Chapter 1

Introduction

Petri nets are one of the most popular formal models of concurrent systems, used by both theoreticians and practitioners. The latest compilation of the scientific literature related to Petri nets, dating from 1991, contains 4099 entries, which belong to such different areas of research as databases, computer architecture, semantics of programming languages, artificial intelligence, software engineering and complexity theory. There are also several introductory texts to the theory and applications of Petri nets (see the bibliographic notes).

The problem of how to analyze Petri nets – i.e., given a Petri net and a property, how to decide if the Petri net satisfies it or not – has been intensely studied since the early seventies. The results of this research point out a very clear trade-off between expressive power and analyzability. Even though most interesting properties are decidable for arbitrary Petri nets, the decision algorithms are extremely inefficient. In this situation it is important to explore the analyzability border, i.e., to identify a class of Petri nets, as large as possible, for which strong theoretical results and efficient analysis algorithms exist.

It is now accepted that this border can be drawn very close to the class of free-choice Petri nets. Eike Best coined the term 'free-choice hiatus' in 1986 to express that, whereas there exists a rich and elegant theory for free-choice Petri nets, few of its results can be extended to larger classes. Since 1986, further developments have deepened this hiatus, and reinforced its relevance in Petri net theory.

The purpose of this book is to offer a comprehensive view of the theory of free-choice Petri nets. Moreover, almost as important as the results of the theory are the techniques used to prove them. The techniques given in the book make very extensive and deep use of nearly all the analysis methods indigenous to Petri nets, such as place and transition invariants, the marking equation, or siphons and traps. In fact, the book can also be considered as an advanced course on the application of these methods in Petri net theory.

1.1 Petri nets

The Petri net is a mathematical model of a parallel system, in the same way that the finite automaton is a mathematical model of a sequential system. Petri nets have a faithful and convenient graphical representation, which we shall use in this informal introduction.

A Petri net has two components: a *net* and an *initial marking*. A net is a directed graph with two sorts of nodes such that there is no edge between two nodes of the same sort. The two sorts of nodes are called *places* and *transitions*. Places are graphically represented by circles, and transitions by boxes.

Places can store *tokens*, represented by black dots. A distribution of tokens on the places of a net is called a *marking*, and corresponds to the 'state' of the Petri net. A transition of a net is *enabled* at a marking if all its input places (the places from which some edge leads to it) contain at least one token. An enabled transition can *occur*, and its occurrence changes the marking of the net: it removes one token from each of the input places of the transition, and adds one token to each of its output places. Figure 1.1 shows on the left a Petri net containing an enabled transition, whose occurrence changes the marking to the one shown on the right.[1]

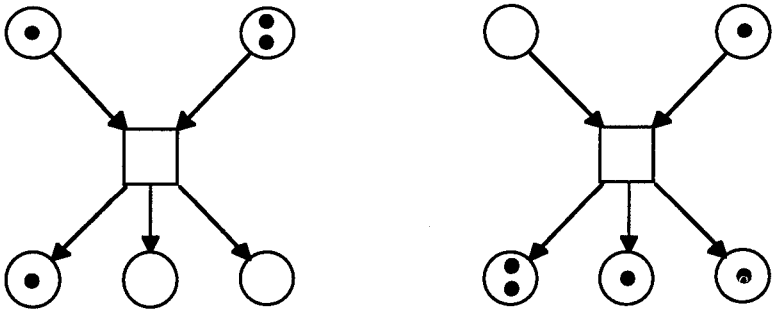

Fig. 1.1 A Petri net before and after the occurrence of a transition

With this simple occurrence rule, Petri nets can be used to model dynamic systems. Consider as an example the Petri net of Figure 1.2, which models a vending machine. At the marking shown in the Figure – called the *initial marking* – the machine is waiting for a coin to be inserted. This is modelled by the token on the place `ready for insertion`, which enables the transition `insert coin`. When this transition occurs, the machine can choose to reject or to accept the coin. In the first case, the machine returns to the initial marking; in the second it gets ready to dispense

[1] Petri nets with this occurrence rule are sometimes called marked nets, place/transition systems, or just systems.

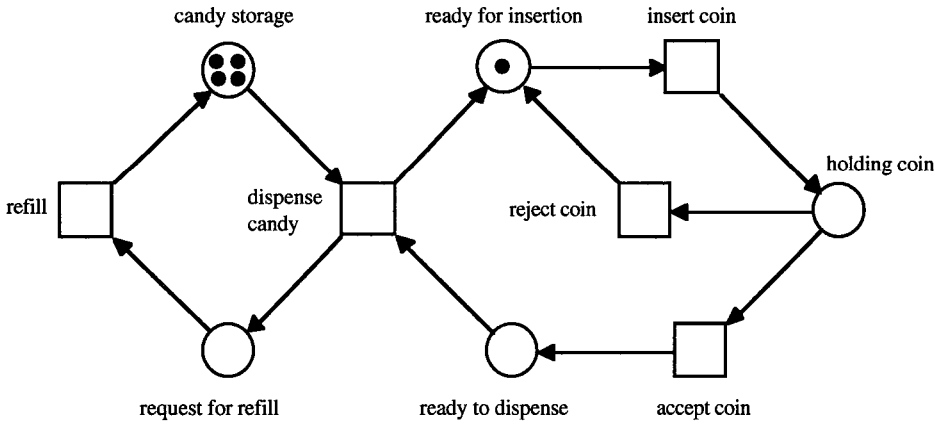

Fig. 1.2 A Petri net model of a vending machine

a candy. However, candies can only be dispensed if there are some available. The available candies are modelled by the tokens in the place `candy storage`. The storage contains initially four candies. When a candy is dispensed, the marking shown in Figure 1.3 is reached. At this point, transitions `refill` and `insert coin`

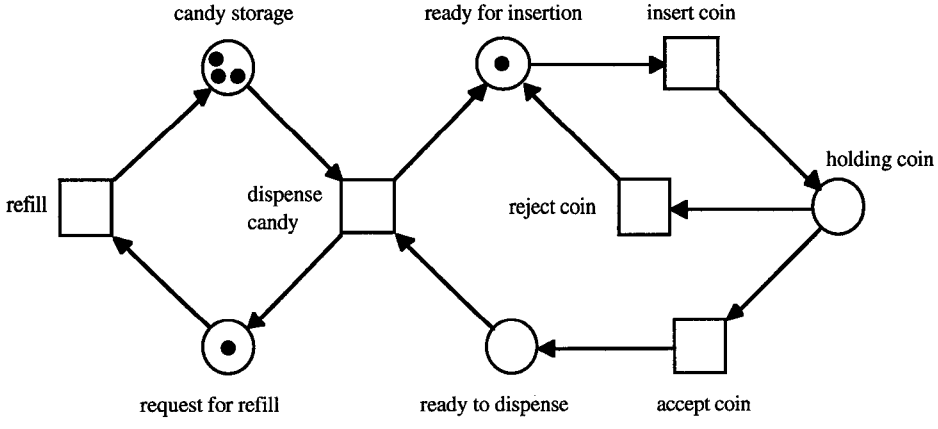

Fig. 1.3 Marking reached after the first candy is dispensed

are enabled. With the given initial marking, the machine can deliver up to four candies without having to refill the storage.

This example can be used to show how Petri nets model a variety of dependency relations between the events of a dynamic system. At the initial marking, the

transition `accept coin` can only occur after `insert coin` has occurred: these two transitions are in *causal* relation. After `insert coin` occurs, both `reject coin` and `accept coin` are enabled, but the occurrence of one of them disables the other: they are in *conflict*. At the marking shown in Figure 1.3, transitions `refill` and `insert coin` can occur independently of each other: they are *concurrent*.

Our vending machine can be seen as composed of a storage unit, which takes care of removing and adding candies to the storage, and a control unit, which takes care of the coins. The storage unit can only deliver a candy if, simultaneously, the control unit changes its state from `ready to dispense` to `ready for insertion`; in other words, the delivery of the candy and this change of state have to be *synchronized*. Figure 1.4 shows the Petri net models of these units. The synchronization is modelled by merging the transitions `dispense candy` of the two units into a single new transition, which has as input (output) places all the input (output) places of the two old transitions. Since – according to the occurrence rule – a transition is enabled if all its input places are marked, the new transition is enabled if the two old transitions are enabled. Moreover, its occurrence has the same effect as the simultaneous occurrences of the old transitions.

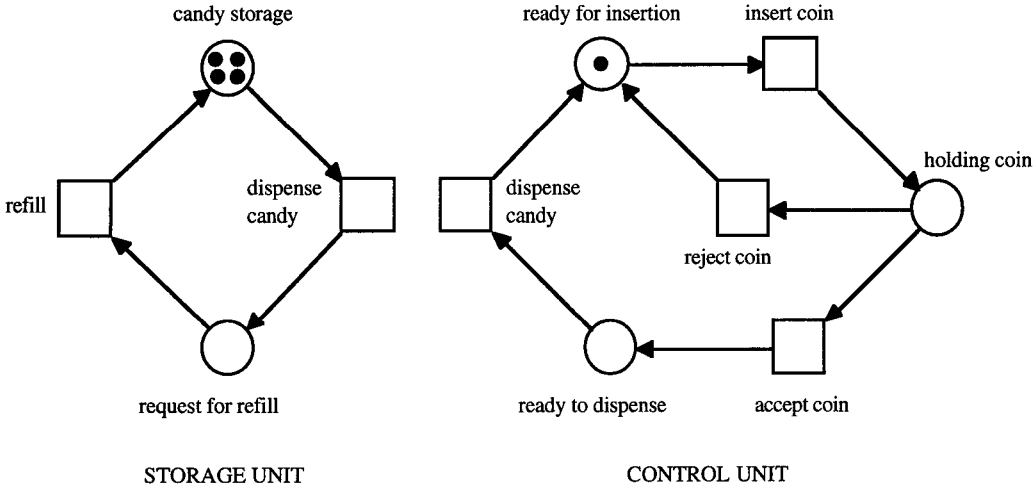

Fig. 1.4 Units of the vending machine

1.2 Free-choice Petri nets

Petri nets have a large expressive power, which makes them suitable to model a rich variety of dynamic systems. As a consequence, the analysis algorithms for arbitrary Petri nets are bound to have a high complexity (when they exist), and it is not possible to develop a comprehensive theory that relates the structure of a Petri net to its behaviour.[2] These obstacles can be removed if we restrict our attention to classes of Petri nets in which – by means of constraints on the graphical structure of the net – certain behaviour is ruled out. In Chapter 3 two of these classes are studied, called S-systems and T-systems. In S-systems, every transition has one input place and one output place, and therefore synchronizations are ruled out. Both the storage unit and the control unit of the vending machine are examples of S-systems[3]. In T-systems, every place has one input transition and one output transition: conflicts are ruled out.[4] The Petri net obtained from the vending machine by removing the transition `reject coin` and its adjacent arcs is a T-system.

The theory of S-systems is very simple. T-systems have been studied since the early seventies, and are today very well understood. These two classes are well within the analyzability border. To get closer to the border, we allow both synchronization and conflict, but in such a way that they do not 'interfere'. A typical situation of interference between synchronization and conflict is shown in the Petri net of Figure 1.5.

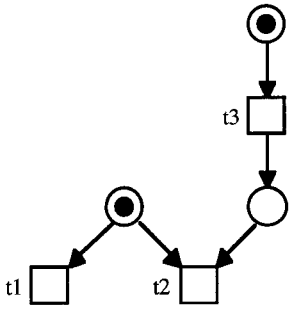

Fig. 1.5 A Petri net in which conflicts and synchronizations interfere

Transitions t_1 and t_2 are not in conflict, because t_2 cannot occur, but will be in conflict if t_3 occurs before t_1. Roughly speaking, due to the synchronization at t_2,

[2]There exist high lower complexity bounds and even undecidability results concerning analysis algorithms for arbitrary Petri nets.

[3]The reason of the name 'S-systems' is that places play in them a more important role than transitions, and places are called *Stellen* in German – the language in which Petri nets were originally defined.

[4]The converse does not hold, see Exercise 1.4.

the transition t_3 influences which one of the transitions t_1 and t_2 can occur. The class of free-choice Petri nets is defined to rule these situations out: in them, the result of the choice between two transitions can never be influenced by the rest of the system – in other words, choices are *free*. The easiest way to enforce this is to keep places with more than one output transition apart from transitions with more than one input place. More precisely, if there is an arc from a place s to a transition t, then either t is the only output transition of s (which implies that t cannot be in conflict with any other transition) or s is the only input place of t (which implies that there is no synchronization at t). In this way, whenever an output transition of s is enabled, *all* output transitions of s are enabled, and therefore the choices in which t takes place are free. The vending machine is an example of a Petri net satisfying this condition.

There is a slightly more general way to achieve the same effect: if there is an arc from a place s to a transition t, then there must be an arc from any input place of t to any output transition of s. We call the Petri nets satisfying this weaker condition *free-choice Petri nets*[5]. The net of Figure 1.6 is free-choice; for every token distribution, either the two transitions t_1 and t_2 are enabled, or none of them is enabled.

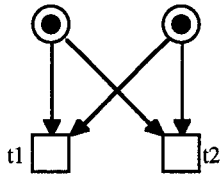

Fig. 1.6 A free-choice Petri net

We show how free-choice Petri nets can be used to model the flow of control in networks of processors, which provides some insight into their expressive power. We model a processor as a computing entity having input and output ports. Processors are connected through unidirectional channels. A channel connects an output port to an input port. Figure 1.7 shows a graphical representation of a processor.

When a processor receives a value through each of its input ports, it computes a result. The processor then selects nondeterministically one of its output ports, and sends the result through all the channels connected to it.

Figure 1.9 shows how to translate a network of processors into a free-choice Petri net. It is easy to see that the behaviour of a network of processors corresponds to

[5]Historically, the Petri nets satisfying the stronger condition have been called *free-choice*, and those satisfying the weaker *extended free-choice*. Since we only consider the weaker condition in this book, the distinction between free-choice and extended free-choice is not necessary.

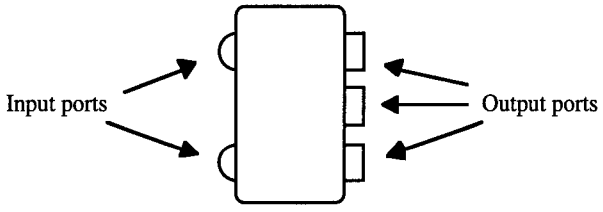

Fig. 1.7 Graphical representation of a processor

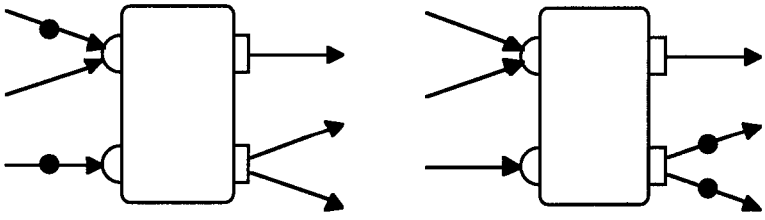

Fig. 1.8 Behaviour of processors

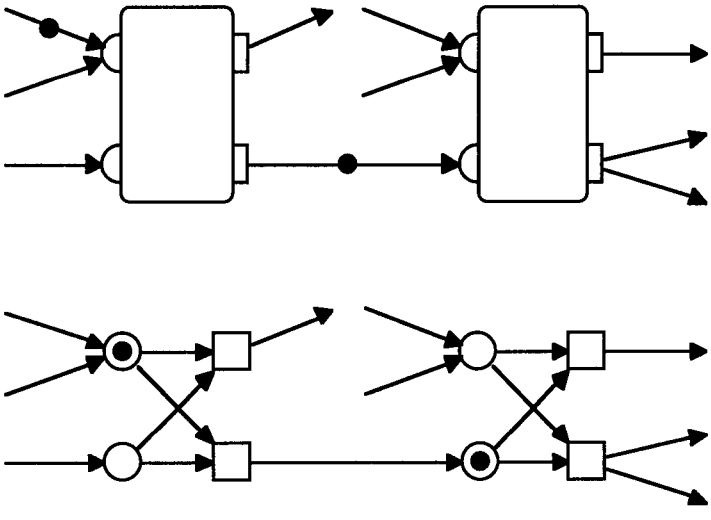

Fig. 1.9 A free-choice Petri net model of networks of processors

the behaviour of the respective free-choice Petri net (notice, however, that we do not model the data manipulated by the processors: for us they are only black dots).

Conversely, every free-choice Petri net can be seen as the Petri net model of a network of processors. The readers are invited to convince themselves of this; it suffices to show that the places and transitions of an arbitrary free-choice Petri net can be grouped into clusters, each of which corresponds to a processor.

1.3 Properties

We describe in this section, in an informal way, some of the properties of Petri nets that are studied throughout the book. The first one we consider is *liveness*. A Petri net is *live* if every transition can always occur again. More precisely, if for every reachable marking (i.e., every marking which can be obtained from the initial marking by successive occurrences of transitions) and every transition t it is possible to reach a marking that enables t. The Petri net model of the vending machine is live, but the Petri net of Figure 1.10 is not live: after the occurrence of the transition t a marking is reached from which t cannot become enabled again.

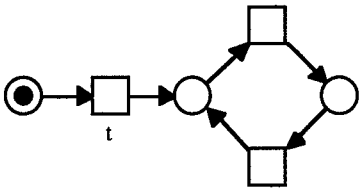

Fig. 1.10 The transition t is not live

Deadlock-freedom is a weaker property than liveness. A Petri net is deadlock-free if every reachable marking enables some transition. The non-live Petri net of Figure 1.10 is deadlock-free.

A Petri net is *bounded* if there exists a number b such that no reachable marking puts more than b tokens in any place. Places in a Petri net are often used to model buffers and registers for the storage of data – this is the case, for instance, of the place `candy storage` of the vending machine. If a Petri net is unbounded, then overflows can occur in these buffers or registers. The vending machine is bounded (no place can ever contain more than four tokens), while the Petri net of Figure 1.11 is unbounded.

A marking is a *home marking* if it is reachable from every reachable marking. A Petri net is *cyclic* if its initial marking is a home marking. The vending machine is an example of a cyclic Petri net. In general, systems which remain in their initial

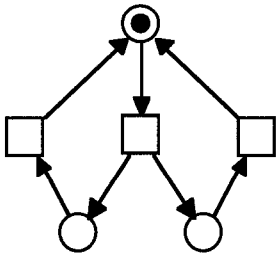

Fig. 1.11 A Petri net in which no place is bounded

state until some user interacts with them, and after the interaction can return to this same state, are modelled by cyclic Petri nets. The Petri nets of Figures 1.10 and 1.11 are not cyclic.

Liveness, boundedness and cyclicity are independent of each other. For instance, there exist Petri nets that are live and bounded but not cyclic. Exercise 1.3 proposes to find Petri nets showing this independence.

Liveness, boundedness, cyclicity, or the reachability of a marking are *behavioural* or *dynamic* properties, i.e., properties of the behaviour of a Petri net, as defined by the rule which governs the occurrence of transitions. In this book, we study the connection between behavioural and *structural* properties for the classes of S-systems, T-systems and free-choice Petri nets. By structural properties we mean those which do not refer to the dynamic aspects of a Petri net, but only to its syntactic description as a graph. For instance, the property 'every circuit of the net contains a place which is marked at the initial marking' is structural. One of the results of Chapter 3 is that a T-system is live if and only if this structural property holds. This is an example of what we call a *structural characterization* of a behavioural property (liveness) for a class of Petri nets (T-systems).

1.4 Structure of the book

Chapter 2 introduces formal definitions and some basic results about arbitrary Petri nets. In particular, it contains five lemmata, namely the Monotonicity, Marking Equation, Exchange, Boundedness, and Reproduction Lemma, and the Strong Connectedness Theorem, all of which are very often used in the next chapters. The chapter also introduces analysis methods for Petri nets based on linear algebra: S- and T-invariants, and the Incidence Matrix.

Chapter 3 studies S- and T-systems. For each of these two classes four theorems are obtained. The first three are structural characterizations of behavioural properties: the Liveness Theorem characterizes the live systems; the Boundedness Theorem characterizes the live systems which are moreover bounded; the Reachability Theorem characterizes the set of reachable markings. The fourth theorem, called the Shortest Sequence Theorem, gives an upper bound on the length of the shortest sequences of transitions that lead to a given marking.

The rest of the chapters develop the theory of free-choice Petri nets. In particular, they generalize the theorems of Chapter 3.

Chapter 4 introduces siphons and traps. They are used to prove Commoner's Theorem, which generalizes the Liveness Theorem for both S- and T-systems. It is shown that deciding non-liveness of free-choice systems is an NP-complete problem.

Chapter 5 contains the S-coverability and T-coverability Theorems, which show that every live and bounded free-choice Petri net can be decomposed into special S-systems and also into special T-systems.

Using these results, Chapter 6 proves the Rank Theorem, which characterizes the free-choice nets which admit a live and bounded marking. It follows from this characterization that live and bounded free-choice Petri nets can be recognized in polynomial time. Another consequence of the Rank Theorem is the Duality Theorem, a classical result of free-choice theory.

Chapter 7 gives reduction rules which reduce all and only free-choice nets which admit a live and bounded marking to very simple nets with just one place and one transition. This provides another algorithm to recognize live and bounded free-choice Petri nets, which is not as efficient as the one of Chapter 6, but gives more information about why a given free-choice Petri net is not live and bounded. The reduction rules can be reversed to yield synthesis rules which generate all and only the live and bounded free-choice Petri nets starting from simple Petri nets.

Chapter 8 studies and characterizes the home markings of live and bounded free-choice Petri nets. It is proved that the problem of deciding if a reachable marking of a live and bounded Petri net is a home marking can be solved in polynomial time.

In Chapter 9, the reachable markings of live, bounded and cyclic Petri nets are characterized. This result generalizes the Reachability Theorem for S- and T-systems. A generalization of the Shortest Sequence Theorems is presented as well.

Finally, Chapter 10 shows how weakened versions of Commoner's Theorem and the Rank Theorem also hold for Petri nets which are not free-choice.