Wolfgang Reisig

PETRI NETS

An Introduction

Wolfgang Reisig

PETRI NETS

An Introduction

Figures

Springer-Verlag
Berlin Heidelberg New York Tokyo

Dr. Wolfgang Reisig
GMD
Postfach 12 40, Schloß Birlinghoven
5205 St. Augustin 1, Germany

Prof. Dr. Wilfried Brauer
FB Informatik der Universität
Rothenbaum-Chaussee 67–69, 2000 Hamburg 13, Germany

Prof. Dr. Grzegorz Rozenberg
Institut of Applied Mathematics and Computer Science
University of Leiden, Wassenaarseweg 80, P.O. Box 9512
2300 RA Leiden, The Netherlands

Prof. Dr. Arto Salomaa
Department of Mathematics, University of Turku
20500 Turku 50, Finland

Translation of the German original edition: W. Reisig, Petrinetze
ISBN 3-540-11478-5
Springer-Verlag Berlin Heidelberg New York 1982

ISBN 3-540-13723-8 Springer-Verlag Berlin Heidelberg New York Tokyo
ISBN 0-387-13723-8 Springer-Verlag New York Heidelberg Berlin Tokyo

Library of Congress Cataloging in Publication Data
Reisig, Wolfgang, 1950–
Petri nets.
Based on lectures given by the author at the Technical University of Aachen.
Translation of Petrinetze.
Includes index.
1. Petri nets. I. Title.
QA267.R4513 1985 511 84-26700
ISBN 0-387-13723-8 (U.S.)

Typesetting, printing and bookbinding: K. Triltsch, Würzburg
2145/3140-5 4 3 2 1 0

Preface

Net theory is a theory of systems organization which had its origins, about 20 years ago, in the dissertation of C. A. Petri [1]. Since this seminal paper, nets have been applied in various areas, at the same time being modified and theoretically investigated. In recent time, computer scientists are taking a broader interest in net theory.

The main concern of this book is the presentation of those parts of net theory which can serve as a basis for practical application. It introduces the basic net theoretical concepts and ways of thinking, motivates them by means of examples and derives relations between them. Some extended examples illustrate the method of application of nets. A major emphasis is devoted to those aspect which distinguish nets from other system models. These are for instance, the role of concurrency, an awareness of the finiteness of resources, and the possibility of using the same representation technique of different levels of abstraction. On completing this book the reader should have achieved a systematic grounding in the subject allowing him access to the net literature [25].

These objectives determined the subjects treated here.

The presentation of the material here is rather more axiomatic than inductive. We start with the basic notions of 'condition' and 'event' and the concept of the change of states by (concurrently) occurring events. By generalization of these notions a part of the theory of nets is presented. It would have been possible to proceed in the opposite order by firstly presenting net representations of practical, real systems and then, proceeding by a sequence of abstraction steps, reaching nets consisting of conditions and events. However, the chosen method of presentation corresponds to the usual way of proceeding in the framework of theoretical computer science.

It is not intended, in this book, to give a total overview and summary of the theory and applications of nets. Such an attempt is doomed to failure, not only because of the number of publications in the field, more than 500 are referenced in [25], but also because of the wide spectrum of the topics covered; for example complexity theory, the theory of formal languages, the theory and design of logic circuits, computer architecture, operating systems, the connection of computer processors, process control and real time systems, programming and command languages, databases, communication protocols, software engineering and yet even further into topics outside computer science (administration, jurisprudence, the logic of inter-personal interaction). Also, we are not able here to treat the foundations of net theory which lie in the philoso-

phies of natural sciences, in the classical and non-classical logics, in theoretical physics and in the theories of communication.

A series of lectures for students in the third and fourth year of computer science, which the author gave at the Technical University of Aachen, served as a basis for this book. It might therefore be used in university courses, but it is also intended for the graduate student, the researcher and the professional who want to start within the field of Petri Nets.

The book assumes only an elementary knowledge of the structure, functioning and application of computer based information systems and some elementary mathematics. Using the first chapter as a basis, Part 1 and Part 2 may be read rather independently of each other. Part 3 uses the notions introduced in Chapts. 2, 4, 5 and 6. The computing practitioner should, in addition to the first chapter, find it worthwhile to study, particularly, the example at the start of Chapt. 5 and Sects. 6.3 to 6.5 and 8.1 to 8.3.

At the end of each chapter exercises are given. The more difficult ones are marked with *.

The appendix presents the mathematical notions and notation which are used in this book.

This book was originally published in German by Springer-Verlag in 1982. For the English edition it was revised and the "Further Reading" appendix and the exercises were incorporated.

Acknowledgements

This book could not have been created without the help of a number of people. At the Institut für Informationssystemforschung of the Gesellschaft für Mathematik und Datenverarbeitung in Bonn (West Germany), I received great support in discussing particular topics from C. A. Petri, H. Genrich, K. Lautenbach und P. S. Thiagarajan. Prof. W. Brauer gave many valuable remarks on the German manuscript.

On the occasion of the English translation it was possible to revise the text due to many hints and comments from its readers. Especially I am indepted to Eike Best, Ursula Goltz, Kurt Lautenbach, Roberto Minio, Horst Müller, Leo Ojala, Anastasia Pagnoni, Grzegorz Rozenberg and P.S. Thiagarajan for their many critical and constructive notes. Horst Müller and Dirk Hauschildt mainly contributed to the revision of Lemma 5.3 (d) and Theorem 7.2 (k), respectively.

I am deeply indebted to the translators Ursula Goltz and Dan Simpson, who with remarkable competence, fervour and patience did an excellent job. They also brought up some valuable discussion with regard to the contents of the book.

W.R.
Aachen, Germany
June 1983

To the English Edition

We have retained the notation of the German book (e.g. B for sets of conditions and S for sets of places) corresponding to the standards introduced at the Advanced Course on Net Theory and Application, cf. [17]. Any changes might have induced further problems (e.g. C for conditions would exclude an appropriate notation for cases. P for places would imply the non-standard notion of P-invariant).

U.G., D.S., Aachen and Sheffield

Contents

Introduction

(a) *Petri nets*, the subject of this book, are a model for procedures, organizations and devices where regulated flows, in particular information flows, play a role.

This language of nets arose from the intention of devising a *conceptual* and theoretical basis "for the description, in a uniform and exact manner, of as great as possible a number of phenomena related to *information transmission and information transformation*" [1]. We shall restrict ourselves to such applications of this theory as lie in the area of the design and use of computer based information systems.

In comparison with other system models, the major characteristics of Petri nets are the following:

- Causal dependencies and independencies in some set of events may be represented explicitly. Events which are independent of each other are not projected onto a linear timescale; instead, a non-interleaving, partial order relation of *concurrency* is introduced. This relation is fundamental for the whole conceptual basis of net theory.
- For some systems it may not be sensible to try to describe them as sequential functions. To do so only leads into unnecessary distracting detail. Examples are a query answering system of a distributed database, a real time system for production control, the control of processes in an operating system or a communication protocol.
- Systems may be represented at different *levels of abstraction* without having to change the description language. These levels of abstraction range from the change of single bits in computer memories to the embedding of a computer system into its environment.
- Net representations make it possible to *verify system properties* and to do correctness proofs in a specific way. Once a system has been modelled as a net, properties of the system may be represented by similar means, and correctness proofs may be built using the methods of net theory. Logical propositions are obtained as static components of dynamic net models.

Two objections may be raised here. One is that other methods which are well-known and established aim for the same goals. The other point is made by considering processes which run independently of each other (for example: processes in the central memory and in peripheral processing units of some computer). Such processes take particular states and perform state changes. The argument is that such states or changes which are coincidential may be

combined into a global state or a global state change which covers these. Thus, a new theory is not required. Here we are not able to discuss in full the reasons why the specific ways of thinking of net theory are sufficiently important to justify the construction of a whole new theory. We simply note two points in reply: first, that the above proposed combination of coincident states or changes gives rise to the problem of determining whether they are really simultaneous. Secondly, a purely sequential model does not truly reflect the real *causal structure* of processes. In any sequentializing view we can not differentiate whether two events occur one after the other because the first is a prerequisite of the second or whether this order in time is solely by chance. But, in fact, the causal relations are those which, to a large extent, characterize a system.

(b) In the first chapter we shall present, by means of several examples, different net models. This gives a first insight into the structural patterns and representation methods typical for nets. The mathematically oriented reader may start at 1.1 and skip to 1.5.

Systems consisting of *conditions and events*, which are introduced in Part 1 of this book, constitute the most detailed description level of marked nets. Here, the fundamental notions of non-sequential processes are studied: viz., the relations of causal dependency and independency of events; the relationship between non-sequential processes and their set of possible sequential realizations; the metric of synchronic distances as a measure for the dependency between events; and, finally, the formulation of system properties in the language of logic and their integration into the net calculus.

In the second part of the book we consider nets consisting of *places and transitions*. Such nets are particularly suited to the formulation of blocking problems. For the investigation of such nets we introduce coverability graphs, which allow conclusions to be drawn about the behaviour of the system we are modelling.

We concentrate our presentation on those investigation methods which do not rely on the set of all possible sequential executions. A particular one of these is the calculus of invariants involving linear algebraic techniques. By means of several examples we show how this calculus may be used for the verification of system properties. For particular place/transition-nets, we derive particular methods of analysis.

In the third part of the book we consider *individuals, predicates* and *relations* on nets; we thus reach a level which yields a relationship between nets and universal algebra. We show how, on this level too, system properties which are formulated in the language of logic may yet again be represented in the net calculus. The verification of system properties so represented is again aided by an invariant calculus generalized from place/transition-nets.

Chapter 1

Introductory Examples and Basic Definitions

1.1 Examples from Different Areas

In the preface and the introduction, we have already used the terms "system organization", "system model", "condition", "event" and "information transformation" without explaining them. These notions are of fundamental importance in net theory. However, as they are concepts from the real world, we shall not try to give precise definitions of them but rather appeal to the intuition and general understanding of the reader. But, we shall have to consider properties of objects of this kind, and also the relationships between such objects. We shall say, for instance, that "system models" represent real systems more or less adequately, that "events" occur and that "conditions" do or do not hold.

(a) Let us first consider systems comprising *conditions* and *events*. Figure 1 shows a system in which the conditions are: "it is spring", "it is summer", "it is autumn" and "it is winter"; the events are: "start of spring", "start of summer", "start of autumn" and "start of winter". We see that each condition is represented by a circle and each event by a box. Each condition which holds is marked by a dot (a *token*) (in Fig. 1, it is "spring"). The set of conditions which hold in some configuration is called a *case*. In the system represented in

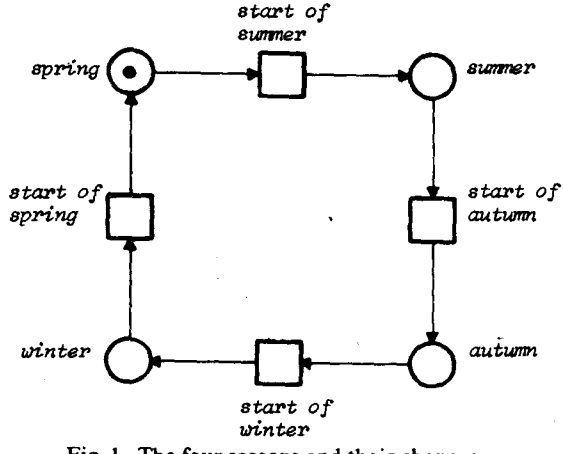

Fig. 1. The four seasons and their changes

Fig. 1, each case has only one element. Whenever an event occurs, another case results. A condition, b, and an event, e, may be related with each other as follows:

(1) b starts to hold when e occurs. b is then called a *postcondition* of e. Graphically, this relationship is represented as an arc from e to b.
(2) b ceases to hold when e occurs. b is then called a *precondition* of e. Graphically, this relationship is represented as an arc from b to e.

If b is not affected by the occurrence of e there is no arc between b and e at all.

So, in our system of the four seasons, when an event occurs the token is moved to the next season.

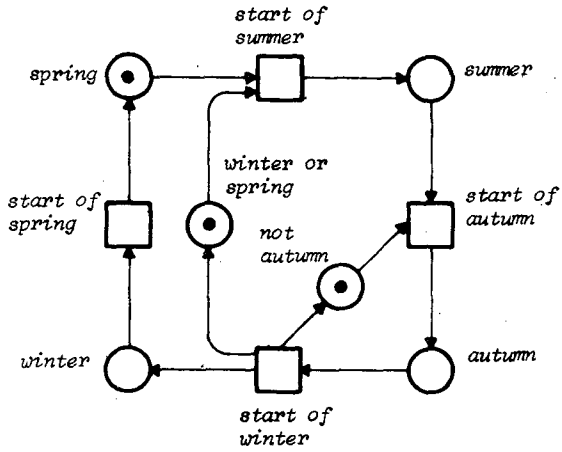

Fig. 2. Addition of two conditions to Fig. 1

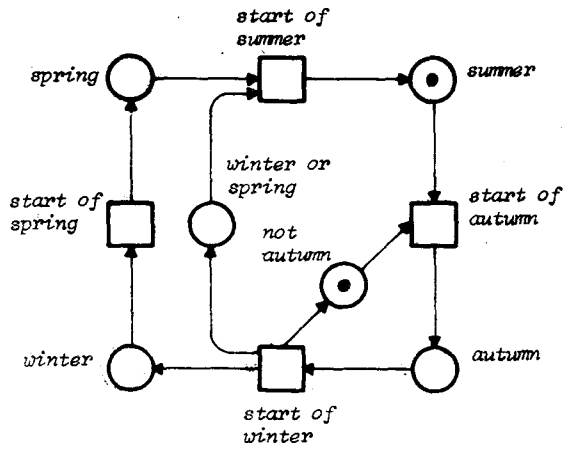

Fig. 3. The system of Fig. 2 after start of summer

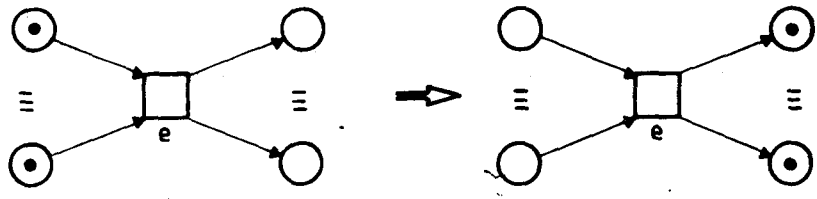

Fig. 4. The occurrence of an event e

When modelling the four seasons and their changes we may wish to represent additional conditions and events. When we add the conditions "winter or spring" and "not autumn", we obtain the system shown in Fig. 2. Note that now some events have several pre- or postconditions.

In the system represented in Fig. 2, consider now in which case the event "start of summer" may occur. This is when it is both "spring" and "winter or spring", and it is not already "summer". By the occurrence of this event we obtain the configuration shown in Fig. 3. In general, an event may occur if all its preconditions hold and none of its postconditions hold. Figure 4 shows the requirements for, and the result of, an event, e, occuring.

Although it is certainly an interesting event that winter ends, it should not be distinguished from the start of spring because neither of these events can occur without the other. The end of winter and the start of spring are *coincident* events, they are represented by one single box.

(b) When describing systems, at some levels, it is not always appropriate to use the notions of "condition" and "event". For example, when looking for

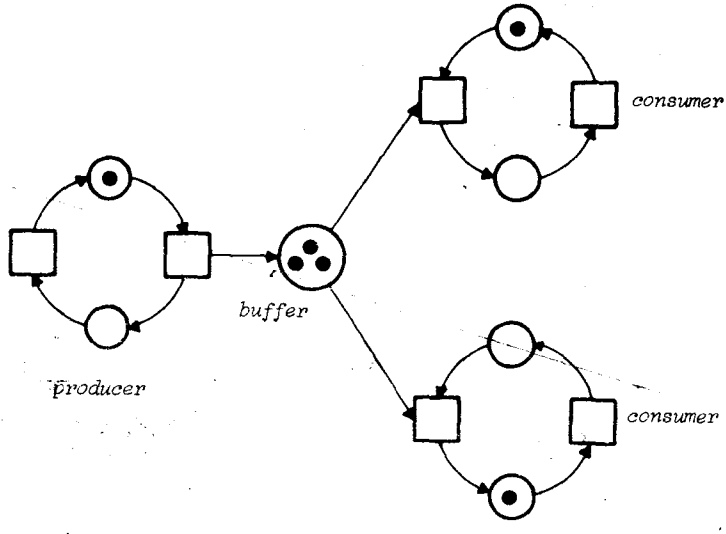

Fig. 5. A system consisting of one producer and two consumers

bottlenecks in manufacturing processes, it may only be the total number of goods produced which is of interest and not their individual identities. In the representation of a store, a set of conditions ("the places $s_1, \ldots, s_n$ are used") may then be combined into one item which is marked n-times ("n places are used"). Figure 5 shows a system of one producer and two consumers using a buffer as their store. The producer generates items (represented as tokens), which are placed in the buffer. The consumers may remove items which are in the buffer. In such nets, we say the elements are *places* ○ and *transitions* □. Places may, in contrast to conditions, carry more than one token. Arcs again indicate the flow of tokens. A transition *fires* by removing a token from each *input place* and by adding a token to each *output place* (Fig. 6). If we restrict each place to carry at most one token this firing rule corresponds to the effect of event occurrences described above.

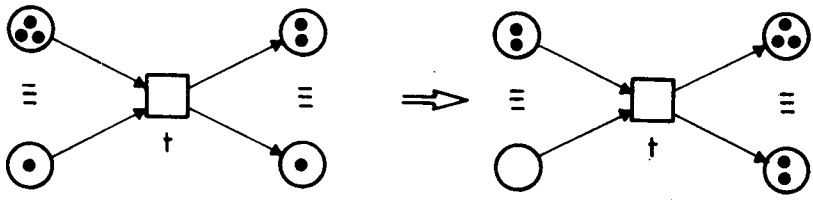

Fig. 6. The firing of a transition t

The two consumers are represented by two tokens in one single consumer part of the net as in Fig. 7. However, now the consumers may no longer be distinguished as individuals.

(c) Nets consisting of places and transitions model system properties concerning the number, the distribution and the flow of objects which are not further distinguished. If we wish to consider individual properties of the objects we must be able to identify particular tokens. Figure 8 shows a fragment of an industrial production system, the operation of which is intuitively clear. This also illustrates the construction of nets. Round nodes (places) represent passive system components. These are those components which may store items, take particular states and make things observable. Rectangular nodes (transitions)

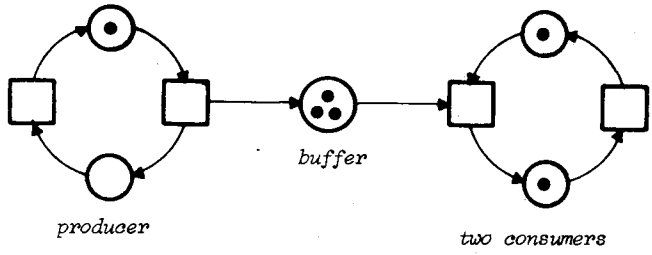

Fig. 7. Combination of the two consumers of Fig. 5 into one part of the net

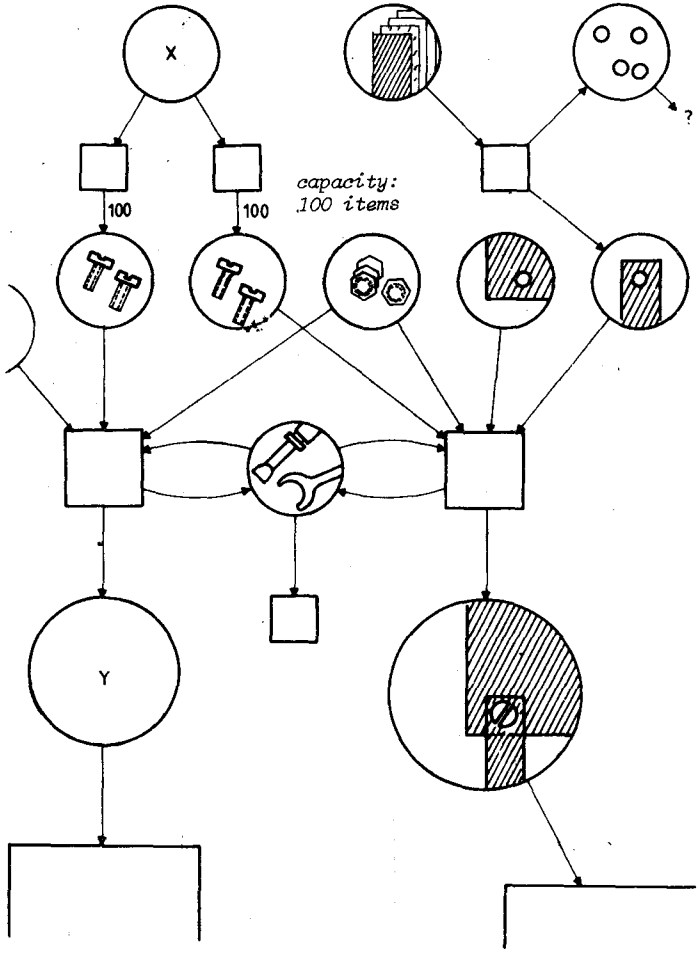

Fig. 8. A fragment of an industrial production system

represent active system components. Such components may produce, transport and change objects. Arcs show which system components are directly coupled with each other and in which direction objects may "flow" through the net. These objects themselves are represented as *individual tokens.*

(d) There are systems where some of the connections between system components are not oriented. Some systems do not have objects which flow. But we shall always adhere to the principle of partitioning the system into active and passive components. This partitioning may often be done in a number of different ways. For example, as a first approximation, a game of chess may be represented as an interaction, t, of two players (holders of states) s_1 and s_2. Alternatively, the board, s, may be considered as a passive object to be accessed by the moves, t_1 and t_2, made by each player. Figure 9 shows the first

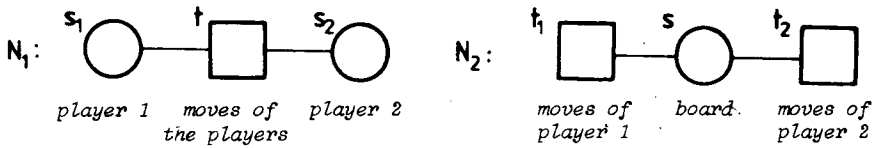

Fig. 9. Two representations of a chess game

view as N_1 and the second view as N_2. These two different partitions stress two different aspects of the same system. Each may be refined so that the aspects of the other view are included. Figure 10 shows the smallest refinement which covers the aspects of both views.

As long as no distinguished flow of objects is to be represented, the arcs of a net may be undirected, as in Fig. 9 and Fig. 10. In this book we will not discuss nets of this kind.

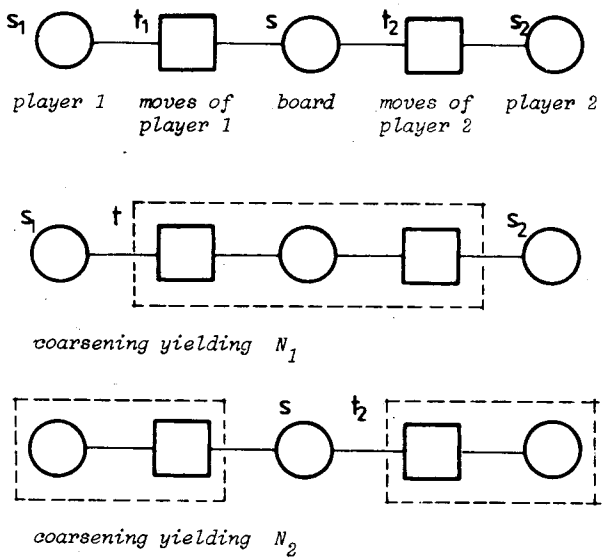

Fig. 10. Common refinement from Fig. 9

1.2 Examples from Logic Circuits and Operating Systems

(a) Let us start with a problem from logic circuits. x and y are two variables, which can take the values "true" and "false". Each is assigned an initial value independently of the other. They are then combined to give the value $x \wedge y$ to the variable x and the value $x \vee y$ to the variable y. These new values are available until they are, again independently, deleted. Then the system returns to the initial configuration and the variables may be given new values. Figure 11 shows this system as a net consisting of conditions and events.