# Book Introduction by the Authors

Invited by

**Luca Aceto**

luca.aceto@gmail.com
President of EATCS
Reykjavik University, Reykjavik, Iceland

# Understanding Petri Nets
# Modeling Techniques, Analysis Methods, Case Studies

Wolfgang Reisig
Humboldt-Universität zu Berlin, Germany
reisig@informatik.hu-berlin.de

Long-time readers of this bulletin may remember my 1984 "Introduction to Petri Nets". I wrote that book at a time when concurrency theory was still evolving as an independent topic of Theoretical Computer Science. In those days, three models for concurrency prevailed: Petri nets, process algebras and parallel programs. A number of other models did not receive that much attention.

The conceptual idea of the 1984 book is the same as that of the new one: to pin down the essentials and the specifics of Petri nets. As time went on, some topics of the 1984 book vanished, and a couple of new ones appeared. The new book reflects this development. Furthermore, I fundamentally updated the style of presentation, based on the observation that Petri net literature tends towards extensive formalism. In the new book I introduce many concepts colloquially and by means of examples, referring to the formal presentation in the appendix.

As the book's title indicates, *understandability* of concepts is the central concern of this book. This governs the book's three parts, Modeling Techniques, Analysis Methods, and Case Studies.

It took me a while to understand that *Modeling* must start with *distinguishable* tokens. For example, "here you see a Euro coin" and "there is a cookie packet" are more intuitive than "a black dot on this place denotes a Euro coin, a black dot on that place denotes a cookie packet." The essentials and specifics of Petri net modeling also include distributed runs (partially ordered behaviors, executions), and the synthesis of Petri nets from sequential observations. The last topic in the "Modeling Techniques" part addresses the frequent criticism that Petri nets are "not compositional". As loosely coupled systems are most important in modern system architectures (such as service-oriented computing), I suggest some fundamental notions for this issue.

Analysis and verification of distributed systems have been dominated by temporal logic and model checking since the mid 1980s. The reachability graph of

a Petri net is a transition system. Thus, it is apt for classical analysis and verification methods. Nevertheless, specific analysis methods for Petri nets are most useful. They are particularly expressive and more efficient than general transition system-based methods. The best-known Petri net-specific analysis technique is the place invariant. Each Petri net $N$ is assigned its matrix $N$ and consequently its system $N \cdot x = 0$ of linear equations. Together with the initial state of $N$, each solution to this system of equations contributes a non-trivial invariant property written as an equation. This kind of invariants is particularly expressive, because Petri net transitions are reversible (in a step $M \xrightarrow{t} M'$ you can gain $M$ from $M'$ and $t$) and linear (with twice as many tokens, you can fire a transition twice as often). Furthermore, you gain inequalities from the plain structure of a net (traps and co-traps). Finally, you can lift all these techniques from black dot tokens to distinguishable tokens. Liveness properties can be deduced from the structure of the net. We stick to "leads-to" properties of the form "$G(\phi \rightarrow E\psi)$", where $\phi$ and $\psi$ are formulae decidable for each marking separately.

The third part presents a variety of case studies. I start with the classic mutual exclusion of the critical states of two processes. Petri nets allow - and enforce - you to explicitly model some assumptions on fairness (that other modeling techniques assume just implicitly, e.g., infinite reading of a variable does not prevent writing) and progress (a process may remain ideal forever, but must leave its critical state). Furthermore, this example shows the advantages of scenarios, which are based on partially ordered runs. A further case study describes a high-grade concurrent hardware processor, synthesized form sequential observations. Finally, a choice of algorithms on networks is discussed, including versions of leader election, mutex on networks, and consensus algorithms. Dijkstra's "Beauty is our Business" inspired the selection of those case studies. A brief history of Petri nets, some speculative questions, as well as references to other system models and analysis techniques conclude the book. I am happy that I could include the foreword by Carl Adam Petri himself, translated from the German version of this text, which appeared in 2010 before Carl Adam passed away.

My 1984 book has been a frequently quoted reference for quite a while. Hopefully, my *Understanding Petri Nets* will play a similar role in the future. You can find more information on this book, and, in particular, comments from David Harel and many other colleagues at

`http://u.hu-berlin.de/understandingpetrinets`