Petri Nets

Fundamental Models, Verification and Applications

Edited by
Michel Diaz

iSTE

WILEY

This page intentionally left blank

Petri Nets

This page intentionally left blank

Petri Nets

Fundamental Models, Verification and Applications

Edited by
Michel Diaz

iSTE

WILEY

First published in France in 2001 and 2003 by Hermes Science/Lavoisier entitled *Les réseaux de Petri* and *Vérification et mise en œuvre des réseaux de Petri* © Hermes Science Ltd, 2001 © LAVOISIER 2003
First published in Great Britain and the United States in 2009 by ISTE Ltd and John Wiley & Sons, Inc.

ISTE Ltd
27-37 St George's Road
London SW19 4EU
UK

www.iste.co.uk

John Wiley & Sons, Inc.
111 River Street
Hoboken, NJ 07030
USA

www.wiley.com

Library of Congress Cataloging-in-Publication Data

Réseaux de Petri and vérification et mise en œuvre des réseaux de Petri. English
Petri nets : fundamental models, verification, and applications / edited by Michel Diaz.
p. cm.
Includes bibliographical references and index.
ISBN 978-1-84821-079-0
1. Electronic data processing--Distributed processing. 2. Parallel processing (Electronic computers) 3. System design. 4. Petri nets. I. Diaz, Michel, 1945- II. Title.
QA76.9.D5P4813 2009
511.3'5--dc22
2009017412

British Library Cataloguing-in-Publication Data
A CIP record for this book is available from the British Library
ISBN: 978-1-84821-079-0

Printed and bound in Great Britain by CPI Antony Rowe, Chippenham and Eastbourne.

FSC
Mixed Sources
Product group from well-managed forests and other controlled sources
Cert no. SGS-COC-2953
www.fsc.org
© 1996 Forest Stewardship Council

Table of Contents

Preface

Future advanced architectures, such as embedded systems, having a greater complexity and new quality requirements, will need a more precise specification and better control of their design process. In order to acquire the corresponding fundamental knowledge, it is essential to rely upon approaches based on the use of adequate system models. In particular, such approaches need to acquire a deep understanding of the system, including its local behaviors and its communications, based on a well-defined representation of the designed architecture. This representation should be used as early as possible to analyze and validate the design. The goal of this volume is to present a family of formal specification models, based on Petri nets and extensions of Petri nets, because they are defined by simple and clear semantics, allow easy modeling of system key mechanisms, and are supported by strong analysis methods and tools. Furthermore, this set of models can be used for all design aspects, i.e. to specify functional behaviors, and to include temporal or stochastic requirements.

The main results related to this approach are given in this volume, in two parts, one presenting the fundamental models, and the other being dedicated to verification and applications. We have tried to highlight the important characteristics and the main properties of these models, and to show how they lead to the emergence of a full design methodology, which is both complete, in terms of all possible functional and other analysis, and integrated, because the same basic semantics are used for the full design support. We think that this volume should greatly help any designer to build the new forthcoming generation of distributed systems.

Lastly, I would like to thank all the authors who contributed to this book, for their expertise, their seriousness, their technical inputs, and for the great job they have done.

Michel DIAZ

This page intentionally left blank

Introduction

New technologies in processors and networks allow system designers to conceive and build advanced and sophisticated parallel and distributed architectures, which need to integrate non-functional real time and stochastic constraints with functional distributed processing and communication.

The global behavior of such systems depends first on the local activities and data, but also on the messages sent and received by the various interconnected sub-systems. As a matter of fact, understanding, expressing, specifying and validating such global behaviors proves to be a problem of very high complexity, leading to many design and implementation difficulties and bugs. For example, when considering *n* connected processors, they can run, at a given instant in time, using 2 x 2, 3 x 3 communications, etc., or a full communication, in which all *n* processors interact. The sum of the resulting combinations, of the order of 2^n, shows the complexity of the resulting conceptual problems, and explains in particular the increasing difficulty obtained when passing from an interconnection of a few processors to an interconnection of a large number of processors: when the number of processors varies from 2 to 10, the difficulty coefficient goes from 4 to about 1,000.

It should then be clearly understood that designing such distributed architectures leads to a very complex conceptual task, which has to be based on a well-defined methodology to be able to manage all system requirements and behaviors.

Design and specification

The design process starts by giving the different functions and agents which are required, and the way they are structured; second, the designers define the behaviors of the various processes and entities, and the way they communicate; then, if they want to analyze the correctness of the design as soon as possible, an adequate approach is needed to represent, in an explicit way, the (full) system global behavior, in particular to be able to check potential unanticipated sub-behaviors.

To check the design correctness, it is essential to use a precise *model* of all critical mechanisms, functions, sub-systems, etc., and then, whenever possible, to use a *formal model*, to define a mathematical representation of the system. Checking the *correctness* validation of the design at this step is then conducted by checking the behavior of this system model.

Note that, after a given adequate sequence of more or less formal validation steps based on models, the system will be defined as 'fully designed' and will be implemented using adequate tools and languages.

Formal approaches have been used for many years for the verification of communication protocols. Two principal approaches have been used., i.e. basic formal models, such as automata, Petri nets, process algebras, etc., and formal description techniques for protocols, such as Estelle, LOTOS, SDL, etc.

This volume proposes and develops a design and validation methodology that relies on the use of a family of basic formal models that are rather easy to understand, and able to:

– describe the semantics of all basic building mechanisms;

– clearly specify the interconnection and communication semantics;

– unambiguously describe the resulting behaviors;

– validate the system during the first phases of its design by using support tools.

In general, basic non-language oriented graphical models, that do not include language-specific operators and statements, lead to the simplest solutions for representing basic mechanisms in a very abstract and integrated way.

For this reason too, this volume selected a basic, language-independent set of models to represent and manipulate the fundamental concepts of communicating architectures.

Selecting a *model*

Several models exist, and each model has particular characteristics, more or less relevant for a specific design. Consequently, the choice of the right model depends on the designed system and on the properties to be analyzed, as the model must be able to describe the design, and also to allow the designer to check the validity of the required properties.

In general, the designer must have a good understanding of the fundamental semantics of the system, i.e. of its basic building mechanisms. Thus, for simple architectures, modeling will be able to represent in a faithful way all details of the system. However, for complex systems, it will generally be impossible, for economic reasons, to represent the details of all existing functions, and it will become necessary to select and validate certain building blocks, i.e those most likely to lead to erroneous behaviors.

Of all existing models, *Petri nets* (PN) and their extensions are of undeniable fundamental interest, because they:

– provided the first modeling approaches for the semantics of concurrent systems, and were used to model the behaviors of the first parallel and distributed basic mechanisms;

– define easy graphic support for the representation and the understanding of these basic mechanisms and behaviors;

– prove to be, starting from state machines, an easy extension of previous approaches and handle, at the same time, the creation and the analysis of models;

– express very simply the main basic concepts in communication, including waiting and synchronization, and furthermore take into account their temporal and stochastic parameters;

– ensure, being unrelated to a particular implementation language, the independence of the specification with respect to its implementation.

Furthermore, many validation methods have been developed, using a great number of theoretical results and support tools, able to manipulate functional, temporal, and stochastic behaviors. Finally, models based on PNs will help us to understand, define and analyze the behavior of these systems, in the preliminary and first steps of their design.

For all these reasons, a set of Petri net models was selected in this book to represent and manipulate the fundamental concepts of communicating architectures.

Petri nets

PNs were introduced by A.C. Petri in 1962 to synchronize communicating automata, and were then extended to define a large set of models, with increasing complexity and capabilities.

As will be seen, this family of PNs, starting from the simple traditional state machines, now allows system designers to handle in an integrated way the functional (qualitative) and the non-functional (e.g. quantitative) temporal and stochastic capabilities of systems.

Extensions of PNs were proposed according to two important axes:

a) for *qualitative properties* and behaviors, to use simpler and more compact models, by high level PNs, for handling generic behaviors (e.g. individual) and data, predicates and functions;

b) complementing this first axis, for *quantitative properties* and behaviors, to extend the previous models by integrating quantitative constructs and parameters related to temporal and stochastic requirements.

It is significant to note that all first and conceptual studies in these quantitative fields were carried out using PN-based models.

Functional qualitative properties

The first PN model, called the Condition–Event PN, was based on the use of Boolean values: true or false. It was generalized by Place–Transition PNs, now simply called PNs, which can use integers. This volume will begin with their presentation and validation.

Non-functional quantitative properties

The fundamental contributions of the second axis considers:

– *time PNs*, or TPNs, used for systems whose behaviors depend explicitly on temporal values;

– *stochastic PNs*, or SPNs, for which distributions are attached to the model, in particular for performance evaluation and reliability.

Families of PNs

When applied to the modeling of systems, it rapidly becomes apparent that these models do not have the same application power, in terms of:

– definition and description of the concepts for parallelism, distribution, and synchronization;

– understanding and using the temporal and stochastic semantics;

– analyzing the possibly different mechanisms and behaviors, in very different contexts and applications.

Figure 1 represents some of the principal models of this family. In this figure, an arrow means that the model at the end of the arrow was proposed after the model at the beginning of the arrow, and so gives the steps followed by the research to propose and develop these principal PN-based models.

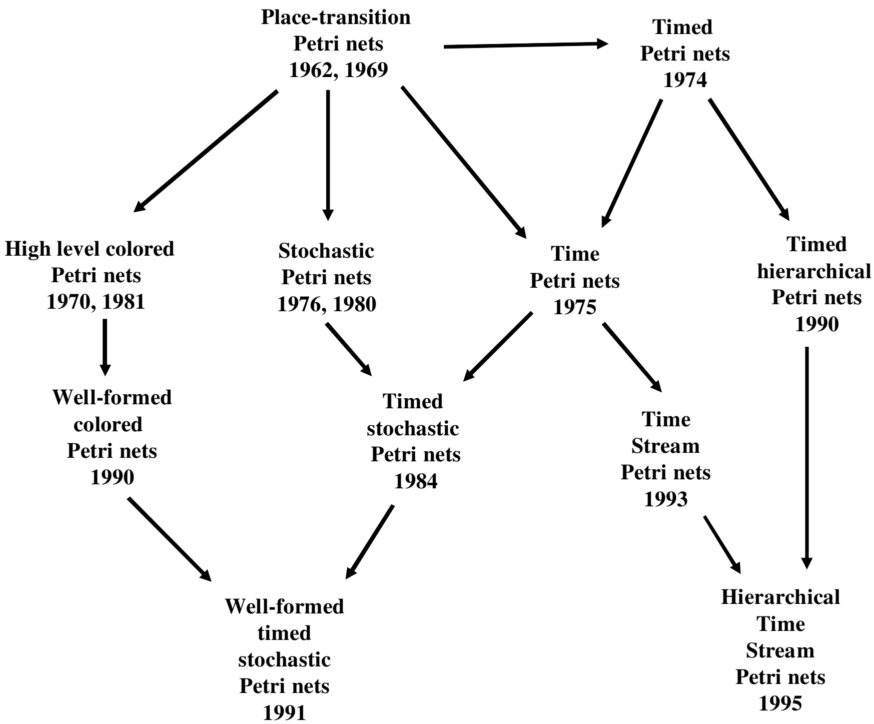

Figure 1. *The main Petri net models*

Figure 2 gives a more conceptual view of these models, by clarifying their syntactic and semantic relationships. In this figure, three fields are respectively defined by:

– a discrete state semantics, for non-temporal and non-stochastic nets, behaviors being represented by a finite graph of all model states;

– a semantics on continuous time, for extended behaviors based on dense time models;

– and stochastic semantics, for behaviors including distributions.

Let us emphasize that these models have three models of reference, respectively PN, TPN, and SPN. Moreover, each model is a pure extension of a previous one, as it can by simplified to become a basic PN model.

As seen in the figure, the models derived from the reference models:

– *lead to more compact models, i.e. are* ***abbreviations***, that do not increase the expressiveness of the model, but simplify the model and the system specification;

– *or are more powerful in terms of expressive power*, i.e. are able to describe mechanisms which could not be described by the unextended models (e.g. introducing time parameters, stochastic distributions, etc., for real-time or dependable systems).

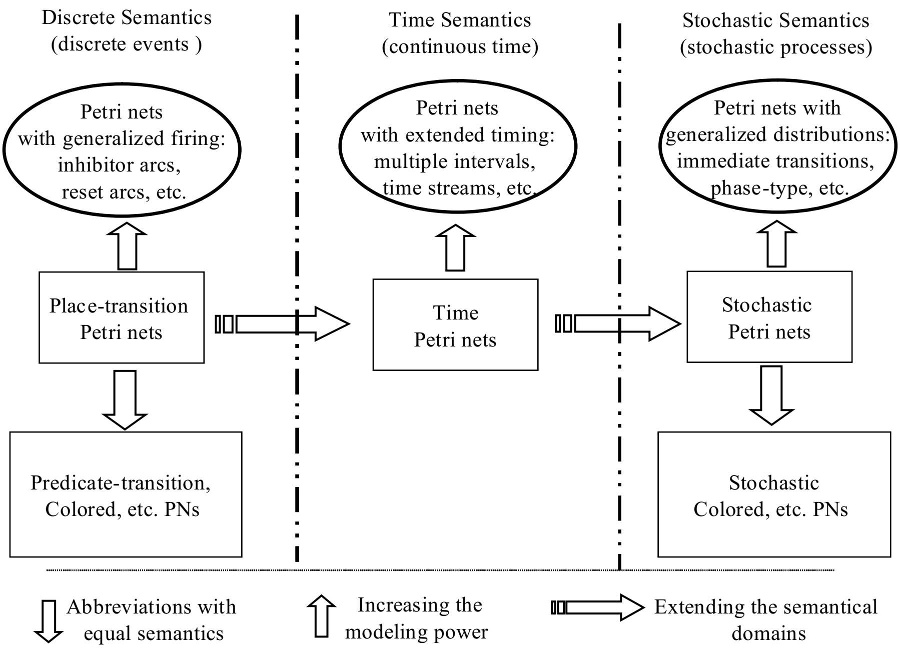

Figure 2. *Semantics domains of the Petri net-based models*

For example:

– PN led to PN with inhibitor arcs (to test the presence of zero token in one place), PN with reset arcs, etc.;

– TPN led to TPN with streams to compose and synchronize independent behaviors with independent temporal constraints, etc.;

– SPN led to SPN with immediate transitions in order to manage the case where transition cannot be delayed, etc.

Consequently, many different models exist, of different power and for different fields of application, but they follow the same semantics basis, and will allow the designers to carry out coherent complementary analyses to validate the correct operation of the modeled (and designed) systems.

The semantics of these models and their properties were used to select, define, and study the most important members of the PN family in the two parts of this volume.

Table of contents of the volume

Part 1 is dedicated to fundamental models and contains 11 chapters. Part 2 addresses verification and applications, and contains the last 7 chapters.

Part 1

Chapter 1 introduces Place–Transition PNs, more simply called Petri Nets (PNs). It gives their fundamental definitions, presents some basic models and clarifies their interest.

Chapter 2 illustrates an application in a very important area: communication protocols; simple PN examples show at the same time the power of the model and the interest of the formal analysis.

Chapter 3 first introduces the general properties that can be checked using PNs (blocking, reachability or accessibility, etc.) and the verification approach that uses the graph of the reachable (or accessible) states. The set of reachable (or accessible) states is the set of states that are reachable or accessible from a given initial state. Two optimization methods of analysis are then presented, one based on linear algebra techniques, and the other one exploiting the topological structure of the PNs.

Chapter 4 deals with the decidability and complexity problems related to checking these general properties.

Chapters 5 and 6 consider models and behaviors based on explicit values of time, and show how to model temporal mechanisms.

Chapter 5 presents the general model, Time PN or TPN, which associates a given interval (minimum, maximum) to each transition; this gives the first semantics for handling time and verifying temporal behaviors.

Chapter 6 presents a general model for composing temporal behaviors and systems. It gives the semantics of temporal composition by a new model, Time Stream PN, for composing autonomous (temporal) flows. It emphasizes their interests and applications for systems having independent temporal constraints, which sometimes interact.

Chapters 7 and 8 again consider PNs, i.e. non-temporal PN models, but define an abbreviation of a PN by a general model, which becomes able to represent, in a very compact way, a given set of similar parallel behaviors. The problems associated with this abbreviation are, on the one hand, to define a compact formalism and, on the other hand, to propose new validation techniques to handle this model, i.e. to avoid the obvious solution that consists of unfolding it into a very large PN.

Chapter 7 presents the main PN abbreviations, while concentrating on Colored PNs, which is the most frequently used model.

Chapter 8 gives one well-defined version of this formalism, Well-formed Colored PNs, which allows the development of efficient analysis techniques.

Chapters 9, 10 and 11 introduce distributions, which take into account probabilistic properties of systems. They introduce stochastic PNs, or SPNs, and define their semantics in terms of stochastic processes, and, for some classes of models, their relationships with Markov chains. The principal methods of analyzing SPNs are then presented. Chapter 9 introduces stochastic PNs.

Chapter 10 introduces well-formed SPNs by combining the formalisms presented in Chapter 7 (Well-formed Colored PNs) and Chapter 9. Modeling a multiprocessor architecture illustrates the expressivity of this formalism and its interest for performance evaluation. Chapter 11 develops a tensorial composition of classical and well-formed SPNs, showing that such a compositional approach reduces the complexity of the corresponding validation.

Part 2

The second part of this volume presents important advanced analysis techniques and finally gives some significant and illustrative case studies.

Chapters 12 and 13 address checking and verifying non-temporal behaviors. They present the main approaches that are based on building and manipulating the (system) accessibility or reachability graph, i.e. the graph representing all possible behaviors of the model. Checking these properties, by algorithms applied to the accessibility graph, suffers from the problems of combinatorial explosion. The general problems to be solved to control such an increase in the number of states, as well as general solutions, are then given. Three specific techniques, based respectively on the unfolding of the colored PNs, on symmetries, and on partial orders, are then presented.

Chapters 14 and 15 focus on the temporal validation of behaviors. Chapter 14 analyzes the relationships existing between symmetry and temporal logic for the verification of properties that depend on the specificities of the system. Chapter 15 introduces a parallel–serial hierarchy of temporal behaviors. This hierarchy simplifies the description of complex systems and is very well adapted for modeling complex multimedia and hypermedia objects, documents, and systems.

Chapter 16 presents how to use the main relationships that exist between linear logic and Petri nets for specification and validation. Logical reasoning is constructed based on the behavior of PNs that does not need to produce the reachability graph. The interest of linear logic is illustrated by showing in particular how to handle symbolic temporal intervals (minimum, maximum).

Chapters 17 and 18 present two important case studies that are illustrative while being manageable and easily understandable. Chapter 17 is devoted to the modeling and design of a multilayered, multimedia architecture that is able to guarantee temporal properties at the application level. Chapter 18 presents the application of PN-based models to performance evaluation in the field of computer-integrated manufacturing systems.

Finally, a conclusion summarizes the contents of this volume.

This page intentionally left blank

Part 1

Fundamental Models

This page intentionally left blank