper, Petri conjectured that synchronic distance between clocks can be derived from the structure of the net, without considering any marking—a conjecture disproved by himself using the counterexample shown in Fig. 5. However, many years later it was shown by various authors that, for particular classes of Petri nets, the existence of a finite synchronic distance between two sets of transitions can be deduced from the incidence matrix of the net (see, e.g., [12] and, for a different definition of synchronic distance, [1]).

## 4 Formal Definitions of Synchronic Distance

As mentioned before, the definition of synchronic distance originates from the behavior of clocks or similar devices that continuously repeat their activities. Unfortunately, the definition is not so obvious in system models with only finite runs, and not even for all fundamental building blocks *sequence*, *concurrency*, and *conflict*, as will be discussed in the following section. Before starting this discussion, we have to provide a formal definition of synchronic distance and some notations. We consider occurrence sequences of a Petri net  $N$ , starting with the initial marking of  $N$ . We denote the set of all finite occurrence sequences of  $N$  by  $L_N$ . If  $\tau_1 \in L_N$  is a prefix of  $\tau \in L_N$ , then we write  $\tau_1 \prec \tau$ . Also,  $\#_\tau(a)$  denotes the number of occurrences of a transition name  $a$  in the sequence  $\tau$ .

The following definition of synchronic distance formalizes the intuition based on the maximal respective leads of a transition  $a$  and a transition  $b$ :

$$\sigma_1(a, b) = \max_{\tau \in L_N} \left( \max_{\tau_1 \prec \tau} (\#_{\tau_1}(a) - \#_{\tau_1}(b)) + \max_{\tau_1 \prec \tau} (\#_{\tau_1}(b) - \#_{\tau_1}(a)) \right)$$

if this maximum exists, and  $\sigma_1(a, b) = \infty$  otherwise.

We will show that the definition of  $\sigma_1$  is equivalent to the following definition of  $\sigma_0$  from [10], which considers the maximal deviation between transitions  $a$  and  $b$ :

$$\sigma_0(a, b) = \max_{\tau_1 \tau_2 \in L_N} |\#_{\tau_2}(a) - \#_{\tau_2}(b)|$$

Our new formulation of the definition will make it easier to compare variants of synchronic distance in the following section.

The equivalence between  $\sigma_1$  and  $\sigma_0$  will be shown next. It is immediate to see that  $\sigma_1(a, b) = \sigma_1(b, a)$ , by symmetry of the definition of  $\sigma_1$ . Therefore, transition name  $a$  can be replaced by a variable  $x$  in the defining term of  $\sigma_1$ , and  $b$  by a variable  $y$ , or vice versa, resulting in the equivalent term

$$\max_{\tau \in L_N} \left\{ \max_{\tau_1 \prec \tau} (\#_{\tau_1}(x) - \#_{\tau_1}(y)) + \max_{\tau_1 \prec \tau} (\#_{\tau_1}(y) - \#_{\tau_1}(x)) \mid \{x, y\} = \{a, b\} \right\}$$

where  $\{x, y\} = \{a, b\}$  means that either  $x = a$  and  $y = b$  or that  $x = b$  and  $y = a$ . Now assume, without loss of generality, that the first prefix  $\tau_1$  in this term is not longer than the second. Then we can rewrite the entire sequence  $\tau$  as  $\tau = \tau_1 \tau_2 \tau_3$  such that  $\tau_1$  is the first, shorter prefix from above, and such that  $\tau_1 \tau_2$  is the second, longer one. We obtain

$$\begin{aligned} \sigma_1(a, b) &= \max_{\tau_1 \tau_2 \tau_3 \in L_N} \{((\#_{\tau_1}(x) - \#_{\tau_1}(y)) + (\#_{\tau_1 \tau_2}(y) - \#_{\tau_1 \tau_2}(x))) \mid \{x, y\} = \{a, b\}\} \\ &= \max_{\tau_1 \tau_2 \in L_N} \{((\#_{\tau_1}(x) - \#_{\tau_1}(y)) + (\#_{\tau_1 \tau_2}(y) - \#_{\tau_1 \tau_2}(x))) \mid \{x, y\} = \{a, b\}\} \\ &= \max_{\tau_1 \tau_2 \in L_N} \{(\#_{\tau_1 \tau_2}(y) - \#_{\tau_1}(y) - \#_{\tau_1 \tau_2}(x) + \#_{\tau_1}(x)) \mid \{x, y\} = \{a, b\}\} \\ &= \max_{\tau_1 \tau_2 \in L_N} \{(\#_{\tau_2}(y) - \#_{\tau_2}(x)) \mid \{x, y\} = \{a, b\}\} = \sigma_0(a, b) \end{aligned}$$

Finally, notice that the definition of  $\sigma_0$  from [10] is not exactly equivalent to Petri's previous definition of synchronic distance. Actually, [10] does not consider occurrence sequences  $\tau_1 \tau_2$  starting at the initial marking, but rather just sequences  $\tau_2$  enabled by arbitrary reachable markings, i.e., by markings reached from the initial marking by some occurrence sequence  $\tau_1$ . Petri's view of reachable markings was slightly different. Assume that a marking  $m'$  is reached from a marking  $m$  by the occurrence of a transition. Then today's notion of reachability is based on the axiom:

If  $m$  is reachable, then  $m'$  is reachable as well.

Petri additionally postulated:

If  $m'$  is reachable, then  $m$  is reachable as well,

because he insisted on symmetry between past and future. We use the simple generalization of synchronic distance to sets of transitions to show the difference in Fig. 6.

Taking the marking depicted by stars as initial marking, it is easy to see that  $\sigma_0(\{a, c\}, \{b, d\}) = 1$ . According to Petri's notion of reachability, the marking depicted by hearts is *backward* reachable from the stars-marking, because from the hearts-marking the stars-marking is reached by occurrence of the transitions  $b$  and

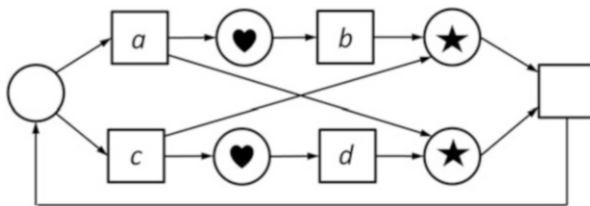


Fig. 6 Synchronic distance depends on the definition of reachability

*d*. So, since from this marking *b* and *d* can occur without any transition of  $\{a, c\}$ , we obtain  $\sigma_0(\{a, c\}, \{b, d\}) = 2$  according to Petri's definition.

## 5 Synchronic Distance in Fundamental Situations

We consider three fundamental behavioral situations: *Sequence*, *Concurrency*, and *Conflict*, as depicted in Figs. 7, 8, and 9, respectively. In all these three figures, the dotted net elements and arcs represent an embedding of the situation in a cyclic setting.

### 5.1 Sequence

Two transitions *a* and *b* are *sequentially ordered* if *b* can only occur after *a* has occurred; see Fig. 7. This is a special variant of strict alternation. After *a*, the difference between the numbers of occurrences is 1; after *a b* it is 0, whence  $\sigma(a, b) = 1$ . So, if *a b* and its prefixes are the only occurrence sequences, then  $\sigma(a, b) = 1$ . This does not change in the cyclic setting, in which *a* and *b* strictly alternate.

### 5.2 Concurrency

Two transitions *a* and *b* occur *concurrently* if each can occur independently of the other; see Fig. 8.

There are various ways to model concurrent runs, the most prominent given by *processes*. For the sake of this chapter, let us assume that a process  $\pi$  of a net is a *set of occurrence sequences* that represents all sequential observations of the process. That is, if *a* and *b* occur concurrently after an occurrence of *c*, then both *c a b* and *c b a* belong to this set. The set of all processes of a marked Petri net *N* is denoted by  $\Pi_N$ .

In the cyclic constellation, we can see the first occurrence of *a* before the first occurrence of *b* and the second occurrence of *b* before the second occurrence of *a*.

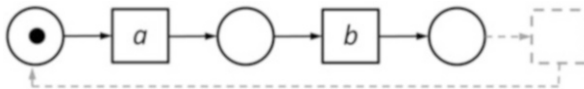


Fig. 7 Sequence

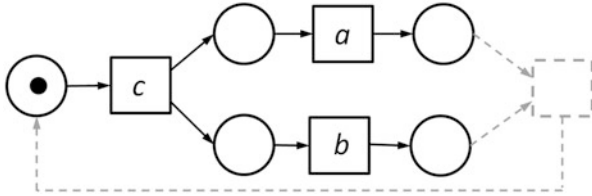


Fig. 8 Concurrency

So this process contains a sequence...  $a b \dots b a \dots$ . According to the definition  $\sigma_1$  of synchronic distance given above, the synchronic distance between  $a$  and  $b$  is 2.

If, however, both  $a$  and  $b$  occur only once, as in Fig. 8, we still would like to obtain the same synchronic distance, but unfortunately the same argument does not work! The process contains sequences  $c a b$  and  $c b a$ . Each of these sequences gives rise to a synchronic distance of 1 only. Only considering the possible lead of  $a$  and also the possible lead of  $b$ , both possible within one process, but in different sequences, results in the desired distance 2. Thus, here is a second definition of synchronic distance (see [3] and [1]), solving this problem:

$$\sigma_2(a, b) = \max_{\pi \in \Pi_N} \left( \max_{\tau_1 < \tau \in \pi} (\#_{\tau_1}(a) - \#_{\tau_1}(b)) + \max_{\tau_1 < \tau \in \pi} (\#_{\tau_1}(b) - \#_{\tau_1}(a)) \right)$$

if this maximum exists, and  $\sigma_2(a, b) = \infty$  otherwise.

### 5.3 Conflict

Now consider the case that either a transition  $a$  can occur or a transition  $b$  can occur, but each of these two transitions excludes the other one, as in Fig. 9. Petri called this situation a *conflict* between  $a$  and  $b$ .

In the cyclic constellation,  $a$  can occur arbitrarily often without any occurrence of  $b$  (and vice versa), whence then there is no finite synchronic distance between the two transitions. However, if we only have a singular conflict, then the only possible maximal occurrence sequences are given by (only)  $a$  and (only)  $b$ . Since, in both

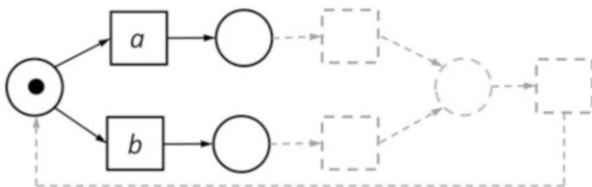


Fig. 9 Conflict

cases, at most one transition can occur, an obvious suggestion is  $\sigma(a, b) = 1$ . However,  $a$  can lead by 1 in one sequence (and in one process) and  $b$  can lead by 1 in a different sequence (and in a different process). The difference with the concurrency case is that the considered sequences  $a$  and  $b$  exclude each other.

The core of the discussion within Petri's research group was the question of whether in this case the synchronic distance between  $a$  and  $b$  should be 1 or 2. As the authors of [4] report in their paper, Petri decided that both results are possible. In particular, if no further control is assumed and both  $a$  and  $b$  are therefore really possible, he decided that the synchronic distance should be 2. This is obtained by the following, much easier definition  $\sigma_3$  of synchronic distance, which does not maximize the mutual deviation of the considered transitions  $a$  and  $b$  within one sequence or within one process, but within all occurrence sequences:

$$\sigma_3(a, b) = \max_{\tau \in L_N} (\#_{\tau}(a) - \#_{\tau}(b)) + \max_{\tau \in L_N} (\#_{\tau}(b) - \#_{\tau}(a))$$

if these maxima exist, and  $\sigma_3(a, b) = \infty$  otherwise.

If, however, some control is assumed that decides how to choose between  $a$  and  $b$ , then Petri voted for synchronic distance 1. Hence, at this point he introduced synchronic distance between transitions not as a behavioral property of the net, but as a *specification* of a property, which the net does not necessarily satisfy. Instead, it is assumed that the modeled system is embedded in an environment such that the composed system will have the specified property. By providing the specification of a synchronic distance, we thus consider the behavior of one Petri net component restricted to the behavior of the entire composed net.

## 6 What Happened to Synchronic Distance?

### 6.1 Places as Specifications

Petri did not like nets where the flow of control is established by places with multiple tokens. Instead, he considered only signals between transitions that cannot accumulate or overtake each other. Actually, he considered all original places of a net as *conditions*, which can be *true* (one token, existing signal) or *false* (no token). According to his firing rule, a transition can only be enabled if all its post-conditions are unmarked; thus multiple tokens on a place cannot occur.

However, Petri also used places with arbitrary markings, but for a different purpose. He added such places to nets with conditions and transitions, but distinguished these places from the conditions (actually, he used dotted lines for these places and their in-and out-going arcs; see [4]). The purpose of such an additional place was *not* to restrict the behavior of transitions, but to count the number of occurrences of transitions in the pre-set *minus* the number of occurrences of transitions in the post-set of the additional place. This way, the place can model a buffer, a resource

or suchlike. If the number of tokens on that place ranges between a lower and an upper limit, then the difference between these numbers is the synchronic distance between two sets of transitions, the pre-set and the post-set of the place. Again, this view assumes that, for some initial marking of the additional place, this place will never run out of tokens, which would disable the transitions in its post-set.

Hence, Petri did not only invent Petri nets with conditions and transitions, but also arbitrary places in nets which can carry more than one token. He distinguished these places from conditions and motivated these places by the concept of synchronic distance. He explicitly mentioned that the synchronic distance between the pre- and post-set of a condition is always 1 (assuming that all transitions can fire at some reachable marking), and thus the condition can be replaced by a place in the above sense. The *synchronic structure* of the net is given by all transitions and all places representing a finite synchronic distance. Petri considered this view, a place/transition net, as an important alternative way to represent the behavior of a distributed system.

## 6.2 Hybrid Specifications

As reported in detail in [4], Petri assumed that each modeled system has some environment that influences or controls the system's behavior. By means of declarative specifications, assumptions about the environment's behavior can be specified.

Petri first added places as described above to depict a behavioral property of a Petri net, namely the synchronic distance between the respective pre- and post-set. Later, he suggested adding such places as a *declarative specification*. More specifically, an additional place states that the behavior of the net should respect this synchronic distance, even if the net without the place does not. Thus, he used a mixture of procedural specification (the net itself) and declarative specification (additional places expressing an assumed synchronic distance). Whereas, for many years, other modeling formalisms concentrated either only on procedural specifications or only on declarative specifications, *hybrid* specifications such as the ones invented by Petri became hip only in this century, particularly in the application area of business process models. See, for example, [2] and [9].

## 6.3 Synthesis Based on Synchronic Structure

In the last few decades, a rich theory of *Petri net synthesis* from a behavioral specification has been developed. The specification of behavior ranges from transition systems, sets of occurrence sequences, and terms of occurrence sequences (expressing infinite behavior, too) to sets of causally ordered runs such as process nets. The core concept of synthesis is to construct a Petri net by taking all transitions from the input descriptions and successively adding places, which restrict the

behavior of the net. The restriction caused by such a place may never exclude a part of the input behavior, i.e., the specified behavior must still be possible in the net constructed so far. Adding *all* such *feasible* places respecting the input behavior results in a Petri net which has exactly this behavior, if such a net exists. However, allowing arbitrary arc weights, the set of all such *feasible* places is usually infinite, and most of these feasible places are redundant. Therefore, research about Petri net synthesis concentrates on efficient algorithms for creating a finite, or even minimal, subset of feasible places such that the net with only these places is behaviorally equivalent to the net with all feasible places.

How is this connected to synchronic distance? Each place to be added can be viewed as a synchronic distance between its pre- and post-set. So Petri net synthesis is about identifying the synchronic structure from a given behavioral description. Whenever there is a finite synchronic distance between two sets of transitions, the corresponding place is feasible. Petri also already thought about small feasible sets, by identifying the previously mentioned relation  $D$  expressing a synchronic distance 1. However, he realized that in general it is not possible to synthesize a net by just using places representing elements of  $D$ , as mentioned earlier.

## 7 Conclusion

I have collected thoughts and findings of Carl Adam Petri about the concept of synchronic distance in Petri nets. Sources have been early papers of Petri, papers reporting on discussions with Petri, and my own recollection of fascinating discussions between Petri and members of his research group.

The precise definition of synchronic distance has changed slightly over the years, and so has its interpretation. Reformulating the respective definitions that can be found in the literature, I gave the most important ones in a (hopefully) better-comparable way. Although synchronic distance seems to be a forgotten concept in Petri net theory, I provided examples to prove that the core ideas are still relevant, namely the interpretation of a bounded place, the use of special places in hybrid specifications, and finally the synthesis of Petri net places from behavioral specifications.

Synchronic distance between two sets of transitions describes a very strict mutual dependency. There have been generalizations in various ways. In particular the existence of a finite synchronic distance between transitions  $a$  and  $b$  implies that neither of the transitions can occur infinitely often if the other transition does not occur, which is in turn a special case of a *fairness assumption*, as introduced much later within Temporal Logics. For an attempt to define a synchronic distance-related fairness notion, see, e.g., [11].



## References

1. J. Desel, Synchronie-Abstand in Stellen-Transitionssystemen. Arbeitspapiere der GMD No. 341, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin (1988)
2. J. Desel, T. Erwin, Hybrid specifications: looking at workflows from a run-time perspective. *Comput. Syst. Sci. Eng.* **155**(5), 291–302 (2000)
3. H.J. Genrich, K. Lautenbach, P.S. Thiagarajan, Elements of general net theory, in *Net Theory and Applications*, ed. by W. Brauer. Lecture Notes in Computer Science, vol. 84 (Springer, Heidelberg, 1979), pp. 21–163
4. U. Goltz, Y. Chong-Yi, Synchronic structure: a tutorial, in *Advances in Petri Nets 1985*, ed. by G. Rozenberg. Lecture Notes in Computer Science, vol. 222 (Springer, Heidelberg, 1985), pp. 233–252
5. C.A. Petri, Kommunikation mit Automaten. Ph.D. thesis, Institut für Instrumentelle Mathematik, Bonn, 1962
6. C.A. Petri, Grundsätzliches zur Beschreibung diskreter Prozesse, in *3. Colloquium über Automatentheorie* (Birkhäuser, Basel, 1967), pp. 121–140
7. C.A. Petri, Concepts of net theory, in *Mathematical Foundations of Computer Science* (Mathematical Institute of the Slovak Academy of Sciences, Bratislava, 1973), pp. 137–146
8. C.A. Petri, Interpretations of net theory. ISF-Report 75.07, Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin (1975)
9. H.A. Reijers, Declarative modeling-an academic dream or the future for BPM?, in *Business Process Management*, ed. by F. Daniel, J. Wang, B. Weber. Lecture Notes in Computer Science, vol. 8094 (Springer, Heidelberg, 2013), pp. 307–322
10. W. Reisig, *Petri Nets: An Introduction*. EATCS Monographs on Theoretical Computer Science, vol. 4 (Springer, Heidelberg, 1985)
11. M. Silva, T. Murata, B-fairness and structural B-fairness in Petri net models of concurrent systems. *J. Comput. Syst. Sci.* **44**(3), 447–477 (1992)
12. M. Silva, J.M. Colom, On the computation of structural synchronic invariants in P/T nets, in *Advances in Petri Nets 1988*, ed. by G. Rozenberg. Lecture Notes in Computer Science, vol. 340 (Springer, Heidelberg, 1987), pp. 386–417

# How Carl Adam Petri Deeply Influenced My Understanding of Invariance and Parallelism



Gerard Memmi

We were a group of eight researchers and had recently published the first book in French on Petri Nets<sup>1</sup> summing up the main results of the French research in the field (both theoretical and applied). A few months later, I received a letter of appraisal from Carl Adam Petri. We were not expecting such a letter and we were touched and indeed very pleased about it.

Thanks to this book and this letter I was able to meet with Carl Adam Petri a few times and have short but meaningful conversations with him. This was the early 1980s and every informal meeting took place at European Workshops on Application and Theory of Petri Nets. I remember him as a fine man, astonishingly reserved given his reputation and the high esteem he enjoyed among his colleagues. I was at the time a young researcher, somewhat clumsy and certainly quite introverted. We talked about the book and I remember one comment in particular from him as it concerned the heart of my own research: Petri had not anticipated how linear algebra could be applied to his model to deduce a variety of behavioral properties. He encouraged me to dig further in this direction. In what follows, I propose to focus on the notion of invariance and how linear algebra relates to this fundamental notion. I will conclude with some additional thoughts on the issue.

---

<sup>1</sup>G. W. Brams “Réseaux de Petri, Théorie et Pratique”, Masson, 1983. GW Brams is a nom de plume made up from the first letter of each author’s name in the following order: C. Girault, R. Valette, G. Vidal-Naquet, G. Berthelot, G. Roucairol, C. André, G. Memmi, and J. Sifakis. The W standing for Valette and Vidal-Naquet is a pun: in French the letter W is pronounced ‘double V’ and not ‘double U.’ A good number of the results described in this short presentation can be found in this book.

---

G. Memmi (✉)

Laboratoire Traitement et Communication de l’Information (LTCI), Télécom ParisTech,  
Université Paris-Saclay, Paris, France  
e-mail: [gerard.memmi@telecom-paristech.fr](mailto:gerard.memmi@telecom-paristech.fr)

## 1 By the Way, What Exactly Is an Invariant?

In computer science or in software engineering, an invariant is a system behavioral property or relationship the value of which does not vary over time, over the evolution of the said system. Each time you use words like *'is constantly,' 'must always,'* or *'never will'* you can suspect that your sentence can be associated with an invariant. In any system model, invariants will be used to describe behavioral requirements or system exceptions which are often used to support the detection of a possible physical failure and provide valuable alerts. Invariants can be described in a temporal logic; they have a constant truth value that is always satisfied.

With regard to Petri Nets, these properties or relationships will be expressed or deduced by using places and transitions. Most of the time, places will model variables and transitions will model equations over these variables. Then, the bipartite graph connecting places and transitions supports the system structure. Together with the “token game,” which in turn models the dynamic evolution of the system under consideration, we have two distinctive elements that in my opinion make Petri Nets so unique and famous. Sometimes, researchers would find Petri Nets too low level to describe complex systems and would compare Petri Nets to Turing machines, which is not to say that they have the same power of algorithmic description (we know since the reachability problem has been solved for Petri Nets that Petri Nets are less powerful in terms of algorithmic description than Turing machines) but rather in order to stress the need for a more concise model, providing a higher level of abstraction. Hence, the many proposals introduced to enrich the token game, making the enabling rule more elaborate using, for instance, priorities, time, or probabilities or making tokens more complex with, for instance, colors, letters, or abstract data types.<sup>2</sup> However, the bipartite structure between places and transitions would remain and the notion of invariance likewise. In a way, considering markings and tokens, an invariant can also be seen as a constant function over the distribution of tokens in a Petri Net independent of the sophistication of its enabling rule or the complexity of the structure of its tokens.

A classic example of an invariant is the relationship expressing mutual exclusion between two tasks of two entities. By modeling these two entities and their interactions by a Petri Net and the fact that their tasks are active by the presence of tokens in two places  $A$  and  $B$ , respectively, proving that  $M(A) \times M(B) = 0$  for any reachable marking from the initial  $M_0$  will show exactly what we are looking for: the two entities cannot be active at the same time (i.e., at the same marking). The set of all reachable markings from the initial marking is traditionally called the reachability set. Then, an invariant is a property true for all elements of the reachability set.

---

<sup>2</sup>The interested reader can consult the excellent book edited by Kurt Jensen and Grzegorz Rozenberg: «High-level Petri Nets, Theory and Application», Springer, 1991.

## 2 Invariants Lead to Liveness and Boundedness

With regard to Petri Nets, invariants are interesting not only because they express a property that is looked for itself (for instance, to comply with the system specification) but also because they allow the exclusion of a vast number of markings that can't be reached from the initial marking without violating the invariant. If a marking does not satisfy an invariant, then it is not reachable. This aspect is critical to analyze and prove many important behavioral Petri Net properties such as liveness or boundedness.

To explore this idea on a deeper level, we associate invariants with sets of markings: An invariant  $I$  can be associated with a set  $I_M$  of all markings satisfying the invariant property or relationship (i.e., for which the invariant is true). A property  $P$  will be an invariant if the reachability set is included in its associated set  $I_P$ . Then, we introduced the notion of home space.

A set of markings,  $HS$ , is a *home space* if and only if for any evolution of the Petri Net, it is always possible to go back to an element of  $HS$  via a sequence of transitions. It is not easy to prove that a given set of markings is a home space since it requires verifying the property for all reachable markings from the initial marking of the Petri Net under consideration. Actually, it is worth noting that any home space has at least one element in each sink strongly connected component of the reachability graph.

When  $HS$  is reduced to one single marking, we say that  $HS$  is a home state, which is a well-known critical notion for physical systems and is usually associated with the reset function of the system. If the initial marking  $M_0$  is a home state, then any reachable marking also is a home state and the reachability graph is strongly connected. It is then obvious that any transition that can be enabled once is live.

Now, we can see that a set of markings associated with an invariant is a trivial home space since it includes the entire reachability set.

These observations lead us to look for the smallest set of markings associated with invariants, or at least to look for the smallest one that can be easily handled. From there, we try to characterize how tokens are distributed over places and, for each possibility of this token distribution, to prove that a specific given marking  $M$  ( $M_0$  being the usual case) is reachable. When this is possible, it can easily be deduced that  $M$  is a home space. Then, it is possible to conclude and be able to prove whether the net under study is live or not.

Sometimes, the token distribution that I just mentioned can allow us to prove that some places are bounded. When places are bounded for any initial marking, they are said to be *structurally bounded*. I will come back to this point soon, but first, let us understand how to proceed to discover some of them.

### 3 Invariant Calculus

In general, invariants can be proven as such by using a model checker. However, the question is not only to prove that a given proposition is an invariant (which is no small undertaking), but also to design algorithms able to discover as many of them as possible with as little guidance as possible and understand which ones are the most potent.

If  $I$  and  $J$  are two invariants associated with their associated marking sets  $I_M$  and  $J_M$ , then it is easy to define a third invariant  $K$  such that its associated marking set is  $I_M \cap J_M$ . As we can see, the intersection is stable over the associated sets of invariants; this is important since we are looking for as small as possible an associated set. It is also worth noting that it is easy to combine invariants in various ways and to generate many of them. It is therefore important to understand whether invariants are organized and how to develop some computational rules over invariants relative to a given net.

Until now, all these definitions have been pretty general and can be applied to any transition system model.

For a Petri Net, any transition  $t$  can be defined by its *Pre* and *Post* conditions and if  $M'$  is reached by enabling  $t$  from  $M$ , then we can compute  $M'$  from  $M$  such that

$$M' = M + Post(.,t) - Pre(.,t)$$

If an invariant  $I$  can be defined as a function over markings, then we have

$$I(M') = I(M + Post(.,t) - Pre(.,t))$$

Moreover, if  $I$  has the good taste to be a homomorphism that preserves addition, then we may rewrite our equation to obtain  $I(M') = I(M) + I(Post(.,t)) - I(Pre(.,t))$ . Since  $I$  is an invariant,  $I(M') = I(M)$  and  $I$  must verify the following system of equations:

$$I(Post(.,t)) = I(Pre(.,t)) \quad \text{for every transition } t \text{ of the net.}$$

This system of equations is fundamental; it sets some constraints over the topology of the net and expresses a law of conservation over the net and its evolution. This law of conservation says that during the dynamic evolution of the net, a function of token distribution remains constant: when a transition is enabled, the evaluation of this function over what is consumed by the transition (described by *Pre*) is equivalent to the evaluation of the function over what is produced (described by *Post*). H. Genrich and K. Lautenbach were among the first ones, if not the first ones, to describe such a concept for Petri Nets.

In many papers, an invariant is associated with an integer weight function  $f$  over the set of places, named semiflow, that verifies *Kirchhoff's law* for each transition. We then have the following invariant:  $f^T M = f^T M_0$  for any marking  $M$  reachable

from  $M_0$ , where  $f^T M$  is the scalar product of  $f$  and  $M$  and it can be computed by solving the following system of equations:

$$f^T Post(.,t) = f^T Pre(.,t) \quad \text{for every transition } t \text{ of the net.}$$

The vast corpus of linear algebra results and algorithms can be applied to analyze a net and compute semiflows. However, the most interesting semiflows from a behavioral point of view are defined over natural numbers instead of integers. A first reason for particularly defining  $f$  over non-negative integers is that it can be proven that if

$$f > 0 \text{ and } f^T Post(.,t) \leq f^T Pre(.,t) \quad \text{for every transition } t \text{ of the net,}$$

then the set of places (the net) is structurally bounded. The reverse is also true: if the net is structurally bounded, then there exists a strictly positive solution for the system above. This characteristic property is indeed false for a semiflow that verifies the system and has at least one negative element.  $\|f\|$  usually denotes the support of  $f$ , that is to say the subset of places that have a non-null weight relative to  $f$ . Any subset of places included in  $\|f\|$  is structurally bounded.

A second reason for particularly defining  $f$  over non-negative integers is that the inequation

$$f^T Pre(.,t) \geq f^T M_0$$

becomes a necessary condition for  $t$  to stand a chance to be enabled or live.

Of course, a linear combination of semiflows is still a semiflow. Unfortunately, the set of semiflows over natural numbers is not a space vector. However, it is still possible to characterize this set by the following result: any semiflow is a linear combination (with rational coefficients) of the family of minimal semiflows of minimal support. Several algorithms have been published to compute such a family of semiflows. This finite family is critical since once we have it we can generate any other semiflow.

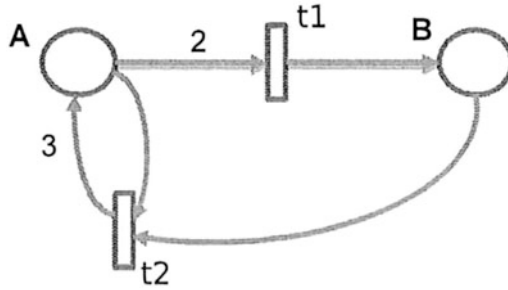
These results have been cited and utilized many times in various applications going beyond computer science, electrical engineering, or software engineering. For instance, they have very recently been used in the domain of biomolecular chemistry for chemical reaction networks, which brings us back to the original vision of C. A. Petri when he claimed that his nets could be used in chemistry. I remember that at the time I was among the skeptical on . . . and of course, I was wrong!

Let us illustrate some of the above concepts through an example.

## 4 A Tiny Example

The Petri Net below models a machine semi-deciding whether a given number  $n$  is even assuming it is greater than 1. That is to say, this net will be live only if the initial marking is such that  $M_0(A) = n = 2k + 1$  for  $k \geq 1$ ; and on the other hand, it will always be possible to find a sequence of transitions such that the net stops ( $t_1$  and  $t_2$  are not enabled anymore) only if  $M_0(A) = n = 2k$ , whatever the initial marking of B.

We have:  $Pre(.,t_1)^T = (2,0)$ ;  $Pre(.,t_2)^T = (1,1)$ ;  $Post(.,t_1)^T = (0,1)$ ;  $Post(.,t_2)^T = (3,0)$ .



$f^T = (1, 2)$  is a semiflow: for  $t_1$  and  $t_2$  Kirchhoff's law is easily verified. Moreover, it can be proven that  $f$  is a minimal semiflow and its support is minimal as well.

The scalar product  $f^T M$  defines an invariant: for any marking  $M$  reachable from an initial marking  $M_0$ ,  $f^T M = M(A) + 2M(B) = f^T M_0$  is constant. This can be rewritten as  $M(A) = f^T M_0 - 2M(B)$ , which means that the parity of  $M(A)$  will never change (considering that zero is even). It is then easy to prove that if  $f^T M > 2$  and is odd, then the Petri Net is live; if  $f^T M$  is even, then the Petri Net is not live ( $t_1$  can be enabled until A has no token which means that it is not possible to enable  $t_1$  and  $t_2$  anymore).

Because  $f > 0$ , we have A and B bounded by  $f^T M_0$  for any initial marking  $M_0$ . A and B are therefore structurally bounded.

What is remarkable about this tiny example is that it was not necessary to develop the reachability Artless graph in order to decide whether or not the net is live or bounded. We could analyze the net even partially ignoring the initial marking [namely,  $M_0(B)$ ].

An interesting behavioral lesson to draw from this tiny example has to do with liveness: once a net is live, adding more tokens does not guarantee that liveness will persist; as a matter of fact, adding one token to A will make the net not live while adding a second token will make it live again. Adding more tokens does not necessarily result in liveness. This is a false "good idea" that I encountered with many students and engineers.

We encourage the reader to similarly design another Petri Net to semi-decide whether a number is odd.

## 5 A Few Words to Conclude

About nets, I can never sufficiently stress the combination of their two key concepts that set Petri Nets apart from many other models. First, a bipartite graph captures and models so well the relationship and connections between actions and resources. Second, the “token game” allows simulations and development of a reachability graph given an initial marking. Clearly, this combination is able to depict concurrency and, in my opinion, plainly supports the notion of parallelism. We have seen how invariants create a link from the static structure (the bipartite graph) to the dynamic evolution (the tokens) of the net. They express constraints over all possible markings, which greatly helps the analysis and discovery of behavioral properties which can be somewhat independent from the initial marking of the net under consideration. Last but not least, they can be computed despite some level of parameterization, avoiding a painstaking symbolic development of the reachability graph.

Well, sometimes just a few words at the right time suffice. As I said in the introduction, I met Carl Adam Petri only a few times, however, I keep a vivid and precious recollection of his encouragements.



# Toward Distributed Computability Theory



Roberto Gorrieri

## 1 Introduction

Petri nets were introduced by Carl Adam Petri through his Ph.D. dissertation [19] in 1962, the year I was born. Unfortunately, I never had the opportunity to talk to Petri about his important computational model. As a matter of fact, I became acquainted with Petri nets in the late 1980s in Pisa, during my Ph.D. studies, under the illuminating supervision of Ugo Montanari and Pierpaolo Degano. At those times, they were studying, together with Rocco De Nicola, how to find a suitable distributed semantics for CCS in terms of safe Petri nets [7]. This line of research can be defined by the motto *Petri nets for Process Algebras*. More recently, I approached the reverse problem of finding suitable process algebras to represent specific classes of Petri nets, whence the motto *Process Algebras for Petri Nets* [9]. The problem described in this short chapter is a sort of by-product of this line of research; this note emphasizes the prominent role that, in my opinion, Petri nets should play in the theory of computation.

In fact, classic computability theory is based on sequential models of computation, such as Turing machines [6, 14, 23]. It is sometimes argued that Turing-complete models of computations are equally expressive. However, I argue that there are problems in distributed computing, such as the Last Man Standing problem [9, 24], that are not solvable in the Turing-complete model CCS(25,12) [10], while they are solvable within finite, nonpermissive [9] Petri nets, a class of nets conservatively extending finite Petri nets with inhibitor arcs. Hence, I argue that Petri nets, in their many facets, are more suitable than sequential models of computation for assessing the relative expressive power of different languages for

---

R. Gorrieri (✉)

Dipartimento di Informatica, Scienza e Ingegneria, Università di Bologna, Bologna, Italy  
e-mail: [roberto.gorrieri@unibo.it](mailto:roberto.gorrieri@unibo.it)

distributed systems. In doing so, I lay the foundation for *distributed* computability theory, as a generalization of *sequential* (or Turing) computability theory.

## 2 Turing Computability

Classic computability theory was developed in the 1930s and 1940s starting from sequential models of computation, such as Turing machines [6, 14, 23], lambda calculus [1, 4], and the like [5, 17, 22]. The computable objects of these formalisms are partial functions over natural numbers. The famous Church-Turing thesis [15] states that any function that is algorithmically computable in some finite formal system is computable by means of a Turing machine. Hence, the supposed validity of the Church-Turing thesis ensures that two sequential models of computation (or two sequential languages) are equally expressive if they are both Turing-complete, i.e., if they both compute all the Turing-computable functions.

However, when considering concurrent models of computations, such as process algebras, the situation is a bit different. First of all, it is well known that a function may not be the suitable semantic model for a concurrent program (this dates back at least to the work of Bekič [2] at the beginning of the 1970s; see also [12, 16] for a more recent and accessible discussion of the problem), and so Turing-completeness does not seem to be the right criterion for comparing the expressive power of two concurrent languages. As a matter of fact, the class of problems that a concurrent language can solve may include the Turing-computable functions, but it may also include many problems that have nothing to do with functions, as illustrated in the next section.

## 3 Last Man Standing Problem

The *Last Man Standing* (LMS, for short) problem, originally introduced in [24], can be solved in a process algebra if there exists a process  $p$  able to detect, in a finite amount of time, the presence or absence of other copies of itself. We need to identify a process  $p$  such that  $p$  is able to execute an action  $a$  only when there is exactly one copy of  $p$  in the current system, while  $p$  is able to perform an action  $b$  only when there are at least two copies of  $p$  in the current system. To be precise, if  $q_i$ , is the system where  $i$  copies of  $p$  are enabled, we require that all of its computations will end eventually (i.e., no divergence is allowed) and that the observable content of each of these computations is  $a$  if  $i = 1$  and  $b$  if  $i > 1$ , where  $a \neq b$ . In other

words, if  $\xRightarrow{\ell}$  stands for a, possibly empty, sequence of silent transitions followed by an  $\ell$ -labeled transition, then

$$\begin{array}{l}
 \text{while} \\
 q_1 = p \qquad q_1 \xRightarrow{a} q'_1 \not\rightarrow \quad q_1 \not\xrightarrow{b} \\
 q_2 = p \mid p \qquad q_2 \not\xrightarrow{a} \qquad q_2 \xRightarrow{b} q'_2 \not\rightarrow \\
 \dots \\
 q_n = \underbrace{p \mid p \mid \dots \mid p}_n \quad q_n \not\xrightarrow{a} \qquad q_n \xRightarrow{b} q'_n \not\rightarrow
 \end{array}$$

where  $\mid$  is the parallel composition operator of the calculus under scrutiny. The operational rules for parallel composition of CCS [12, 16] state that any process  $p$ , able to execute some action  $a$ , can perform the same action in the presence of other processes as well, so that if  $p \xRightarrow{a} p'$ , then also  $p \mid p \xRightarrow{a} p \mid p'$ , which contradicts the requirement that  $q_2 \not\xrightarrow{a}$ . As a matter of fact, CCS is *permissive*: no parallel process can prevent the execution of an action of another process. Hence, CCS cannot solve the LMS problem.

However, the LMS problem can be solved in other calculi that possess some ability to make contextual checks. In fact, [24] proposes a simple, non-Turing-complete calculus, called FAP, which is able to express some form of priority among its actions and to solve the LMS problem. Furthermore, [9] proposes a Turing-complete calculus, called NPL, which is able to perform atomic tests for absence of certain concurrent processes, which can also solve the LMS problem.

Hence, there exists a problem in concurrency theory (i.e., the LMS problem) that a Turing-complete language (e.g., CCS(25,12), i.e., CCS using at most 25 constants and 12 actions [10]) cannot solve, while it can be solved by a non-Turing-complete calculus (i.e., FAP). Therefore, what is the analog of computable function for concurrency? And what is the analog of Turing-completeness for concurrency? New definitions are necessary. Here I present my own proposal as a possible new foundation for *distributed* computability theory—as a generalization of classic, *sequential* (or Turing) computability theory.

## 4 Distributed Computability Theory

A *process* is a semantic model, up to some behavioral equivalence. For instance, if the chosen models are labeled transition systems and the chosen equivalence is (weak completed) trace equivalence, then a process is nothing but a formal language [6, 14]; if, instead, the chosen models are Petri nets [18, 19, 21] and the chosen equivalence is net isomorphism, then a process is a Petri net, up to net isomorphism. In other words, the notion of computable function on the natural numbers for

sequential computation is to be replaced by a model with an associated equivalence relation, which we call a process, for concurrent computation.

A process algebra is *complete* w.r.t. a given class of models if it can represent all the elements of that class, up to the chosen behavioral equivalence. A class of models is Turing-complete if it can compute all the computable functions; if a process algebra is complete w.r.t. that class of models, then it is also Turing-complete.

Different process algebras usually have different expressive power. In [9] I present a list of six increasingly expressive process algebras, each one *complete* w.r.t. some class of Petri nets, up to *net isomorphism*. The most expressive one is NPL, which is the only language studied in [9] to be Turing-complete, as it can represent all finite, nonpermissive Petri nets (a class of nets conservatively extending the class of nets with inhibitor arcs), and which can also solve the LMS problem. However, there are other process algebras, such as CCS(25,12) [10], that are Turing-complete, but that can neither represent all the finite, nonpermissive Petri nets, nor solve the LMS problem. Nonetheless, CCS(25,12) can represent many infinite Place/Transition Petri nets, and so NPL and CCS(25,12) are incomparable, at least if the considered semantic equivalence is net isomorphism.

## 5 Why Not Use Labeled Transition Systems?

If a process is a semantic model, up to some behavioral equivalence, we might think, as many process algebra scholars do, that labeled transition systems, equipped with some bisimulation-like equivalence, are more than enough for the purpose.

As a matter of fact, it is often assumed that the only relevant part of the behavior of a system is given by its *interaction capabilities*, i.e., the actions the system performs with which an observer can interact. This argument is based on the assumption that a system, like a vending machine, is actually a *black box*, and that the observer of the system can only interact with this system by means of the externally visible buttons or slots, but without any knowledge of the internal structure of the machine. This idea is quite appealing and has an obvious consequence that an interleaving model such as labeled transition systems, showing which visible actions can be performed and when, is more than enough to completely describe the behavior of a system. However, by modeling a reactive, distributed system with a transition system, we are forced to make one fundamental abstraction of its structure: the states of the reactive system are monolithic entities that cannot be inspected and each transition from, say, state  $q$  to state  $q'$ , with label  $a$ , transforms the state  $q$  as a whole, even if only some components of the system are actually involved in the execution of action  $a$ . As the structure of the system is abstracted away in interleaving semantics, distributed systems with very different structures and very different properties can be equated. For instance, a deterministic, symmetric, fully distributed, deadlock-free solution to the well-known dining philosophers problem (originally proposed by Dijkstra [8], and then

elaborated on by Hoare [13] in its current formulation) can be provided in Multi-CCS (see Chapter 6 of [12]). Let  $p$  be the Multi-CCS solution to this problem; its interleaving semantics is a finite-state labeled transition system, which is also the semantics of a purely sequential process  $q$ ; since  $p$  and  $q$  are interleaving equivalent, we may wrongly conclude that  $q$  is also a deterministic, symmetric, fully distributed, deadlock-free solution to the dining philosophers problem! Of course, this is not the case. Since the properties of interest for a distributed system are often related to the structure of the system, the semantics cannot abstract away from this aspect of the system. Not surprisingly, the Petri net semantics of the Multi-CCS process  $p$  mentioned above is expressive enough to show that the solution that  $p$  offers to the dining philosophers problem is deterministic, symmetric, fully distributed and deadlock-free, indeed.

## 6 Conclusion and Future Research

If a process is a Petri net, up to some behavioral equivalence, which behavioral equivalence is the most adequate? In my study [9] I have used net isomorphism. However, this equivalence is very concrete and one may wonder whether some weaker equivalence relation can be more suitable instead. Process algebra scholars usually compare the expressive power of different process algebras by showing the existence (or nonexistence) of suitable encodings, up to (weak) bisimulation equivalence. If we desire a similar technique, a suitable candidate behavioral equivalence for Petri nets should be the distributed generalization of (sequential) bisimulation for labeled transition systems. Even though some concurrent bisimulation-like definitions for Petri nets have been proposed in the literature (see, e.g., [3, 20]), I think that further research is needed to yield a really satisfactory definition in this direction [11].

## References

1. H.P. Barendregt, *The Lambda Calculus: Its Syntax and Semantics*. Studies in Logic and the Foundations of Mathematics, vol. 103 (revised ed.) (North Holland, Amsterdam, 1984)
2. H. Bekič, The semantics of parallel processing, in *Programming Languages and Their Definition - Hans Bekič (1936–1982)*, ed. by C.B. Jones. Lecture Notes in Computer Science, vol. 177 (Springer, Berlin, 1984), pp. 215–229
3. E. Best, R. Devillers, A. Kiehn, L. Pomello, Concurrent bisimulations in Petri nets. *Acta Inform.* **28**(3), 231–264 (1991)
4. A. Church, An unsolvable problem of elementary number theory. *Am. J. Math.* **58**(2), 345–363 (1936)
5. M. Davis (ed.), *The Undecidable, Basic Papers on Undecidable Propositions, Unsolvability Problems And Computable Functions* (Raven Press, Hewlett, 1965)
6. M.D. Davis, E.J. Weyuker, *Computability, Complexity and Languages* (Academic, New York, 1983)

7. P. Degano, R. De Nicola, U. Montanari, A distributed operational semantics for CCS based on C/E systems. *Acta Inform.* **26**(1–2), 59–91 (1988)
8. E.W. Dijkstra, Hierarchical ordering of sequential processes. *Acta Inform.* **1**(2), 115–138 (1971)
9. R. Gorrieri, *Process Algebras for Petri Nets: The Alphabetization of Distributed Systems*. EATCS Monographs in Theoretical Computer Science (Springer, Berlin, 2017)
10. R. Gorrieri, CCS(25,12) is Turing-complete. *Fund. Inform.* **154**(1), 145–166 (2017)
11. R. Gorrieri, Verification of finite-state machines: a distributed approach. *J. Logic Algebraic Methods Program.* **96**, 65–80 (2018)
12. R. Gorrieri, C. Versari, *Introduction to Concurrency Theory: Transition Systems and CCS*. EATCS Texts in Theoretical Computer Science (Springer, Berlin, 2015)
13. C.A.R. Hoare, *Communicating Sequential Processes* (Prentice Hall, Upper Saddle River, 1985)
14. J.E. Hopcroft, R. Motwani, J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*, 2nd edn. (Addison-Wesley, Boston, 2001)
15. S.C. Kleene, Recursive predicates and quantifiers. *Trans. Am. Math. Soc.* **53**(1), 41–73 (1943)
16. R. Milner, *Communication and Concurrency* (Prentice Hall, Upper Saddle River, 1989)
17. M.L. Minsky, *Computation: Finite and Infinite Machines* (Prentice Hall, Upper Saddle River, 1967)
18. J.L. Peterson, *Petri Net Theory and the Modeling of Systems* (Prentice Hall, Upper Saddle River, 1981)
19. C.A. Petri, Kommunikation mit Automaten. Ph.D. Dissertation, University of Bonn, 1962
20. L. Pomello, G. Rozenberg, C. Simone, A survey of equivalence notions for net based systems, in *Advances in Petri Nets: The DEMON Project*, ed. by G. Rozenberg. Lecture Notes in Computer Science, vol. 609, pp. 410–472 (Springer, Berlin, 1992)
21. W. Reisig, *Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies* (Springer, Berlin, 2013)
22. H. Rogers Jr., *Theory of Recursive Functions and Effective Computability* (MIT Press, Cambridge, 1987) (original edition published by McGraw-Hill, 1967)
23. A.M. Turing, On computable numbers, with an application to the Entscheidungsproblem, in *Proceedings of the London Mathematical Society*. Series 2, vol. 42 (1936), pp. 230–265
24. C. Versari, N. Busi, R. Gorrieri. An expressiveness study of priority in process calculi. *Math. Struct. Comput. Sci.* **19**(6), 1161–1189 (2009)

# Petri Inheritance: The Foundation of Nondeterministic, Concurrent Systems



Roberto Bruni and Ugo Montanari

## 1 A Personal Experience

In September 1973 I (the second author) attended one of the first (actually the second) conference on *Mathematical Foundations of Computer Science* at Štrbské Pleso, High Tatras. I was travelling by car and it was quite an adventure since there were some problems at the borders due to prevention measures for cholera. At Štrbské Pleso I met for the first time Carl Adam Petri. Actually, on this occasion I also heard about Petri nets for the first time. There was a lot of excitement about nets, which were anyway already more than 10 years old. I was very impressed by Petri: his taste for mathematical foundations of concurrency, reminiscent of Dedekind, was quite appealing to me. Thus I decided to invite him for the first Italian *Convegno di Informatica Teorica*, which I coorganised at Mantova, in November 1974. Later on, a thread of collaboration was developed: a former Pisa student, Pippo Torrigiani, worked for a few years at Schloss Birlinghoven, Sankt Augustin, where I also visited for two 1-month periods at the end of the 1970s.

At Schloss Birlinghoven I had several interesting discussions with Hartmann Genrich, Kurt Lautenbach and P. S. Thiagarajan. With Petri I had just one session, a full afternoon long, about general net theory. As a main goal, I was interested in possible semantic domains for concurrency consisting of unfolded occurrence nets representable in three dimensions: causality, nondeterminism and concurrency. Here *maximal* sections having as dimensions nondeterminism and concurrency would represent states of the nondeterministic computation, causality  $\times$  nondeterminism would show the views of certain sequential observers, while causality  $\times$  concurrency would model concurrent computations. Some results were

---

R. Bruni (✉) · U. Montanari  
Dipartimento di Informatica, Università di Pisa, Pisa, Italy  
e-mail: [bruni@di.unipi.it](mailto:bruni@di.unipi.it); [ugo@di.unipi.it](mailto:ugo@di.unipi.it)

presented in a joint paper with Andrea Simonelli. I learned quite a lot during my stay at Schloss Birlinghoven, in particular concepts and constructions very useful for my subsequent work on concurrency. Slightly later, the work by Nielsen, Plotkin and Winskel on event structures and domains appeared at a conference on *Semantics of Concurrent Computation* and Glynn Winskel wrote his thesis at the University of Edinburgh on events in computation, establishing a strong basis for domain-based concurrency theory. In both cases the formal developments were explicitly motivated and made possible by Petri net models and results.

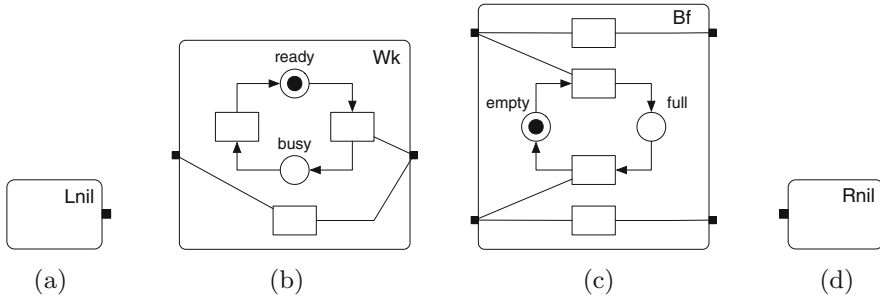
A renewed interest in Petri nets matured during my sabbatical at SRI in 1987. José Meseguer and I wrote a paper (published at LICS 1988 and in *Information and Computation*) titled *Petri Nets are Monoids*, which gave for the first time a presentation of the semantics of Place Transition (PT) nets in terms of symmetric monoidal categories (SMC). Later on, together with Vladimiro Sassone, a full picture connecting SMC with event structures and prime algebraic domains was developed. Finally, further work, in collaboration with the first author, made the connection stronger, in fact functorial.

More recently, a variety of networks were represented as arrows of SMC enriched with signatures of operations equipped with axioms, concisely represented as string diagrams. This research suggested a line (by both authors, Hernán Melgratti and Pawel Sobocinski) where compositional versions of Petri nets with observations and connector algebras have been defined. Particularly interesting is the enabling mechanism, defining a distributed choice in the presence of distribution, concurrency and nondeterminism. An abstract definition of such mechanisms has shown that it is the most general possible. This universal result confirms the importance and generality of Petri nets, also in comparison with other formalisms (BIP, REO etc.). This fundamental result is presented succinctly in the remainder of this chapter.

## 2 Petri Nets as a Connector Algebra

Net composition is a useful tool for modelling systems and proving their properties as well as for comparing the expressive power of Petri nets with other formalisms. Several notions of composition have appeared in the literature, often based on interfaces made of places, of transitions or of both. Notable examples are the approaches by Nielsen, Priese and Sassone, based on combinators, by Best, Devillers and Koutny, based on the Petri box algebra, by Baldan, Corradini, Ehrig and König, based on open nets, by Reisig, based on interface nets, by Katis, Sabadini and Walters, based on the bicategory of  $\text{Span}(\text{Graph})$ , by van der Aalst and others, based on net modules, and the very recent Springer book by Roberto Gorrieri that surveys suitable process algebras that can represent specific classes of Petri nets. Here we overview nets with boundaries, as introduced by Pawel Sobocinski and





**Fig. 1** Some nets with boundaries. (a) Lnil. (b) Worker model. (c) Buffer model. (d) Rnil

studied in our joint work *Connector algebras for C/E and P/T nets' interactions*<sup>1</sup> Interestingly, they have been exploited to compare the expressiveness of Petri nets with other formalisms for modelling distributed decisions, such as Reo and BIP.

Nets with boundaries extend ordinary Petri nets with left and right interfaces that can be used for composition. An interface is just a list of ports, to which transitions can be attached. Names of ports are not important, so they are named as natural numbers, according to the position they occupy in the interface. The distinction of left and right interfaces is convenient for defining sequential composition of nets with boundaries, but it would be misleading to think about an input/output distinction. The general idea is that a transition attached to some ports is a sort of fragmented transition that must be completed with other fragments via an interaction on shared ports. Graphically, this intuition is rendered by using undirected arcs to connect ports with transitions.

The operational semantics of nets with boundaries can be expressed as a labelled transition system whose labels are pairs modelling the observable interactions on the ports of the left and right interfaces. Interestingly, the corresponding bisimilarity equivalence is a congruence w.r.t. parallel and sequential composition of nets with boundaries. A concise formal account follows.

For a natural number  $n \in \mathbb{N}$ , we let  $\underline{n} = \{0, 1, \dots, n - 1\}$ . Given  $m, n \in \mathbb{N}$ , a *net with boundaries*  $N : m \rightarrow n$  is a tuple  $N = (S, T, \overset{\circ}{-}, \overset{\circ}{-}, \bullet{-}, -\bullet)$  where  $S$  is the set of places,  $T$  is the set of transitions, the functions  $\overset{\circ}{-}, \overset{\circ}{-} : T \rightarrow 2^S$  assign sets of places, called respectively the pre-set and the post-set, to each transition and the functions:  $\bullet{-} : T \rightarrow 2^{\underline{m}}$  and  $-{\bullet} : T \rightarrow 2^{\underline{n}}$  transitions to the left and right boundaries of  $N$ , respectively.

Figure 1 presents some nets with boundaries that we use as a running example. The net Wk:  $1 \rightarrow 1$  in Fig. 1b models a worker (either a producer or a consumer) that is *ready* to perform some action to enter the *busy* state. Its left and right interfaces each consist of just one port. The transition from ready to busy is attached to the unique port of the right interface. The bottom transition is attached to both ports: it

<sup>1</sup>Logical Methods in Computer Science 9(3) (2013).

will be used to compose the worker with other workers. The transitions that share a port compete for interaction with that port.

The net Bf:  $2 \rightarrow 2$  in Fig. 1c models a buffer to store the item produced by some worker. The buffer has only one slot: it can be *empty* and ready to accept an item, or *full* and ready to dispense the stored item. In this case the left and right interfaces have two ports each: the topmost port in the left interface will be used to combine the buffer with some producer; the bottom port in the left interface will be used to combine the buffer with some consumer, the two ports in the right interface can be used to combine the buffer with other buffers.

The nets Lnil:  $0 \rightarrow 1$  in Fig. 1a and Rnil:  $1 \rightarrow 0$  in Fig. 1d are empty: they can be used to restrict interaction on some ports.

Given two nets  $N_1 : m_1 \rightarrow n_1$  and  $N_2 : m_2 \rightarrow n_2$ , their parallel composition is the net  $N_1 \otimes N_2 : m_1 + m_2 \rightarrow n_1 + n_2$  obtained by the disjoint union of the two nets, up to an obvious rearrangement of their left and right interfaces (the ports of  $N_1$  precede those of  $N_2$ ).

Sequential composition is defined when the right interface of one net with boundaries matches the left interface of another net. Given  $N_1 : m \rightarrow k$  and  $N_2 : k \rightarrow n$ , their sequential composition is the net  $N_1 ; N_2 : m \rightarrow n$  whose places are the disjoint union of the places of  $N_1$  and  $N_2$  and whose transitions are all the possible (minimal) synchronisations of (mutually independent) sets of transitions from  $N_1$  and  $N_2$  that interact over the  $k$  shared ports.

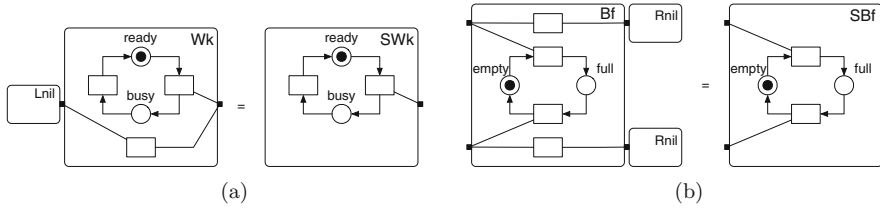
Formally, two transitions  $t, u$  are said to be *independent* when their sources as well as their targets are separated, i.e. when

$${}^\circ t \cap {}^\circ u = \emptyset \wedge t^\circ \cap u^\circ = \emptyset \wedge {}^\bullet t \cap {}^\bullet u = \emptyset \wedge t^\bullet \cap u^\bullet = \emptyset$$

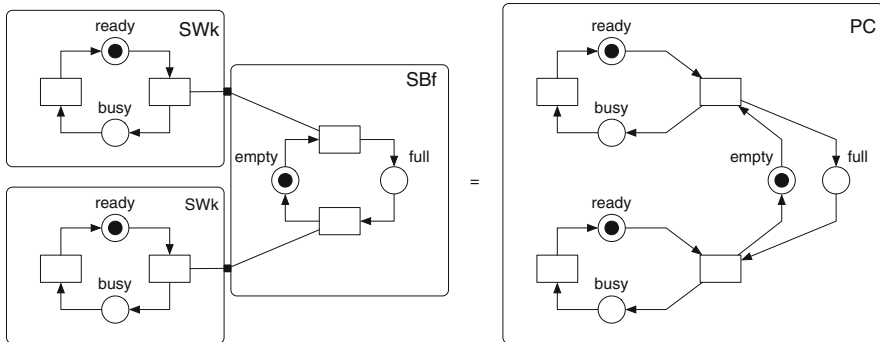
and a set  $U$  of transitions is mutually independent when, for all  $t, u \in U$ , if  $t \neq u$  then  $t$  and  $u$  are independent. We say that a *synchronisation* is a pair  $(U_1, U_2)$  with  $U_1$  and  $U_2$  mutually independent (disjoint) sets of transitions of  $N_1$  and  $N_2$ , respectively, such that: (1)  $U_1 \cup U_2 \neq \emptyset$  and (2)  $U_1^\bullet = {}^\bullet U_2$ . The set of synchronisations inherits an ordering from the subset relation, i.e.  $(U_1, U_2) \subseteq (U'_1, U'_2)$  when  $U_1 \subseteq U'_1$  and  $U_2 \subseteq U'_2$ . A synchronisation is *minimal* when it is minimal with respect to this order. The intuition is that a minimal synchronisation cannot be broken in simpler synchronisations. As a special case, note that any transition  $t_1$  in  $N_1$  (respectively  $t_2$  in  $N_2$ ) not connected to the shared boundary  $k$  defines a minimal synchronisation  $(\{t_1\}, \emptyset)$  (respectively  $(\emptyset, \{t_2\})$ ).

Next, we discuss some compositions of the nets of our running example. First we can model a single worker by restricting the left interface of a worker: the corresponding net SWk = Lnil; Wk is shown in Fig. 2a. Similarly, we can model a single buffer by restricting the right interface of a composable buffer: the corresponding net SBf = Bf; (Rnil  $\otimes$  Rnil) is shown in Fig. 2b. Then, we can assemble two workers (a producer and a consumer) with a buffer: the corresponding net PC = (SWk  $\otimes$  SWk); SBf is shown in Fig. 3.

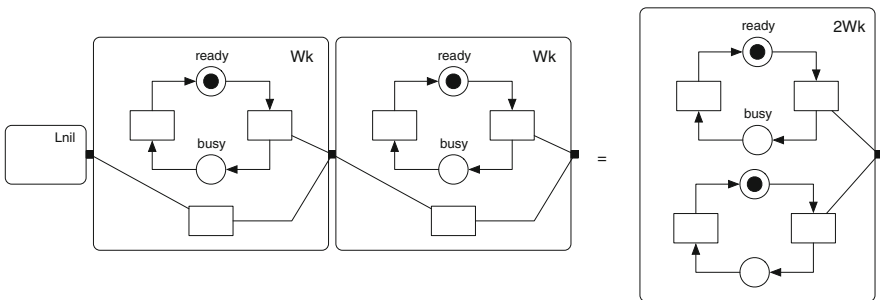
It is interesting to observe what happens when we compose several workers or several buffers together. The net 2Wk = Lnil; Wk; Wk is given in Fig. 4, while the



**Fig. 2** Some composed nets. (a) The net  $SW_k = Lnil; W_k$ . (b) The net  $SB_f = B_f; (Rnil \otimes Rnil)$



**Fig. 3** The net  $PC = (SW_k \otimes SW_k); SB_f$



**Fig. 4** The net  $2W_k = Lnil; W_k; W_k$

net  $2B_f = B_f; B_f; (Rnil \otimes Rnil)$  is given in Fig. 5. We can then compose a system with two producers, one consumer and two buffers as shown in Fig. 6. Of course, the composition immediately generalises to any number of workers and buffers, with the advantage that the notion of synchronisation automatically accounts for the combinatorial explosion of cases (any producer can post an item in any empty buffer and any consumer can retrieve an item from any full buffer).

Parallel and sequential compositions of nets with boundaries have been useful in showing several results. First, it has been shown that Petri nets with boundaries are equivalent to an algebra of connectors that is freely generated by a small set of

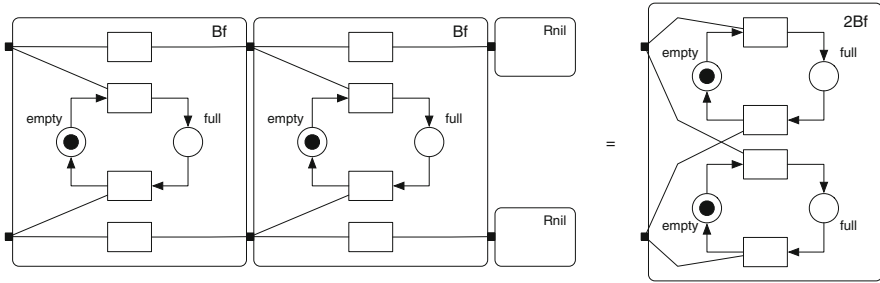


Fig. 5 The net  $2Bf = Bf; Bf; (Rnil \otimes Rnil)$

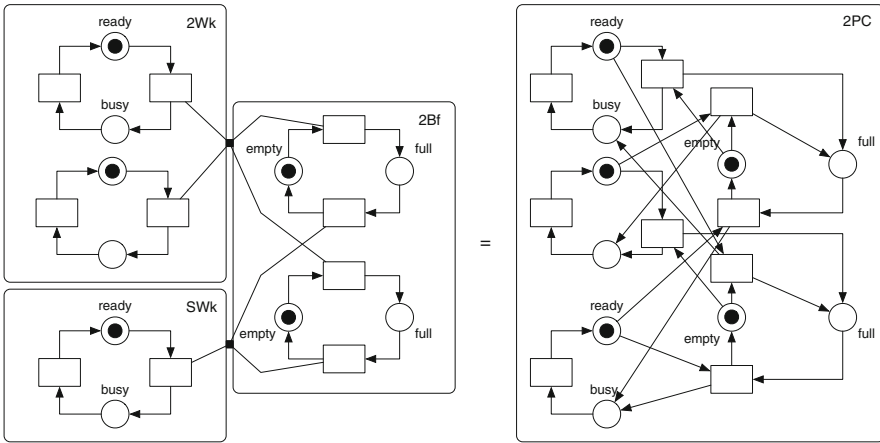


Fig. 6 The net  $2PC = (2Wk \otimes Wk); 2Bf$

basic stateless components (identity, swap, synch, mutex, hiding and noact) together with a one-position buffer component. Exploiting this correspondence, Petri nets with boundaries have been shown to be equivalent to the exogenous coordination framework Reo proposed by Arbab et al. Moreover, exploiting compositionality at the semantic level, Petri nets with boundaries have also been shown to be equivalent to the BIP framework (in the absence of priorities), even in the hierarchical case. An informal account of the above results can be found in our joint paper with Hernán Melgratti *A Survey on Basic Connectors and Buffers*,<sup>2</sup> where interesting connections with the tile model and the wire calculus are also made.

Such semantical equivalences provide further evidence that Petri nets (with boundaries) offer a core stateful model that accounts for distribution, concurrency and nondeterminism, and that Carl Adam Petri's, vision was very insightful in this respect.

<sup>2</sup>Proc. of FMCO 2011, LNCS vol.7542, pp. 49–68, 2013.

However, decades of studies have not been sufficient to solve in a fully satisfactory manner all the issues related to Petri nets. In fact they give rise to a still very active and lively research area, with many interesting open problems. Having the occasion to pick one of them, we direct the reader's attention to the replacement of nondeterminism with probability distributions, i.e. to the coexistence of concurrency and probability. A number of probabilistic versions of Petri nets have been proposed, but most of them introduce time-dependent stochastic distributions, thus giving up the time and speed independence feature typical of proper truly concurrent models. In the absence of confusion, a solution has been proposed by Varacca and Winskel based on event structures. In the more general case, Abbes and Benveniste's branching cells provide a more complex solution but they miss compositionality and a description of the probabilistic execution in terms of some concurrent transition system. The authors and Hernán Melgratti have recently achieved substantial progress about these aspects (LICS'18).

We foresee that the connection between Petri nets (with boundaries) and algebras of connectors can be instrumental in devising a compositional approach to the definition of a general model where concurrency and probability are accounted for in a fully satisfactory manner. Also, remarkable analogies between probabilistic Petri processes without confusion and Bayesian networks may suggest how to extend well-known analysis techniques from the latter to the former model.



Ekkart Kindler

## 1 Introduction

I met Carl Adam Petri in person only a few times. But I grew up, scientifically speaking, in the group of Wolfgang Reisig who came directly from Petri's group at GMD in Sankt Augustin, bringing along with him Jörg Desel as my, so to speak, older brother and some other Ph.D. students who had been working at GMD. In this group at TU München, we had many discussions about Petri's philosophy and I learned about Petri's way of thinking. I also learned how much time Petri spent coming up with good titles with the right level of ambiguity—or, more precisely, with a meaning on different levels—such as his Ph.D. thesis [1]. Most of all, I learned about Petri nets and the philosophy behind them.

In this short paper, I briefly discuss a notation which I came up with much later in my career, the *Event Coordination Notation (ECNO)*, which allows modelling of the behaviour of larger software and generation of the code for that software fully automatically from such models [2, 3]. ECNO's concepts have been inspired by Petri's philosophy and the discussions in my early scientific life with people who had been exposed to Petri's ideas. I know that Petri would not have liked some of ECNO's concepts, but I hope that he would have liked some of its core concepts, and I am sure he would have liked the “meta”-aspect of it, in the sense that ECNO can be used for defining the semantics of Petri nets and even the semantics of ECNO itself.

For lack of space, ECNO is introduced by example only, and the example is actually the semantics of Petri nets. To be precise, it is the semantics of *Place/Transition Systems* [4], which would probably be Petri's first disappointment,

---

E. Kindler (✉)  
DTU Compute, Technical University of Denmark, Lyngby, Denmark  
e-mail: [ekki@dtu.dk](mailto:ekki@dtu.dk)

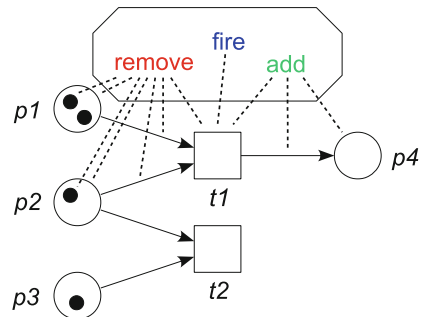
since he did not particularly like *Place/Transition Systems*. Section 2 introduces *Place/Transition Systems* and defines their intuitive semantics, which, along the same lines, is formalised in Sect. 3 using ECNO. Finally, in Sect. 4, I discuss some of ECNO's main ideas and what, I believe, Petri would have made of them.

## 2 Petri Nets

Figure 1 shows a Petri net in which transition  $t_1$  is about to fire, and the events involved in firing  $t_1$  are indicated in an octagon. Carl Adam Petri would probably not be happy with this example for two reasons: firstly, the example is a *Place/Transition System*, where more than one token can be on a place, which violates his idea of elementary states being based on logic; secondly, we break down the firing of a transition into smaller parts – something which he considered to be atomic is broken down to sub-atomic parts. Anyway let us see where this will lead us and hope that, in the end, Petri would be happy again.

When transition  $t_1$  in the Petri net of Fig. 1 fires, we say that the transition participates in a *fire* event, indicated by the dashed line from the *fire* event in the octagon to the transition. The *fire* event is broken down into two other events: *remove* and *add*. So, when the transition participates in the *fire* event, the transition also participates in these two events. And the requirements on the *remove* and *add* events will actually define when a *fire* event can happen and what its effect will be. When a transition participates in a *remove* event, it is required that all incoming arcs also participate in this *remove* event; in the situation of Fig. 1, this is indicated by the two dashed lines from the *remove* event to the incoming arcs of  $t_1$ . If an arc participates in a *remove* event, this requires the source place of the arc to participate in this event too; in our example, these are the two places  $p_1$  and  $p_2$ . And for each of these places, one token on each place also needs to participate in the *remove* event—again indicated by dashed lines. For place  $p_1$ , we would actually have two choices, where the top left token was chosen in the *interaction* shown in Fig. 1. These two tokens are the ones removed when the transition fires. Likewise, when the transition participates in an *add* event, all outgoing arcs also need to participate

**Fig. 1** A Petri net example with an interaction firing transition  $t_1$



in this *add* event; in our example indicated by the dashed line again. Finally, when an arc participates in an *add* event, the target place of that arc needs to participate in the *add* event too. And the place participating in the *add* event will create a new token on the place. Together, these rules define the firing rule of Petri nets: we need a token on each of the incoming places, which will be removed when the transition fires, and one new token will be created on each outgoing place.

For Petri, this was to happen atomically. But above, we have broken this down into smaller, simpler and local rules, which together have this effect, and will be executed atomically (we say as an *interaction*) if and when all the above elements participating in the events fall into place.

### 3 Modelling Petri Nets

In order to make the firing rules from Sect. 2 explicit, we formalise them in ECNO. We distill the underlying principles of Petri nets and how their behaviour is coordinated into an ECNO model.

Figures 2, 3, 4, 5, and 6 completely define the behaviour of Place/Transition systems. Figures 3, 4, 5, and 6 model the life cycles of the different Petri net elements as so-called *ECNO nets*, and Fig. 2 shows the coordination of the behaviour among the different Petri net elements as a so-called *ECNO coordination diagram*. Here, we use the example to explain some of ECNO's concepts and notations, and show how they can be used to rephrase our informal description of the firing rule from Sect. 2.

The ECNO coordination diagram from Fig. 2 defines the three types of events *fire*, *remove* and *add* that we used in the informal discussion. They are shown as rounded rectangles. Moreover, the coordination diagram shows the four types of elements of Petri nets and their relation and how the different elements coordinate their participation in the events.

Let us now assume that a transition participates in a *fire* event. The life cycle of a transition shown in Fig. 3 says that a transition can participate in a *fire* event at any time, but, when that happens, the transition will also participate in a *remove* and an *add* event.

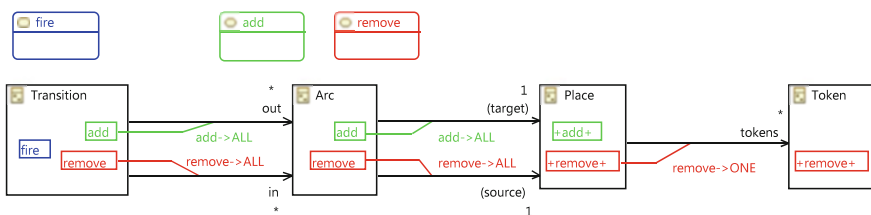


Fig. 2 ECNO coordination diagram for P/T-systems



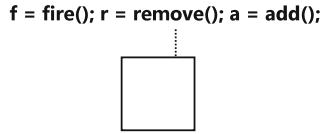


Fig. 3 Life cycle of a Transition

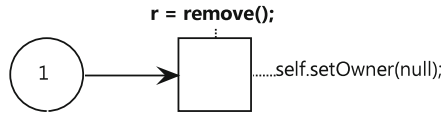


Fig. 4 Life cycle of a Token

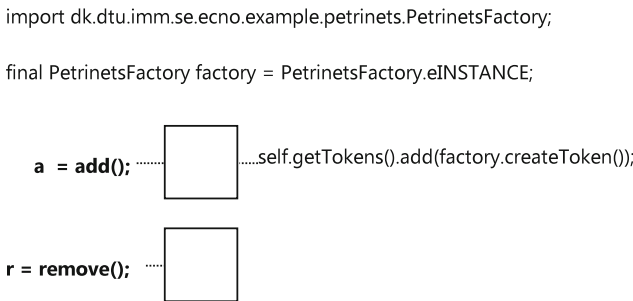


Fig. 5 Life cycle of a Place

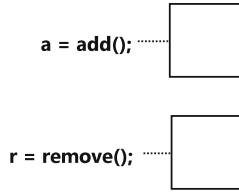
The coordination diagram in Fig. 2 says that, when a transition participates in a *remove* event, all its *incoming* arcs will participate in this *remove* event too. Likewise, it says that, when an arc participates in a *remove* event, all<sup>1</sup> its *source* places will participate in the *remove* event, too. For a place participating in a *remove* event, the coordination diagram requires one of its tokens to participate in the *remove* event, too: the one that will be removed from the place.

Similarly, the coordination diagram in Fig. 2 says that, with a transition participating in an *add* event, all its *outgoing* arcs and indirectly also the associated *target* places need to participate in the *add* event.

The life cycles of the arc and place elements of Figs. 6 and 5 show that the arc and place are always ready to participate in *add* and *remove* events.

The life cycle of the place shown in Fig. 5 is slightly more interesting: when a place participates in an *add* event, this is associated with an *action*: the piece of Java code `self.getTokens().add(factory.createToken())`. This will actually create a new token on the place when the place participates in an *add* event.

<sup>1</sup>Note that the arc has exactly one source place, so all means one in this case.



**Fig. 6** Life cycle of an Arc

Similarly, the life cycle of the token shown in Fig. 4 has an action associated with a *remove* event: `self.setOwner(null)`. This removes the token from the place it is contained in.

Altogether, this formally defines the behaviour of Petri nets as informally discussed in Sect. 2. Figure 4 shows that the life cycles of the Petri net elements are actually a special kind of Petri net. The others are very degenerate Petri nets (transitions without any places in their pre- and post-set). Only for the life cycle of tokens, there is a precondition: a token can be used (removed) only once in its lifetime. This is an aspect that might have fascinated Carl Adam Petri: in the definition of the behaviour of Petri nets, Petri nets are used again—just very simple ones.

## 4 Discussion

Altogether, ECNO allows us to define the behaviour of a system by two kinds of models: first, the *life cycles* for the system’s elements defining at which points an element can participate in which events and which actions are executed when they do; second, the *coordination diagram*, which defines rules for which other elements need to participate in an event when a certain element participates in this event. The life cycles are local to the elements; the coordination requirements are local too, in that they require elements adjacent to an element to participate too. This locality would certainly be appealing to Petri. What Petri probably would not like is that the coordination requirements apply transitively, and, this way, a corresponding *interaction* can involve many not directly related elements—and there is no upper bound on the number and distance of elements participating in an interaction. For our ECNO model of Petri nets, this would be the pre-set and the post-set of the transitions and the respective tokens, which is local. But, in general it could be much more.

Petri was, however, very much interested in the fundamental principles of “communication with automata” being practical [1]. The question is whether the way “behaviour is coordinated” should be governed by the principles of physics or by what an engineer would want to achieve even if it is not directly achievable by physics—as long as it can be enacted by some underlying mechanism. The

implementation of ECNO shows that interactions can be enacted *atomically* and *independently* of each other. And as long as the sets of participating elements of two interactions are disjoint, they can even be executed *concurrently*. This would probably appeal to Petri.

Petri would probably be surprised that ECNO nets (a kind of Petri nets) are used for defining the life cycles of the elements, which are local and not coordinating behaviour at all. Interestingly enough, it turns out that it is crucial for defining some behaviours that the life cycle of an element can participate in different events concurrently: for example, in order to fire a transition with a loop, the involved place needs to participate in an *add* and a *remove* event at the same time.

Very much in the spirit of Petri nets, ECNO does not mandate what will actually happen in a modelled system. ECNO defines only which interactions can happen in a given situation, very much like conflicting enabled transitions. What will actually happen in an ECNO application is non-deterministic—or driven by some external controller.

Here, we have shown that ECNO can model the behaviour of Petri nets, and this works also for other approaches for modelling behaviour, such as process algebras. This way, ECNO distills the underlying principles of “coordinating behaviour”. In particular, ECNO comes with a notion of atomicity: the corresponding interactions. And I hope that Petri would have liked that. Finally, I believe that ECNO can be used to define its own semantics including its notion of atomicity. And I am sure that Carl Adam Petri would have had fun with that idea.

## References

1. C.A. Petri, Kommunikation mit Automaten. Technical Report Schriften des IIM, Nr. 2, Institut für Instrumentelle Mathematik, Bonn, 1962
2. E. Kindler, An ECNO semantics for Petri nets. Petri Net Newsletter **81**, 3–16 (2012). Cover Picture Story
3. J. Jepsen, E. Kindler, The event coordination notation: behaviour modeling beyond Mickey Mouse, in *Behavior Modeling—Foundations and Applications. International Workshops, BMFA 2009–2014*, ed. by E. Roubtsova, A. McNeile, E. Kindler, C. Gerth. Revised Selected Papers. Lecture Notes in Computer Science, vol. 6368 (Springer, Berlin, 2015)
4. W. Reisig, Place/Transition systems, in *Petri Nets: Central Models and Their Properties*, ed. by W. Brauer, W. Reisig, G. Rozenberg. Lecture Notes in Computer Science, vol. 254 (Springer, Berlin, 1987), pp. 117–141

# Inductive Counting and the Reachability Problem for Petri Nets



Peter Chini and Roland Meyer

## 1 A Tough Homework

Being late for a class usually has negative consequences for a student. Your professor will dart an angry glance at you, for sure you will have missed an important argument, and you may have missed details of your homework. It rarely happens that coming late turns into something good. The following is a legend that proves this belief wrong.

Back in 1987, in Bratislava, then in Czechoslovakia, a student is running through the hallways of Comenius University: It is Róbert Szelepcsényi, and he is late for a class. Only after minutes of athletics he reaches the classroom, enters, and sits down. Just arrived, he realizes that the blackboard is already full of text. In a hurry, he unpacks paper and pencil, but there is not enough time to copy everything from the board. Instead, he tries to catch the important bits. Luckily, he manages to copy the exercises.

When doing his homework a few days later, everything works fine, as always: The ideas compose into a solution at a speed that only allows for poor handwriting so as not to slow down the creative process. But on one exercise, his thoughts start stuttering. This exercise is so hard that it cannot be solved right away. But giving up is not an option: Provoked by the hardness, Szelepcsényi considers it his duty to solve the exercise. He tries again and again. More unsuccessful attempts. But still there are so many ideas to follow. He has to put in more hours! He is thinking about the problem every free minute; even in his dreams the problem chases him.

After 2 weeks of obsession, Szelepcsényi finds a solution and hands it in. To his surprise, rather than returning the corrected sheets, the professor calls him in for

---

P. Chini (✉) · R. Meyer

Institute of Theoretical Computer Science, Technische Universität Braunschweig, Braunschweig, Germany

e-mail: [p.chini@tu-bs.de](mailto:p.chini@tu-bs.de); [roland.meyer@tu-bs.de](mailto:roland.meyer@tu-bs.de)

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,  
[https://doi.org/10.1007/978-3-319-96154-5\\_21](https://doi.org/10.1007/978-3-319-96154-5_21)

161

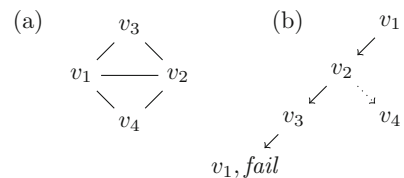
a discussion. What he had solved, the professor explains, was not meant to be his homework. It was a research problem that had been open for more than 20 years! The professor had checked the proposed solution and found it to be correct. He is the first to congratulate him, and the whole computer science community would follow. Altogether, the story seems unlikely, but to phrase it with the words of the famous computer scientist Richard J. Lipton: *Theory is like that sometimes*. The quote is taken from Lipton and Regan’s blog where they describe Szelepcsényi’s contribution [5] and explain its relation to the work of Lipton [6]. We will elaborate on this relation below.

## 2 Complexity Theory

Róbert Szelepcsényi solved a problem in a branch of computer science called complexity theory. The subject of complexity theory is computational problems formulated as a combination of input and computation task: Given an input from a well-defined domain, compute some specified information over this input. A computational problem that is immediately recognized as such is the task of determining the integral over a given curve. Often, however, the computation task is formulated as determining a yes-or-no answer. An example of such a *decision problem* is checking whether a given graph contains a Hamiltonian cycle. A graph is a set of elements, called vertices, that are connected via so-called edges, Fig. 1a. Think of the vertices as a number of cities and the edges as direct train connections between the cities. A path is a sequence of vertices, every neighboring pair of which are connected by an edge. A path is a cycle if the first and last vertex coincide. It is Hamiltonian if it contains every vertex of the graph exactly once (except the first vertex, which occurs twice). In the example,  $v_1.v_3.v_2.v_4.v_1$  is a path, even a cycle, and Hamiltonian. If the graph represents the rail network, a Hamiltonian cycle corresponds to an itinerary that is optimal in that it visits every city once.

Complexity theory aims at understanding the resources that are required to solve a computational problem. What does it mean to solve a computational problem of the form input/task? It means to give a step-by-step description of a computation process such that, for every input, by following the description the corresponding calculation will accomplish the task at hand. Such a step-by-step description is called an algorithm, and typically written in the form of a program a computer can execute. Indeed, computer scientists often consider programming, describing

**Fig. 1** A graph (a) and a depth-first search in this graph (b)



computation processes, the heart of their discipline. The most fundamental resources consumed by an algorithm are time and space. The time requirement of an algorithm is a function  $t(n)$  that, depending on the size  $n$  of the input, returns the maximal time a computation may take on an input of this size. Phrased differently, the function value is a guaranteed time limit for the computation. Technically, the time is measured as the number of basic operations that the computation executes on a representative processor. Assume the input consists of  $n$  numbers, then an algorithm requiring  $n^2$  additions will be able to finish relatively quickly, within seconds, even for many numbers, whereas an algorithm performing  $n^n$  additions will soon compute for weeks. The space requirement  $s(n)$  of an algorithm is the maximal amount of memory that is allocated during the computation on an input of size  $n$ .

For the problem of finding a Hamiltonian cycle, a possible algorithm proceeds as follows, Fig. 1b. Mark the first vertex in the list of vertices, here  $v_1$ . Give an ordering on the list of edges connected to each vertex. In the example, we assume the ordering is given by the indices, so  $v_2$  is ordered before  $v_3$ , which in turn is ordered before  $v_4$ . Then do a depth-first search on the graph from the marked vertex. By a depth-first search, we mean the algorithm first tries to extend the path from every vertex that is reached, and only later tries out alternatives. In the example, the algorithm extends the path from  $v_1$  by  $v_2$ , then by  $v_3$  because it is ordered before  $v_4$ , and then again by  $v_1$ . Whenever a vertex is visited that already belongs to the path, the algorithm checks whether it is the initial vertex. If so, it checks whether the path contains all vertices. If the vertex belongs to the path but is not initial or the path does not contain all vertices, the algorithm tries a different path from the preceding vertex. This is called backtracking. In the example, the initial vertex is found but  $v_4$  is missing on the path. The algorithm thus backtracks to  $v_3$ , which does not have alternatives to choose from. It then backtracks to  $v_2$  and continues the computation with  $v_4$ , as indicated by the dots. Also this computation will be unsuccessful and the algorithm will backtrack to  $v_1$ , from where it will try  $v_3$ . As the algorithm backtracks over and over again, the time requirement is  $n^n$ , where  $n$  is the number of vertices.

There are two kinds of results that complexity theorists strive for: Upper and lower bounds. Upper bounds show that a problem can be solved within given resource constraints. Establishing an upper bound thus means designing an algorithm, writing a program, that solves the problem and whose resource requirements meet the constraints. Lower bound results, in turn, show that a problem requires a certain amount of time or space. No algorithm working with less resources can solve the problem. There are two points of view to this. From the problem's perspective, a lower bound shows that the problem has some inherent complexity, like an unavoidable physical law of computation. From an algorithm's perspective, a lower bound gives information about the nature of efficient algorithms: Their behavior is such that it does not allow them to solve the problem. Proving lower bounds has turned out extremely complicated, and close to none are known. The famous  $P \neq NP$  conjecture can be cast as the belief that Hamiltonian cycles cannot be found within polynomial time.

The reader may object to the conjecture and argue that judging whether a given path is a Hamiltonian cycle can be done quickly. We check whether we visited all

vertices precisely once, say by marking a list of all vertices, and in the end verify that we arrived at the vertex we started from. Hence, all that would be needed to solve the Hamiltonian cycle problem efficiently is appropriate signs on the graph that guide the way. We would then follow the signs and check whether the resulting path is a Hamiltonian cycle. The reason why the community thinks the problem cannot be solved efficiently is that the assumption of guidance is a strong one. It means relaxing the postulate of determinism. The algorithm is allowed to make a choice whenever there are alternative ways to continue the computation. A different choice yields a different computation on the same input. The resulting *non-deterministic* algorithm would return yes if one of the computations returns yes, and no otherwise. To find a Hamiltonian cycle, a non-deterministic algorithm would guess the signs on the graph and test the corresponding path.

### 3 The Problem

When confronted with a decision problem, the task is to find an algorithm that returns yes if the input is a yes-instance, say a graph containing a Hamiltonian cycle, and no if the input is a no-instance, say a graph without such a cycle. Interestingly, one may also view this as the task of finding two algorithms. We need one algorithm that returns yes for the yes-instances. A second and independent algorithm should answer the no-instances. Given these two algorithms, we let them run in parallel and obtain an algorithm that solves the problem.

If the focus is on the no-instances, we talk about the *complement problem*. Consider the task of finding a path from a vertex  $s$  to a vertex  $t$  in a given graph. To be precise, from now on we drop the Hamiltonicity requirement but assume the edges to be directed (indicated by arrows like in Fig. 1b). We refer to this computational problem as PATH. The complement problem  $\overline{\text{PATH}}$  is to show that there is no path from  $s$  to  $t$  in a given directed graph. The thing to note is that the complement problem asks for a guarantee about all paths, namely that none of them connects  $s$  to  $t$ .

Complement problems seem to be difficult to solve with non-deterministic algorithms. The former ask about universal statements whereas the latter existentially guess guidance for the computation. A bad match! But is this really true? Or does non-determinism actually help to solve complement problems, detecting no-instances? If complexity theory aims at understanding the nature of computation, it has to give a definite answer.

Róbert Szelepcsényi answered the question in the case of space complexity, where it is formulated as follows. Can we construct from a non-deterministic algorithm with space requirement  $s(n)$  a non-deterministic algorithm that solves the complement problem and has the same space requirement? Intuitively, can we detect no-instances without using more space? Technically, consider a decision problem  $P$  for which there is a non-deterministic algorithm with space requirement  $s(n)$ . Is there a non-deterministic algorithm for the complement problem  $\overline{P}$  that also requires

$s(n)$  space? If we write  $\text{NSPACE}(s(n))$  for the class of all problems that can be solved with  $s(n)$  space, and  $\text{coNSPACE}(s(n))$  for the complement problems, the question is

$$\text{NSPACE}(s(n)) = \text{coNSPACE}(s(n))?$$

This equality must have been on the blackboard when Róbert Szelepcsényi entered the class. He took it to be his homework, and solved it.

## 4 Science Beats Borders

After more than 20 years of unsuccessful investigations, the solution to the problem came unexpectedly: The equality does hold! Every non-deterministic algorithm can be turned into a non-deterministic algorithm for the complement problem that does not require more space. As we will see in a moment, the solution is not even complicated. Why was the problem not solved earlier? There are two plausible answers to this. First, the community was searching in the wrong direction. The consensus belief was that the equality does not hold, and people were looking for ways to prove so. The second reason may be the demonization of the problem: If the classes were different, a lower bound would be needed, and that should be really hard to establish. This may have scared researchers away and they did not even start working on the problem.

Despite the apparent difficulty, a leading researcher tackled the problem: Neil Immerman, in the mid-1980s associate professor at Yale University, today among the leading figures in theoretical computer science and a key person in the development of a research branch called descriptive complexity theory. In 1987, Neil Immerman came up with his most famous result. Independently of Róbert Szelepcsényi, he proved the equality. Together, the two were awarded the Gödel prize in 1995, the highest award in theoretical computer science, for their publications [3, 11].

That Immerman and Szelepcsényi solved the same problem in the same year already seems unlikely. Even worse, their proofs rely on the same technique. Still, they have the best conceivable proof of the independence of their work. In 1987, the Cold War divided the world into East and West, and the two lived on competing sides with close to no possibility of communication. Phrased positively scientific insight is independent of political opinions and borders. A technique as natural as theirs seems to be guaranteed to be found. Again, quoting Lipton: *Theory is like that sometimes.*



## 5 The Solution

We elaborate on Immerman and Szelepcsényi's idea, called inductive counting. The first thing to note is that PATH can be solved with a non-deterministic algorithm requiring space  $s(n)$  roughly equal to the *memory needed to store a vertex*. The algorithm simply enumerates the vertices one by one, and stores the current vertex and a counter. The counter is used to stop the computation when the path has become too long and thus contains a cycle. Indeed, if  $n$  is the number of vertices in the given graph, then any path longer than  $n$  will contain a cycle. As for correctness, there is a path from the given vertex  $s$  to the given  $t$  if and only if there is an acyclic one. By non-determinism, the algorithm will find it.

The no-instances of PATH are those where there is no path from  $s$  to  $t$ . Checking this requires us to go through all potential paths between  $s$  and  $t$ . This can be done by a depth-first search like the one in Sect. 2. The resulting algorithm would need memory to store every vertex on the path, and hence require space  $n \cdot s(n)$ . Since our goal is to obtain a space usage of  $s(n)$ , detecting no-instances this way is too space consuming. The consequence is that we cannot store full paths. But the algorithm is also a bad idea conceptually. It is fully deterministic and does not take advantage of the power of non-determinism. How can we make use of non-determinism to reduce the space consumption when detecting no-instances of PATH?

The key idea of Immerman and Szelepcsényi is to assume that we have an additional input. Let  $N$  be the *number of vertices that are reachable* from  $s$  by a path. Surprisingly, having  $N$  is sufficient to show the absence of a path from  $s$  to  $t$  with a non-deterministic algorithm. The algorithm works as follows. It maintains a counter *count* keeping track of how many vertices have been found reachable so far, initially 0. The algorithm goes through all vertices of the graph in a predefined order, like in Sect. 2. For a vertex  $v$ , it makes a guess at whether or not the vertex is reachable from  $s$ . If the guess is unreachable, the computation proceeds to the next vertex in the ordering. If the guess is reachable, the algorithm tries to find a path using the above one-by-one enumeration that is stopped by a counter. This enumeration may guess any path of at most  $n$  vertices starting in  $s$ , leading to  $v$  or not. If the path does not reach  $v$ , the computation stops with output no. This should be interpreted as an unsuccessful branch in the non-deterministic computation. If the path reaches  $v$  and moreover  $v$  happens to be the vertex  $t$  of interest, the output is also no, meaning there actually is a path from  $s$  to  $t$  (recall that we consider the complement problem). If the path reaches  $v$  and  $v$  is not  $t$ , another reachable vertex has been found. In this case, *count* is increased by 1.

If the computation did not stop while going through the list of vertices, we cannot yet conclude that there is no path from  $s$  to  $t$ . We may have guessed incorrectly that a vertex is unreachable although it is reachable. If this happened for  $t$ , we would give an incorrect answer. To overcome the problem, the algorithm compares *count* to the initially given  $N$ . If  $count = N$ , all reachable vertices were guessed correctly. If  $t$  was not among them, the algorithm returns yes,  $t$  is not reachable from  $s$ . Otherwise, the algorithm returns no, indicating a mistake in the non-deterministic choices.

It remains to check the space requirement. The algorithm needs memory for the counters *count* and  $N$ , for the counter in the path exploration, and for two vertices, the one that is currently being explored, and the one that is guessed to be on a path. Altogether, this sums up to roughly  $s(n)$  space.

We still have to compute  $N$ . Interestingly, the above algorithm can also be used for this task. Assume we have computed  $N_i$ , the number of vertices that are reachable in at most  $i$  steps. Using this number and the above algorithm, we can compute  $N_{i+1}$  non-deterministically. Then  $N$  is  $N_n$ , where  $n$  is the number of vertices. The method is called *inductive counting* as the numbers  $N_i$  are computed inductively.

## 6 Lipton

Like structural analysis checks the stability of buildings, a branch of computer science called verification checks the correctness of computer programs. For a braking system in a car, verification would compute the response time. For an online-banking application, verification would ensure that a secure connection is established. Like structural analysis, verification is conducted on a model of the overall system. This model is often a Petri net; see, e.g., [2, 8–10]. Petri nets reflect well the interaction among the system components, and today close to every program either interacts with an environment, like the braking system, or is concurrent by nature, like the banking application.

Concerning safety, verification amounts to showing that the system cannot reach an unsafe state. A defect in the braking system, for example, would lead to a state where the pedal has been pushed, a certain amount of time has elapsed, and the brakes have not been applied. For the banking application, an unsafe state would contain a seemingly secure connection between a client and the bank to which a third person has access. Formulated as a decision problem for Petri nets, the reachability problem is as follows. Given a Petri net, an initial and a final marking, the task is to check whether the final marking can be obtained from the initial one by executing a sequence of transitions.

A person who helped to understand the reachability problem is Richard J. Lipton. He is one of the founders of complexity theory and came up with results so fundamental that they are now standard in computer science education. In 1976, Lipton was a researcher at Yale University and proved a result that would turn out to be important: The first lower bound on the reachability problem for Petri nets. He did not publish his proof; it only appeared as a technical report [4]. Lipton must have thought the result could be improved and would not persist. He was wrong: Even today, his lower bound is the best known.<sup>1</sup> This is even more remarkable since

---

<sup>1</sup>After writing this chapter, a new lower bound for the reachability problem was found [1]. This is the first improvement of Lipton's result after more than 40 years.

in 1976 no upper bound for the problem was known. Hence, it was not clear where to start from when searching for a lower bound.

The upper bound remained open until 1981, when Ernst W. Mayr gave the first algorithm to solve the reachability problem for Petri nets [7]. Also Mayr's result has not been improved until today. That researchers unsuccessfully tried to improve the lower and the upper bound in particular means the two do not match. There is a gap, a canyon, between them. The difference is astronomically large! Lipton showed that every algorithm solving the reachability problem requires at least exponential space, in numbers  $2^{\sqrt{n}}$ . Mayr's algorithm may consume a non-primitive-recursive amount of memory, in numbers

$$2^{2^{\cdot^{\cdot^{2^n}}}}$$

and the height of the tower of exponentials depends on the size of the input. Closing the gap between upper and lower bound is among the most important problems in theoretical computer science.

We elaborate on the idea behind Lipton's proof. He showed that Petri nets can simulate *bounded* programs. These programs only have integer variables, which they manipulate by increment and decrement, and which they test for holding value zero. Bounded means that the variable values never exceed  $2^{2^n}$ . For this class of programs, an exponential-space lower bound was known, and with Lipton's construction it carries over to Petri nets. In his construction, each variable is translated into a place. The value of the variable is represented by the number of tokens in that place. The hard part is to simulate a zero-test: Does place  $p$  hold 0 tokens? Petri nets cannot model this check right away. To simulate it, Lipton introduced *complement places*: For a place  $p$  holding  $x$  tokens, the complement place  $\bar{p}$  is guaranteed to contain exactly  $2^{2^n} - x$  tokens. Testing whether  $p$  holds 0 tokens then amounts to checking  $\bar{p}$  for  $2^{2^n}$  tokens.

The main difficulty is to generate  $2^{2^n}$  tokens. Lipton used an inductive argument. Assume a Petri net  $\text{Inc}_i$  generating  $2^{2^i}$  tokens has already been constructed. Lipton showed how to construct, out of  $\text{Inc}_i$ , the Petri net  $\text{Inc}_{i+1}$  generating  $2^{2^{i+1}}$  tokens. The idea is to use the equation  $2^{2^i} \cdot 2^{2^i} = 2^{2^{i+1}}$ , and repeatedly execute  $\text{Inc}_i$ , namely  $2^{2^i}$  times.

Taking a step back, Lipton's idea is inductive counting in an exponential manner [6]. He showed how to generate  $2^{2^{i+1}}$  tokens provided the smaller number of  $2^{2^i}$  tokens has already been produced. Lipton may have been the first to propose inductive counting—11 years before Immerman and Szelepcsényi did. But *Theory is like that sometimes*.

## References

1. W. Czerwinski, S. Lasota, R. Lazic, J. Leroux, F. Mazowiecki, The reachability problem for Petri nets is not elementary (extended abstract). CoRR, abs/1809.07115. arXiv:1809.07115 (2018)
2. J. Esparza, K. Heljanko. *Unfoldings - A Partial-Order Approach to Model Checking* (Springer, Berlin, 2008)
3. N. Immerman, Nondeterministic space is closed under complementation. *SIAM J. Comput.* **17**(5), 935–938 (1988)
4. R.J. Lipton, The reachability problem requires exponential space. Technical Report 62, Yale University, 1976
5. R.J. Lipton, K.W. Regan, We all guessed wrong. Gödel’s lost letter and  $P = NP$  (2009). <https://rjlipton.wordpress.com/2009/02/19/we-all-guessed-wrong/>
6. R.J. Lipton, K.W. Regan, An EXPSPACE lower bound Gödel’s lost letter and  $P = NP$  (2009). <https://rjlipton.wordpress.com/2009/04/08/an-expspace-lower-bound/>
7. E.W. Mayr, An algorithm for the general Petri net reachability problem. in *STOC* (ACM, New York, 1981), pp. 238–246
8. R. Meyer, T. Strazny, Petruccio: from dynamic networks to nets, in *CAV*, ed. by T. Touili, B. Cook, P. Jackson. Lecture Notes in Computer Science, vol. 6174 (Springer, Berlin, 2010), pp. 175–179
9. W. Reisig, *Petri Nets: An Introduction* (Springer, Berlin, 1985)
10. P.H. Starke, *Analyse von Petri-Netz-Modellen* (Teubner, Leipzig, 1990)
11. R. Szelepcsényi, The method of forced enumeration for nondeterministic automata. *Acta Inform.* **26**(3), 279–284 (1988)

**Part IV**  
**Connecting to Other Areas**

# On Petri Nets in Performance and Reliability Evaluation of Discrete Event Dynamic Systems



Gianfranco Balbo and Gianfranco Ciardo

## 1 Preface

We began using Petri Nets for Performance Evaluation of Discrete Event Dynamic Systems in the early 1980s, after learning of Mike Molloy's work at UCLA and finding that his Stochastic Petri Net extension was a valuable tool for describing and analyzing complex computer systems behavior. At that time, Gianfranco Balbo was working with Marco Ajmone Marsan and Gianni Conte on the analysis of the prototype architecture for a multiprocessor system being developed at Politecnico di Torino, and Gianfranco Ciardo was a student in the Informatics Department of the University of Torino, developing a software tool for evaluating models of this prototype as part of his thesis.

Importing the network theory developed by Carl Adam Petri to the area of the Performance Evaluation of Computer Systems was an important step in the development of this field, since modeling concurrency, synchronization, cooperation, and competition in Discrete Event Dynamic Systems became straightforward exploiting the binary relations of concurrency and causality of Petri's theory. Adding time to Petri Nets by assuming a stochastic duration of transition firings is compatible with the essence of Petri's theory, as long as the firing-time distributions have infinite support and are memoryless, since this only changes the nondeterminism of standard Petri Nets into a probabilistic choice among the transitions enabled in a marking. The negative exponential distributions in Molloy's Stochastic Petri Nets satisfy these requirements, so the timed model is still based on concurrency and causality

---

G. Balbo (✉)

Dipartimento di Informatica, Università di Torino, Torino, Italy

e-mail: [balbo@di.unito.it](mailto:balbo@di.unito.it)

G. Ciardo

Computer Science Department, Iowa State University, Ames, IA, USA

e-mail: [ciardo@iastate.edu](mailto:ciardo@iastate.edu)

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,

[https://doi.org/10.1007/978-3-319-96154-5\\_22](https://doi.org/10.1007/978-3-319-96154-5_22)

relations. This feature of Petri's net theory remained a guiding reference we tried to respect in further Petri Net extensions we used in our career, mainly to keep the models of complex systems as simple as possible and to develop computationally efficient analysis and solution methods.

Gianfranco Balbo met Petri for the first time in 1999, when Petri was awarded a Doctorate *Honoris Causa* by the University of Zaragoza. In a private conversation, Petri graciously stated that he understood our desire to use continuous times in our models and that, indeed, the Markovian assumption of the Stochastic Petri Nets we were using did not destroy the qualitative properties of the net, even though he still believed that there was no need to represent time in this manner. But what turned out to be extremely interesting during this event was the "Lectio Magistralis" he delivered after receiving his Doctorate, in which he recalled that the graphical representation of Petri Nets was not part of his Ph.D. thesis, but was instead originally proposed years before, when developing "his second network" for the representation of chemical reactions. Seven years later we started using Petri Nets in Systems Biology to study biological pathways usually defined as sets of biochemical reactions, and it was reassuring to know that they had indeed originally been developed precisely for that purpose.

## 2 Introduction

Discrete event dynamic systems (DEDSs) [12] are ubiquitous and we often analyze them to assess their logical (reachability, model checking) or timing (scheduling, performance, and reliability) properties. To study a DEDS, we first *model its behavior* (with an appropriate formalism), then we *describe the measures of interest* (either in a formal language or by choosing from a menu of predefined choices), and finally we *solve it* (by using a tool implementing algorithms that can compute these measures).

For performance analysis, *queuing networks*, where *customers* move between *stations* to request *service*, gained acceptance in the 1970s [22, 26], Predefined *scheduling policies* (defining the order in which customers are served) and *routing policies* (defining where a customer goes after completing service) were sufficient to model the nascent timesharing computer systems, where the most important goal was dimensioning (e.g., deciding how many CPUs and disks suffice to obtain an acceptable *response time* while maintaining a high *utilization*—the typical predefined measures for such systems). For reliability or availability analysis, *fault trees* and *reliability block diagrams* were used to express the conditions under which a system would fail, or be operational [17]. Each component has a failure time distribution, and the model connectivity specifies how to combine these failure probabilities to obtain the overall reliability over time.

However, as systems and analysis needs grew, the specification and solution ease offered by specialized formalisms and algorithms could not offset their inability to model common situations [8, 13]. For example, classic queueing networks

cannot model non-memoryless routing or fork-and-join behavior, while failures affecting multiple components or shared repair facilities in a fault tree require additional specifications and, worse yet, violate the independence needed for a simple combinatorial analysis.

When activities have independent exponentially distributed random durations, these DEDS can be described by continuous-time Markov chains (CTMCs), whose behavior can in principle be analyzed numerically. One might manually list the CTMC state space and transition rates, but this is not practically feasible, as they can be huge. A general-purpose high-level formalism was needed to describe these, and future, systems. In the 1980s, stochastic extensions to Petri Nets (SPNs) were introduced as a viable solution, and they are still an excellent choice decades later [19, 31, 39].

### 3 Modeling Discrete Event Dynamic Systems

A system is often defined as a collection of *objects* and *relations* among them. Objects have *attributes*, which describe their *local* states, and the collection of the states for all objects constitutes a (*global*) system *state*. Relations describe the possible state changes, or *transitions*. In particular, Discrete Event Dynamic Systems (DEDSs) have a discrete, i.e., finite or countably infinite, set  $S$  of states, or *state space*, and their evolution is not directly due to the passage of time, but to the occurrence of *events*.

DEDSs arise in diverse application domains such as flexible manufacturing, computing, telecommunications, traffic, chemistry, and biology. Regardless of the meaning of different components, understanding the behavior of DEDSs is challenging because of the intrinsic complexity ( $S$  is usually huge) and the many and sometimes subtle interactions among components, often leading to unforeseen and counterintuitive behaviors.

Reasoning about such systems, understanding and learning their operation, improving their performance, and making decisions about their design and operation requires us to properly describe the system, so that the relations among its components become clear. This is best done by building a model of the actual system to capture the important features of its structure and to provide ways of quantifying its properties, and requires choosing an appropriate level of abstraction, deciding which aspects to include in the model and which details to neglect, so that the representation is sufficiently faithful while still amenable to analysis. A model is thus a simplified representation of reality, and the effort required to fit a complex situation within the constraints of a formalism is rewarded with a better understanding of the system, which helps remove incompleteness and contradictions, identify properties, and discover possible improvements [7].

The mathematical nature of a formal model allows us to rigorously reason about the behavior of the system. It is also a prerequisite for its automated analysis, provided the appropriate algorithms are implemented in a software tool. Furthermore,



models are usually parametric; this not only highlights the key numerical inputs affecting the system size and behavior, but also allows one to easily ask “what-if” questions and perform sensitivity analysis by simply varying the values of these parameters.

## 4 Analyzing Discrete Event Dynamic Systems

While most DEDSs are studied to optimize their operation, real systems are fault prone, thus their models must also capture failure and repair aspects to examine how the system performs under less-than-perfect conditions. Performance and dependability (or *performability* [30]) evaluation uses mathematical models to compute performance indices such as resource utilization, system productivity, and system response time, while accounting for system failures. Time plays a crucial role in the analysis of such systems, since the occurrence of certain desired (or undesired) events within a certain finite time horizon, or the efficiency of the system in the long run, are key topics to be addressed.

To capture these aspects, the modeling formalism must be able to specify the timing of events (usually in terms of the distribution of the time to complete a given action) and the probability of different outcomes when an event occurs. The resulting probabilistic model describes a *stochastic process*, so that performance and reliability indices are defined in terms of the probability of finding the DEDS in each of its possible states.

A stochastic process  $\{X(t), t \in T\}$  is a family of random variables  $X(t)$  taking values over the state space  $S$  and indexed by time parameter  $t$ , where  $T$  is the natural numbers  $\mathbb{N}$ , for discrete-time models, or the non-negative reals  $\mathbb{R}^{\geq 0}$ , for continuous-time models. Special classes of stochastic processes are of particular interest because their probabilistic characterization and their analysis are simpler. In a Markov process, for example, the future behavior depends only on the present state; there is no memory of the trajectory that led to the present state. Markov chains are Markov processes with discrete state spaces; they can be discrete-time (DTMCs) or continuous-time (CTMCs).

The formalism of Queueing Networks (QNs) is particularly suited to modeling DEDSs when the focus is on competition for the use of shared resources and the congestion arising in the system. Under commonly adopted distributional assumptions, the process underlying a QN is a CTMC, but the greatest appeal of this formalism is the existence of *product form solutions* for a class of QNs. The BCMP theorem [6] provides sufficient conditions for a model to be a Product Form Queueing Network (PFQN). Computationally efficient algorithms for PFQN analysis do not require to generate the state space  $S$ , whose size is  $O(N^M)$  where  $N$  is the number of queues and  $M$  is the number of customers; their time and memory complexity is instead polynomial in  $N$  and  $M$ .

Many DEDSs exhibiting queueing behavior, however, cannot be modeled by traditional QNs due to synchronizations, blocking, splitting and fusion of customers,

or dependencies across queues. Proposed QN extensions seek to keep the model description concise, but claim no completeness. The lack of descriptive power becomes apparent, even with these extensions, when trying to capture the increasingly common behaviors just listed; the gap between the DEDS and the model is sometimes bridged with natural language specifications, and ad hoc solution methods must be devised for each individual case. Unless the solution relies on numerical analysis of the underlying CTMC, the results obtained may even just be (hopefully good) approximations.

In the reliability field, *fault trees* (FTs) and *reliability block diagrams* (RBDs, the dual of FTs, not discussed further) are two commonly used formalisms. The leaves of an FT correspond to atomic components with a known failure probability, while internal nodes are boolean gates describing how the failure of a subsystem depends on that of its children, up to the root node, describing the overall system failure. For example, if the failure of any one of  $A$ ,  $B$ , and  $C$  composing a subsystem  $X$  causes its failure, then the three leaves  $A$ ,  $B$ , and  $C$  are the children of an OR gate  $X$ , while, if a second subsystem  $Y$  is functionally equivalent to  $X$  so that the overall system works as long as either one does,  $X$  and  $Y$  are the children of an AND root node. Assuming independence in the failed/repaired status of each component, efficient combinatorial algorithms can compute the overall reliability. In the simplest case, we specify the probability of any component being *up* or *down* at any one time, and we seek the long-term availability (probability of the system working). Another amenable case arises if there are no repairs, components have exponentially distributed failure times, and we seek the reliability at a finite time  $t$  (without repairs, failure is certain in the long run).

However, just as for PFQNs, both FT specifications and their solution algorithms become inadequate as soon as we attempt to include certain realistic aspects. For example, failure of a component may stress other components and increase their fault rate; or the repair facility may have finite capacity, so that only a few components can be repaired concurrently (this gives rise to queueing, since failed components not being repaired must wait, with a complex service policy, since failed components critical to the overall system reliability should have repair priority). Also in these models, the solution approach relies on the generation and solution of the underlying CTMC.

## 5 Stochastic Petri Nets

If we are willing to pay the price of generating and solving the underlying CTMC, there is little reason to restrict ourselves to formalisms inherently limited in description power and requiring ad hoc algorithms to cope with situations not satisfying the requirements for efficient solutions. A natural alternative then is to adopt a general formalism, such as Petri Nets (PNs) [1, 35–37], which are ideally suited to modeling concurrency and synchronization. PNs are more powerful than finite-state machines, thus can describe any finite state space  $S$  and any graph

specifying arbitrary transitions between states. More importantly, they are usually concise and rely on a minimal set of primitives and dynamic rules. PNs can even model many DEDSs with infinite state spaces, such as those modeled by “open” QNs. Indeed, PNs with inhibitor arcs or transition priorities become Turing complete, thus able to model the structure of any DEDS, but at the cost of decision power (mostly a theoretical problem, since we use these extensions merely to ease the modeling of a finite DEDS, not to attain Turing completeness).

PNs are not concerned with actual event durations, but several ways to add timing have been proposed [3, 29, 33, 34], *Stochastic PNs* (SPNs) [19, 31, 39] are arguably the most successful extension, and the most appropriate for performance and reliability analysis, where the time between the enabling of a transition and its firing is a random variable. If these *firing times* are exponentially distributed, the SPN defines a CTMC.

### 5.1 Simple Primitives, Precise Semantics, Powerful Modeling Capabilities

PNs pair a simple formalism with a precise semantics: their static structure and dynamic behavior can be described in a paragraph. A PN is a tuple  $(P, T, I, O, \mathbf{i}_{init})$  where  $P$  and  $T$  are disjoint sets of *places* and *transitions*,  $I, O: P \times T \rightarrow \mathbb{N}$  describe the *cardinality* of the *input arcs* from places to transitions and of the *output arcs* from transitions to places, and  $\mathbf{i}_{init}: P \rightarrow \mathbb{N}$  is the initial *marking* (assignment of *tokens* to places). A transition  $t \in T$  is enabled in a marking  $\mathbf{i}$  if  $I(p, t) \leq \mathbf{i}(p)$ , for each  $p \in P$ . An enabled transition  $t$  in a marking  $\mathbf{i}$  can fire, leading to a new marking  $\mathbf{j}$  satisfying  $\mathbf{j}(p) = \mathbf{i}(p) - I(p, t) + O(p, t)$ , for each  $p \in P$ . Researchers from different backgrounds can easily grasp this formalism and use it to model the structure of a DEDS.

The additional specifications to define an SPN may also be simple, or quite complex, depending on the particular type of SPN. If all firing times are described by random variables with support  $[0, \infty)$  and we assume a *race policy* with *atomic firing* and *preemptive-repeat-different* policy [5] (firing times are resampled once the transition fires or becomes disabled), the state space  $S$  of the (untimed) PN coincides with that of the SPN, and the results and algorithms for logical PN analysis apply to the SPN. Then, if all firing times are exponentially distributed, we simply specify the firing rate  $\lambda(t)$  of each transition  $t$ , and the *race semantics* implies that transition  $t$  enabled in marking  $\mathbf{i}$  fires next with probability  $\lambda(t) / \sigma(\mathbf{i})$ , where  $\sigma(\mathbf{i})$  is the sum of the firing rates of the enabled transitions in  $\mathbf{i}$ , thus the rate of the exponentially distributed sojourn time in  $\mathbf{i}$ .

The specification and the solution algorithms become instead more complex if we have non-memoryless distributions and policies such as *preemptive-resume* (if a transition becomes disabled, its remaining firing time continues when it becomes enabled again) or *preemptive-repeat-identical* (if a transition becomes disabled, its

previous entire firing time restarts once it becomes enabled again) [5]. Additional information is needed if the SPN has discrete-time distributions, since multiple conflicting transitions may attempt to fire at the same time; such situations require us to specify priorities or probabilities over the transitions to avoid nondeterminism. Even with these additions, though, SPNs retain the advantage of a general-purpose formalism with a clear semantics, allowing automated analysis using software tools, many of which exist today.

## 5.2 *Correctness of the Untimed Model*

Another advantage of SPNs is that the analysis of the corresponding untimed PN can yield important insights into the structural or logical aspects of the DEDS behavior, such as the presence of deadlocks, the reachability of certain conditions, the liveness of portions of the system, or the boundedness of the state space. This is particularly true when the state spaces of the SPN and of the PN coincide, which is true when all firing times have infinite support, as already mentioned. However, even when the behavior of the SPN is affected by timing specifications, its state space is guaranteed to be a subset of that of the PN; thus analysis of the PN can provide valuable information. For example, if the PN has no deadlock, then neither does the SPN; on the other hand, a deadlock in the PN *might* also correspond to a deadlock in the SPN, so it might be worth exploring whether this is the case or whether the SPN timing makes this deadlock impossible.

This ability to perform logical verification is new in performance and reliability, because the formalisms traditionally used for these fields were limited, while SPNs can more accurately describe the behavior of complex systems, which may have logical design flaws worth discovering before we undertake a timing and probabilistic analysis.

## 5.3 *SPN Extensions*

After the introduction of SPNs with exponentially distributed timing and an underlying CTMC, many SPN variants and extensions have been proposed in the literature (still respecting the key idea that places can be interpreted as resources, transitions as activities) with the aim of simplifying the construction of models for complex systems without completely losing the connection between the timed and the untimed models.

One alternative is *discrete time* SPNs, with geometrically distributed firing times [32] or more generally firing times with a “defective discrete phase-type distribution” (any distribution corresponding to the absorption time for a finite DTMC) [15]. The underlying process is a DTMC, but the SPN description is complicated by the need to arbitrate simultaneous firing of conflicting transitions,

so this is best used for DEDSs whose behavior itself exhibits such timing conflicts, e.g., clocked systems.

A widely employed extension of SPNs, *generalized SPNs* (GSPNs) [2], adds instead *immediate* transitions with zero firing times. Immediate transitions are often devoted to the representation of logical actions that do not consume time, but they can also model situations where branching probabilities are independent of timing specifications (the effect of an exponentially distributed timed transition is otherwise deterministic). When timed and immediate transitions are simultaneously enabled in the same marking, immediate transitions are assumed to fire first. This implies that, when timed specifications are disregarded, the underlying untimed models of GSPNs are PNs extended with a priority structure that can be used for a preliminary qualitative analysis. Timescale differences captured by timed vs. immediate transitions are related to the concepts of visible and invisible transitions in classical PN theory. When time is taken into consideration instead, the exponential distributions of the firing times of timed transitions insures that the underlying stochastic process of a GSPN remains a CTMC, as long as pathological situations such as *vanishing loops* are avoided.

A further extension allows SPN firing times to have any continuous phase-type distribution [16], possibly in addition to immediate transitions. Just as for the discrete phase-type case, the benefit of using these more complex distributions is a more faithful representation of the actual duration of DEDS activities, and the price is a larger underlying state space, since the state must encode both the SPN marking and the phase of each enabled SPN transition. One important advantage of using continuous phase-type firing times, though, is that the logical behavior of the untimed PN still reflects that of the SPN, while the same does not hold for discrete phase-type firing times.

The SPN of a complex system may be cluttered due to the need to capture system peculiarities, thus the need for more powerful constructs has been felt since the early use of SPNs in the analysis of multiprocessor architectures. Timed extensions of high-level Petri Nets (colored PNs [24]) have been proposed, to allow researchers already familiar with the basic formalism to address more difficult problems with the little additional effort needed to use these high-level extensions. For example, DEDSs may contain internal symmetries, which can be exploited to achieve a more compact specification, and a class of Colored SPNs called Stochastic Symmetric Nets (SSN) (previously known as *Stochastic Well-formed Nets*) has been proposed, to represent the system in a way that naturally leads to a *lumped* model [14]. Usually, this compact representation is exploited only when specifying the model [28], but the compact colored net is “unfolded” into a much larger (uncolored) SPN before it is analyzed.

## 5.4 Analysis Algorithms

The **numerical solution** of a generic CTMC underlying the SPN (or any other formalism for that matter) modeling a DEDS requires the following steps:

- Generate the state space  $\mathcal{S}$ , which must be finite but can be huge in practice.
- Build the infinitesimal generator  $\mathbf{Q} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{S}|}$  by summing in  $\mathbf{Q}[\mathbf{i}, \mathbf{j}]$  the rate of events whose occurrence in state  $\mathbf{i}$  leads to  $\mathbf{j}$ , then letting  $\mathbf{Q}[\mathbf{i}, \mathbf{i}] = -\sum_{\mathbf{j} \neq \mathbf{i}} \mathbf{Q}[\mathbf{i}, \mathbf{j}]$ .
- Compute the desired solution, either the transient probability vector  $\pi_\tau$  satisfying  $d\pi_\tau/d\tau = \pi_\tau \cdot \mathbf{Q}$  with initial condition given by the initial state, or the stationary probability vector  $\pi$  solution of  $\pi \cdot \mathbf{Q} = 0$ , well defined if the CTMC is ergodic, i.e., every state can reach every other state, as is often the case for Performance Evaluation models.

The memory requirements for a traditional sparse encoding of  $\mathbf{Q}$  are proportional to the number  $\eta(\mathbf{Q})$  of its nonzero entries, usually a reasonably small multiple of  $|\mathcal{S}|$ . In case of CTMCs with very large state space, the use of decision diagrams [42] may allow the model to be stored in main memory. However, the time to compute  $\pi_\tau$  or  $\pi$  numerically depends on the nature of  $\mathbf{Q}$ ; *stiff* matrices may require many iterations, each with cost  $\eta(\mathbf{Q})$ . Once  $\pi_\tau$  or  $\pi$  has been computed, most measures of interest can be expressed as simple weighted sums. For example, if a place  $p$  models jobs waiting to be processed, their expected number in steady state is computed as  $\sum_{\mathbf{i} \in \mathcal{S}} \pi[\mathbf{i}] \cdot \mathbf{i}(p)$ ; the probability of having a specific number of jobs waiting can be similarly computed, just as easily.

If the state space  $\mathcal{S}$  is infinite, or even just too large to fit in memory, or if the firing-time distributions are such that the underlying process is not a CTMC and is not amenable to a numerical solution, the SPN can still be analyzed using **discrete event simulation** [18, 20]. This technique is completely general and has been used to study DEDSs in many application areas, as in principle it is always applicable: any formalism defining how each event modifies the state and how its duration can be randomly sampled can use a simulation “engine.” The reality, however, is that this generality makes it even more important to have a formal model semantic (here, again, SPNs have a clear advantage), and the results from a simulation study need to be carefully validated, as it is possible that the model has flaws that may be hard to discover by looking at just a handful of numerical outputs (here, too, PN analysis tools can help find logical errors in the model). In any case, because of its statistical-sampling nature, simulation can at best provide *confidence intervals* for the measures of interest (e.g., “with 95% confidence the response time is between 3.2 and 3.7 seconds”). However, if the process being simulated is *Markov regenerative*, i.e., we can define instants at which the evolution of the process statistically depends only on the current state, the simulation results have more desirable statistical properties, since we are now essentially able to sample multiple truly independent runs. This holds for the *regenerative SPNs* of Haas and Shedler [23].

Somewhere between SPNs with an underlying CTMC and the full generality of SPNs requiring simulation, there are classes of non-Markovian SPNs that admit a numerical solution based on **embedding**. These are extensions of the original SPNs, but we discuss them here because the reason for their definition was the availability of new numerical solutions. The *deterministic and stochastic PNs* (DSPNs) [4] were the first such class to be proposed. Firing times can be deterministic or exponentially distributed, but at most one deterministic transition can be enabled at any time, ensuring that the SPNS has an underlying Markov renewal process whose kernel can be computed numerically. *Phase-delay PNs* [25] instead have firing times with discrete or continuous phase-type distribution; conditions about when both types can be enabled exist but, if these are not satisfied, at least regenerative simulation can be employed. *Markov regenerative SPNs* [21] allow generally distributed instead of deterministic firing times, but are still affected by the number of enabled non-exponential transitions: if there is only one, the solution generalizes the one for DSPNs; otherwise, the method of *supplementary variables* is used, but complexity and numerical stability can become problematic.

Finally, when the SPN models large groups of elements with similar behavior (e.g., internet users, human populations, or reagent molecules), exploiting these symmetries for lumpability may be insufficient, and performance indices may have to be estimated with discrete event simulation instead of computed with numerical methods. Alternatively, we may approximate the stochastic model with a deterministic one where the time evolution of quantities of interest is captured by a system of ordinary differential equations (ODEs) whose solution provides the expected values of these quantities as a function of time [9, 11, 38]. The convergence of the solution to the correct expected values when the size of the involved populations grows has been studied by many researchers. Starting from the work of Kurtz [27], it has been shown that the accuracy of the approximation is acceptable when the model can be considered as the representation of a system of interacting population quantities [40, 41]. The approach is based on the idea of replacing the integer vector representation of the markings of the SPN model with a vector of real values, thus **fluidifying** the tokens of the SPN.

## 6 Conclusions

After some initial reluctance to accept this new formalism, SPNs are now widely used for performance and reliability evaluation of many practical systems by researchers with considerably different backgrounds. Key features that made SPNs popular are:

**Graphical representation.** SPNs have a clear, precise, and compact semantics, but also a graphical representation, which helps us to discuss and communicate models in an intuitive manner.

**Generality.** SPNs can model DEDSs irrespective of the application field, and are not limited to choosing from sets of predefined building blocks as required by more specialized formalisms.

**Fidelity.** The automatic construction of the underlying stochastic model relieves the analyst from the burden of specifying it by hand, ensuring that it is truly equivalent to the specification of the system, and avoiding errors due to misinterpreting or oversimplifying system details.

**Structural analysis.** The analysis of the PN obtained from the SPN by ignoring timing and probabilistic information can provide valuable insight into the structure of the DEDS by revealing the existence of unexpected behaviors such as deadlocks, livelocks, or unboundedness.

**Analysis tools.** Performance and reliability experts can rely on many tools that have been developed to model and analyze SPNs. These tools employ state-of-the-art techniques to tackle SPNs with ever-larger underlying CTMCs, or resort to simulation when the underlying stochastic process is not amenable to a numerical solution.

Despite their success, much work is still needed to make SPNs the formalism of choice to model and analyze current DEDSs. One obstacle is the difficulty of building large models comprising several sub-models, which call for better divide-and-conquer approaches, even if *compositionality* is not an inherent feature of PNs.

A much more fundamental need, however, is the ability to solve models with extremely large state spaces, where, again, compositionality should be explored further. Once we build a complex model from small building blocks, this structure should be exploitable for the solution. Hierarchical techniques for specific problems have been proposed [10], but a general methodology that can be safely and automatically applied to SPNs to speed up their solution, be it numerical or simulative, is needed.

Decision diagrams have been shown to greatly help the generation and storage of the state space and infinitesimal generator of Markovian SPNs, but not their numerical solution. Approximations exploiting the decision diagram structure have been introduced [42], but more desirable would be algorithms to *bound* the measures of interest.

Another way of tackling the solution of large SPNs is the already mentioned *fluidification*, where the discrete SPN marking is replaced by a continuous approximation. This is justifiable when the number of tokens in the model becomes very large or tends to infinity [38] but much more difficult are intermediate situations where the population is large, yet discrete system features cannot be completely neglected.

In conclusion, we can say that SPNs are another example of the impact that the original work of Carl Adam Petri had (and is still having) on many application and research fields, but we must also observe that research on performance and reliability evaluation of DEDSs must still refer to the ideas of Carl Adam Petri to address the challenges coming from application domains with ever increasing complexity.



## References

1. T. Agerwala, Putting Petri nets to work. *IEEE Computer* **12**(12), 85–94 (1979)
2. M. Ajmone Marsan, G. Conte, G. Balbo, A class of Generalised Stochastic Petri Nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comp. Syst.* **2**(2), 93–122 (1984)
3. M. Ajmone Marsan, G. Balbo, K. Trivedi (eds.), *Proceedings of the International Workshop on Timed Petri Nets*, Torino, July 1985 (IEEE-CS Press, Los Alamitos, 1985)
4. M. Ajmone Marsan, G. Chiola, A. Fumagalli, Improving the efficiency of the analysis of DSPN models, in *Proc. 9<sup>th</sup> Enrop. Workshop Appl. and Theory of Petri Nets*, ed. by G. Rozenberg. LNCS, vol. 424 (Springer, Berlin, 1988), pp. 30–50
5. M. Ajmone Marsan, G. Balbo, A. Bobbio, G. Chiola, G. Conte, A. Cumani, The effect of execution policies on the semantics and analysis of stochastic Petri nets. *IEEE Trans. Softw. Eng.* **15**(7), 832–846 (1989)
6. F. Baskett, K. Chandy, R. Muntz, F. Palacios, Open, closed, and mixed networks of queues with different classes of customers. *J. ACM* **22**(2), 248–260 (1975)
7. G. Balbo, M. Silva (eds.), *Performance Models for Discrete Event Systems with Synchronizations: Formalisms and Analysis Techniques II*, vol. 2 (Kronos, Zaragoza, 1998)
8. J. Bechta Dugan, S. Bavuso, M. Boyd, Fault trees and Markov models for reliability analysis of fault-tolerant systems. *Reliab. Eng. Syst. Safe.* **39**(3), 291–307 (1993)
9. A. Bobbio, M. Gribaudo, M. Telek, Analysis of large scale interacting systems by mean field method, in *Proceedings of the QBST'08* (IEEE CS Press, Los Alamitos, 2008), pp. 215–224
10. G. Bolch, S. Greiner, H. Meer, K. Trivedi, *Queueing Networks and Markov Chains* (Wiley, New York, 1998)
11. J. Bradley, R. Hayden, W. Knottenbelt, T. Suto, Extracting response times from fluid analysis of performance models. in *Proceedings of the SIPEW'08*, ed. by S. Kounev, I. Gerton, K. Sachs. LNCS, vol. 5119 (Springer, Berlin, 2008), pp. 29–43
12. C.G. Cassandras, S. Lafortune, *Introduction to Discrete Event Systems* (Springer, New York, 2008)
13. K.M. Chandy, C.H. Sauer, Approximate methods for analyzing queueing network models of computer systems. *ACM Comput. Surv.* **10**(3), 281–317 (1978)
14. G. Chiola, C. Dutheillet, G. Franceschinis, S. Haddad, Stochastic well-formed coloured nets for symmetric modelling applications. *IEEE Trans. Comput.* **42**(11), 1343–1360 (1993)
15. G. Ciardo, Discrete-time Markovian stochastic Petri nets, in *Computations with Markov Chains*, ed. by W.J. Stewart. (Kluwer, Dordrecht, 1995), pp. 339–358
16. A. Cumani, ESP - A package for the evaluation of stochastic Petri nets with phase-type distributed transition times, in *Proc. Intern. Work. Timed Petri Nets* (IEEE-CS Press, Los Alamitos, 1985), pp. 144–151
17. C. Ericson, Fault tree analysis - a history, in *Proceedings of the 17th International System Safety Conference* (1999)
18. G. Fishman. *Principles of Discrete Event Simulation* (Wiley, New York, 1978)
19. G. Florin, S. Natkin, Les reseaux de Petri stochastiques. *Technique et Science Informatiques* **4**(1), 143–160 (1985)
20. R. Gaeta, Efficient discrete-event simulation of colored Petri nets. *IEEE Trans. Softw. Eng.* **22**(9), 629–639 (1996)
21. R. German, D. Logothetis, K. Trivedi, Transient analysis of Markov regenerative stochastic Petri nets: a comparison of approaches, in *Proceedings of the PNPM* (IEEE CS Press, Washington, 1995), pp. 103–112
22. G. Graham, Queueing network models of computer system performance. *ACM Comput. Surv.* **10**(3), 219–224 (1978)
23. P. Haas, G. Shedler, Regenerative stochastic Petri nets. *Perf. Eval.* **6**(3), 189–204 (1986)
24. K. Jensen, *Coloured Petri nets. Basic Concepts, Analysis Methods and Practical Use*, vols. 1–3 (Springer Inc., New York, 1997)

25. R. Jones, G. Ciardo, Regenerative simulation of stochastic Petri nets with discrete and continuous timing, in *Proceedings of the PMCCS* (2003), pp. 23–26
26. H. Kobayashi, *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology* (Addison-Wesley, Boston, Reading, 1978)
27. T.G. Kurtz, Solutions of ordinary differential equations as limits of pure jump Markov processes. *J. Appl. Probab.* **1**(7), 49–58 (1970)
28. F. Liu, M. Heiner, M. Yang, An efficient method for unfolding colored Petri nets, in *Winter Simulation Conference* (IEEE CS Press, Los Alamitos, 2012), p. 295
29. P.M. Merlin, D.J. Farber, Recoverability of communication protocols: implications of a theoretical study. *IEEE Trans. Commun.* **24**(9), 1036–1043 (1976)
30. J. Meyer, On evaluating the performability of degradable computing systems. *IEEE Trans. Comput.* **29**(8), 720–731 (1980)
31. M.K. Molloy, Performance analysis using stochastic Petri nets. *IEEE Trans. Comput.* **31**(9), 913–917 (1982)
32. M.K. Molloy, Discrete time stochastic Petri nets. *IEEE Trans. Softw. Eng.* **11**(2), 417–423 (1985)
33. M. Molloy, T. Murata, M. Vernon (eds.) *Proceedings of the International Workshop on Petri Nets and Performance Models*, Madison, Wisconsin, August 1987 (IEEE-CS Press, Washington, 1987)
34. J. Noe, G. Nutt, Macro e-nets representation of parallel systems. *IEEE Trans. Comput.* **31**(9), 718–727 (1973)
35. J. Peterson, *Petri Net Theory and the Modeling of Systems* (Prentice Hall, Upper Saddle River, 1981)
36. C. Petri, Communication with automata. Technical Report RADC-TR-65–377, Rome Air Dev. Center, New York, 1966
37. W. Reisig, *Petri Nets: An Introduction* (Springer, Berlin, 1985)
38. M. Silva, J. Julvez, C. Mahulea, C. Vazquez, On fluidization of discrete event models: observation and control of continuous Petri nets. *Discrete Event Dynam. Syst. Theory Appl.* **21**(4), 427–497 (2011)
39. F. Symons, The description and definition of queueing systems by numerical Petri nets. *Austr. Telecommun. Res.* **13**(2), 20–31 (1980)
40. M. Tribastone, Scalable differential analysis of large process algebra models, in *Proceedings of the QEST'10* (IEEE CS Press, Los Alamitos, 2010), p. 307
41. M. Tribastone, S. Gilmore, J. Hillston, Scalable differential analysis of process algebra models. *IEEE Trans. Softw. Eng.* **38**(1), 205–219 (2012)
42. M. Wan, G. Ciardo, A. Miner, Approximate steady-state analysis of large Markov models based on the structure of their decision diagram encoding. *Perf. Eval.* **68**(5), 463–186 (2011)

# Modelling Time Using Petri Nets



**Michel Diaz**

When we started our research on testing the first computer systems, including computer peripherals, which were already rather complex to understand and design, we were using hardware and logical descriptions. Later when the complexity increased, it became obvious to us that a formal behavioural description would be of great help. Looking at the existing proposals we discovered Petri Nets (PNs) and it appeared to us that they would provide an excellent model and very useful support to represent the systems' behaviours, as they include intrinsic parallelism and synchronisation. We then designed Petri-net-based simulators to derive test sequences. Still later, the behaviours of communicating systems depending on time and delays proved to be more and more important, making the designs more complex, and so we devoted a part of our research to the description of systems whose behaviour depends on explicit values of time. Again, we found that Petri Nets provide an excellent basic model to represent and handle behaviours based on explicit values of time.

During our research we actually met Carl Adam Petri and discussed with him our ideas for modelling systems and the selected ways to extend Petri Nets to represent and handle explicit values of time. He was really quite interested in these new efficient extensions of the basic model and he found them quite simple, elegant and attractive.

We will now discuss how these models can express fundamental and complex temporal requirements by extending Place-Transition (PT) PNs. To represent and analyse systems relying on explicit values of time, three of the proposed extensions of PTs will be briefly discussed, viz., Timed PNs, Time PNs and Timely Synchronised PNs.

---

M. Diaz (✉)

Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS-CNRS), Toulouse, France

e-mail: [michel.diaz@laas.fr](mailto:michel.diaz@laas.fr)

## 1 Timed PNs

In their basic semantics PNs do not have time values. Nevertheless, when going from a marking to a successor marking the model assumes the indivisibility of the firing of transitions, which means that a firing takes no (i.e. zero) time. In PTs the behaviour of a timed system is represented in an ad hoc way, for instance in protocols by adding a place that is marked when a message is lost. Such a simple solution allows the validation of rather complex algorithms, such as a virtual ring protocol, e.g. showing that its logical design is correct not only with respect to the loss of messages, but also when an interface cannot send any message and when an interface cannot receive any message [1].

The first time extension of PTs was called Timed PNs [2] and used for performance analysis. This model added to PTs a time value for each transition, to represent the time needed for a firing. Here a transition is fired as soon as it enabled.

## 2 Time PNs (TPN)

A more general definition was given by [3]. It defines Time Petri Nets by adding a two-valued time interval  $[\tau_i\text{min}, \tau_i\text{max}]$  to each transition  $t_i$ , (Fig. 1). The semantics is that when a transition  $t_i$  becomes enabled, for instance at time  $\theta$ , it cannot be fired before the time  $(\theta + \tau_i\text{min})$  and it must be fired by the time  $(\theta + \tau_i\text{max})$ . Also here, firing a transition takes zero time.

Note that any significant time values in the system must be explicitly represented in the model.

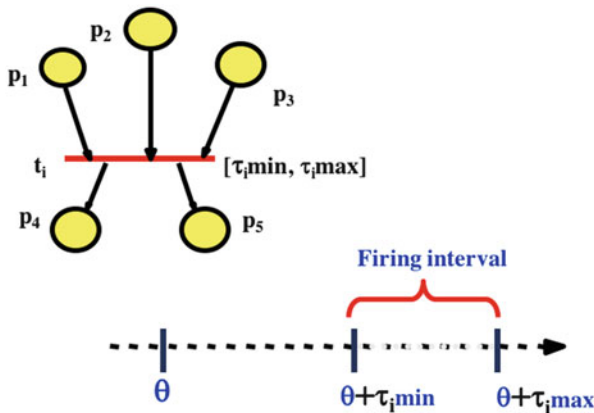


Fig. 1 A Time PN

Clearly, Time PNs are able to exactly express real timers, i.e. a delay value together with its potential maximum drift, but also the min and max times of a given computation. The formal analysis of the behaviours of TPNs is given in [4, 5].

We had an opportunity to discuss the TPN model with Carl Adam Petri, twice, during two PN conferences. Although he was not working on time representation and analysis, he found this approach quite appealing and easy to use and understand. Unfortunately, we had no opportunity to discuss with Carl Adam our next model, TSPN.

### 3 Timely Synchronised PNs (TSPNs)

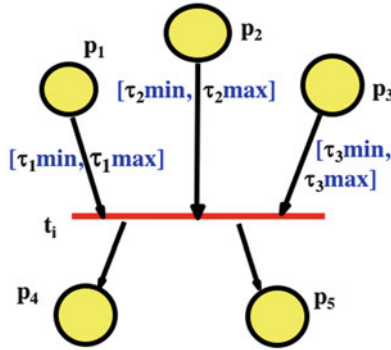
When carefully considering TPNs, it appears that the behaviour of a transition can be decomposed into two steps. Step 1: The transition waits to be enabled, i.e., there exists, for instance between the first token and the last token marking the input places of the transition, an undefined waiting time. Step 2: At the time when the transition becomes enabled, the model starts counting the time (of the time interval). These two steps imply a subtle underspecification that must be well understood: *the time behaviours of the enabling input places are not explicitly handled before the enabling of the transition, as counting the time starts only at the moment when the transition is enabled. Consequently, as the input places do not have an explicit temporal behaviour, fully autonomous behaviours cannot be modelled.*

This led to the definition of temporal synchronisation and composition in [6–8]. The idea is that, as counting time is local to all behaviours, when a place is marked, its temporal behaviour has to be enabled. It follows then that the [min, max] value intervals must be assigned, not to the synchronisation transition, but rather to the arcs connecting places with transitions. In such a model, the time interval related to the place can start as soon as it is marked. As a consequence, the temporal synchronisation begins as soon as the first place of a given transition receives a token, and continues until all the places receive a token, of course together with their arc-related intervals.

As can be seen from Fig. 2, the time arcs can lead to very different semantics for the firing of transitions in TSPNs, and many cases are possible.

For instance, it may happen, with just two places  $p_1$  and  $p_2$  and  $\theta_i$  being the time at which place  $p_i$  is marked, that the time value  $(\theta_1 + \tau_1 \max)$  is smaller than  $(\theta_2 + \tau_2 \min)$ . In such a case, the transition is then not really enabled (Fig. 3), as the second token has not arrived before the end of the time existence of the first token. Of course, one possible semantics is to state that there is an error in the modelled behaviour, and the simulation or verification can be stopped.

Nevertheless, in real life, missing a time synchronisation (rendezvous) often does not stop the behaviour, for instance of the first arrived token. This shows that this simple error semantics has to be refined, for instance by tagging the transition to be



- If Arc  $a_j = (p_j, t_i)$
- is enabled by a token in  $p_j$  at time  $\theta_j$  then
  - it is firable between  $\theta_j + \tau_{jmin}$  and  $\theta_j + \tau_{jmax}$

Fig. 2 A TSPN

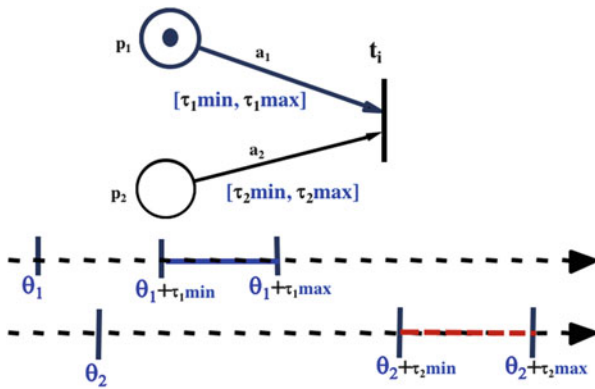


Fig. 3 Missed timed synchronisation

temporally Possibly Missed (PM), and by sending in such a case a warning message, while still continuing the simulation or validation.

Furthermore, even for the classical enabling behaviour, in which all places enable the transition, it appears that different semantics are still possible and acceptable. As an example, Fig. 4 gives four different temporal firing semantics that were proposed for multimedia systems (in the papers [6–8] cited already): AND from max of the min to min of the max (all arcs enabled), WEAK-AND, STRONG-OR, OR from min of the min to max of the max (at least one arc enabled).

This shows that several relevant semantics of composition can be defined, in particular in different application areas. A paramount aspect of these different semantics is that temporal synchronisation *cannot be defined a priori*. In fact, the

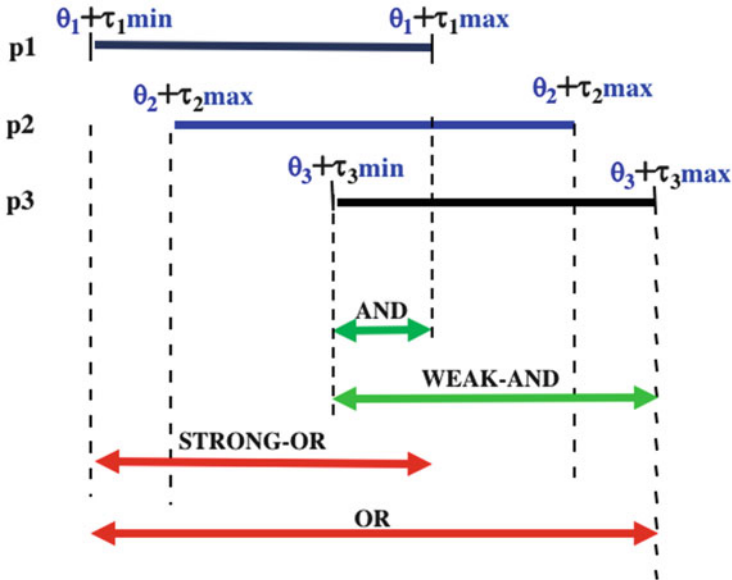


Fig. 4 Possible firing semantics for TSPNs

semantics to be selected depends on the meaning of the composition, defined by the application context in which the composition occurs. The semantics will depend on the (higher) identity of the tokens, as the waiting time for the tokens may depend on the context. For instance, to represent the reality, if the synchronisation concerns two people, then a token representing the behaviour of one person may stay (wait) longer or shorter, depending on the other person that the first arrived is waiting for.

These different semantics have a very important implication: *The choice of the semantics of a temporal synchronisation is a pragmatic decision* (in the chain syntax, semantics, pragmatics), as it comes from a higher design level than the considered level, e.g., from the application level.

This implies that a simple automatic (programmed) composition semantics cannot be used for defining full time synchronisation, for instance in contradistinction to the normal non-timed rendezvous that is defined by merging related transitions, independently of the higher levels.

## 4 Conclusion

Nowadays, there exist more and more systems, such as real-time systems and embedded systems, whose behaviours depend on explicit values of time.

Of course, present autonomous temporal behaviours greatly complicate the models and the specifications of these systems.

Nevertheless, it has been shown that simple, easy to understand models are able to represent and analyse sophisticated and complex time behaviours, showing again the potential descriptive power and relevance of PNs in this area.

## References

1. J.-M. Ayache, J. Courtiat, M. Diaz, Rebus, a fault tolerant distributed system for industrial real-time control. *IEEE Trans. Comput.* **C-31**(7), 637–647 (1982)
2. C. Ramchandani, *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets*. Ph.D. Thesis, Dept. Electrical Engineering, MIT, Cambridge, MA, 1974
3. P.M. Merlin, D.J. Farber, Recoverability of communication protocols: implications of a theoretical study. *IEEE Trans. Commun.* **24**(9), 1036–1043 (1976)
4. B. Berthomieu, M. Menasche, A state enumeration approach for analyzing time Petri nets, in *Proceeding of Applications and Theory of Petri Nets (ATPN'82)*, Como, Italy, 1982, pp. 27–56
5. B. Berthomieu, M. Diaz, Modeling and verification of time dependent systems using Time Petri Nets. *IEEE Trans. Softw. Eng.* **17**(3), 259–273 (1991)
6. M. Diaz, P. Senac, Time Stream Petri Nets, a multi-media model for streams synchronization, in *Proceedings of the First International Conference on Multi-Media Modelling*, vol. 1, ed. by T.-S. Chua, T.L. Kunii (World Scientific Singapore, Singapore, 1993), pp. 257–273
7. M. Diaz, P. Senac, Time Stream Petri Nets, a model for timed multi-media information, in *15th International Conference on Application and Theory of Petri Nets, June 1994*, ed. by R. Valette. LNCS, vol. 815 (Springer, Berlin, 1994), pp. 219–238
8. P. Senac, M. Diaz, A. Light, P. De Saqui Sannes, Modelling logical and temporal synchronization in hypermedia systems. *IEEE J. Selected Areas Commun.* **14**(1), 84–103 (1996)



# All True Concurrency Models Start with Petri Nets: A Personal Tribute to Carl Adam Petri



Wojciech Penczek

## 1 Starting from Petri Nets

Paraphrasing ‘All roads lead to Rome’ one might say ‘All true concurrency models start with Petri nets.’ So, in my case, Petri nets have been my first research topic. Initially, I investigated their algebraic properties, but later my work focused on verification of Petri nets using various model reduction techniques and symbolic model-checking approaches.

### 1.1 *Petri Nets Have Always Been Around*

My first contact with Petri nets was in 1984–1985 during my studies at Warsaw University of Technology and the University of Warsaw. Then I learned the basic definitions of this true concurrency model for distributed systems and could enjoy its attractive graphical representations.

In 1986 when I joined the Institute of Computer Science of the Polish Academy of Sciences, my work, supervised by Prof. Antoni Mazurkiewicz, was focused on prime event structures, traces, and trace systems. I have been impressed by the match between trace systems and condition-event systems, their elegant composition and decomposition methods, and how the trace theory is used as a tool for reasoning about the behavior of Petri nets [5]. Later, I could also see how extensions of traces,

---

W. Penczek (✉)

Institute of Computer Science, PAS, Warsaw, Poland

Siedlce University of Natural Sciences and Humanities, Institute of Computer Science (ICS),  
Siedlce, Poland

e-mail: [penczek@ipipan.waw.pl](mailto:penczek@ipipan.waw.pl)

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,  
[https://doi.org/10.1007/978-3-319-96154-5\\_24](https://doi.org/10.1007/978-3-319-96154-5_24)

193

such as infinite traces or semi-traces, are used for modeling behaviors of more involved variants of Petri nets.

## 1.2 Meeting Carl Adam Petri at GMD

My first foreign research trip was to the ‘Gesellschaft für Mathematik und Datenverarbeitung’ (GMD, Society for Mathematics and Information Technology) in 1988. Invited by Prof. Ursula Goltz I spent a few weeks at GMD, having the privilege of meeting there famous people working on Petri nets. These were Wolfgang Reisig, Eike Best, Kurt Lautenbach, and finally I was introduced to Carl Adam Petri. This was a very short meeting and we only exchanged a few sentences, but I do remember this contact very well, being under a very strong impression of the personality of Prof. Petri who was kindly interested in my research. He asked about my scientific interests and the aim of my visit to GMD. This personal contact with Prof. Petri has strongly influenced my scientific career and the topics of my interest. Most of my further research results have been concerned with verification of Petri nets.

## 2 Verification of Time Petri Nets

My work on verification of (time) Petri nets started with several results concerning abstractions and partial order reductions [8, 9]. In 2001 these were the main methods for alleviating the state explosion problem of state spaces. Regardless of the true concurrency nature of Petri nets, their state spaces built on marking graphs may grow exponentially in the number of transitions or can be infinite in the case of time Petri nets. So, efficient verification was possible only after abstracting or reducing the state spaces. Many verification methods for time Petri nets were adapted from those developed for Timed Automata and vice versa [10]. One of the most important examples is minimization algorithms for time Petri nets [12] exploiting *partitioning refinement*. Then, we put forward methods based on bounded model checking using BDDs as well as SAT- and SMT-solvers, applied to both concrete and abstract state spaces [4, 6, 7, 11, 13].

In order to discuss verification methods of time Petri nets in more detail, it is useful to recall their definition.

**Definition 1** A time Petri net (TPN) is a six-tuple  $\mathcal{N} = (P, T, F, m^\circ, Eft, Lft)$ , where

- $P = \{p_1, \dots, p_{np}\}$  is a finite set of *places*,
- $T = \{t_1, \dots, t_{nT}\}$  is a finite set of *transitions*, where  $P \cap T = \emptyset$ ,
- $F: (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$  is the *flow function*, where  $\mathbb{N}$  is the set of natural numbers, including 0,
- $m^0: P \rightarrow \mathbb{N}$  is the *initial marking* of  $\mathcal{N}$ ,

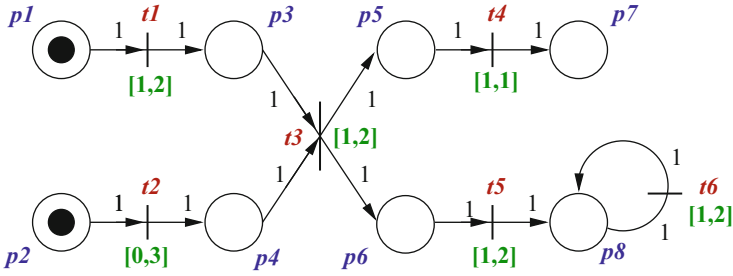


Fig. 1 A time Petri net

- $Eft: T \rightarrow \mathbb{N}, Lft: T \rightarrow \mathbb{N} \cup \{\infty\}$  are functions describing respectively the *earliest firing time* and the *latest firing time* of the transitions, where clearly  $Eft(t) \leq Lft(t)$  for each  $t \in T$ .

A four-tuple  $(P, T, F, m^0)$  is called a Petri net.

An example of a time Petri net is shown in Fig. 1. The values of the functions  $Eft$  and  $Lft$  are given by  $Eft(t) = 1$  and  $Lft(t) = 2$  for  $t \in \{t_1, t_3, t_5, t_6\}$ ,  $Eft(t_2) = 0$ ,  $Lft(t_2) = 3$ , and  $Eft(t_4) = Lft(t_4) = 1$ .

Intuitively, a marking specifies how many tokens (denoted by the black dots) are stored in particular places. Firing of a transition consumes a number of tokens from its input places and produces a number of tokens in its output places; the above numbers are given by the corresponding values of  $F$  (represented in the figure by the numbers decorating the arrows connecting those pairs of a place and a transition for which the value of  $F$  is not equal to zero).

Intuitively, a transition  $t$  is enabled if all its input places contain a sufficient number of tokens to be consumed by the flow function. The earliest and latest firing times of a transition  $t$  specify the timing interval in which  $t$  can be fired. If the time passed since the transition  $t$  has become enabled reaches the value  $Lft(t)$ , the transition has to be fired, unless disabled by a firing of another transition. In order to simplify some subsequent notions, we consider only nets satisfying the following two conditions:

- the flow function  $F$  maps onto  $\{0, 1\}$ , so  $F$  can be identified with the relation  $F \subseteq (P \times T) \cup (T \times P)$ , and
- the number of tokens that can be stored in a place is limited to 1, so  $m^0$  can be identified with the set of places  $m^0$  assigns to 1. Introducing such a limit changes slightly the conditions under which a transition is enabled, which can be seen below.

In order to describe models for time Petri nets we need to give some basic definitions.

Let  $t \in T$  be a transition.

- By the *preset* of  $t$  we mean the set of its input places  $\bullet t = \{p \in P | F(p, t) = 1\}$ .

- By the *postset* of  $t$  we mean the set of its output places  $t\bullet = \{p \in P \mid F(t, p) = 1\}$ .
- By a *marking* of  $\mathcal{N}$  we mean any subset  $m \subseteq P$ .
- A transition  $t$  is *enabled* at  $m(m[t])$  for short) if  $\bullet t \subseteq m$  and  $t\bullet \cap (m \setminus \bullet t) = \emptyset$ , i.e.,  $m$  contains each input place of  $t$ , and  $m$  does not contain any output place of  $t$  except for these places which are both input and output places of  $t$ .
- By  $en(m) = \{t \in T \mid m[t]\}$  we mean the set of transitions enabled at  $m$ .
- If  $t$  is enabled at  $m$ , then  $t$  leads from  $m$  to the marking  $m' = (m \setminus \bullet t) \cup t\bullet$ .

## 2.1 Concrete Models of TPNs

A concrete state records a snapshot of the behavior of a TPN. In order to verify whether a system modeled by a TPN satisfies some properties, we need to represent the set of all its concrete states, defined below, together with their valuation, with propositions, which are used for formulating the properties. Such a structure is called a concrete model. Typically, one of two approaches to building concrete models is applied, depending on whether the flow of time is recorded by real variables, called clocks, or firing intervals. We discuss the clock approach in more detail.

A *concrete state* of a TPN is a pair  $\sigma = (m, clock)$ , where  $m$  is a marking and  $clock$  is a function that gives values to the clocks that can be associated with the transitions or the places. If a net is distributed, i.e., composed of well-defined sequential processes, then the clocks can be also assigned to its processes [10]. The explanation below is for the clocks assigned to the transitions. The initial state is denoted by  $\sigma^0 = (m^0, (0, \dots, 0))$ . The concrete state changes because of either the firing of an enabled transition  $t \in T$  for which  $Eft(t) \leq clock(t) \leq Lft(t)$ , denoted  $\sigma \xrightarrow{t}_c \sigma'$ , or the passing of some time provided this does not disable any enabled transition, denoted  $\sigma \xrightarrow{\tau}_c \sigma'$ , where  $\tau$  is a special symbol. Notice that  $\tau$  does not specify how much time passed, but it is used to label the time transitions.

Since our aim is to verify temporal (i.e., changing in time) properties of markings of a TPN  $\mathcal{N} = (P, T, F, m^0, Eft, Lft)$ , we use the propositional variables corresponding to its places. Formally, let  $PV = \{\wp_p \mid p \in P\}$  be a set of propositional variables, where the propositional variable  $\wp_p$  corresponds to a place  $p \in P$ .

**Definition 2 (Concrete Model for TPN)** A *concrete model* for a time Petri net  $\mathcal{N} = (P, T, F, m^0, Eft, Lft)$  is a tuple  $M_c(\mathcal{N}) = ((\Sigma, \sigma^0, \rightarrow_c), V_c)$ , where

- $\Sigma$  is the set of all the concrete states of  $\mathcal{N}$ ,
- $\sigma^0$  is the initial state,
- $\rightarrow_c$  is the transition relation, and
- $V_c: \Sigma \rightarrow PV$  is a *valuation function* such that  $V_c((m, \cdot)) = \{\wp_p \mid p \in m\}$  i.e.,  $V_c$  assigns the same propositions to concrete states with the same markings.

To abstract from the flow of time in the transition relation, in the definition of a concrete model instead of  $\rightarrow_c$  one can use also the *discrete transition relation*  $\rightarrow_d \subseteq \Sigma \times T \times \Sigma$ , defined as:  $\sigma \xrightarrow{t}_d \sigma'$  iff  $(\exists \sigma_1, \sigma_2 \in \Sigma) \sigma \xrightarrow{\tau^*}_c \sigma_1 \xrightarrow{t}_c \sigma_2 \xrightarrow{\tau^*}_c \sigma'$ .

Unfortunately, concrete models are typically infinite. Therefore, we need to abstract them into preferably finite abstract ones, i.e., models whose nodes are sets of concrete states. The transitions of the abstract models are labeled with elements of the set  $B = T \cup \{\tau\}$ , consisting of the transitions  $T$  of  $\mathcal{N}$  and the special symbol  $\tau$ , as defined below.

**Definition 3 (Abstract Model for TPN)** A structure  $M_a(\mathcal{N}) = ((W, w^0, \rightarrow), V)$  is an *abstract model* for a concrete model  $M_C(\mathcal{N}) = ((S, s^0, \rightarrow), V_c)$ , where

- each node  $w \in W$  is a set of states of  $S$  and  $s^0 \in w^0$ ,
- $V(w) = V_c(s)$  for each  $s \in w$ ,
- $\rightarrow \subset W \times B \times W$  such that

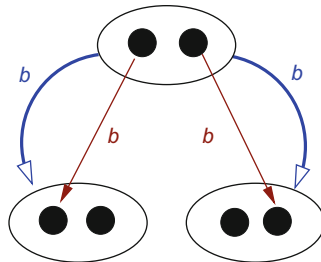
$$EE: (\forall w_1, w_2 \in W) (\forall b \in B) : w_1 \xrightarrow{b} w_2 \text{ if } (\exists s_1 \in w_1)(\exists s_2 \in w_2) \text{ s.t. } s_1 \xrightarrow{b} s_2.$$

The condition EE is illustrated in Fig. 2.

Definition 3 is very general, so it needs to be refined in order to ensure that abstract models preserve the properties expressible in a given temporal logic. Moreover Definition 3 does not give any clue how to build abstract models. We elaborate on these two issues in the next two sections.

## 2.2 Abstract Models Preserving Temporal Logics

Properties of timed systems are usually expressed using temporal logics. In this section our focus is on the logics that are most commonly, used, i.e., Linear Time Temporal Logic (LTL) and Computation Tree Logic\* (CTL\*); see Chapter 4 of [10]. LTL is interpreted over all the state sequences of a model starting at the initial state, while CTL\* is interpreted over the tree resulting from unwinding a model starting



**Fig. 2** An example of an abstract model satisfying the condition EE. The ovals represent the abstract states, the black dots stand for the concrete states, and the red (blue) arrows represent the concrete (abstract, resp.) transitions. Valuations of the nodes are omitted. The same graphical conventions apply to Figs. 3, 4, and 5

from the initial state. The formulas of LTL and of CTL\* are built from propositional variables of  $PV$  using standard Boolean operators  $\neg$ ,  $\vee$ ,  $\wedge$ , and the state operators  $X$  (neXt),  $U$  (Until), and  $R$  (Release), the meanings of which correspond to their names. Since CTL\* is interpreted over trees, its language contains also the two path quantifiers  $A$  (for All sequences) and  $E$  (there Exists a sequence) that allow us (to apply the state operators to some sequence or to all sequences starting at some state. We also use the restriction of CTL\*, called ACTL\*, such that negation can only be applied to the propositional variables and the operator  $E$  cannot be used. For example,  $X\{\wp_p U \wp_q\}$  is an LTL formula,  $AG(EX(\wp_p U \wp_q))$  is a formula of CTL\*, while  $AG(AX(\wp_p U \wp_q))$  is an ACTL\* formula.

Three more properties, EA, AE, and U, of abstract models are defined below in order to ensure the preservation of the three temporal logics, LTL, CTL\*, and ACTL\*, respectively. *Surjective models* are abstract models that satisfy the condition EA, given below.

These models preserve the logic LTL.

$$EA: (\forall w_1, w_2 \in W) (\forall b \in B) : w_1 \xrightarrow{b} w_2 \Rightarrow (\forall s_2 \in w_2) (\exists s_1 \in w_1) s_1 \xrightarrow{b} s_2.$$

The condition EA is illustrated in Fig. 3.

*Bisimulating models* are abstract models that satisfy the condition AE, defined below. These models preserve the logic CTL\*.

$$AE: (\forall w_1, w_2 \in W) (\forall b \in B) : w_1 \xrightarrow{b} w_2 \Rightarrow (\forall s_1 \in w_1) (\exists s_2 \in w_2) s_1 \xrightarrow{b} s_2.$$

The condition AE is illustrated in Fig. 4.

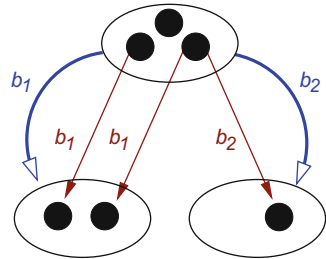
*Simulating models* are abstract models that satisfy the condition U, defined below. These models preserve the logic ACTL\*.

U: For each  $w \in W$  there is a nonempty  $w^{cor} \subseteq w$  such that  $s^0 \in (w^0)^{cor}$ , and

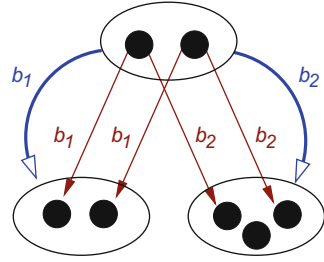
$$(\forall w_1, w_2 \in W) (\forall b \in B) : w_1 \xrightarrow{b} w_2 \Rightarrow (\forall s_1 \in w_1^{cor}) (\exists s_2 \in w_2^{cor}) s_1 \xrightarrow{b} s_2.$$

The condition U is illustrated in Fig. 5, where the blue ovals denote subsets  $w^{cor}$  of the abstract states and the blue-dashed-violet arrows represent the abstract transitions.

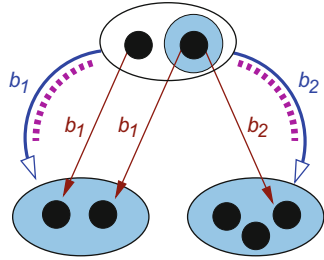
**Fig. 3** An example of an abstract model satisfying the condition EA



**Fig. 4** An example of an abstract model satisfying the condition AE



**Fig. 5** An example of an abstract model satisfying the condition U



There are different methods of generating abstract models, descriptions of which can be found in [10]. One of the main methods, called *partitioning refinement*, starts with a small abstract model preserving only the propositional variables of a given formula. Then, this abstract model is refined by splitting states until one of the properties EA, AE, or U begins to hold, depending on which temporal logic the formula belongs to. Abstract models constructed this way are finite, thus they allow the standard model-checking method to be applied.

### 2.3 Partial Order Reductions (POR) [8]

However, it may turn out that the abstract models defined above are still too large to be efficiently verified. Then, we can weaken languages of temporal logics by disallowing the use of the next-step operator. This is indicated with  $-X$  added to the name of each logic. Then, the equivalence to be preserved by a reduced model is weakened as well. Such an equivalence relation preserving  $CTL^*-X$  ( $ACTL^*-X$ ,  $LTL-X$ ) is called *stuttering bisimulation* (*simulation*, *trace equivalence*, *respectively*).

The intuitive idea behind partial order reductions consists in generating reduced abstract models such that for each maximal path  $p$  in the abstract model, the reduced one contains at least one (but as few as possible) maximal path  $p'$ , which differs from  $p$  only in the ordering of independent transitions. This independence relation is symmetric and is defined for Petri nets as follows. Two transitions  $t, t'$  are independent iff  $(\bullet t \cup t \bullet) \cap (\bullet t' \cup t' \bullet) = \emptyset$ , i.e.,  $t$  and  $t'$  have no common input and no common output places. In the case of time Petri nets one can define an asymmetric covering relation by  $t$  covers  $t'$  iff  $t, t'$  are independent and  $Eft(t) = 0$ .

There are many algorithms that use this independency or covering relation to generate a reduced model without building the full one. At each node  $w$  of the generated reduced state space, the algorithm computes a subset (called *stubborn* or *ample*) of the set of all the enabled transitions and generates the successor nodes for the transitions of this subset only.

## 2.4 Symbolic Model Checking for TPN

At the Institute of Computer Science, Polish Academy of Sciences we have developed several new methods for verification of (parametric) time Petri nets. We considered two symbolic approaches, SAT- and BDD-based, to bounded model checking (BMC) of (parametric) time Petri nets and focused on the properties expressed in Linear Temporal Logic and the existential fragment of Computation Tree Logic. More specifically, we developed a SAT-based (parametric) reachability method for a class of distributed time Petri nets [11], which are TPNs composed of sequential processes, BMC approaches for verification of distributed time Petri nets [7], bounded parametric model checking for Petri nets [4], and a BDD-based BMC method for temporal properties of 1-Safe Petri nets [6]. Moreover, we proved and exploited the fact that the discrete-time semantics is sufficient to verify the properties formulated in the existential and the universal version of CTL\* of time Petri nets with the dense semantics [2]. Recently, an SMT-based reachability-checking method for bounded time Petri nets was defined [13].

## 2.5 Verics

Our tool Verics [3] is a model checker composed of several independent modules aimed at verification of (parametric) time Petri nets, timed automata, and multiagent systems. The verification engine is mainly based on translations of the model-checking problem into the SAT problem. Depending on the type of considered system, the verifier enables us to test various classes of properties—from reachability of a state satisfying certain conditions to more complicated features expressed with formulas of (timed) temporal, epistemic, or deontic logics. The implemented model-checking methods include SAT-based ones as well as these based on generating abstract models for systems.

## 2.6 SAT-Based Bounded Model Checking

Bounded model checking is a symbolic method aimed at verification of temporal properties of distributed (timed) systems. It is based on the observation that some



properties of a system can be checked using only a part of its model. In order to apply SAT-based BMC to testing whether a system satisfies a certain (usually undesired) property, the transition relation of a given system is unfolded up to some depth  $k$  and encoded as a propositional formula. The formula expressing the property of interest is encoded as a propositional formula as well and satisfiability of the conjunction of these two formulas is checked using a SAT-solver. If the conjunction is satisfiable, one can conclude that a witness was found. Otherwise, the value of  $k$  is incremented. The above process is terminated when the value of  $k$  is equal to the diameter of the system, i.e., to the maximal length of a shortest path between its two arbitrary states.

## 2.7 Encoding for PN and TPN

Below we show how to encode the paths of length  $k$ . A set of global states is represented by a single *symbolic state*, i.e., as a vector of propositional variables  $\mathbf{w}$  (called a *state variable vector*). Then,  $k+1$  state variable vectors stand for a symbolic  $k$ -path, where the first symbolic state encodes the initial state of the system while the last one corresponds to the last states of the  $k$ -paths. The formula encoding the symbolic  $k$ -path is defined as follows:

$$path_k(\mathbf{w}^0, \dots, \mathbf{w}^k) = \mathcal{I}(\mathbf{w}^0) \wedge \bigwedge_{i=0}^{k-1} \mathcal{T}(\mathbf{w}^i, \mathbf{w}^{i+1}) \quad (1)$$

where  $\mathcal{I}(\mathbf{w}^0)$  encodes the initial state of the system, and  $\mathcal{T}(\mathbf{w}^i, \mathbf{w}^{i+1})$  encodes a transition between symbolic states represented by the global state vectors  $\mathbf{w}^i$  and  $\mathbf{w}^{i+1}$ .

In what follows we briefly describe how to define  $\mathcal{T}(\mathbf{w}^i, \mathbf{w}^{i+1})$  for PN and TPN.

**Implementation for PN** Consider a Petri net  $\mathcal{N} = (P, T, F, m^0)$ , where the places are denoted by the integers smaller than or equal to  $n = |P|$ . We use the set  $\{p_1, \dots, p_n\}$  of propositions, where  $p_i$  is interpreted as the presence of a token in the place  $i$ . The states of  $S$  are encoded by valuations of a vector of state variables  $\mathbf{w} = (\mathbf{w}[1], \dots, \mathbf{w}[n])$ , where  $\mathbf{w}[i] = p_i$  for  $0 \leq i \leq n$ . The initial state and the transition relation  $\rightarrow$  are encoded as follows:

- $\mathcal{I}(\mathbf{w}^0) := \bigwedge_{i \in m^0} \mathbf{w}^0[i] \wedge \bigwedge_{i \in P \setminus m^0} \neg \mathbf{w}^0[i]$ ,
- $\mathcal{T}(\mathbf{w}, \mathbf{v}) := \bigvee_{t \in T} \left( \bigwedge_{i \in \bullet t} \mathbf{w}[i] \wedge \bigwedge_{i \in (t \bullet \setminus pre(t))} \neg \mathbf{w}[i] \wedge \bigwedge_{i \in (\bullet t \setminus t \bullet)} \neg \mathbf{v}[i] \wedge \bigwedge_{i \in t \bullet} \mathbf{v}[i] \wedge \bigwedge_{i \in (p \setminus (\bullet t \cup t \bullet)) \cup (\bullet t \cap t \bullet)} \mathbf{w}[i] \Leftrightarrow \mathbf{v}[i] \right)$ .

A more detailed description and the encoding of the formulas of the existential fragment of CTL can be found in [11].

**Implementation for Distributed TPN** The main difference between the symbolic encoding of the transition relation of PN and TPN consists in the time flow. Below, we give some details of the encoding for distributed TPN. A current state of a TPN  $\mathcal{N}$  is given by its marking and the time passed since each of the enabled transitions became enabled (which influences the future behavior of the net). Thus, a *concrete state*  $\sigma$  of  $\mathcal{N}$  can be defined as an ordered pair  $(m, clock)$ , where  $m$  is a marking and  $clock: I \rightarrow \mathbb{R}_+$  is a function, which for each index  $i$  of a process of  $\mathcal{M}$  gives the time elapsed since the marked place of this process became marked most recently [12]. In order to deal with countable structures instead of uncountable ones, we can use *extended detailed region graphs*, i.e., abstract models based on the well-known detailed region graphs introduced by Alur et al. for timed automata [1].

To apply the BMC approach we deal with a model obtained by a *discretization* of its extended detailed region graph. The model is of an infinite but countable structure, which, however, is sufficient for BMC (which deals with finite sequences of states only). Instead of dealing with the whole extended detailed region graph, we *discretize* this structure, choosing for each region one or more appropriate representatives. The discretization scheme is based on the one for timed automata [14], and it preserves the qualitative behavior of the underlying system. The details and the formal definitions can be found in [11].

## 2.8 Bounded Parametric Model Checking

BMC is also applied to verification of properties expressed in the existential fragment of the logic PRTCTL, which is an extension of Computation Tree Logic (CTL) by allowing the formulation of properties involving lengths of paths in a model. For example, consider  $EG^{\leq 5}\wp$ , which expresses the fact that there is a path such that in the first six states of this path  $\wp$  holds. Another example is  $\forall_{\Theta_1 \leq 1} \exists_{\Theta_2 \leq 2} EF^{\leq \Theta_1 + \Theta_2} \wp$ , which expresses the fact that for each value of the parameter  $\Theta_1$  not greater than 1 there is a value of the parameter  $\Theta_2$  not exceeding 2 such that  $\wp$  can be reached at a prefix of length at most  $\Theta_1 + \Theta_2 + 1$  of some path. The propositions appearing in these formulas correspond to the places of the net considered. In order to apply verification using BMC, the qualitative properties expressed in PRTECTL are directly encoded as propositional formulas.

## 3 Final Remarks

We have discussed some features of verification that are specific to Petri nets, in particular model abstraction techniques, partial order reductions, and SAT-based bounded-model-checking methods for (time) Petri nets.

Verification of (time) Petri nets is a very active area of research with many new verification methods emerging every year. This is motivated by broad practical applications of time Petri nets to model concurrent systems, real-time systems, stochastic systems, and hybrid systems. It is also important to mention that the efficiency of tools for (time) Petri nets is improving every year (see <http://mcc.lip6.fr/>).

**Acknowledgements** The author is grateful to Dr. Agata Pórola for reading and commenting on this paper, which is mainly based on the joint book [10] and joint research.

## References

1. R. Alur, C. Courcoubetis, D. Dill, Model checking in dense real-time. *Inf. Comput.* **104**(1), 2–34 (1993)
2. A. Janowska, W. Penczek, A. Pórola, A. Zbrzezny, Using integer time steps for checking branching time properties of time Petri nets. in *Trans. Petri Nets and Other Models of Concurrency*, ed. by M. Koutny, W.M.P. van der Aalst, A. Yakovlev. LNCS, vol. 8100(8) (Springer, Berlin, 2013), pp. 89–105
3. M. Knapik, A. Niewiadomski, W. Penczek, A. Pórola, M. Sreter, A. Zbrzezny, Parametric model checking with VerICS, in *Trans. Petri Nets and Other Models of Concurrency*, ed. by M. Knapik et al. LNCS, vol. 6550(4) (Springer, Berlin, 2010), pp. 98–120
4. M. Knapik, M. Sreter, W. Penczek, Bounded parametric model checking for elementary net systems, in *Trans. Petri Nets and Other Models of Concurrency*, ed. by M. Knapik et al. LNCS, vol. 6550(4), 42–71 (Springer, Berlin, 2010)
5. A. Mazurkiewicz. Trab theory, in *Advances in Petri Nets 1986*, ed. by W. Braner, W. Reisig, G. Rozenberg. LNCS, vol. 255 (Springer, Berlin, 1986), pp. 279–324
6. A. Męski, W. Penczek, A. Pórola, BDD-based bounded model checking for temporal properties of 1-safe Petri nets. *Fundam. Inform.* **109**(3), 305–321 (2011)
7. A. Męski, A. Pórola, W. Penczek, B. Woźna-Szcześniak, A. Zbrzezny, Bounded model checking approaches for verification of distributed time Petri nets, in *Proceedings of the International Workshop on Petri Nets and Software Engineering (PNSE'11)* (2011), pp. 72–91
8. W. Penczek, A. Pórola, Abstractions and partial order reductions for checking branching properties of time Petri nets, in *Proceedings of the 22nd International Conference on Applications and Theory of Petri Nets (ICATPN'01)*, ed. by J.M. Colom, M. Koutny. LNCS, vol. 2075 (Springer, Berlin, 2001), pp. 323–342
9. W. Penczek, A. Pórola, Specification and model checking of temporal properties in time Petri nets and timed automata, in *Proceedings of the 25th International Conference on Applications and Theory of Petri Nets (ICATPN '04)*, ed. by J. Cortadella, W. Reisig. LNCS, vol. 3099 (Springer, Berlin, 2004), pp. 37–76
10. W. Penczek, A. Pórola, *Advances in Verification of Time Petri Nets and Timed Automata: A Temporal Logic Approach*. Studies in Computational Intelligence, vol. 20 (Springer, Berlin, 2006)
11. W. Penczek, A. Pórola, A. Zbrzezny, SAT-based (parametric) reachability for a class of distributed time Petri nets, in *Trans. Petri Nets and Other Models of Concurrency*, ed. by M. Knapik et al. LNCS, vol. 6550(4) (Springer, Berlin, 2010), pp. 72–97

12. A. Póřrola, W. Penczek, Minimization algorithms for time Petri nets. *Fundam. Inform.* **60**(1–4), 307–331 (2004)
13. A. Póřrola, P. Cybula, A. Meski, SMT-based reachability checking for bounded time Petri nets. *Fundam. Inform.* **135**(4), 467–882 (2014)
14. A. Zbrzezny, SAT-based reachability checking for timed automata with diagonal constraints. *Fundam. Inform.* **67**(1–3), 303–322 (2005)

# Petri Nets for BioModel Engineering: A Personal Perspective



**Monika Heiner**

**Preamble** This paper gives my personal reflections on an exciting field of research which started its journey about 20 years ago. I'm convinced that it will still carry us a very long way, in our quest to gain deep understanding of living organisms by elucidating the components and their interactions which govern their behaviour. This will finally permit us, among other things, to cautiously direct organic evolution, correct unwanted (spontaneous) mutations or to derive recommendations for personalised treatment of certain diseases.

## 1 How Everything Began

**The Begin of the Beginning** I fell in love with Petri nets in the late 1970s, while searching for a Ph.D. subject. My supervising professor arranged for me to get access to the best library in town, usually reserved for the company's employees only. Reading my way through the hidden gems, I stumbled over two Ph.D. theses, one written by K. Lautenbach, introducing what is known today in Petri net theory as transition/place invariants [16], the other by O. Herzog, analysing PL/I programs for deadlocks by mapping the programs' control structure onto Petri nets [14]. These two treasures, well-protected in the company's library, were the begin of the beginning of my professional orientation. A cross-check in the library of the local technical university didn't reveal a single item with the Petri net buzzword—not yet.

---

M. Heiner (✉)

Department of Computer Science, Brandenburg Technical University, Cottbus, Germany

Department of Computer Science, Brunel University, London, UK

e-mail: [monika.heiner@b-tu.de](mailto:monika.heiner@b-tu.de); <https://www-dssz.informatik.tu-cottbus.de>

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,  
[https://doi.org/10.1007/978-3-319-96154-5\\_25](https://doi.org/10.1007/978-3-319-96154-5_25)

205

But this was about to change, with my Ph.D. thesis [6] becoming one tiny piece of the ever-growing puzzle.

Thus, I started my career by applying Petri nets for verifying technical systems—hardware/software structures alike, writing compilers to do the boring part automatically—the mapping onto Petri nets, so one can concentrate on the interesting part—how to analyse the model, and even more importantly, how to interpret the results gained by Petri net analysis in terms of the modelled technical system. But there were and still are tight limits to this pretty simple idea; and all models are more or less similar in their basic characteristics, with minor variations by changing the programming language, from P1/1 to Algol 68, Ada, OCCAM, CHILL, Java, you name it—so I got tired.

Later, when turning my attention from human-made systems to systems created by organic evolution, I quickly realised that biologists think differently than hardware/software engineers, maybe because they encounter different problems, want to solve different riddles and are hunting different phenomena; anyway, my professional life got exciting again.

**Petri Net Versus Petri Dish** No, the two notions are not related, to answer one of the top FAQs right from the start. Petri dishes were invented by Julius Richard Petri (1852–1921), and Petri nets by Carl Adam Petri (1926–2010); so the two even didn't have a chance to meet each other.

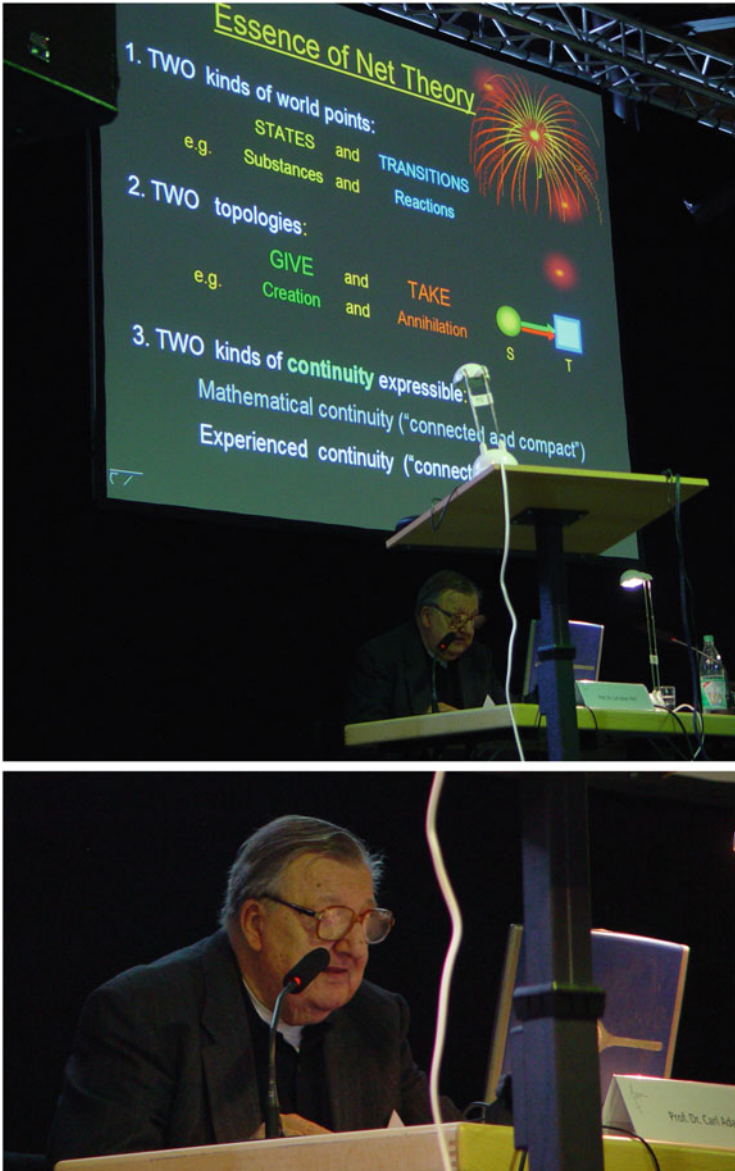
According to some email communication which I had with C.A. Petri in the first decade of this century, he drew some inspiration from chemistry, with his first ideas of how to visualise stoichiometric equations sketched on his high school chemistry textbook. However, his textbook copy got lost in WW2, so it's up to you whether you buy the story.

Anyway, written proofs do exist. C.A. Petri himself mentioned the analogy between the abstract notions of places/transitions and chemical substances/reactions in one of his numerous internal research reports [20], and continued to do so, see Fig. 1. Similarly, very early scientific publications about Petri nets employed simple stoichiometric equations, such as

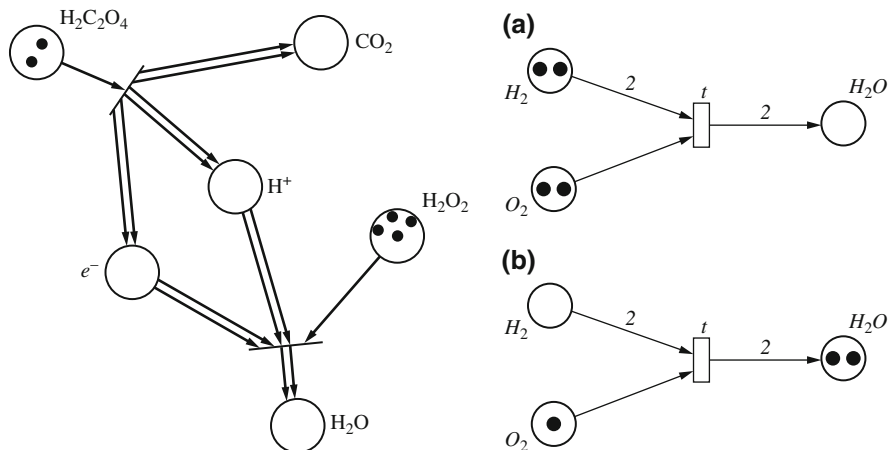


as introductory illustrations to explain the key idea, e.g. the very first textbook about Petri nets in English by J.L. Peterson [19] and the seminal journal paper by T. Murata [18]; see Fig. 2. This doesn't come as a big surprise, as there is a one-to-one correspondence—if you understand one of the two notations, you immediately understand the other one as well. Both describe exactly the very same behaviour; there is zero leeway for interpretation.

This might be the explanation for a remarkable effect I experienced more than one time: biologists or physicians looking for the very first time in their life at a Petri net, with the Petri net describing one of their favourite subjects (e.g. a specific pathway, some very specific proteins or protein interactions), and they immediately



**Fig. 1** C.A. Petri at the Deutsches Technikmuseum in Berlin, November 2006; photographs taken by the author

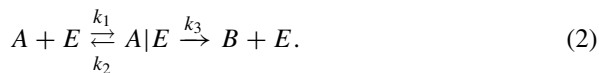


**Fig. 2** Petri net representations for stoichiometric equations in early publications; **(left)** reproduced from [19], **(right)** Petri net for Eq. (1) reproduced from [18]

start to comment: *here something is missing, this can't be right, this is probably meant to be ... , etc..*

So, it was just a matter of time before the business got more serious. First papers re-discovering the key modelling ideas and applying them to metabolic pathways were published in the early 1990s [15, 21], while my first talks addressing this subject go back to the late 1990s [7, 10], followed by my own reviewed publications starting in this century.

**Growing Complexity** These days, systems biology and Petri nets employ a number of related terms, e.g. reactions/transitions, metabolites/places, which are often used interchangeably; see Table 1 for a quick reference, and Fig. 3 for three equivalent Petri net representations of a typical building block of which biochemical pathways are often made—metabolic, signal transduction and gene regulatory pathways alike:



Small Petri net components of this kind can then be composed together to form arbitrarily complex networks, and we are sooner or later bound to encounter a problem—how to cope with the growing complexity of the Petri nets to be constructed?

Being a software engineer myself, help is just around the corner. So, let's re-use construction principles, well established in many engineering disciplines: composition of model components (see Fig. 3) and hierarchical structuring (see

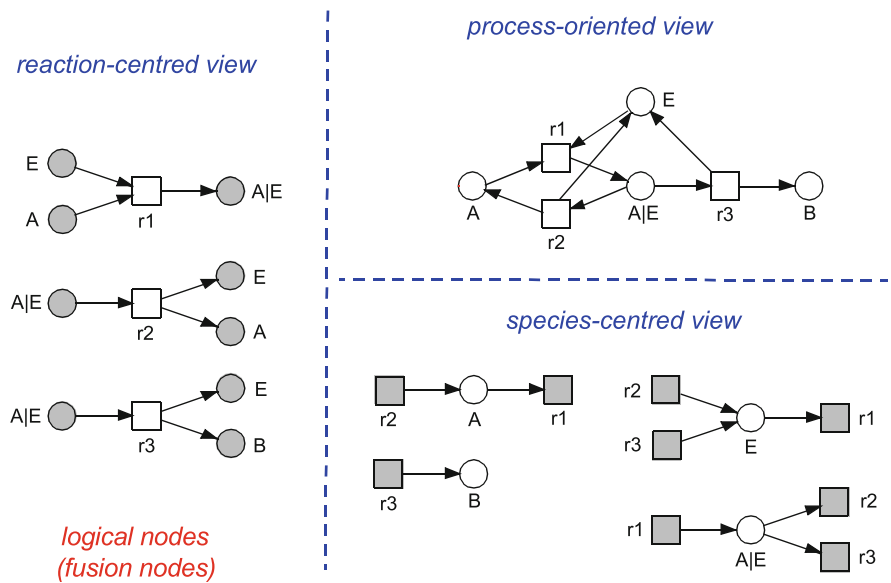


**Table 1** Terminology analogies in Petri nets vs systems biology

Petri nets	Systems biology
Place	Species (metabolite, enzyme, gene, mRNA)
Transition	Reaction (transport step, conformational change)
Arc weight	Stoichiometry
Tokens	Molecules, mass
Token number	Concentration <sup>a</sup>
Marking	State
(Firing) rate	Flux
Incidence matrix	Stoichiometric matrix <sup>b</sup>
P-invariant	Mass conserving subnet
Minimal T-invariant	Elementary mode, extreme pathway <sup>b</sup>

<sup>a</sup> Up to obvious normalisation

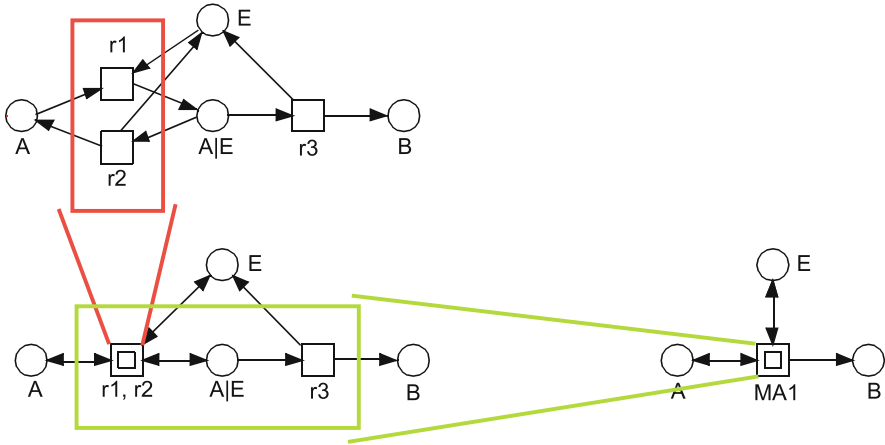
<sup>b</sup> Up to reversible reactions



**Fig. 3** Three graphically different, but mathematically equivalent Petri net representations for the basic enzymatic reaction according to Eq. (2)

Fig. 4). Both principles postpone the problem and work quite well up to Petri nets of a few hundred nodes (assuming the right tool support).

It may even carry us beyond, when committing ourselves to a strictly modular modelling discipline, as demonstrated in [1] for biomolecules involved in the JAK/STAT signalling pathway. Here, each biomolecule gets its own model component (module), composable by shared interface subnets. This allows professionals with expertise in specific biomolecules to concentrate and curate exactly those mod-



**Fig. 4** Hierarchical Petri net representations for the process-oriented view in Fig. 3

ules they feel competent in. Modular modelling also permits us to systematically mimic *in silico* mutations that a biomolecule may undergo *in vivo* or *in vitro*, and then to automatically generate model versions corresponding to a very specific constellation of mutations (gene profile).

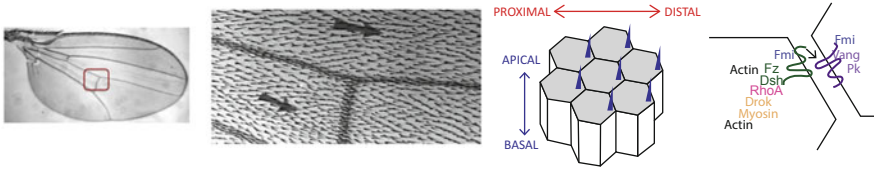
But what next, if all these techniques don't help?

## 2 And Then There Was Colour

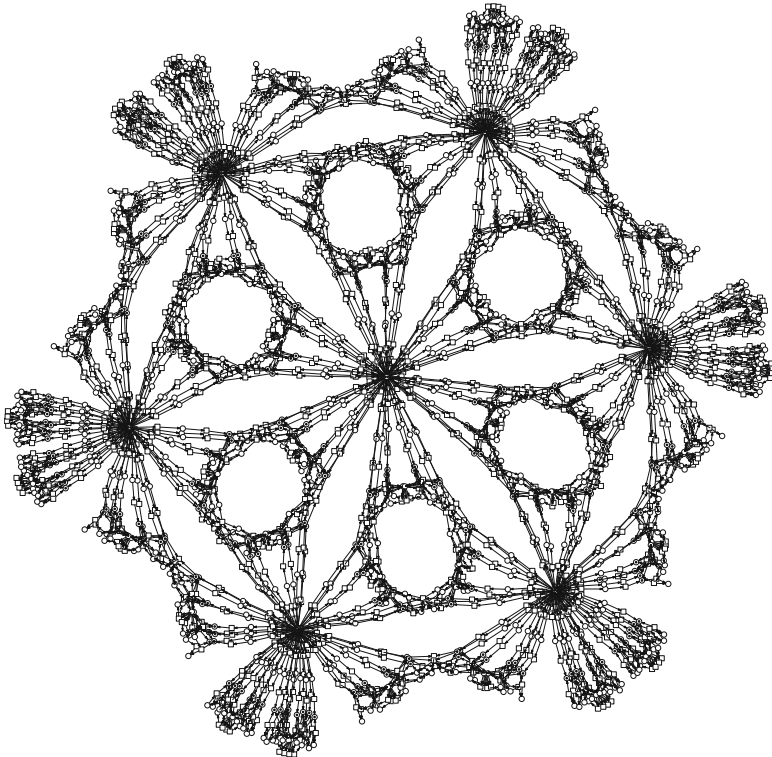
**Basics** Coloured Petri nets offer a modelling concept entirely orthogonal to the engineering principles of compositionality and hierarchical structuring. They can be considered to be a shorthand notation for (plain) Petri nets. The reduction in the visible model size is achieved by folding similar net components into one component; the different components are then technically distinguished by coloured tokens (instead of simply having black tokens). This just requires us to borrow a few concepts from standard high-level programming languages. Consequently, modelling with coloured Petri nets requires some basic programming skills, too.

Coloured Petri nets can be automatically unfolded into the underlying (plain) Petri nets, as long as we confine ourselves to finite colour sets. In return for this concession, we can do everything with coloured Petri nets (animation, simulation, analysis) that we are able to do with (plain) Petri nets, a compromise that is paid off over the long term.

**What for** From a practical point of view, the capabilities of coloured Petri nets exceed by far those of (plain) Petri nets. Coloured Petri nets can cope with large biological models, which could hardly be handled by Petri nets without colour. For



**Fig. 5** Planar Cell Polarity in *Drosophila*, from left to right: wing, wing tissue, cell arrangement, inter- and intracellular signalling cartoon; reproduced from [9]



**Fig. 6** Wallpaper-sized Petri net, obtained by unfolding a coloured Petri net modelling molecular interactions between seven neighbouring cells of a fly wing tissue; by courtesy of Q. Gao; for more details see [3]

instance, to explore planar cell polarity in fly wing tissue, we developed a coloured Petri net modelling the molecular interactions between neighbouring cells [3]; see also Fig. 5. This model can be configured by the number of cells the tissue shall be made of. This could be seen as a nice and flexible wallpaper generator, see Fig. 6; e.g. a tissue comprising 400 cells yields by unfolding an uncoloured Petri net consisting of 164,000 places and 229,669 transitions.

Likewise, colour permits us to encode spatial attributes, and thus opens the door to multilevel modelling. Spatial grid structures of arbitrary dimension and neighbourhood relations can be prepared and later used off the shelf in different settings; e.g. to explore phase variation in multistrain bacterial cell colonies [5]. Assuming a disc-like universe for our growing colonies and choosing a resolution of  $101 \times 101$  for the corresponding 2D grid generates by unfolding a Petri net comprising 30,603 places and 362,404 transitions.

In any case, crucial points advocating the use of coloured Petri nets for our application scenarios are the modelling comfort in describing structural regularities and their scalability, which permits convenient model configuration while preserving the power of all Petri net techniques to explore the model behaviour.

### 3 The Big Pros

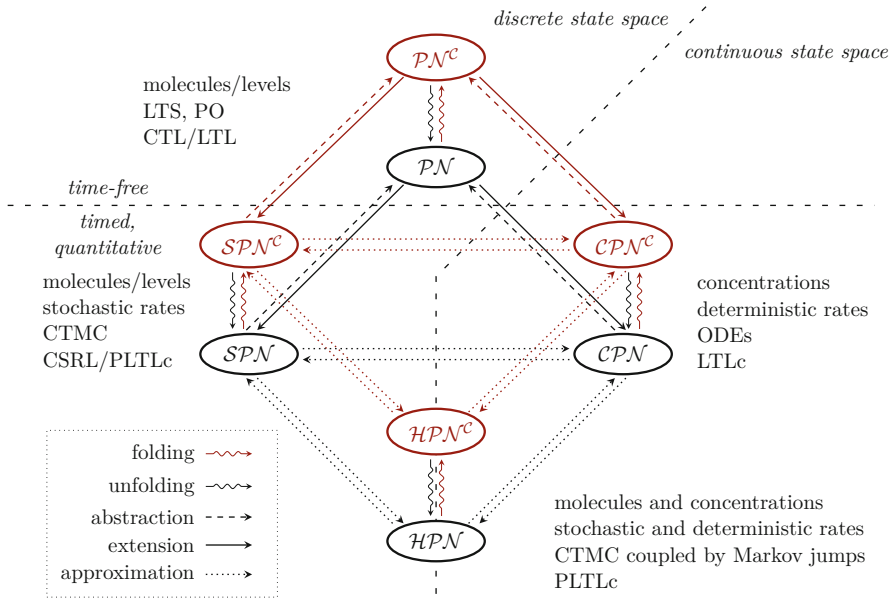
Looking through my Petri net glasses I can see a lot of very good reasons to adopt Petri net technologies for BioModel Engineering. They just simplify our life. I compress them here into two categories.

**Bridging Gaps** Representation style is often a matter of taste. Which kind of representation best serves its purpose and is considered to be most appropriate may depend on many aspects, one of them simply what one is used to. This holds for biochemical network representations, too.

However, systems biology is an inherently interdisciplinary field of research. Thus, communication means are crucial which are easily approachable by professionals having quite diverse backgrounds, while not being specifically addicted to mathematical notations. See [22] for a recent paper with a wide spectrum of authors that would not have been possible without Petri nets and their intuitive graphical approach.

**Unifying Diversity** Having agreed upon the model structure, the next step usually consists in getting the time-dependent behaviour right. This can basically be done in two different ways—stochastically or continuously. Stochastic Petri Nets ( $SPN$ ) seem to be a natural choice, as the behaviour of biochemical networks is inherently governed by stochastic laws. However, if molecules are in high numbers and stochastic effects can be neglected, one can easily switch to a deterministic setting and read the given net structure and its attached kinetics as a Continuous Petri Net ( $CPN$ ). Combining both yields Hybrid Petri Nets ( $HPN$ ). In summary, Petri nets may serve as a kind of umbrella formalism—the models share structure, but vary in their kinetic details. We obtain a family of related models with high analytical power [2, 8]; compare Fig. 7.

So there is no reason to be afraid of ordinary differential equations (ODEs); they are automatically generated out of the given  $CPN$  [4]. Likewise, our Petri



**Fig. 7** The unifying Petri net framework comprising a family of related models:  $\mathcal{PN}$ ,  $\mathcal{SPN}$ ,  $\mathcal{CPN}$ ,  $\mathcal{HPN}$  and their coloured counterparts; adapted from [11], with kind permission of Springer Science + Business Media B.V., Fig. 1, p. 399

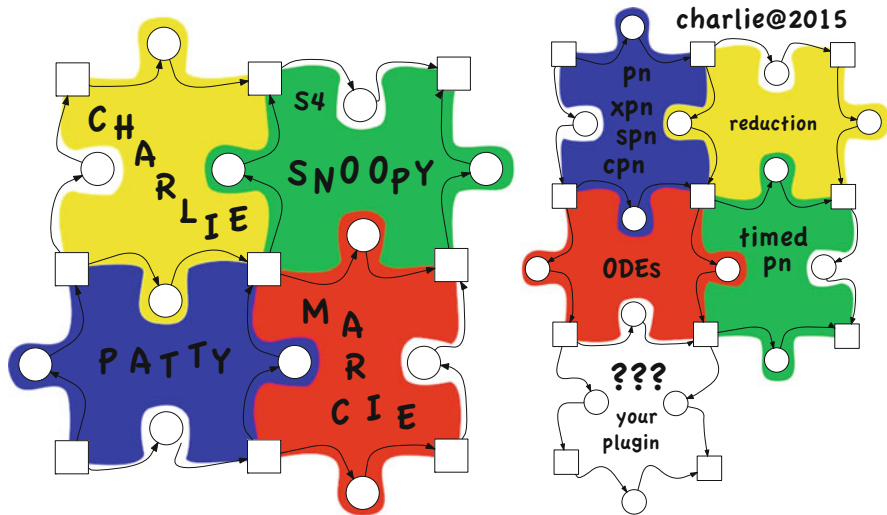
net technology allows us to describe and analyse systems that are elsewhere treated as partial differential equations (PDEs) [17].

## 4 Nothing Without the Right Tools

Having spent I-don't-know-how-many nights drawing Petri nets with pen and ink in my early technical reports, Ph.D. thesis and papers, I was desperate for computer support, so desperate that we started in the early 1990s to develop our own tools and continue to do so.

Long-term software development comes with its own challenges. It's pretty easy to write a simple Petri net editor and animator, able to deal with a few tens of nodes. It's getting more tricky, and thus more interesting from a software engineering point of view, to construct software able to handle up to millions of nodes, while supporting the Petri net design principles sketched above. But to explain this appropriately would make a paper in its own right.

Currently, the main components of our toolkit include *Snoopy*—a Petri net editor, animator and simulator [11], *Charlie*—standard analysis methods of Petri net theory, complemented by explicit model checking [13], *Marcie*—symbolic exact analysis and model checking for  $\mathcal{QPN}$  and  $\mathcal{SPN}$ , as well as approximative and



**Fig. 8** (left) Our Petri net toolkit, comprising a family of close friends, (right) Some of Charlie's components. All tools available at <https://www-dssz.informatik.tu-cottbus.de/DSSZ/Software>

simulative analysis for  $\mathcal{SPN}$  [12], and the tiny *Patty*—for web-based Petri net animation; see Fig. 8 for a quick overview.

By the way, if you want to give it a try by playing the token game yourself, just go to <https://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Sampler>. There you will find *Patty*, a Petri net animator directly executable within any standard web browser; nothing needs to be downloaded or installed.

## 5 Looking Forward

The technologies I sketched above still face many challenges, but their potential is breathtaking. Let me illustrate this by just one example—personalised medicine (also known as precision medicine).

Everybody is familiar with smart cards, one popular application scenario of which is health insurance cards. So far, they just store general patient data, such as name, address, insurance company etc. But they could easily carry any kind of patient-specific data, including—among others—the personal gene profile, which then would allow a GP, assisted by then state-of-the art computing techniques, to automatically generate individualised biochemical networks permitting model-based recommendations for the best diet, drug dosage or therapy. Some readers might find this threatening. But actually, I don't care, as long as my personal biochemical networks will be represented and interpreted as Petri nets.

**Acknowledgements** I owe a lot of credit to many colleagues for their inspiration and support throughout the years; these include above all David Gilbert, professor of computing in the synthetic biology group at Brunel University, London, and Wolfgang Marwan, professor of regulatory biology at Otto-von-Guericke University, Magdeburg.

Software tools don't develop themselves; my gratitude goes to all current and previous students and staff members who contributed to our toolkit now enjoying worldwide use.

## References

1. M. Blätke, A. Dittrich, C. Rohr, M. Heiner, F. Schaper, W. Marwan, JAK/STAT signalling - an executable model assembled from molecule-centred modules demonstrating a module-oriented database concept for systems and synthetic biology. *Mol. BioSyst.* **9**(6), 1290–1307 (2013). <http://dx.doi.org/10.1039/C3MB25593J>
2. M. Blätke, M. Heiner, W. Marwan, BioModel engineering with Petri nets, in *Algebraic and Discrete Mathematical Models for Modern Biology*, chap. 7, ed. by R. Robeva (Elsevier, Amsterdam, 2015), pp. 141–193. <http://www.sciencedirect.com/science/article/pii/B9780128012130000071>
3. Q. Gao, D. Gilbert, M. Heiner, F. Liu, D. Maccagnola, D. Tree, Multiscale modelling and analysis of planar cell polarity in the *Drosophila* wing. *IEEE/ACM Trans. Comput. Biol. Bioinform.* **10**(2), 337–351 (2013). <http://dl.acm.org/citation.cfm?id=2516526>
4. D. Gilbert, M. Heiner, From Petri nets to differential equations - an integrative approach for biochemical network analysis, in *Proc. ICATPN 2006*, ed. by S. Donatelli, P.S. Thiagarajan. *Lecture Notes in Computer Science*, vol. 4024 (Springer, Berlin, 2006), pp. 181–200. <http://www.springerlink.com/content/18516147038t012j/>
5. D. Gilbert, M. Heiner, F. Liu, N. Saunders, Colouring space - a coloured framework for spatial modelling in systems biology, in *Proceedings of the PETRI NETS 2013*, ed. by J. Colom, J. Desel. *Lecture Notes in Computer Science*, vol. 7927 (Springer, Berlin, 2013), pp. 230–249. [http://link.springer.com/chapter/10.1007%2F978-3-642-38697-8\\_13](http://link.springer.com/chapter/10.1007%2F978-3-642-38697-8_13)
6. M. Heiner, A Contribution to Deadlock Analysis Based on a Language-guided Programming Methodology (in German). Ph.D. thesis, Technical University Dresden, CS Department, 1980
7. M. Heiner, Application of Petri nets to metabolic networks (in German) (August 1998). [http://www-dssz.informatik.tu-cottbus.de/publications/slides/1998\\_HUB\\_metabol\\_pn\\_dt.sld\\_2up.pdf](http://www-dssz.informatik.tu-cottbus.de/publications/slides/1998_HUB_metabol_pn_dt.sld_2up.pdf)
8. M. Heiner, D. Gilbert, How might Petri nets enhance your systems biology toolkit, in *Applications and Theory of Petri Nets*, ed. by L.M. Kristensen, L. Petrucci. *Lecture Notes in Computer Science*, vol. 6709 (Springer, Berlin, 2011), pp. 17–37. <http://www.springerlink.com/content/t21n70p613h75m64/>
9. M. Heiner, D. Gilbert, Biomodel engineering for multiscale systems biology. *Progr. Biophys. Mol. Biol.* **111**(2–3), 119–128 (2013). <http://www.sciencedirect.com/science/article/pii/S0079610712001071?v=s5>
10. M. Heiner, R. Heinrich, Metabolic Petri Nets (October 2000). [http://www-dssz.informatik.tu-cottbus.de/publications/slides/2000\\_dfg-treffen\\_metabol\\_pn.sld\\_2up.pdf](http://www-dssz.informatik.tu-cottbus.de/publications/slides/2000_dfg-treffen_metabol_pn.sld_2up.pdf)
11. M. Heiner, M. Herajy, F. Liu, C. Rohr, M. Schwarick, Snoopy - a unifying Petri net tool, in *Application and Theory of Petri Nets Proc. PETRI NETS 2012*, ed. by S. Maddad, L. Pomello. *Lecture Notes in Computer Science*, vol. 7347 (Springer, Berlin, 2012), pp. 398–407. <http://www.springerlink.com/content/127w488386w03m45/>
12. M. Heiner, C. Rohr, M. Schwarick, MARCIE - Model checking And Reachability analysis done effiCIently, in *Proc. PETRI NETS 2013*, ed. by J. Colom, J. Desel. *Lecture Notes in Computer Science*, vol. 7927 (Springer, Berlin, 2013), pp. 389–399. [http://link.springer.com/chapter/10.1007/978-3-642-38697-8\\_21](http://link.springer.com/chapter/10.1007/978-3-642-38697-8_21)

13. M. Heiner, M. Schwarick, J. Wegener, Charlie - an extensible Petri net analysis tool, in *Proc. PETRI NETS 2015*, ed. by R. Devillers, A. Valmari. Lecture Notes in Computer Science, vol. 9115 (Springer, Berlin, 2015), pp. 200–211. [http://link.springer.com/chapter/10.1007/978-3-319-19488-2\\_10](http://link.springer.com/chapter/10.1007/978-3-319-19488-2_10)
14. O. Herzog, Analysing the control structure of parallel programs using Petri nets (in German). Ph.D. thesis, University of Dortmund, CS Department: Report 24/76, 1976
15. R. Hofestädt, A Petri net application to model metabolic processes. *J. Syst. Anal. Model. Simul.* **16**(2), 113–122 (1994). <http://dl.acm.org/citation.cfm?id=195799>
16. K. Lautenbach, Exact Liveness Conditions of a Petri Net Class (in German). Ph.D. thesis, GMD Report 82, Bonn, 1973
17. F. Liu, M. Blätke, M. Heiner, M. Yang, Modelling and simulating reaction-diffusion systems using coloured Petri nets. *Comput. Biol. Med.* **53**, 297–308 (2014). <http://www.sciencedirect.com/science/article/pii/S0010482514001693>
18. T. Murata, Petri nets: properties, analysis and applications. *Proc. IEEE* **77**(4), 541–580 (1989)
19. J. Peterson, *Petri Net Theory and the Modeling of Systems* (Prentice Hall PTR, Upper Saddle River, 1981)
20. C.A. Petri, Interpretations of net theory. Tech. Rep. 75-07, GMD, Internal Report, 1976, 2nd improved edition
21. V. Reddy, M. Mavrovouniotis, M. Liebman, Petri net representations in metabolic pathways, in *Proc. of the Int. Conf. on Intelligent Systems for Molecular Biology (ISMB 93)* (AAAI, Palo Alto, 1993), pp. 328–336. <http://www.aaai.org/Papers/ISMB/1993/ISMB93-038.pdf>
22. J. Somekh, M. Peleg, A. Eran, I. Koren, A. Feiglin, A. Demishtein, R. Shiloh, M. Heiner, S. Kong, Z. Elazar, I. Kohane, A model-driven methodology for exploring complex disease comorbidities applied to autism spectrum disorder and inflammatory bowel disease. *Biomed. Inf.* **63**, 366–378 (2016). <http://www.sciencedirect.com/science/article/pii/S1532046416300855>



# Petri Nets in Systems Biology: Transition Invariants, Maximal Common Transition Sets, Transition Clusters, Mauritius Maps, and MonaLisa



Ina Koch

## 1 Preface

The first time I met Carl Adam Petri was in January 1992 in St. Augustin close to Bonn—at that time still the capital of Germany. Our small computational biology group, later called the bioinformatics group, moved to the former GMD (“Gesellschaft für Mathematik und Datenverarbeitung mbH”). We came from the Institute of Cybernetics and Information Processes of the Academy of Sciences of the former GDR (German Democratic Republic) in East Berlin. Under the terms of the unification treaty, our institute was closed. Our group was offered the chance to proceed with our research at the GMD in St. Augustin. At that time, we were working on secondary structure prediction of proteins using graph-theoretic methods.

Arriving at the GMD, we were warmly welcomed by the colleagues of the Petri institute. Carl Adam Petri was already an emeritus professor, but was still doing research. His presence in the institute was indicated by a fuming door; he was sitting behind it in a cold room with opened windows, wearing outdoor clothes, smoking, and, of course, working. His group had the nice tradition of meeting in the library after lunch for tea and coffee. These meetings were not only relaxing, but also informative and inspiring. I remember that we discussed protein structures and biology without thinking about an application of Petri net theory to model biochemical systems. Carl Adam Petri was interested in all topics and discussed every point in a lively manner.

---

I. Koch (✉)

Molecular Bioinformatics, Institute of Computer Science, Faculty of Computer Science and Mathematics and Cluster of Excellence ‘Macromolecular Complexes’, Johann Wolfgang Goethe-University, Frankfurt am Main, Germany  
e-mail: [ina.koch@bioinformatik.uni-frankfurt.de](mailto:ina.koch@bioinformatik.uni-frankfurt.de)

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,  
[https://doi.org/10.1007/978-3-319-96154-5\\_26](https://doi.org/10.1007/978-3-319-96154-5_26)

217

Nearly seven years later, when I was working in the group of Jens Reich at the MDC (Max Delbrueck Center for Molecular Medicine) in Berlin-Buch, it was Stefan Schuster who gave me the paper by Reddy et al. [1], after my talk about application of graph theory to protein structure analysis. I read the paper with interest and discussed it with colleagues, among them Stefan Schuster, Falk Schreiber, and Monika Heiner. In particular, the concept of invariants and its relation to the concept of elementary flux modes or elementary modes (EMs) defined by Schuster for metabolic systems [2] formed one focus of our work. We were able to show that minimal semi-positive transition invariants (TIs) defined by Lautenbach in 1973 [3] correspond to elementary modes [4]. We started to apply Petri nets, first to model only metabolic pathways, and obtained interesting results. We illustrated dependencies of EMs or TIs, respectively, for the central carbon metabolism of potato tubers [5]. Moreover, we could easily find a modeling error in the underlying kinetic model as first formulated, which did not converge. We extended the application of PNs and started to consider also signal transduction systems (e.g., [6]) and gene regulatory systems (e.g., [7]).

In the following, we will consider molecular or biochemical Petri nets, discuss the special importance of minimal semi-positive transition invariants, define maximal common transition sets, and introduce transition clusters, Mauritius maps, and MonaLisa as software tools. At the end, we will give an overview of further developments and challenges.

## 2 Molecular Petri Nets

Now, we introduce the basic definitions that are necessary to follow the paper. We use the standard Petri net notations according to Reisig [8] and Koch et al. [9].

A *Petri net (PN)* is a bipartite, directed graph. A Petri net consists of two types of nodes, *places*  $P = \{p_1, \dots, p_k\}$  and *transitions*  $T = \{t_1, \dots, t_l\}$ . Places, drawn as circles, model passive system elements such as conditions, states, or biological species (or chemical compounds, such as proteins, RNA, DNA, and protein complexes). Transitions, drawn as squares or rectangles, stand for active system elements, such as events or chemical reactions, e.g., de-/phosphorylations and complex formation/degradation. Nodes of different type are connected by directed, positively weighted arcs. The arcs describe the relation between active and passive elements. *Pre-places* of a transition are those places that are connected by arcs which start at these places and end at the considered transition. *Post-places* of a transition are those places that are connected by arcs which start at the considered transition and end at these places. Analogously, we define *pre-transitions* and *post-transitions* of a place. A transition without pre-places (post-places) is called an *input (output)* transition, and is often drawn as a rectangle.

Transitions describe events. An event takes place if its pre-conditions represented by the pre-places and post-conditions represented by its post-places are fulfilled. The fulfillment of the pre-conditions is realized via movable objects called *tokens*,

which are located on the pre-places. Tokens represent a discrete amount, e.g., one molecule or one mole of a chemical compound. The distribution of tokens over all places describes a certain *system state* and is called a *marking*,  $m$ . The *initial marking*,  $m_0$ , defines the system state before any firing takes place. In this paper, the post-conditions are always fulfilled, because we do not restrict the number of tokens on a place.

The *firing rule* defines the dynamic behavior of a PN. It determines under which conditions and how the event takes place. We consider discrete place/transition nets without any time properties, i.e., the firing rule is *timeless*. If the preconditions of a transition are fulfilled, i.e., each pre-place carries at least as many tokens as indicated by the incoming arc weight, the transition has *concession* or is *activated* or *enabled*, and can *fire*. Then, the required number of tokens is consumed on the pre-places, and as many tokens are produced on the post-places as dictated by the outgoing arc weights. This takes place instantaneously since the firing rule is timeless. If tokens on the pre-places are not consumed during the firing, but are required to enable the transition, we use bidirectional arcs called *read arcs* or *test arcs*. In biochemical systems, we model catalytic activities using read arcs.

### 3 Transition Invariants

The *incidence matrix*  $C$  of a given PN is a  $(k \times l)$  matrix, where  $k$  is the number of places and  $l$  is the number of transitions. Every matrix element,  $C_{i,j}$ , corresponds to the number of tokens changed on place  $p_i$  by firing of transition  $t_j$ . Invariant properties of the net hold independently of any firing in each system state. For chemical stoichiometric reaction systems in chemistry and for metabolic systems in biology, the incidence matrix is known as *stoichiometric matrix*. The stoichiometric factors in the chemical stoichiometric equations correspond to the integer numbers of the entries of the stoichiometric matrix. For a PN of a metabolic system, these stoichiometric factors define the arc weights for incoming and outgoing arcs of the transitions.

Based on the incidence matrix we define *transition invariants* (TIs) and *place invariants* (PIs) [3]. A TI is a vector,  $\mathbf{x} \geq \mathbf{0}$ ,  $\mathbf{x} \in \mathbb{N}'_0$ , that satisfies the equation  $C \cdot \mathbf{x} = \mathbf{0}$ , representing a multiset of transitions. The firing of these transitions reproduces a given marking. The vector,  $\mathbf{x}$ , defines a *Parikh vector*, which indicates the firing frequency for each transition in the TI to reproduce a given marking.

In biology, the invariant condition expressed by the equation  $C \cdot \mathbf{x} = \mathbf{0}$  corresponds to the *steady state* of a metabolic system. A steady state, also called steady-state equilibrium or flux balance, is a stationary state, in which the substances, coming into the system and going out of the system, are balanced, such that their weighted sums remain constant during the considered time. Interestingly, in 1994, nearly 20 years later, the biological meaning of TIs for metabolic networks was proven by Stefan Schuster who defined *elementary modes* (EMs). His approach is based on the analysis of a pointed convex cone [2], where vectors inside the cone

and on the cone's surface correspond to invariants. Mathematically, the convex cone analysis can be mapped to integer linear programming based on the fundamental theorem of linear inequalities. Schuster showed that EMs can be interpreted as minimal functional units at steady state in metabolic systems. The concept of EMs is well established and widely used in the field of systems biology. Using EMs, new pathways have been predicted that were later validated experimentally, for example the glyoxylate pathway, which was theoretically predicted by Liao et al. [10] and Schuster et al. [11] and later experimentally proven in hungry *Escherichia coli* [12].

For the sake of completeness, we define a place invariant, PI, as a vector  $\mathbf{y} \geq 0$ ,  $\mathbf{y} \in \mathbb{N}^k_0$ , that satisfies the equation  $C^T \cdot \mathbf{y} = \mathbf{0}$ . It stands for a set of places over which the weighted sum of tokens is always constant. Thus, PIs indicate substance conservations of the system. This property is also used in biology, but will not be considered here in the following.

The set of non-zero elements of an invariant  $\mathbf{u}$  is called the *support* of  $\mathbf{u}$ , written as  $\text{supp}(\mathbf{u})$ . An invariant  $\mathbf{u}$  is called *minimal*, if its support  $\text{supp}(\mathbf{u})$  does not contain the support of any other invariant  $\mathbf{z}$ , i.e., there does not exist an invariant  $\mathbf{z}$ , such that  $\text{supp}(\mathbf{z}) \subseteq \text{supp}(\mathbf{u})$  and the largest common divisor of all non-zero entries of  $\mathbf{u}$  is equal to one. Linear combinations of invariants again give invariants. In the following we consider always minimal, semi-positive TIs writing only *TIs*.

A net *covered by TIs* is called *CTI* if every transition participates in at least one TI. A TI defines a subnet, consisting of its support, its pre- and post-places, and all arcs in between. A *trivial TI* describes, for example, reversible reactions. It consists of two transitions, representing the forward and backward reaction. Typically, trivial TIs occur in metabolic networks.

To verify a molecular PN model, we check whether it is connected and CTI. A biochemical PN should be CTI because a transition which is not contained in any TI does not contribute to the system's behavior. Moreover, in a biological PN, each invariant should be biologically reasonable. Otherwise, in most cases, a modeling error has occurred. This makes the CTI property essential during the modeling process.

## 4 Maximal Common Transition Sets and Transition Clusters

The number of TIs increases exponentially with larger and more complex networks. Even for networks with hundreds of nodes and arcs, the computation of TIs becomes not only very expensive in time and space, but hundreds and thousands of TIs are generated, making an exploration of TIs for their biological meaning very difficult and often not manageable. One way to handle large numbers of TIs is their classification and further decomposition. For this purpose, we defined *maximal common transition sets (MCT-sets)* and *transition clusters (T-Clusters)*.

To define MCT-sets, let us consider a PN with  $n$  TIs, and let  $X$  denote the set of all TIs. A transition set  $A \subseteq T$  is called an MCT-set if and only if for each TI  $\mathbf{x} \in X : A \subseteq \text{supp}(\mathbf{x})$  or  $A \cap \text{supp}(\mathbf{x}) = \emptyset$ . Thus, two transitions  $t_i$  and  $t_j$  belong to

the same MCT-set if and only if they participate in exactly the same TIs, i.e., the transitions of an MCT-set always occur together in the same TIs and do not belong to any other TI, for all  $i, j \in \{1, \dots, n\}$ ,  $t_i$  and  $t_j$  correspond to the same MCT-set if and only if for each TI  $\mathbf{x} \in X$ ,  $t_i \in \text{supp}(\mathbf{x}) \iff t_j \in \text{supp}(\mathbf{x})$ .

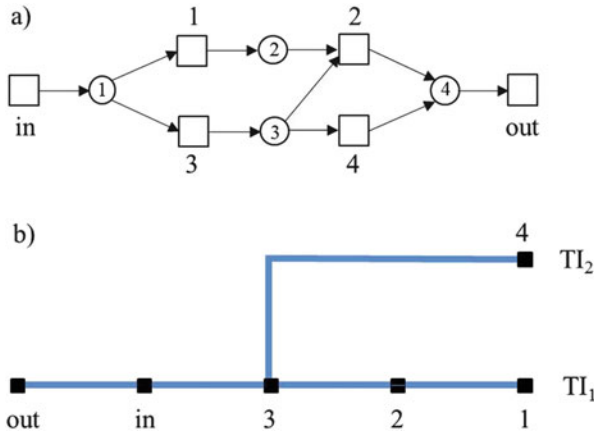
This dependency relation leads to maximal common sets of transitions. An MCT-set defines a disjunctive subnet, which is not necessarily connected. These subnets represent a possible structural decomposition of large biochemical networks into rather small subnets. MCT-sets can be interpreted as smallest biologically meaningful, functional units. They decompose a network into nonintersecting **building blocks** [6].

A transition that models a biochemical reaction is often represented by an enzyme, i.e., a certain protein that catalyzes a chemical reaction. Since a protein can exhibit more than one function, a transition can be part of two or more functional building blocks. Using MCT-sets, the transition would belong to no more than one such building block, because MCT-sets do not overlap. To address the biological reality, we apply clustering techniques to define overlapping subnetworks. We cluster TIs using the well-known UPGMA algorithm. We use a normalized, simple distance measure, the Tanimoto coefficient, which is known from drug design. It is based on the support vector (e.g., [13]). For TIs  $\mathbf{x}_i$  and  $\mathbf{x}_j$ , we define the pairwise similarity  $s_{ij}$  as the ratio of the number of shared transitions and the number of all transitions. A pairwise similarity  $s_{ij}$  can easily be transformed into a distance  $d_{ij} = 1 - s_{ij}$ . For a detailed description see Grafahrend-Belau et al. [14].

## 5 Mauritius Maps and Knockout Matrices

Biochemical reactions (transitions in PN models) often model the protein or the gene that *expresses* that protein, i.e., the gene is *switched on* and protein production starts. In experimental biochemistry and biology, it is common to remove a single gene or more genes, i.e., to knock them out, and to check the behavior of the knocked-out system. This technique, called **knockout analysis**, has been systematically applied to better understand the system's behavior. In a mathematical model, **in silico knockouts** can easily be performed, such that, for example, certain experiments can *a priori* be excluded. Thus, *in silico* knockout analysis is very useful in planning experiments. In this context, it is of great interest to explore also *in silico* the dependencies between TIs.

**Mauritius maps (MMs)** have been developed to visualize dependencies between TIs and the knockout behavior of a biological PN model. We consider the supports of TIs and define an MM as a finite, binary tree. A node represents a transition that belongs to a TI. The root node is located in the lower left corner. A horizontal arc connects nodes of the same TI. A branch in the tree indicates another TI that shares the nodes (transitions) of the left part of the horizontal arc, see Fig. 1. Vertical arcs connect nodes of the left subtree with nodes of the right upper subtree of the same TI.



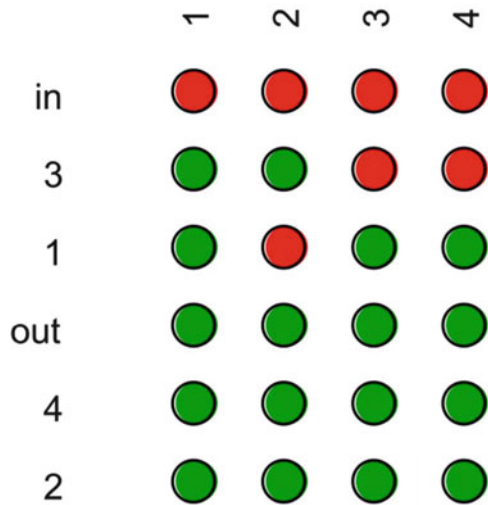
**Fig. 1** A Petri net and its Mauritius map. (a) The Petri net, consisting of six transitions and four places. The PN exhibits two TIs, whose supports are  $\text{supp}(TI_1) = \{\text{out}, \text{in}, 3, 2, 1\}$  and  $\text{supp}(TI_2) = \{\text{out}, \text{in}, 3, 4\}$ . (b) The Mauritius map of the PN in (a). Nodes indicate transitions. Transition *out* represents the root node. The horizontal arcs connect nodes of the same TI, here, for example, the nodes *out*, *in*, *3*, *2*, and *1*. A branch in the tree indicates another TI that shares the nodes of the left part of the horizontal arc. Vertical arcs connect nodes of the left subtree with nodes of the right upper subtree of the same TI, here the nodes *out*, *in*, *3*, and *4*

Starting with the root, the horizontal line until the first branch is labeled by the transitions that are part of every TI and thus, can be considered to be the **most important transitions** of the PN. The protein or gene represented by such a transition would have the highest impact if it would be knocked out. For the PN in Fig. 1, the transitions *in*, *out*, and *3* are then considered to be most important.

The impact of a knockout of a transition can be measured by the number of affected, i.e., destroyed, TIs. Thus, one part of the system remains active and another part loses its biological function. The knockout of a single transition affects all TIs (pathways) that are described by the corresponding right subtree. The knockout of a transition fragments a net covered by TIs into two subnets. One subnet (left child) does not contain the transition knocked out, and represents the function of the model not affected by the knockout. The second subnet (right child) depends on the presence of the transition knocked out. Thus, only those TIs (pathways) that cover the left child and its successors are not affected, maintaining their biological functionality. These subsets of TIs with the corresponding places in between represent sub-PN models, which are, in turn, CTI.

We represent the results of the knockout analysis in a color-coded **knockout matrix**, see Fig. 2. The rows indicate the transitions to be knocked out and the columns the places that are affected (colored red) or not affected (colored green) by the knockout of the corresponding transition [15].

**Fig. 2** The knockout matrix of the PN model in Fig. 1. The transitions (proteins or genes) to be knocked out are shown vertically, and the places (chemical compounds) that are affected (red) or not affected (green) by the knockout of the corresponding transition are shown horizontally



## 6 MonaLisa

MonaLisa is an open-source software tool for creation, visualization, and analysis of PNs that we developed especially for applications to biochemical PNs. It implements a PN editor with numerous functions for creating, removing, moving, zooming, coloring, and labeling of objects as well as several network analysis techniques, such as invariant analysis (implemented in C, see [16]), including the graphical visualization of the resulting functional modules, general topology features, e.g., the distribution of node degrees and cluster coefficients, knockout analysis (single and multiple knockouts), and others, all without any knowledge of kinetic parameters, such as substance concentrations or reaction rates. The analysis methods focus on decomposition methods to identify functional modules at steady state. Besides network visualization, editing, and analysis, the software enables a visual inspection of the analysis results [17].

Moreover, MonaLisa provides stochastic simulation abilities. The intuitive graphical user interface allows the user to focus on modeling and simulation at different scales. Constant places and the usage of mathematical expressions for describing simulation parameters offer improved flexibility compared to other tools, allowing for modeling of non-standard kinetics and complex relationships to the external environment. Useful features, such as built-in plotting, navigation through the simulation history, export of the simulation setups to XML-files, and setting the seed of the random number generator, can help to easily adjust parameters to follow and assess simulation results [18].

MonaLisa implements interfaces to many tools in systems biology, the Petri net world, and graph theory, supporting a broad range of file formats, such as PNML (Petri Net Markup Language), PNT (Petri Net Technology), SPPED, SBML

(Systems Biology Markup Language), KGML (KEGG Markup Language), DAT (Metatool format), and the image file formats Portable Network Graphics (PNG) and Scalable Vector Graphics (SVG). MonaLisa provides the possibility to easily extend its functionality by new modules. It also supports annotation with MIRIAM identifiers and SBO terms [19].

The software is licensed under Artistic License 2.0. It is freely available at <http://www.bioinformatik.uni-frankfurt.de/software.html>. It requires at least Java 6 and runs under Linux, Microsoft Windows, and Mac OS.

## 7 Summary

I'm not sure whether I would have began to use Petri nets for modeling biochemical systems without meeting Carl Adam Petri. His personality, his curiosity, and his ideas inspired our work. Meanwhile, Petri nets represent a well-established approach in the systems biology community. In the last 15 years many papers has been published on applications of Petri nets to biology, e.g., Matsuno et al. [20], Hardy and Robillard [21], Peleg et al. [22], Koch and Chaouiya [23], Rodriguez et al. [24], Minervini et al. [25], and Scheidel et al. [26].

Until now, the work of our group was mainly focused on invariant analysis. We developed interesting extensions such as MCT-sets, T-clusters for network decomposition, and Mauritius maps to represent dependencies between invariants and knockout matrices. Recently, *Manatee* invariants were defined, especially for modeling signaling pathways by Amstein et al. [27]. All these concepts were inspired by questions that arose during modeling of biochemical systems. We have demonstrated that all these concepts can also be used for modeling signal transduction systems and gene regulatory systems. To make these new ideas attractive to users from biology and medicine and to combine them with a powerful editor and with topological network analysis, we developed the open-source software MonaLisa.

Nonetheless, there are still many open problems and challenges that are typical for modeling big biological systems based on incomplete data. The reachability analysis alone, which would be very useful in biology and medicine, is not manageable at the moment. To bring the ideas to biologists and physicians, there is a big need for the graphical representation of the results.

Based on various experiments at different scales, there exist several types of models of biochemical systems for modeling of metabolic systems, signal transduction pathways, and gene regulatory pathways. These models represent different levels of abstraction and exhibit different typical properties that can and should be addressed in the analysis techniques.



## References

1. V.N. Reddy, M.L. Mavrouniotis, M.N. Liebman, Petri net representations in metabolic pathways. *Proc. Int. Conf. Intell. Syst. Mol. Biol.* **1**, 328–336 (1993)
2. S. Schuster, C. Hilgetag, On elementary flux modes in biochemical reaction systems at steady state. *J. Biol. Syst.* **2**(2), 165–182 (1994)
3. K. Lautenbach, *Exact Liveness Conditions of a Petri Net Class*. GMD, Report 82, Bonn (in German) (1973)
4. S. Schuster, T. Pfeiffer, F. Moldenhauer, I. Koch, T. Dandekar, Exploring the pathway structure of metabolism: decomposition into subnetworks and application to, *Mycoplasma pneumoniae*. *Bioinformatics* **18**(2), 351–361 (2002)
5. I. Koch, B.H. Junker, M. Heiner, Application of Petri net theory for modelling and validation of the sucrose breakdown pathway in the potato tuber. *Bioinformatics* **21**(7), 1219–1226 (2005)
6. A. Sackmann, M. Heiner, I. Koch, Application of Petri net based analysis techniques to signal transduction pathways. *BMC Bioinform.* **4**(7), 482 (2006)
7. S. Grunwald, A. Speer, J. Ackermann, I. Koch, Petri net modelling of gene regulation of the Duchenne muscular dystrophy. *BioSystems* **92**(2), 189–205 (2008)
8. W. Reisig, *Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies* (Springer, Berlin, 2013)
9. I. Koch, W. Reisig, F. Schreiber, *Modeling in Systems Biology: The Petri Net Approach* (Springer, Berlin, 2011)
10. I.C. Liao, S.Y. Hou, Y. Chao, Pathway analysis, engineering, and physiological considerations for redirecting central metabolism. *Biotechnol. Bioeng.* **5**(1), 129–140 (1996)
11. S. Schuster, T. Dandekar, D.A. Fell, Detection of elementary flux modes in biochemical networks: a promising tool for pathway analysis and metabolic engineering. *Trends Biotechnol.* **17**(2), 53–60 (1999)
12. E. Fischer, U. Sauer, A novel metabolic cycle catalyzes glucose oxidation and anaplerosis in hungry *Escherichia coli*. *J. Biol. Chem.* **278**(47), 46446–46451 (2003)
13. D. Bajusz, A. RÁC, K. Héberger, Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? *J. Cheminform.* **7**, 20 (2015)
14. E. Grafarend-Belau, F. Schreiber, M. Heiner, A. Sackmann, B.H. Junker, G. Stefanie, S. Astrid, K. Winder, J. Koch, Modularisation of biochemical networks based on classification of Petri net t-invariants. *BMC Bioinform.* **9**, 90 (2008)
15. J. Scheidel, L. Amstein, J. Ackermann, I. Dikic, I. Koch, *In silico* knockout studies of Xenophagic capturing of *Salmonella*. *PLoS Comput. Biol.* **12**(12), e1005200 (2016)
16. I. Koch, J. Ackermann, On functional module detection in metabolic networks. *Metabolites* **3**(3), 673–700 (2013)
17. J. Einloft, J. Ackermann, J. Nöthen, I. Koch, MonaLisa – visualization and analysis of functional modules in biochemical networks. *Bioinformatics* **29**(11), 1469–1470 (2013)
18. P. Balazki, K. Lindauer, J. Einloft, J. Ackermann, I. Koch, MONALISA for stochastic simulations of Petri net models of biochemical systems. *BMC Bioinform.* **16**, 215 (2015)
19. M. Courtot, N. Juty, C. Knüpfér, D. Waltham, A. Zhukova, A. Dräger, M. Dumontier, A. Finney, M. Golebiewski, J. Hastings, S. Hoops, S. Keating, D.B. Kell, S. Kerrien, J. Lawson, A. Lister, J. Lu, R. Machne, P. Mendes, M. Pocock, N. Rodriguez, A. Villeger, D.J. Wilkinson, T. Wimalaratne, C. Laibe, M. Hucka, N. Le Novère, Model storage, exchange and integration. *Mol. Syst. Biol.* **7**, 543 (2011)
20. H. Matsuno, A. Doi, M. Nagasaki, S. Miyano, Hybrid Petri net representation of gene regulatory network. *Proc. Pac. Symp. Biocomput.* **5**, 338–349 (2000)
21. S. Hardy, P.N. Robillard, Modelling and simulation of molecular biology systems using Petri nets: modelling goals of various approaches. *J. Bioinform. Comput. Biol.* **2**(4), 595–613 (2004)
22. M. Peleg, D. Rubin, R.B. Altman, Using Petri Net tools to study properties and dynamics of biological systems. *J. Am. Med. Inform. Assoc.* **12**(2), 181–199 (2005)

23. I. Koch, C. Chaouiya, Discrete modelling Petri net and logical approaches, in *Systems Biology for Signaling Networks*, ed. by S. Choi (Springer, New York, 2010), pp. 821–856
24. E.M. Rodriguez, A. Rudy, R.C. del Rosario, A.M. Vollmar, E.R. Mendoza, A discrete Petri net model for cephalostatin-induced apoptosis in leukemic cells. *Nat. Comput.* **10**(3), 993–1015 (2011)
25. G. Minervini, E. Panizzoni, M. Giollo, A. Masiero, C. Ferrari, S.C. Tosatto, Design and analysis of a Petri net model of the Von Hippel-Lindau (VHL) tumor suppressor interaction network. *PLoS ONE* **9**(6), 96986 (2014)
26. J. Scheidel, K. Lindauer, J. Ackermann, I. Koch, Quasi-steady-state analysis based on structural modules and timed Petri net predict system's dynamics: the life cycle of the insulin receptor. *Metabolites* **5**(4), 766–793 (2015)
26. L. Amstein, J. Ackermann, J. Scheidel, S. Fulda, I. Dikic, I. Koch, Manatee invariants reveal functional pathways in signaling networks. *BMC Syst. Biol.* **11**, 72 (2017)

# From Nets to Circuits and from Circuits to Nets



Jordi Cortadella

## 1 Introduction

It is a mere coincidence that Petri nets came on the scene the same year the author of this paper was born [15]. But it not a coincidence that they have had a strong relationship since the early 1990s. Petri nets have had widespread use in multiple areas where a computational model capable of expressing concurrency, causality and choice is required. One of these areas is Electronic Design Automation [2], since *hardware is inherently concurrent* [10]. Dataflow systems [11], communication protocols [8], hardware/software co-design [12] etc. are examples of domains where Petri nets have been used for specification, synthesis and verification of electronic systems.

This paper reviews the impact of Petri nets in one of the domains in which they have played a predominant role: asynchronous circuits. The paper also discusses challenges and topics of interest for the future.

## 2 Minimalist Petri Nets

When associating certain semantics with events, Petri nets inherit an interpretation that represents the functioning of a particular system. An event can symbolise the initiation of a task, the arrival of a message, the utilisation of a resource etc. Here is an interesting question:

What is the simplest interpretation one can conceive for a Petri net?

---

J. Cortadella (✉)

Department of Computer Science, Universitat Politècnica de Catalunya, Barcelona, Spain  
e-mail: [jordi.cortadella@upc.edu](mailto:jordi.cortadella@upc.edu)

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,  
[https://doi.org/10.1007/978-3-319-96154-5\\_27](https://doi.org/10.1007/978-3-319-96154-5_27)

227

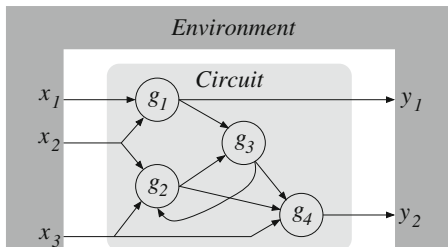


Fig. 1 Asynchronous circuit

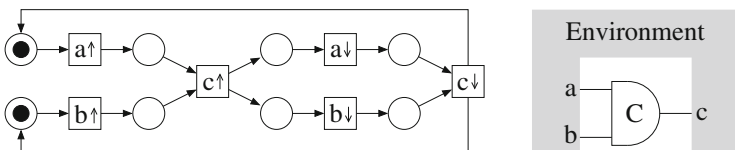


Fig. 2 Signal Transition Graph and implementation

If one bit is the minimum unit of information, changing the state of one bit can be considered the most elementary event. If each bit is implemented by an electronic signal, we end up with a circuit in which each signal may change its state asynchronously according to a certain behaviour. This is precisely what an *asynchronous circuit* is (see Fig. 1): a set of input signals  $(x_1, \dots, x_n)$ , a set of output signals  $(y_1, \dots, y_m)$ , a set of components (logic gates) and a protocol that specifies the interaction the circuit and its environment.

Signal Transition Graphs (STG) [17] are the *minimalist* version of Petri nets that can specify the behaviour of asynchronous circuits. Figure 2 presents an STG specifying a circuit with two input signals ( $a$  and  $b$ ) and one output signal ( $c$ ). The figure also depicts a possible circuit implementation using a Muller C-element [13]. More precisely, the circuit waits for signals  $a$  and  $b$  to go high (in any order). After that, the circuit generates a rising transition of  $c$ . Next, the circuit waits for  $a$  and  $b$  to go low and generates a falling transition of  $c$ . This is iteratively repeated forever.

Fortunately, the author of this paper had the pleasure to briefly talk about minimalist nets with Prof. Carl Adam Petri in 2003 (Eindhoven) and the impact of his contributions in the small community of asynchronous design: the term *Petri net* appears 44 times in the Asynchronous Bibliography<sup>1</sup> [14].

<sup>1</sup>Let us bear in mind that the Asynchronous Bibliography has not been updated since 2004.

### 3 Synthesis and Verification

The marriage between Petri nets and asynchronous circuits gives rise to interesting design automation problems, most of them related to logic synthesis and formal verification. State encoding and logic decomposition have been the most challenging knots in synthesis [5].

Formal verification is also an intricate problem [16] that is computationally expensive when dealing with timed circuits [3]. In this area, Petri net unfoldings have also played a fundamental role when dealing with timed circuits [18].

For most synthesis and verification problems, the reachability graph of the system needs to be generated and transformed. But a fundamental problem arises: the system is specified as a Petri net, but the transformations at the level of reachability graph (e.g. insertion of new events) cannot be observed as a Petri net unless some method exists to retrieve a Petri net from a reachability graph.

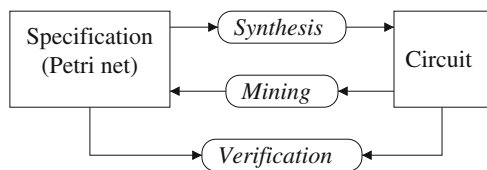
Here is where the theory of regions [9], by Ehrenfeucht and Rozenberg, plays a crucial role in this area. Visualising the transformations of a reachability graph with the same formal model with which it was generated is an extremely useful engine to gain intuition about the algorithms used for synthesis. In fact, the theory of regions was the main motivation for creating one of the state-of-the-art tools for synthesis of asynchronous circuits and Petri nets: *petrify* [4]. The tool not only synthesises circuits, but also Petri nets. For the authors of the tool, the term *petrifying means returning to the Petri net world* from the reachability graph. The tool is still actively used in the community of asynchronous circuit designers.

### 4 Mining: From Circuits to Nets

Figure 3 depicts the main problems that can be envisioned when relating specifications and implementations. In the previous section we discussed synthesis and verification. Process mining [19], also known as process discovery or specification mining, is becoming an area of growing interest in different domains. How can process mining help in asynchronous design?

An interesting problem to solve is as follows: given a circuit, is it possible to discover environments that can safely interact with it? Moreover, can we describe these environments using Petri nets? This is a kind of reverse engineering problem still in its infancy, although some initial effort has been recently undertaken [7].

**Fig. 3** Interesting problems in asynchronous design



Solving this problem might contribute to improve some of the tasks in circuit design and verification. For example, the discovered specifications might be used to re-synthesise and improve the quality of the circuits. Compositional verification might also benefit by substituting fragments of circuits with their mined specifications (potentially more abstract and simpler than the implementations).

In the area of mining, the recent work by Best and Devillers [1] may be extremely useful since it may help to characterise those environments that preserve certain properties (e.g. persistence) and can still be represented as a Petri net (e.g. a choice-free net). We envision more progress in this direction in the next few years.

## 5 The Challenge

In the long journey to introduce asynchronous design in industry, we have realised that most designers have an Electrical Engineering background, whereas Petri nets have been mainly used by an academic community with a Computer Science background. This creates a cultural gap that makes the adoption of this technology difficult, if not impossible in some cases.

The potential users of this technology are mostly familiar with finite-state machines, waveforms, schematics and HDLs such as Verilog or VHDL. One of the challenges for the future is to approach designers with a formalism that can have the expressive power of Petri nets (or some subclass of them) while exposing a friendly specification language similar to the formalisms typically used by electrical engineers.

Along this direction, *Waveform Transition Graphs* (WTG) [6] have recently been proposed as an alternative to STGs. In this formalism, the choice-free fragments of the behaviour are represented as waveforms, which are objects very well known by circuit designers. Additionally, concurrency and choice are mutually exclusive in such a way that choices can only be made when the system has sequential behaviour. Thus, the choice/join places mimic states of the system that glue the waveforms.

An example is shown in Fig. 4 where the left part represents an STG and the right part represents a WTG. The dotted arc in the STG is required to sacrifice some performance and prevent concurrency during the choice represented by  $s0$  in the WTG.

Petri nets have played a remarkable role in concurrent hardware and their computational model will prevail in the future. It is now time for academia to create bridges for engineers that enable a broader adoption of Petri nets so that the footprint of Carl Adam Petri's legacy becomes deeper in this area.

Long live Petri nets!

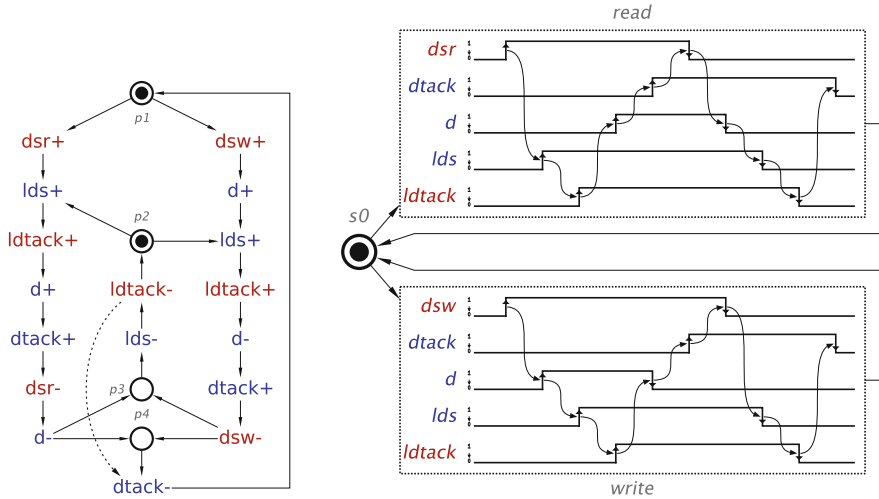


Fig. 4 STG (left) and WTG (right)

## References

1. E. Best, R. Devillers, Synthesis and reengineering of persistent systems. *Acta Inform.* **52**(1), 35–60 (2015)
2. R. Brayton, L.P. Carloni, A.L. Sangiovanni-Vincentelli, T. Villa, Design automation of electronic systems: past accomplishments and challenges ahead. *Proc. IEEE* **103**(11), 1952–1957 (2015).
3. R. Clarisó, J. Cortadella, Verification of timed circuits with symbolic delays, in *Proc. of Asia and South Pacific Design Automation Conference*, January (2004), pp. 628–633
4. J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Yakovlev, Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers. *IEICE Trans. Inf. Syst.* **E80-D**(3), 315–325 (1997)
5. J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, A. Yakovlev. *Logic Synthesis of Asynchronous Controllers and Interfaces* (Springer, Berlin, 2002)
6. J. Cortadella, A. Moreno, D. Sokolov, A. Yakovlev, D. Lloyd, Waveform transition graphs: a designer-friendly formalism for asynchronous behaviours, in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems* (IEEE, Los Alamitos, 2017), pp. 73–74
7. J. de San Pedro, T. Bourgeat, J. Cortadella, Specification mining of asynchronous controllers, in *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems* (IEEE, Los Alamitos, 2016), pp. 107–114
8. M. Diaz, Modeling and analysis of communication and cooperation protocols using Petri net based models. *Comput. Netw.* **6**(6), 419–441 (1982)
9. A. Ehrenfeucht, G. Rozenberg, Partial 2-structures; part I: Basic notions and the representation problem, and part II: State spaces of concurrent systems. *Acta Inform.* **27**(4), 315–368 (1990)
10. M. Kishinevsky, A. Kondratyev, A. Taubin, V. Varshavsky, *Concurrent Hardware: The Theory and Practice of Self-Timed Design*. Series in Parallel Computing (Wiley, New York, 1994)
11. E.A. Lee, D.G. Messerschmitt, Synchronous data flow. *Proc. IEEE* **75**(9), 1235–1245 (1987)
12. P. Maciel, E. Barros, W. Rosenstiel, A Petri net model for hardware/software codesign. *Des. Autom. Embed. Syst.* **4**(4), 243–310 (1999)

13. D.E. Muller, W.S. Bartky, A theory of asynchronous circuits, in *Proceedings of an International Symposium on the Theory of Switching* (Harvard University Press, Cambridge, 1959), pp. 204–243
14. A. Peeters, The ‘Asynchronous’ Bibliography (BIBTEX) database file async.bib. <http://www.win.tue.nl/async-bib/doc/async.bib>
15. C.A. Petri, Kommunikation mit Automaten. Dissertation, Schriften des IIM 2, Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn, Bonn, 1962
16. I. Poliakov, A. Mokhov, A. Rafiev, D. Sokolov, A. Yakovlev, Automated verification of asynchronous circuits using circuit Petri nets, in *2008 14th IEEE International Symposium on Asynchronous Circuits and Systems* (IEEE, Los Alamitos, 2008), pp. 161–170
17. L.Y. Rosenblum, A.V. Yakovlev, Signal graphs: from self-timed to timed ones, in *Proceedings of International Workshop on Timed Petri Nets*, Torino, July 1985 (IEEE Computer Society Press, Los Alamitos, 1985), pp. 199–207
18. A. Semenov, A. Yakovlev, Verification of asynchronous circuits using time Petri-net unfolding, in *Proc. ACM/IEEE Design Automation Conference* (IEEE, Los Alamitos, 1996), pp. 5–63
19. W.M.P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, 1st edn. (Springer, Berlin, 2011)



# Living Lattices



Alex Yakovlev

*Once upon a time there were lattices. They were nice, diamond-like, but they were too big, cumbersome and immobile . . . Then came Petri nets to capture them and to give them life! (from a hypothetical book “The Future Origin of Species”)*

## 1 Prologue

In the third year of our university life we were gradually being absorbed into research activities in our department (Department of Computer Engineering, Leningrad Electrical Engineering Institute, aka LETI—now called St. Petersburg State Electrotechnical University). That was a usual thing in the USSR, for students planning to continue their studies for a doctorate to get involved in research a few years before graduation. I went along to explore ideas of microcomputer implementation of finite automata defined in tabular form, with applications to numerical control of industrial machines. One of my classmates went to study formal modelling of operating systems. He once showed me the Russian translation of a book by D.C. Tsichritzis and P.A. Bernstein, *Operating Systems*, which had an appendix on Petri nets. My friend and I spent a few hours playing with Petri nets describing semaphores. That was 1976 and that was probably the first time I touched this wonderful mathematical object. From the very start it fascinated me with the ease with which one could start playing with a mathematical model. The strange impact that Petri nets had on me from the very beginning was that the model was very unconventional, compared to all previous maths I had studied before. Graphs

---

A. Yakovlev (✉)  
School of Engineering, Newcastle University, Newcastle upon Tyne, UK  
e-mail: [alex.yakovlev@newcastle.ac.uk](mailto:alex.yakovlev@newcastle.ac.uk)

© Springer Nature Switzerland AG 2019  
W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,  
[https://doi.org/10.1007/978-3-319-96154-5\\_28](https://doi.org/10.1007/978-3-319-96154-5_28)

233

even with different types of vertices were too static. Finite automata, or state machines, were more dynamic but they always had one state at a time. Computer programs were “too syntactic”, much like formal languages and grammars. The awe and charm of Petri nets was that they seemed to be free from all these limitations!

But then, with my greater involvement with numerical control and the arrival of microprocessors, which quickly occupied my interests (come on, I was able to play with one of the first USSR microcomputers Elektronika NC-3!), I somehow stopped thinking about Petri nets, for a few more years... little did I know!

The next thing I remember well was that I was joining Victor Varshavsky's group as a Ph.D. student. One of the few starting pieces of reading recommended by Varshavsky was R.M. Karp and R.E. Miller's “Parallel Programs Schemata” (Journal of Computer and Systems Sciences, 1969) in Russian translation. That wasn't an easy read for someone who was trained as an electronic and computer engineer. But that paper opened my eyes to formal models to describe parallel computing. I remember some combinations of terms such as “parallel sequencing” (!) I learned from that paper. Why Varshavsky asked me to read that paper I understood a bit later, when I delved into the world of asynchronous automata, or better to say Aperiodic Automata (this term was born in Varshavsky's group). These automata were quite different from the classical state machines that we studied at the university. Aperiodic automata were very much like parallel programs. My subsequent studies took me through the world of speed-independent circuits, invented by David Muller. I read about Muller's theory and such things as a C-element, semi-modularity and many other puzzling objects and properties from the wonderful book by R.E. Miller “Sequential Circuit Theory”, volume 2. The famous chapter 10 in that book was a real marvel—it talked about the particular type of logic circuits that could coordinate their switching behaviour in time by themselves—without a clock! The circuits mysteriously passed the signal transitions from one element to another, sometimes forking switching threads in parallel and then joining them. The state graphs that described them looked quite peculiar. Their shapes had lots of diamond-like rectangles, where signals firing in parallel were in some form of interleaving.



Victor Varshavsky with members of his group at our reunion in Eilat, Israel, in 2000; left to right: Alex Kondratyev, Alexander Taubin, Michael Kishinevsky, Victor Varshavsky, Alex Yakovlev (me) and Maria Yakovlev (my wife)

These shapes, as Muller proved, had the properties of lattices, and what was most interesting, he showed the link between the correct operation, i.e. independence of the behaviour of the actual values of gate delays (similar to the absence of hazards), and the classes of lattices. So, that way I learnt about semi-modular (distributive) lattices and semi-modular (distributive) circuits. I think the most influential paper by Muller for me was this one: D.E. Muller and W.S. Bartky, “A theory of asynchronous circuits”, *Annals of the Computation Laboratory of Harvard University* vol. 29, Harvard University Press, 1959, pp. 204–243. I recommend it to anyone who really wants to understand fundamental links between mathematical models of concurrency (similar to Petri nets in spirit) and physical behaviour of digital circuits.

## 2 On Handling Complexity

One important aspect of modelling concurrency concerned the question of how to represent concurrency. In my early attempts to represent concurrency in asynchronous circuits I used the so-called Muller Diagrams, which were basically state graphs (or as people usually call them now Labelled Transition Systems). States were labelled with Boolean vectors corresponding to the values of circuit signals and

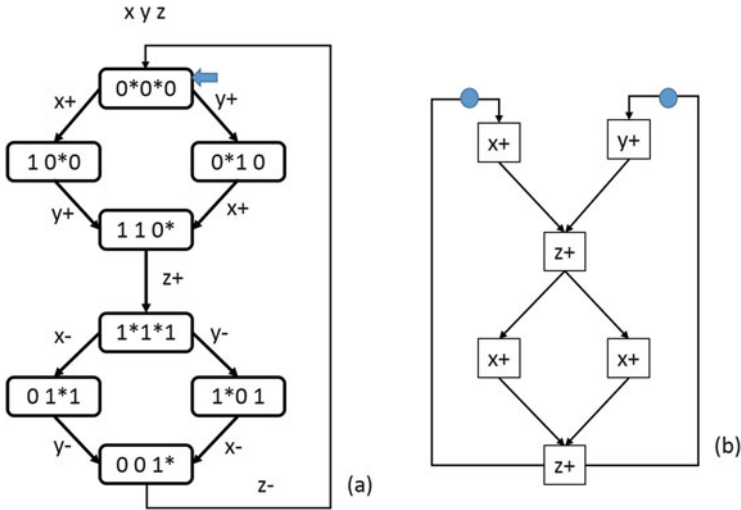
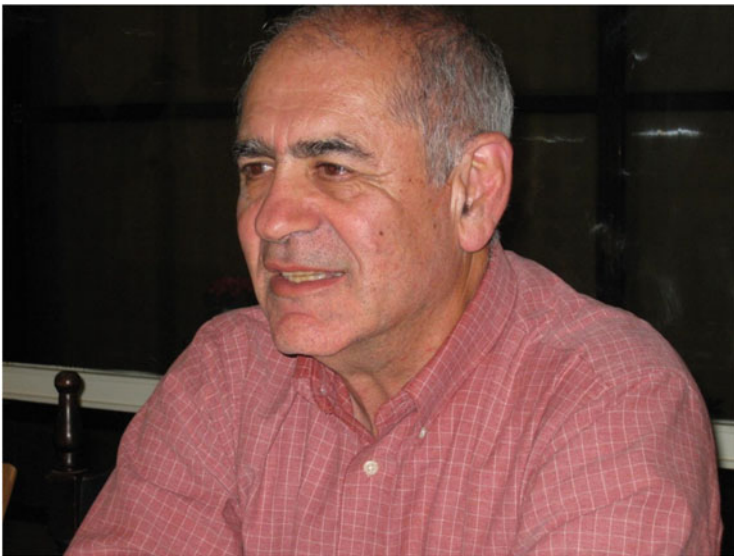


Fig. 1 Muller diagram (a) and corresponding signal graph (b)

transitions were labelled by signal changes (see Fig. 1a). I spent days modelling and synthesising circuits, such as bus interface controllers, say UNIBUS, using Muller diagrams. The difficulty was that with the number of signals becoming greater, the number of states could grow exponentially—so even producing a specification was a challenge. At the same time, another less formal notation was also in use amongst electronic engineers—that was Timing Diagrams, or waveforms, in which signal transitions—from 0 to 1 and back—were connected by arrows, representing causal relationships. At some point, my other PhD advisor, Leonid Rosenblum, suggested to me to use a similar causal representation to capture the behaviour of asynchronous circuits. That gave rise to the so-called Signal Graphs (see Fig. 1b), which could be made equivalent to Muller Diagrams, provided that certain semantical conventions were followed. Basically, the path from Signal Graphs to Muller Diagrams was facilitated via a token game that was adopted from the reachable state construction for Petri nets. At first, we used only a subclass of Petri nets called Marked Graphs (which contained only causality and concurrency but no choice). Using Signal Graphs made my life with circuit design a lot easier and that’s what I called “Rosenblum’s magic”. With Signal Graphs I could achieve linear complexity of the representation of the specification of a circuit with many signals—the exponential size was hidden! The other nice feature of this notation was that the concurrency and causality were captured by it in a true form (not via an interleaving of states).

Leonid Rosenblum was the first in the USSR who used Petri nets to describe asynchronous circuits (called Aperiodic Automata in Varshavsky’s group, as mentioned above)—it was around 1975. Sometime later, in mid-1980s, he wrote a very important article for “Tekhnicheskaya Kibernetika” (Engineering Cybernetics)

of the USSR Academy of Science. It was a comprehensive literature survey on Petri nets. At that time, besides the work of Vadim Kotov and his team at Novosibirsk nobody else in the USSR wrote much on Petri nets. Rosenblum's paper was a tremendous success and made a great impact on young researchers. It opened up a wealth of interesting research problems, not only in Petri net theory but also in its applications, to many people in various Soviet research centres. Naturally, for me this was a way to explore in terms of using them in digital circuit design and connecting them with models like Muller Diagrams and Lattices. In particular, the behaviour of Petri nets could be treated in the same way as that of asynchronous circuits. Namely, a Petri net produced a Reachable State Graph, which could be unfolded into a Cumulative State Graph, in which (cumulative) states were Parikh vectors (built on transition counts) rather than markings. So, I asked myself questions such as what classes of Petri nets corresponded to distributive lattices and to semi-modular lattices on those Parikh vectors? Later, I found an answer to the first question: Safe-Persistent nets, or Marked Graphs, to which Safe-Persistent nets could be unfolded. The second question was however a bit trickier. Why? The answer was in learning how to capture OR-causality!



Leonid Rosenblum, who introduced Petri nets into asynchronous circuit modelling and popularised Petri nets in the USSR in the 1970s and 1980s

Meanwhile, the important lesson learnt from that study was that true causal representations could make the modelling more compact and natural, and could find a much shorter road to the minds of practical engineers.

Here, I cannot help mentioning the following. Today, we have a much more comprehensive understanding of the relationship between state-based models (e.g. labelled transition systems) and event-based models such as different types of Petri nets—thanks to the theory of regions, whose pioneers include Andrzej Ehrenfeucht, Grzegorz Rozenberg, Eric Badouel, Luca Bernardinello, Philippe Darondeau and several other members of the Petri Nets community. The theory of regions underpins many developments in various prolific application areas such as process mining, visualisation and asynchronous circuit synthesis. Tools such as Petrify, developed by Jordi Cortadella, help to automate these steps.

### 3 On AND and OR Causality

I first studied Signal Graphs at the level of Marked Graphs. The two papers that had most effect on me then were: F. Commoner, A.W. Holt, S. Even and A. Pnueli, “Marked Directed Graphs” (Journal of Computer and Systems Sciences, 1971) and J.R. Jump and P.S. Thiagarajan “On the interconnection of asynchronous control structures” (Journal of the ACM, 1975). I would also highly recommend them to anyone who wants to understand the way of causality and concurrency in discrete event systems. One important aspect of this class of models was that the causality was in its strong aspect, e.g. if an event C had two causes A and B, then it was imperative for both of them to have happened before C was enabled. At some point, I realised that in my Muller Diagrams I could also model another form of causality, which I called weak or OR-causality, which was also sometimes associated with the so-called inclusive OR element in logic. In the OR causal case, the above-mentioned event C would be caused by A or B, whichever comes first. This behaviour was still within the class of semi-modular circuits, captured by the Parikh (cumulative state) vector graph as a semi-modular lattice. My colleagues Michael Kishinevsky, Alexander Taubin and Alex Kondratyev, in order to capture semimodular behaviour, invented their own event-based model called Change Diagrams. It basically coincided with Marked Graphs in the class of distributive lattices, but to handle OR causality it required a special type of dashed (weak-causality) arcs and some other ‘tricks’ (e.g. negative marking). I preferred to stay within Petri nets as I eventually found how to represent OR causality in them—I just needed to employ non-1-safe Petri nets, so from the practical point of view the problem was solved. The theoretical problem, however, was about finding the exact class of Petri nets that would correspond to Change Diagrams. It remained open until about 1996 when I published a paper “On the models for asynchronous circuit behaviour with OR causality” with M. Kishinevsky, A. Kondratyev, L. Lavagno and M. Pietkiewicz-Koutny in Formal Methods in System Design, 1996. This paper proved (using languages generated by the models and an observation equivalence notion) that within finite Petri nets we could not capture some behaviours of Change Diagrams. On the other hand Change Diagrams could not model processes with non-

determinism and conflicts. As a result a new model, called Causal Logic Nets, was proposed in that paper.

As it happened in the early 1990s, after I came to England, for me there was no alternative but to start calling our Signal Graph model a Signal Transition Graph, or STG for short. STGs in this nomenclature were proposed by Tam-Anh Chu (a Ph.D. student of Jack Dennis at MIT), who published his first paper on STGs “A design methodology for concurrent VLSI systems” in ICCD-85 in October 1985, while our paper with Leonid Rosenblum “Signal Graphs: from Self-Timed to Timed Ones” was published in the Workshop on Timed Petri Nets in July 1985. Incidentally, Chu, in his work, limited the class of STGs for which he developed his circuit synthesis method to the so-called persistent STGs. I felt that was an unnecessary limitation (see below).

Meanwhile an important lesson learnt from this study was that there was an important link between the algebraic structure of the state semantics of a concurrent system and the “causal fabric” of the system, which ultimately reflected the functionality (use of AND and OR logic). It is also worth mentioning that the properties of distributivity and semi-modularity, closely connected with the use of only AND causality and AND-plus-OR causality respectively, were also linked with the idea of extendibility of the concurrency relation. As shown in our paper “A look at concurrency semantics through lattice glasses”, published in the Bulletin of the EATCS in 1989, the class of distributive behaviours allowed the generalisation of the relation of concurrency between a set of  $N$  events from being pairwise concurrent to being  $N$ -way concurrent. This showed a way to distinguish between the notion of independence between events or actions and their parallelism in execution. The latter was connected with the availability of sufficient resources to run them in parallel.

## 4 On Persistence

Persistence is an interesting property of Petri nets. It refers to the behaviour of a net in which an enabled transition cannot be disabled by another transition’s firing. This property has an inherent analogy with the property of semi-modularity in Muller’s asynchronous circuits, where a gate if excited cannot be disabled from firing by the fact that one of its inputs (concurrently excited) changes its state. This sort of persistence was natural to me and clearly separated good behaviour from bad. Tam-Anh Chu defined another notion of persistence (specific to an STG) that had nothing to do with the persistence of the Petri net (any of its transitions) underlying the STG model under consideration. Instead, what Chu called the violation of persistence was the following. Suppose a transition  $a+$  causes a transition  $b+$ . Suppose  $a+$  also causes another transition  $c+$  immediately followed by  $a-$ , in which case  $b+$  can be concurrent with  $a-$ . This sort of effect my colleagues and I used to call “takeover”. Takeover has a certain influence on the way logic circuits are derived and the fact that they normally require the use of OR gates

for positively switching logic. However, for some reason Chu required that STGs had to be persistent in this sense. This condition was a limitation which was not necessary for deriving a logic implementation from the STG. Methodologically, though, it reflected an interesting discrepancy between the way concurrency and causality was perceived at the level of the behavioural graph specification (Petri net, STG) and how it was perceived at the circuit level. Again, deeper understanding of the semantics of Petri nets as a model of a system by itself, its interpretation in terms of the STG, as well as the relationship between these semantics, was very important. My papers “On limitations and extensions of STG model for designing asynchronous control circuits” (ICCD 1992) and “A unified signal transition graph model for asynchronous control circuit synthesis” (with Lavagno and Sangiovanni-Vincentelli at ICCAD-92, which in full appeared in *Formal Methods in System Design*, 1996) resolved this and a few other important modelling discrepancies with significant impact on practical asynchronous design, including a characterisation of the classes of delay-insensitive circuits due to Jan-Tijmen Udding in terms of STG classes.

Basically, an important lesson was learnt from the study of persistence: one had to pay attention to the details of models and their semantical interpretations. In the last few years thanks to the efforts of Eike Best and other colleagues, theoretical interest in capturing persistent behaviours has increased and produced interesting papers in Petri net synthesis.

## 5 Epilogue

Many other exciting avenues concerning ways to represent concurrency in a compact form were explored (e.g. Petri net unfoldings) and it is only due to space limitations that they are omitted here.

I would like to finish this essay by saying how much effect the event of meeting with Carl Adam Petri in person for the first and last time in my life had on me. It was in June 2005, at the 26th Conference on Petri nets in Miami, where Petri gave a keynote lecture. His lecture contained very interesting links between information theory and physics. Namely, what fascinated me was the notion of information preservation in the course of information processing. Later I only heard Petri’s lectures given by his associates, in particular, by Rudiger Valk (2008, Xi’an). And they continued to intrigue me. With my increased interest in the last 10 years in studying ways to mathematically model energy flow, or what Oliver Heaviside called “energy current”, in electronic circuits, everything seems to be gradually connecting together. I am sure that Petri nets will be at centre of some unification theories and practical applications linking information processing and physics!

Finally, I would like to thank Professors Wolfgang Reisig and Grzegorz Rozenberg for inviting me to contribute to this collection. It’s a great honour and pleasure.





Our recent book on modelling of concurrent processes using Petri nets (by L. Rosenblum, V. Marakhovsky and me), published in Russian, Saint-Petersburg, 2014

# The Road from Concurrency to Quantum Logics



Luca Bernardinello and Lucia Pomello

In this contribution we revisit the influence that Carl Adam Petri has had, either through personal interaction or through his work, on our research over almost 35 years.

Given the nature of the present volume, we will often refer to our personal memories and experiences, and will use abbreviations (LP and LB) to refer to each of us.

## 1 Carl Adam Petri in Milano

We had the luck to meet Carl Adam Petri several times, not only because LP spent 10 months during her Ph.D. course at GMD when C. A. Petri was director of the FI institute in 1983/84, or because Carl Adam Petri often attended the Petri Net Conferences as well as the advanced courses on PN, but also because Petri visited the University of Milano in 1989 and in 1997, and the universities of Milano and Milano-Bicocca in 2004, staying for quite a long period of time, giving lectures and always being willing to meet and discuss with students and researchers. In fact, in the 1970s Petri nets had been introduced at the Istituto di Cibernetica of the University of Milano by the director Giovanni Degli Antoni, who was firmly promoting net theory, stimulating people to do research on Petri nets, and invited Carl Adam Petri, with whom he was very close.

---

L. Bernardinello (✉) · L. Pomello

Dipartimento di Informatica, Sistemistica e Comunicazione (DISCo), Università degli studi di Milano-Bicocca, Milano, Italy

e-mail: [bernardinello@disco.unimib.it](mailto:bernardinello@disco.unimib.it); [lucia.pomello@unimib.it](mailto:lucia.pomello@unimib.it)

© Springer Nature Switzerland AG 2019

W. Reisig, G. Rozenberg (eds.), *Carl Adam Petri: Ideas, Personality, Impact*,  
[https://doi.org/10.1007/978-3-319-96154-5\\_29](https://doi.org/10.1007/978-3-319-96154-5_29)

243

## 2 From Concurrent Programming to Petri Nets

During her Master's thesis in Mathematics at the Istituto di Cibernetica, under the supervision of Giorgio De Michelis and Carla Simone, LP learned about Petri nets and afterwards, also together with Fiorella De Cindio, started to do research in net theory. Since the group had a background in program semantics and in languages for concurrent programming, we started by modeling with Predicate Transition Nets both Hoare's Communicating Sequential Processes and Milner's Calculus of Communicating Systems. In contrast to the interleaving semantics, we proposed models in which any sequential process is described by a state machine and the whole CSP program/CCS system is obtained by *superposing* the nets modeling the different sequential processes on the basis of the specification of a *synchronic distance zero* between corresponding transitions. As a natural result of this research a subclass of Petri nets, called Superposed Automata nets, was introduced. Superposed Automata nets are indeed obtained by considering systems to be composed of autonomous sequential components interacting with one another, where the interactions are modeled by the identification of transitions.

Inspired by Petri's proposal to *use nets to model organizational systems* and in particular *the information flow* among the different system components, this class of nets was proposed, with success, to model different aspects of organizational systems in the context of negotiation of organizational changes. Thanks also to the simplicity of the net language, the different actors involved in the changes were even able to construct the model and to discuss within the model itself aspects such as the impact of the proposed change, e.g., the degree of autonomy of a system component on the basis of the locality of the solution of net conflicts.

The study of Milner's approach, and in particular his proposal of observational equivalence as supporting organizational abstraction on the basis of which a sequential component can be substituted by the composition of different interacting subcomponents without changing its behavior, led us to introduce, also in the framework of net theory, observational equivalences as a tool to verify the correctness of each step in the model construction process.

## 3 A Whole Afternoon with Carl Adam Petri

When, during my stay at GMD, I (LP) gave a talk on my work on equivalence notions for concurrent systems modeled by nets, Carl Adam Petri, with great patience and generosity, spent a whole afternoon till late evening complaining about my use of event labeled nets, where different events, with different *extensionality*, could be labeled by the same observable action name. I tried, without success, to explain the intuition on organizational abstraction and to justify the presence of transitions with the same label.

During that afternoon, Petri explained to me a lot of different aspects of net theory, suggesting that I should take more deeply into account not only the extensionality principle, but also the *duality events-conditions*, the *non-sequential processes* as behavioral models with their basic relations of *causal independence* and *causal dependence*, the notions of *morphisms* between nets, . . .

Taking into account these suggestions, and also encouraged by Eike Best and César Fernández, the various considered equivalence notions have been defined also on partial order semantics on the basis of non-sequential processes.

This led to a contribution to the controversy, which was taking place in those years, between *interleaving* and *true concurrency* semantics. On the basis of a simple example, we pointed out that interleaving semantics strongly depend on the *axiom of action atomicity*. This in turn led to an investigation of *refinement-preserving equivalence notions* based on partial orders.

Concerning the *duality events-conditions*, the notion of observability has been applied also to conditions. In this way, it is possible to abstract, through the corresponding introduced equivalence relations, from the level of action description and instead to preserve the transformations of the observable local states.

## 4 Local Structures: LST Algebras, Regions

One of the recurrent themes in our research, and one where the influence of Petri is crucial, is the search for “local” structures in models of distributed or concurrent systems. This search has taken several shapes over time.

In particular, around the 1990s, in collaboration with Carla Simone, in order to capture the *locality* of system evolution, the system state space was characterized by a relational algebraic structure, called a Local State Transformation (LST) algebra, in which both global and local states as well as local state transformations are explicitly taken into account.

A class of injective *morphisms* between such algebras was introduced, allowing the comparison of system models at different levels of abstraction of the action description, together with a notion of state observability and of preorder and equivalence based on state observability.

A different way of looking at relations between local elements of a model relies on the notion of a *region* of a transition system, and on the corresponding duality between global states and local properties of a system.

In 1989, Grzegorz Rozenberg visited Milano and Mantova on two occasions, introducing, in a couple of talks, 2-structures and regions; in 1990 a series of papers on the subject, by Rozenberg and Andrzej Ehrenfeucht, appeared in Acta Informatica.

Although 2-structures were introduced as a very general kind of mathematical object, we immediately became interested in the use of regions as the key to solving the synthesis problem for Petri nets: given a transition system, decide whether there

exists a net of conditions and events such that its case graph is isomorphic to the given transition system.

We started working on this subject, and the first result was a proof that minimal regions (minimal with respect to set inclusion) are sufficient to solve the synthesis problem.

This result prompted us to look more deeply into the structure of the family of regions of a transition system. In the same period, LB and Giorgio De Michelis met, in Leiden, Eric Badouel and Philippe Darondeau, who were working on the notion of region, in relation to trace languages and trace nets. During that meeting, the idea of a new class of regions, corresponding to places in PT-nets, started to take shape, and, not surprisingly, proved to be closely related to the notion of weighted synchronic distance, which Petri and others had introduced with the idea of measuring the degree of independence between transitions.

Our work on regions proceeded since then along two paths. On one hand, following the major results obtained by Badouel and Darondeau on the complexity of the synthesis problem, and on the notion of *type of nets*, defined as a transition system describing the possible states, and state changes, of a single “place”, leading to a very general notion of *local state*.

On the other hand, we showed, working with Carlo Ferigato, that elementary regions, partially ordered by set inclusion, form an algebraic structure known as an orthomodular poset, quantum logic, or partial Boolean algebra. The link with quantum logics prompts one to reflect on Petri’s original claim that the conceptual basis of his theory was constructed with a clear reference to modern physics.

It is interesting to note that the family of regions of a single sequential component is a Boolean algebra, while the presence of concurrency disrupts the global Boolean character, producing a family of partially overlapping Boolean subalgebras corresponding to sequential components. This observation leads us to conjecture that the proper logic for distributed systems should not be classical.

## 5 Morphisms and Composition of System Models

In *General Net Theory* (1977), Petri states that concepts and results of the theory are transformed into “concepts and results on higher (and lower) levels of system description, by means of certain kinds of *net-mapping*,” (*morphisms*).

We have introduced *abstraction* and *composition* notions based on morphisms between elementary system models, which preserve the logic and algebraic structure of the system’s local properties.

Through a morphism, a model can be seen as the refinement of a more abstract one, where some regions/conditions are refined by some others, which are contained in set terms, and imply the more abstract ones in logical terms. Two different refinements of the same abstract model can therefore be composed by identifying those conditions representing the same observable property in the two different models. The result is a new model comprising the details of both operands and

indeed such that there are morphisms into the result from both the operands and the abstract starting model.

The intuition underlying this morphism-based composition is again due to the *duality conditions-events* applied to the *extensionality principle*, where any condition is completely specified by the events changing its truth value: its pre- and post-events. Thus, the composition of two different elementary net models containing the same condition is obtained by identifying not only that condition, but also the events changing its value, and this is given to an appropriate pairwise event superimposition.

## 6 Closure Operators Based on Concurrency

We have already noted that Petri has always referred to modern physics as a source of inspiration for his theory of information flow. We believe that the tightest link is that with relativity theory. The relativity of simultaneity, and the corresponding notion of space-like relation between events, have a direct correspondence in Petri nets.

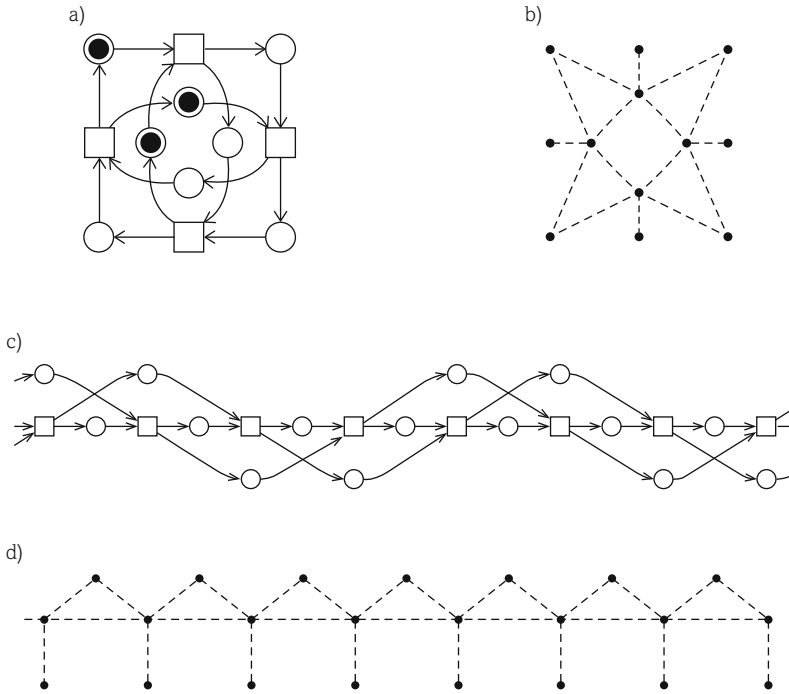
We started looking at occurrence nets as a discretization of relativistic spacetime, focusing on some properties of the concurrency relation.

Our starting point was a well-known result of lattice theory: given a symmetric and irreflexive binary relation on a given set, one can define a closure operator so that the corresponding closed sets form a complete lattice with an orthocomplement. It turned out that this lattice, in the case of the concurrency relation in occurrence nets, is orthomodular. This gives the discrete counterpart of a series of results obtained by several authors on continuous Minkowski spacetime. In our case, the closed sets can be seen as “causally closed” subprocesses, and the orthocomplement of a closed set as the maximal independent subprocess.

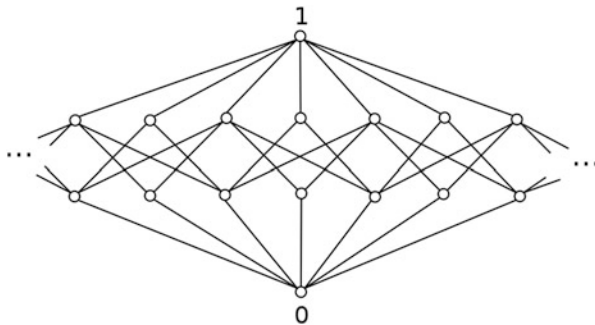
To illustrate this, we consider an example frequently used by Petri, and shown in Fig. 1 (taken from C.A. Petri, *Concurrency*, in LNCS 84, Springer, 1980). The figure shows a Condition/Event net system, a portion of its unfolding, and the corresponding concurrency relations. Figure 2 shows a portion of the infinite lattice of closed sets of the unfolding.

Going further along this direction, we proved that the orthomodularity of the lattice holds for a wider class of posets, namely for N-dense posets. Further interesting properties of the lattice of closed sets turn out to depend on K-density of the occurrence net. In particular, in K-dense nets, any line, which can be interpreted as a worldline, crosses either a given closed set or its orthocomplement.

Also in this case, like for regions, concurrency makes the overall structure non-Boolean.



**Fig. 1** From C.A. Petri, *Concurrency*, 1980. (a) A Condition-Event-System. (b) The concurrency structure of the system (a). (c) Section of the cycle-free occurrence net of the System (a). (d) The concurrency structure of (c)



**Fig. 2** The lattice of closed sets of the occurrence net in Fig. 1c

## 7 Teaching with Petri's Examples

Besides their major role in our research, for us Petri nets are also an effective tool in teaching. We believe that their effectiveness comes from the combination of a simple conceptual basis (the ideas of local states, local changes of states, and clear

representation of fundamental situations, such as causal dependence, conflict, and independence), together with a rigorous formal apparatus (and the rich theory that comes with it), and the graphical representation. This allows one to present several concepts in a gradual way, from simple and intuitive examples to more elaborate formal results.

When explaining to the students the difference between synchronous, asynchronous, and alternating events we always find very useful and impressive the following simple examples Petri often used during his lectures. A pair of *asynchronous events* is two hands slapping together (in parallel) on the table: if we do it in slow motion we can perceive that one is slapping before the other, and if we repeat it twice, or even more times, between two subsequent slaps on the table of the same hand it is possible to register zero, one, or even two, but no more than two, slaps of the other hand. That means that their *synchronic distance* is two and that they are *concurrent*. Only if we deliberately alternate the slap of the left with the slap of the right hand, then we can register exactly one slap of one hand between two subsequent slaps of the other one: their synchronic distance is one and indeed there is a *causal dependence* between the two different *alternating* events. An example of *synchronous events* is given instead by two hands clapping each other. The clap of the left hand synchronizes with the clap of the right one: their synchronic distance is zero, no clap of one hand is possible without the other one, they are just *one unique event*. These examples are really illuminating, the students appreciate them very much and like modeling the different situations with Petri nets.