
Introducing Petri nets

M. Silva

1.1 INTRODUCTION

Modern manufacturing systems are highly parallel and distributed. They need to be analyzed from qualitative and quantitative points of view. Qualitative analysis looks for properties like the absence of deadlocks, the absence of (store) overflows, or the presence of certain mutual exclusions in the use of shared resources (e.g. a robot). Its ultimate goal is to prove the correctness of the modeled system. Quantitative analysis looks for performance properties (e.g. throughput), responsiveness properties (e.g. average completion times) or utilization properties (e.g. average queue lengths or utilization rates). In other words, the quantitative analysis concerns the evaluation of the efficiency of the modeled system.

As in many engineering fields, the design of manufacturing systems can be carried out using **models**. Petri nets allow the construction of models amenable both for correctness and efficiency analysis. Moreover they can be implemented using many different techniques (hardware, micro-programmed, software). Because of the graphical nature of net models, they are mostly self-documented specifications, making easier the communication among designers and users. Net models can be used during the entire life cycle of manufacturing systems.

A Petri net (PN), like a differential equation, is a mathematical formalism. Petri nets find their basis in a few simple objects, relations and rules, yet can represent very complex behaviors. More precisely, Petri nets can be considered as a graph theoretic tool specially suited to model and analyze **discrete event dynamic systems (DEDS)** which exhibit parallel evolutions and whose behaviors are characterized by synchronization and sharing phenomena. Their suitability for modeling this type of system has led to their application in a wide range of fields. Examples of such DEDS are communication networks, computer systems and, the purpose of this book, discrete part manufacturing systems.

To be able to use a Petri net for modeling a given type of application, we must enrich it with an adequate interpretation. That is, we must associate

a semantics (i.e. a 'physical' meaning), to the net's entities (places, transitions, tokens), evolution conditions and, eventually, define the actions generated by the evolutions. Broadly speaking, the interpretation gives a meaning to the net system and defines its relationships with the external world (i.e. the interpretation considers the environment in which the net model will be exercised).

The interpretation of graph theoretic tools is nothing new. A graph (in its theoretical sense) is a set of objects (nodes) with relations (see, for example, Deo (1974); Gibbons (1985)). With a graph the connectivity between sites (towns, points in a circuit, ...) can be represented using obvious interpretations. Another kind of interpretation on graphs allow us to model discrete and finite dynamic systems: the nodes represent the states of the system, the arcs represent transitions between states. Particularizing a little more the state-based interpretation, **state diagrams** (SD) (see, for example, Breeding 1989), and **state transition diagrams** (STD) (see, for example, Ajmone *et al.* (1987)) are widely used interpreted graphs: SDs allow the modeling of finite state sequential switching systems, while STDs allow the modeling of homogeneous finite Markov chains. For both formalisms, SDs and STDs, the evolution of the system can be done in **continuous time** (asynchronous state diagrams; state transition rate diagrams) or in **discrete time** (synchronous state graphs; discrete time state transition diagrams).

Provided with adequate interpretations, PNs are able to model 'distributed state diagrams', the control flow of concurrent programs or queuing networks with synchronizations, among other possibilities. The evolution of a fully uninterpreted net system is said to be **autonomous**. An interpreted net system is said to be **non-autonomous** because its evolution depends also on the **state of the environment** considered by the associated interpretation. For example, the **timing** of a net is a particular interpretation by which its evolution depends also on time.

There exists a very rich body of knowledge around Petri nets theory and applications. The purpose of this chapter is to briefly overview in a semi-formal and illustrative way the basic modeling concepts and the main techniques for qualitative analysis. It can be said that Petri nets are suited for parallel systems even more than are automata for sequential systems. Anyhow, the main practical argument for employing PNs should be the use of a graphical, easy to understand single family of formalisms through all the different stages from the design until the implementation and operation.

Although many recent results are integrated in this text, the main line of argument closely follows Silva (1985). The chapter is basically organized in two parts. The first one (up to section 1.5) is devoted to different modeling issues. The second part, sections 1.6 and 1.7, is mainly devoted to qualitative analysis. More precisely, the chapter is structured as follows. Net structure and the dynamics of net systems are introduced in section 1.2.

illustrative way some possible interpretations of net systems models. The existence of pathological behaviors on concurrent systems leads to the introduction of some basic qualitative properties in section 1.5. Their analysis is done in section 1.6, overviewing reachability graph (section 1.6.1), net system reductions (section 1.6.2) and linear algebra techniques (section 1.6.3). Section 1.7 is devoted to some basic net subclasses and their analysis. Obviously, the more restrained the net subclass is, the more powerful the analysis techniques are. Concluding and bibliographical remarks end this introductory presentation.

1.2 NETS AND NET SYSTEMS

A Petri net model of a dynamic system consists of two parts:

1. a **net structure**, a weighted-bipartite directed graph, that represents the static part of the system; and
2. a **marking**, representing a **distributed overall state** on the structure.

The above separation allows one to reason on net-based models at two different levels: **structural** and **behavioral**. Reasoning at the structural level we can derive some 'fast' conclusions on the behavior of the modeled system, relating when possible structural and behavioral properties. Purely behavioral reasonings are computationally very complex.

1.2.1 Net structure (what is a Petri net?)

To model a discrete-event-dynamic system we need to take into account its states and the events leading to the state-evolutions. In net systems the state is described by means of a set of **state variables** representing local conditions. Moreover, net models make explicit the existence of **state-transitions**. Therefore net structures are built on two disjoint sets of objects: **places** (represented as circles), and **transitions** (represented as bars or boxes). Places are the support of the state variables.

Places and transitions are related through a **weighted flow relation**, described by an unweighted flow relation, F , and a weighting function on F , W . Let us now give the formal definitions and see some examples.

Definition 1.1. A Petri net is a four-tuple:

$$N = \langle P, T, F, W \rangle$$

where:

P is a finite non-empty set of $n = |P|$ **places**

T is a finite non-empty set of $m = |T|$ **transitions**

$P \cap T = \emptyset$; i.e. places and transitions are disjoint sets

$F \subset (P \times T) \cup (T \times P)$ is the **flow relation** (set of directed arcs):

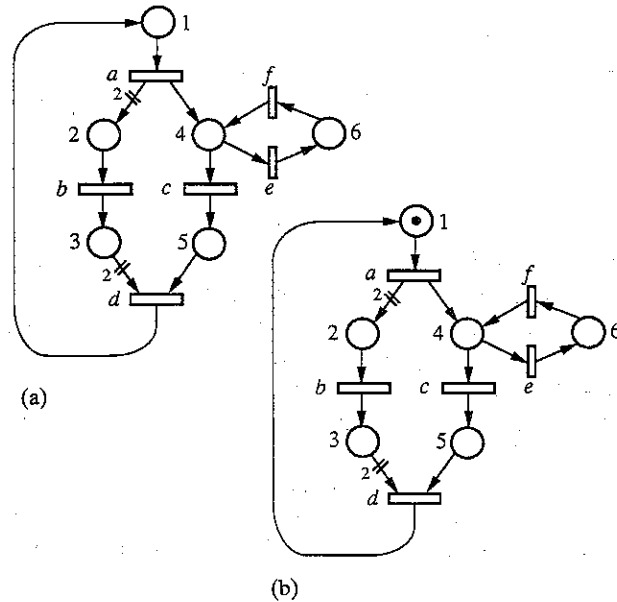


Figure 1.1 Net structure and net system: (a) N , net structure; (b) $\langle N, M_0 \rangle$, net system.

Figure 1.1(a) shows a net structure. Arcs are labeled with natural numbers, $W(p_i, t_j)$ or $W(t_k, p_l)$, the **arc weights**. As will be seen, non-unitary arc weights allow us to model **bulk arrivals** or **bulk services**. By convention, unlabeled arcs are weighted one. All the arc weights in the net of Fig. 1.1 are 1, except for arcs (a, p_2) and (p_3, d) whose weights are 2. In many practical cases there exists neither bulk arrival nor bulk service. Therefore all the arc weights are one. In this case the net is said to be **ordinary**.

A place p is an input (output) place of transition t if there exists an arc going from p to t (output respectively from t to p). In Fig. 1.1, $\{p_3, p_5\}$ are input places of d while $\{p_2, p_4\}$ are output places of a .

An alternative way to see Petri nets is to define the weighted flow relation through two *incidence functions*:

Definition 1.1(a). A **Petri net** is a four-tuple:

$$N = \langle P, T, Pre, Post \rangle$$

where:

P and T are disjoint, finite, non-empty sets of places and transitions, respectively

$Pre: P \times T \rightarrow N$ is the **pre-incidence** or **input function**

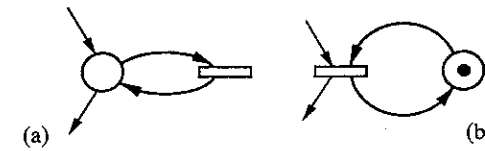


Figure 1.2 Two self-loops.

There is an arc going from the place p_i to the transition t_j iff $Pre(p_i, t_j) \neq 0$. Similarly, there is an arc going from transition t_k to place p_l iff $Post(t_k, p_l) \neq 0$. The arc weight, $Pre(p_i, t_j) = W(p_i, t_j)$ or $Post(t_k, p_l) = W(t_k, p_l)$, labels the corresponding arc. The **pre-** and **post-set** of transition $t \in T$ are defined respectively as ${}^*t = \{p \mid Pre(p, t) > 0\}$ and $t^* = \{p \mid Post(t, p) > 0\}$. The **pre-** and **post-set** of a place $p \in P$ are defined respectively as ${}^*p = \{t \mid Post(t, p) > 0\}$ and $p^* = \{t \mid Pre(p, t) > 0\}$.

A practical way of representing the net structure is to use **incidence matrices**. The incidence functions can be represented by means of **pre-** and **post-incidence matrices**, Pre - and $Post$ -, both having $n = |P|$ rows and $m = |T|$ columns.

The pre- and post-incidence matrices of the net in Fig. 1.1(a) are as follows:

$$Pre = \begin{matrix} & \begin{matrix} a & b & c & d & e & f \end{matrix} \\ \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} & \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \end{matrix} \quad \text{and} \quad Post = \begin{matrix} & \begin{matrix} a & b & c & d & e & f \end{matrix} \\ \begin{matrix} p_1 \\ p_2 \\ p_3 \\ p_4 \\ p_5 \\ p_6 \end{matrix} & \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \end{matrix}$$

A pair of place p and transition t is called a **self-loop** if p is both an input and output place of t . A Petri net is said to be **pure** if it has no self-loops.

Figure 1.2 shows two self-loops. A self-loop can be easily eliminated (e.g. by expanding the transition into a sequence: initial transition – intermediate place – final transition). Pure nets are completely characterized by the (single) *incidence matrix*:

$$C = Post - Pre$$

Positive (negative) entries in C represent the post- (pre-) incidence function. If the net is not pure, the incidence matrix ‘does not see’ the self-loops.

1.2.2 Net systems: marking and token game

The structure of a net is something static. Assuming that the behavior of

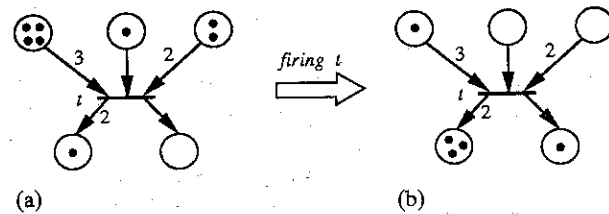


Figure 1.3 Firing transition t : marking evolution.

dynamics of a net structure are created by defining its **marking** and **marking evolution rule**.

Definition 1.2. The marking M of a net N is an application of P on N , i.e. the assignment of a non-negative integer (number of tokens) to each place.

Definition 1.3. A **marked Petri net or net system** is the couple $\langle N, M_0 \rangle$, where N is a Petri net and M_0 is an initial marking.

The number of tokens at a place represents the **local state** of the place (i.e. the value of the state variable). The state of the overall net system is defined by the collection of local states of the places. A marking M is denoted as an $n = |P|$ vector whose p th component, $M(p)$, represents the number of tokens in place p . The vector M is the **state-vector** of the discrete event dynamic system described by the net system. Pictorially, we place $M(p)$ black dots (tokens) in the circle representing place p . Figure 1.1(b) represents a net system with an initial marking $M = (1, 0, 0, 0, 0)^T$.

Once the distributed state is defined, the question is: how does a net system work? The evolution is defined through a **firing** or **occurrence rule**, informally named the '**token game**'. This is because net structures can be seen as 'special checkers', the tokens as 'markers' and the firing rule as the 'game rule'. Transitions represent potential moves in the 'token game'.

Definition 1.4 (token game). A marking in a net system evolves according to the following **firing** (or **occurrence**) rule:

1. A transition is said to be **enabled at a given marking** if each input place has at least as many tokens as the weight of the arc joining them.
2. The **firing** or **occurrence** of an enabled transition is an instantaneous operation that removes from (adds to) each input (output) place a number of tokens equal to the weight of the arc joining the place (transition) to the transition (place).

The pre-condition of a transition can be seen as the **resources required** for the transition to be fired. The post-condition represents the **resources produced** by the firing of the transition.

Transition t (Fig. 1.3(a)) is enabled. Its firing leads to the marking in Fig.

system of Fig. 1.1(b) is a . Its firing leads to the marking $M = (0, 2, 0, 1, 0, 0)^T$, where b , c and e are now enabled.

An important remark concerning the firing rule on our abstract model is that **enabled transitions are never forced to fire**. This is a form of **non-determinism**. In practical modeling the interpretation partially governs the firing of enabled transitions (e.g. depending on whether or not an external event associated to an enabled transition occurs). In section 1.3 we will come back to this important issue.

The enabling and firing of a transition can be represented in a very convenient way using incidence matrices and marking vectors. Let us denote the columns associated to t in the different incidence matrices as $Pre(t)$, $Post(t)$ and $C(t)$:

1. Transition t is enabled at M iff $M \geq Pre(t)$ (1.1)
2. Denoting as $M_1[t] M_2$ the fact that M_2 is reached by firing t at M_1 (M_1 enables t):

$$M_2 = M_1 + Post(t) - Pre(t) = M_1 + C(t) \quad (1.2)$$

Assuming N to be pure (otherwise it can be easily transformed), it is not difficult to derive the following:

$$M_1[t] M_2 \Leftrightarrow M_2 = M_1 + C \cdot e_t \geq 0 \quad (1.3)$$

where e_t is the **characteristic vector** of t : $e_t(x) :=$ if $x = t$ then 1 else 0

The right-hand side of the equivalence in eq. (1.3) is clearly a **state equation**: M_1 is the **present state**, M_2 the **next state**, e_t the input vector. Unfortunately classical control theory is not of great help to us when studying the dynamic behavior of net systems: the state (marking) and input vectors should take their values on non-negative integers.

Integrating the state equation from M_0 along a firing sequence $\sigma = t_i t_j \dots$ leading to M_k (M_k is said to be reached from M_0 by means of σ) we can write:

$$M_0[\sigma] M_k \Rightarrow M_k = M_0 + C \cdot \bar{\sigma} \geq 0, \bar{\sigma} \geq 0 \quad (1.4)$$

where $\bar{\sigma}$ is the **firing count vector** of σ : $\bar{\sigma}(t)$ is the number of times t has been fired in σ .

Equation (1.4) is called the **fundamental equation** or, more frequently, the **state equation** of the net system. (**Remark**: properly speaking the **state equation** is eq. (1.3), while eq. (1.4) is the **transition equation** in control theory terminology.)

The most important remark now is that only the right-hand implication exists in eq. (1.4). Otherwise stated, unfortunately a non-negative integer solution $\bar{\sigma} \geq 0$ of $M_k = M_0 + C \cdot \bar{\sigma} \geq 0$ does not imply there exists a σ such that M_k is reachable from M_0 (i.e. does not imply $M_0[\sigma] M_k$). For example, assuming $M_0 = 0$ for the net in Fig. 1.1(a), $\bar{\sigma} = (1, 1, 1, 1, 0, 0)^T$

integer couple $M_k = 0$, $\bar{\sigma} = (1, 1, 1, 1, 0, 0)^T$ is called a **spurious solution** of the state equation. The existence of spurious solutions is the main problem for the analysis of net systems using linear algebra techniques. However, many practical analysis results can be obtained using these techniques (see later sections 1.6.3 and 1.7.2).

1.3 ON MODELING FEATURES

Petri nets, as introduced thus far, are a mathematical formalism. This section presents a number of features which – in our opinion – make nets an interesting modeling formalism, specially suited for discrete-event dynamic systems with concurrent or parallel events and activities.

The considerations in this section are general, i.e. still on the abstract formalism, valid for any particular interpretation. Before concentrating on our main issue here, **on practical modeling**, it is important to highlight the fact that nets allow a natural **graphical** representation that makes them very much appreciated in engineering circles ('a picture is worth a thousand words!'). Nevertheless, big and not well-structured net models are difficult to understand and analyze. This means in practice that good modeling disciplines are very important.

As a preliminary remark on practical modeling, the reader can easily check the simplicity of representing with nets three basic modeling notions: **causal dependence** (e.g. sequence), **conflict** (decision, choice) and **concurrency**. Going back to our net system in Fig. 1.1(b) it is obvious that the firing of f must be done *after* (causal dependence) that of e . Also, it is clear that twice b and c must precede the firing of d . Moreover, c and e define a conflict. From any marking with $M(p_4) = 1$, transitions c and e are simultaneously enabled, but they cannot be simultaneously fired: **a decision must solve the conflict**.

As already mentioned, a major feature of nets is that they do not define in any way **how** and **when** (i.e. time independence) a given conflict should be solved, leading to **non-determinism** on its behavior.

Sequence and **conflict** are classical notions in sequential systems (e.g. in finite automata). **Concurrency** is a third concept that net systems represent in an extremely natural way. Informally speaking two transitions are concurrent at a given marking if they can be fired 'at the same time', i.e. simultaneously. Once transition a is fired in the net system at Fig. 1.1(b), the marking $M = (0, 2, 0, 1, 0, 0, 0)^T$ is reached. Then transitions b and c can be fired simultaneously. Moreover, because $M(p_2) = 2$, transition b can be fired concurrently to itself (idea of **re-entrancy**): **self-concurrency**.

Synchronizations are very important in the modeling of distributed and concurrent systems. How are synchronizations modeled with nets? Basi-

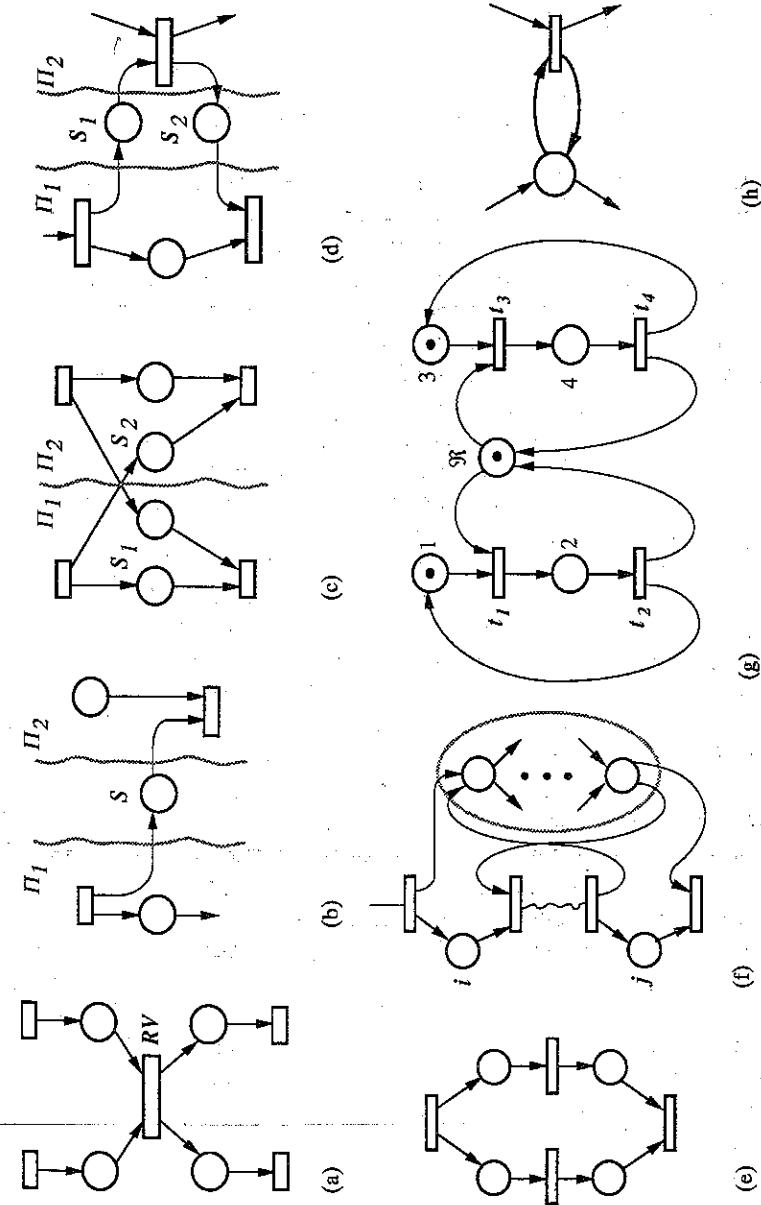


Figure 1.4 Typical synchronization schemes: (a) rendezvous, RV; (b) semaphore, S; (c) symmetric RV/semaphore; (d) asymmetric RV/semaphore (master/slave); (e) fork-join; (f) subprogram (p, p), are in mutual exclusion, mutex); (g) shared-resource (\mathcal{R}); (h) guard (condition reading).

may appear, even if a transition has only one input place, when the arc is weighted (e.g. assuming p_5 was not present in Fig. 1.1, transition d would also represent a synchronization because its firing would require the presence of two tokens on p_3).

Figure 1.4 is self-explanatory. In all cases nets are ordinary (thus synchronizations are on transitions with more than one input place). Just two remarks: (1) the correct behavior of schema (f) is based on the fact that both p_1 and p_3 cannot be simultaneously marked (i.e. they must be in *mutual exclusion*) and (2) the resource \mathcal{R} of schema (g) can be used in place p_2 or in place p_4 , but simultaneous use is impossible (i.e. the use in p_2 and p_4 is in mutual exclusion).

The separation in a bipartite structure and a marking makes the net-based approach very powerful for modeling purposes. In particular, the dichotomy places/transitions leads to a treatment of states and actions on an equal footing. This makes – in our opinion – nets superior to either purely state- or purely transition-oriented formalisms where one of the notions is explicit and the other has to be deduced.

The existence of a **locality principle** on states and actions (transitions) in net models is a direct consequence of its bipartite structure and marking definition. The importance of the locality principle resides in the fact that net models can be **locally modified, refined** or **made coarse**, without altering the rest of the model. This means, in particular, that nets can be synthesized using **top-down** and **bottom-up** approaches. Top-down synthesis is any procedure that, starting with an initial (very abstract) model, leads to the final model through **stepwise refinements**. In a bottom-up approach **modules** are produced, possibly in parallel by different groups of designers, and later **composed**. Restricting the many possible refinements and compositions strategies, we just mention here **place** and **transition refinements** and compositions through merging of transitions (i.e. **synchronization of modules**) and merging of places (i.e. **fusion of modules**). The net in Fig. 1.5 shows a two-level **hierarchical refinement**: p_5 (that defines local states) and θ_2 are refined. The net system in Fig. 1.1 can be obtained synchronizing two modules (Fig. 1.6(a), **synchronizing** transitions a and d) or fusing two modules (Fig. 1.6(b), **fusing** the places p_4).

Summarizing, still at an abstract level, net systems have the following practical features for modeling:

1. **Graphical and equational** representations. Therefore, net systems enjoy some comparative advantages for documentation and analytical studies.
2. Natural expression of **causal dependences, conflicts and concurrency**.
3. Simple, appealing and powerful **synchronization** mechanism making natural the construction of **mutual exclusion** constraints.
4. **Locality of states and actions** which allows the **hierarchical** and the

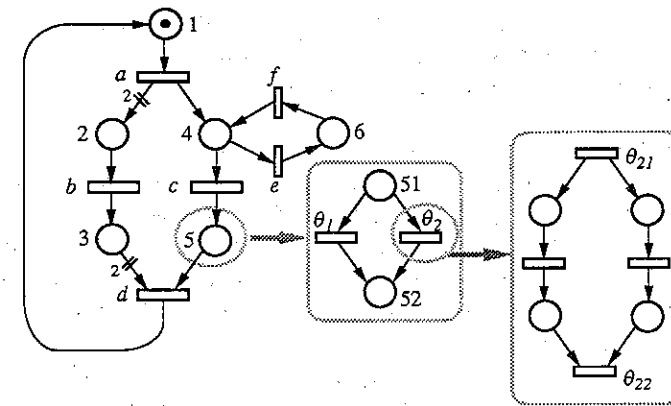


Figure 1.5 Hierarchical definition of a net system.

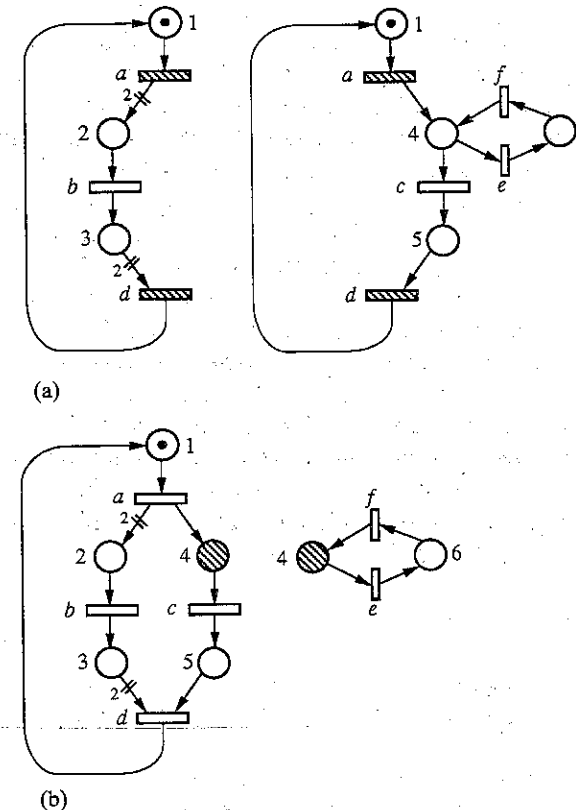


Figure 1.6 Two modular ways of constructing the net in Fig. 1.1: (a) synchronization; (b) fusion.

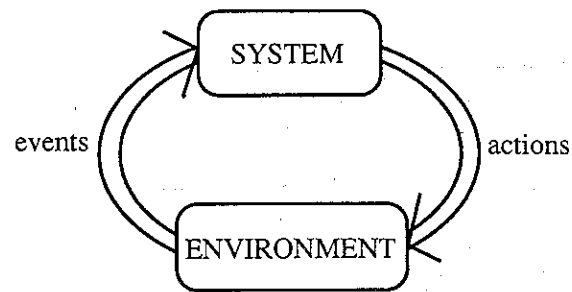


Figure 1.7 Modeled system and its environment mutually interact.

1.4 ON NET SYSTEMS INTERPRETATIONS

A Petri net can be used to model a discrete event dynamic systems assigning a **meaning** to its associated elements (places, transitions and tokens) and relating explicitly the modeled system and its **environment** (Fig. 1.7). In general the behavior of a system is influenced by the environment (through events in our case), while the actions generated by the system influence the behavior of its environment. Therefore, to interpret a net system is to establish a convention which defines:

1. The meaning of places, transitions and tokens.
2. A meaning for the conditions which govern the transition firing. The marking evolution rule is slightly modified by the interpretation, which also becomes a function of the behavior of the modeled system's **environment**.
3. The actions generated by the model.

If the behavior of a net system is not influenced by the environment, it is said to be **autonomous**. Non-autonomous net systems have more constrained behavior than the underlying autonomous net system.

The purpose of this informally written section is not to fix 'good' interpretations, but to show the existence of many possible ones, even for a given class of problems or application domain. Therefore, the reader should not be very much worried about technical details.

Section 1.4.1 introduces two different interpretations, one for modeling the control part of concurrent programs, the second generalizes the classical **state diagram** formalism (see, for example, Ercegovac and Lang (1985); Breeding (1989)) useful to model sequential switching systems. To clearly differentiate PN's as uninterpreted models from their different interpretations, **marking flow charts** and **marking diagrams** are the names given to

1.4.1 Marking flow charts and marking diagrams

The environment of a net modeled system can affect the behavior of the net model and vice versa. Usually the interaction is done through:

- (a) **Events** and/or **predicates** over some 'external states'. These 'guard' the firing of transitions (from the environment to the net modeled system).
- (b) **Actions** that, generated by the net modeled system, cause the state of the environment to 'change'.

Depending more precisely on the application domain (software, hardware controllers, logical automatisms, etc.), many interpretations exist. In some cases, actions are associated with the **firing** of transitions (as in **Mealy-Automata**); in other cases actions are associated with the **marking** of places (as in **Moore-Automata**).

Even for a given application domain, there exist many possible interpretations. Thus no formal or rigid definition of interpretations will be given. Only two cases are considered for illustration purposes. First there is a collection of comments on what net interpretations look like to model the control part of concurrent software systems. Later a discrete-event-controller (a production cell with two machines, one robot and a store) is modeled using marking diagrams.

(a) Marking flow charts

In modeling **software** the more natural interpretation is based on the classical **control part (CP)**–**operative part (OP)** decomposition. Using the CP–OP decomposition principle, the state of a program (sequential or concurrent) can be considered as the concatenation of a **control state** (for the CP) and a **data state** (for the OP). Places will represent parts of the control state (defined by the marking of the net), while predicates over the data state are associated with the transitions. Predicates allow decisions to be taken (i.e. conflicts to be solved).

Therefore the firing of transitions is governed by the net marking and predicates on the data. Data transformations can be associated with the firing of a transition (usually implemented by means of sequential modules activated by the firing of the transition).

For a given program, the **execution of an instruction** (at a more abstract level a **block** of instructions or **module**) will be represented by the firing of a transition. The preincidence (input) function of each transition determines a condition which must be fulfilled for the instruction (or the block) to be executed. In general, with each transition a label, consisting of a 'predicate over data/data transformation' couple, is associated. If a transition is not conditioned by a predicate, that field is omitted in the

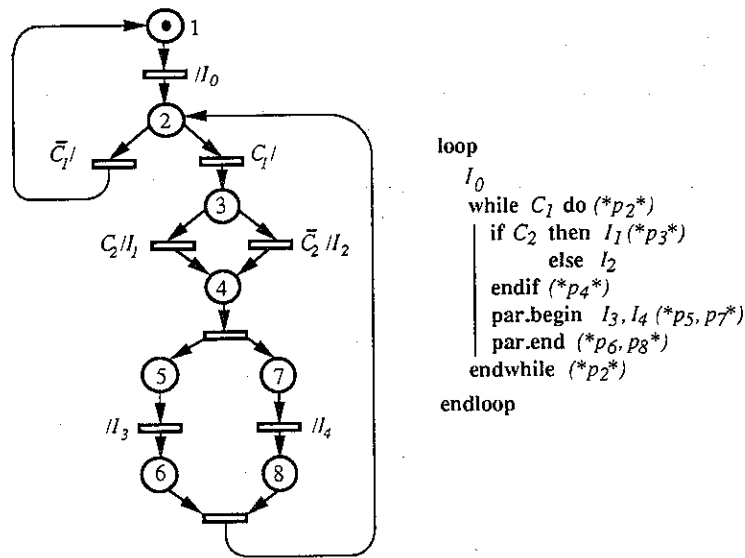


Figure 1.8 Flow control representation of a simple parallel-PASCAL-like program.

corresponding field is omitted. The complete condition for firing a transition is the intersection of the net system enabling condition and, possibly, the associated **predicate over data**.

According to the above comments, at a certain level of abstraction the interpreted net system models the control part, while the operative part is implemented through the modules associated with the firing of transitions. Figure 1.8 is practically self-explanatory.

(b) Marking diagrams

In modeling **discrete-event-controllers**, **state diagrams** (see, for example, Ercegovac and Lang (1985); Breeding (1989)) are classical interpretations of graphs allowing the modeling of **finite sequential switching automata**.

In state diagrams, the nodes of the graph represent the states, while the arcs are labeled with external events and external state conditions. Actions are associated with the transitions (arcs) or the states (nodes). Eventually actions may be conditioned by the external state. State diagram interpretations can be easily applied to Petri net systems leading to some marking diagrams (MD): (a) events and Boolean functions of external variables guard the firing of transitions, and (b) actions can be associated with transitions (level-actions) or to the marking of places.

Let us concentrate now on our manufacturing domain through an example.

consumer (MACHINE 2) schema with a **mutual exclusion semaphore** (R , a robot) (Fig. 1.9(a)). Many production systems can be constructed concatenating different stages of such schema.

The following behavior is assumed for the cell (Fig. 1.9(b)). Raw parts arrive through a conveyor. The arrival of a part is detected by a presence (e.g. photoelectric) sensor: $I_1 = 1$ iff a part is present. When a raw part is present, MACHINE 1 is not loaded and the robot is free, it proceeds to load the machine (*load*; *el*: end_of_load). The machine performs operations op_1 and waits for deposit in the buffer (*wait-dep.*).

The *deposit* is done when there is an empty niche in the buffer and the robot is free again. End_of_deposit, *ed*, is represented by a transition. MACHINE 2 proceeds in an analogous way, but once op_2 has finished, *eop₂* (*end_of_op₂*) waits for the robot to perform the *unloading*, assuming the second sensor detects that there is any part at the beginning of the finished parts conveyor ($I_2 = 0$ iff the conveyor is free).

The net system model in Figure 1.9(b) (let us call these interpreted net models marking diagrams) specifies the above behavior. It is labeled with external conditions at transitions (I_1 label t_1 , I_2 label t_2), and actions at places (*load*, op_1 , *deposit*, op_2 , *unload*, *withdrawal*).

1.4.2 Timed net systems

Uninterpreted Petri nets do not include any notion of time and are aimed to model only the logical behavior of systems by describing the causal relations existing between events. The introduction of a timing specification is essential if we want to use this class of model to consider **performance**, **scheduling** or **real-time control** problems.

Timing and firing process. Since Petri nets are bipartite graphs, historically there have been two ways of introducing the concept of time in them, namely, associating a time interpretation with either places or transitions. Because transitions represent activities that change the state (marking) of the net, it seems 'natural' to associate a duration with these activities (transitions). In order to solve conflicts between transitions many authors tend to define a 'timed firing' of transitions in **three phases**: a first instantaneous phase in which an enabled transition removes tokens from its input places, then a **timed phase** in which the transitions are 'working', and a final instantaneous phase in which tokens are deposited into the output places. If we want to model **pre-emption** of activities after their starting, however, we are forced to associate an enabling time with transitions and define **atomic firing**. In this way conflicts should be solved at the end of the delay of enabled transitions. The solution is always in favor of the first transition that elapses its firing time among the conflicting transitions (Ajmone *et al.*, 1989).

In any case, from the above discussion it follows that the only effect of

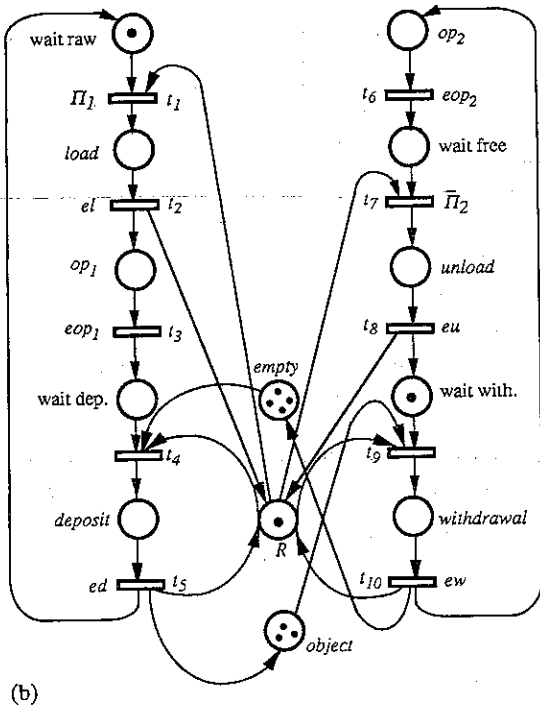
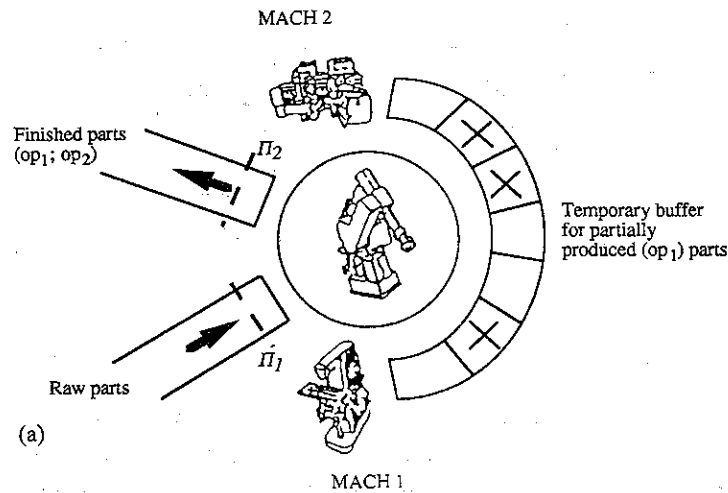


Figure 1.9 A production cell with two machines, one robot and a store:
 (a) Schema of a manufacturing cell; (b) Net system specifying the behavior.

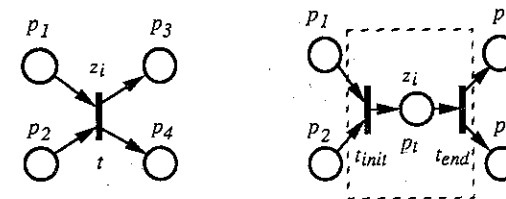


Figure 1.10 Visualization of the 3 phases firing of a timed transition: place p_i will hold each token for z_i time units.

to the different implications that the choices have on the **resolution of conflicts**.

Let us now give in a more precise way two alternative definitions for **deterministically** timed Petri nets. The reader is left to reflect on the relationship between the two. For illustration purposes only **three phases timed firing** is now considered.

Definition 1.5. A (deterministically) transition-timed Petri net (t -TPN) is a couple $\langle N, Z \rangle$ such that $N = \langle P, T, Pre, Post \rangle$ and Z is a function which assigns a non-negative real number, z_i , to each transition in the net: $Z: T \rightarrow \mathbf{R}^+$; $z_i = Z(t_i)$ is called the **firing time** of the transition t_i .

The marking evolution rule of t -TPNs is nearly identical to that of a PN. The only difference is that firing t_i takes z_i time units. Adopting the three phases approach (Fig. 1.10):

1. When the transition t is enabled, a firing is initiated. Conflicts are non-deterministically solved as in the untimed net system. In this phase $Pre(p, t)$ tokens disappear from each input place of t .
2. The firing process (which represents an operation) remains for z_i time units.
3. When the z_i time units have elapsed, the firing ends. In this phase $Post(t, p)$ tokens are added to each output place of t .

Definition 1.5(bis). A (deterministically) place-timed Petri Net (p -TPN) is a couple $\langle N, R \rangle$ such that $N = \langle P, T, Pre, Post \rangle$ and R is a function which assigns a non-negative real number r_i to each place in the net: $R: P \rightarrow \mathbf{R}^+$; $r_i = R(p_i)$ is the minimum **residence time** of a token in p_i .

A token in a p -TPN can be in either of two states: **ready** or **not ready**. If the tokens are ready the marking evolution rule is the same as for an autonomous net system. Not ready tokens do not enable transitions, as though they were not present yet. When a token reaches a place, it goes into the non-ready state, and becomes ready again after an interval of $r_i = R(p_i)$ time units.

In basic timed PN models, conflict resolution strategies are not specified:

Single versus multiple server semantics: degree of self-concurrency. A possible source of confusion in the definition of any timed net model is related to the **self-concurrency** (or **re-entrance**) of a transition. In the case of timing associated with places, it seems quite natural to define **unavailability time** which is independent of the total number of tokens already present in the place. This can be interpreted as an **'infinite server'** policy from the **queuing theory** perspective. In the case of time associated with transitions, the adopted semantics is less obvious. Assume that transition t is k -enabled at a given marking. Then either one firing of t occurs at that time or k firings occur in parallel (i.e. the idea of one or more servers).

Thus **single server** and **infinite server** semantics can be considered in transition-timed net models. Of course an infinite server transition can always be constrained to a ' k -server' behavior by just adding a self-loop place around the transition (i.e. an input and output place) with k -tokens (e.g. the self-loop place in Fig. 1.2(b) constrains the transition to a single server). Therefore, the infinite server semantic appears to be the most general one. However, this generality of the infinite server assumption is usually paid in terms of complexity of the analysis algorithms (e.g. for computing performance figures).

Some comments on stochastic net systems. Definitions 1.5 and 1.5(bis) consider only deterministic timing of transitions. In many cases the timing is not deterministic, being characterized by the probability distribution function (PDF) of a random variable. In this case **stochastic net systems** are defined. The most usual approach is to consider one-phase firing transition-timed models. Stochastic Petri nets are defined in Chapter 4. The following informal comments (that can be skipped without affecting the comprehension of the rest of the material) try to point out that stochastic Petri nets can be viewed as queuing networks provided with a semantically simple and formal way of introducing synchronizations among queues.

The consideration of stochastic Petri nets as mentioned above is limited in practice by the fact that routing probabilities' are not naturally expressible when one-phase firing is assumed for transitions. The advantage of one-phase over three-phase firing is that tokens do not 'disappear' as in the second phase of the three-phase semantics; thus token (e.g. customers) conservation laws of the uninterpreted net system are preserved.

Generalized stochastic Petri nets (GSPN) have (one-phase) stochastically timed transitions and immediate transitions. Immediate transitions fire in zero time (i.e. instantaneous firing). Conflicts among immediate transitions are solved, among other mechanisms, using routing probability schemas. Because a single-phase firing semantics is used, immediate transitions are **prioritized** with respect to timed transitions (see Ajmone *et al.* (1984) for the seminal definition of GSPNs, and Ajmone *et al.* (1989)).

From a conceptual point of view the modeling power of GSPNs is greater than **extended queuing networks (EQN)**, that were introduced to partially

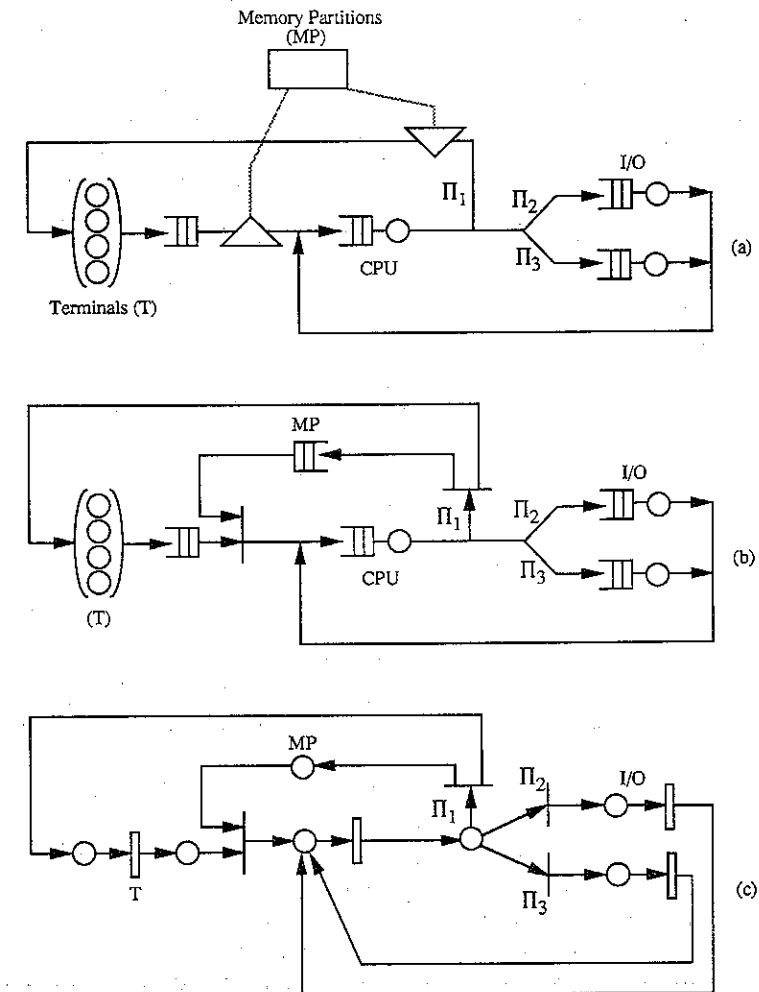


Figure 1.11 Central server representations (Mailles, 1987): (a) extended queuing network; (b) Descriptive queuing network; (c) Generalized stochastic Petri net (a free choice net with three conflicting immediate transitions: Π_1, Π_2, Π_3).

More precisely EQNs allow us to express particular synchronization schemas as:

- fork and joins
- passive resources
- replicated customers.

Figure 1.11 (Mailles, 1987) is almost self-explanatory. As a summary,

with a stochastic interpretation or, as sometimes more convenient (Campos *et al.*, 1991), **synchronized queuing networks (SQNs)**.

1.5 APPROACHING CONCURRENCY QUALITATIVE PROBLEMS

Concurrent and distributed systems are usually difficult to manage and understand. Thus misunderstanding and mistakes are frequent during the design cycle.

A way of cutting down the cost and duration of the design process is to express in a formalized way properties the system should enjoy and to use formal proof techniques. Errors could eventually be detected close to the moment they are introduced, reducing their propagation to subsequent stages.

Only a few qualitative properties will be considered in this introductory chapter.* They are general in the sense that they are meaningful for any concurrent system, not only for those modeled with Petri nets. Nevertheless, their statements with Petri net concepts and objects make them especially 'easy to understand' in many cases. The properties to be considered are:

1. **Boundedness**, characterizing finiteness of the state space.
2. **Liveness**, related to potential firability in all reachable markings. **Deadlock-freeness** is a weaker condition in which only global infinite activity (i.e. firability) of the net system model is guaranteed, even if some parts of it do not work at all.
3. **Reversibility**, characterizing recoverability of the initial marking from any reachable marking.
4. **Mutual exclusion**, dealing with the impossibility of simultaneous **submarkings** (p -mutex) or **firing concurrency** (t -mutex).

Let us consider the net in Fig. 1.12(a). The firing of t_2 allows us to reach the marking $M = (0, 0, 1, 1)^T$ (i.e. p_3 and p_4 have one token). Firing now t_4 , $M^1 = (1, 0, 1, 0)^T$ is reached. Repeating ω times the sequence t_2t_4 the marking $M^\omega = (1, 0, \omega, 0)^T$ is reached. So the marking of p_3 can be arbitrarily high. In practice the capacity of the physical element represented by p_3 should be finite, so an **overflow** can appear. Place p_3 is said to be **unbounded**. Attention must be paid to the above situation because unboundedness can be a pathological situation. System boundedness (i.e. all places bounded) is a good behavioral property.

The maximum number of tokens a place may contain is its (marking) **bound**. A place is bounded if its bound is finite. A net system is bounded if each place is bounded.

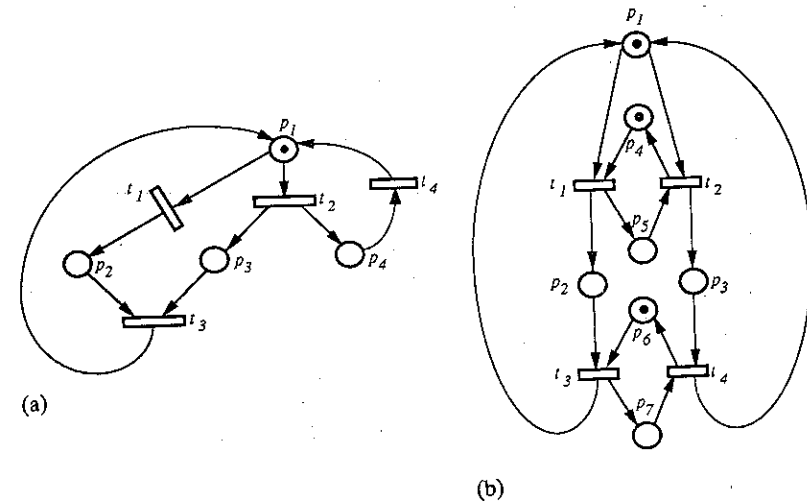


Figure 1.12 On qualitative pathological behaviors: (a) an unbounded, deadlockable (non-live), non-reversible net system; (b) Increasing the initial marking (e.g. $M_0(p_5) = 1$) the live net system is killed!

For any initial marking we can define on the net structure of Fig. 1.1(a) the following token conservation laws hold:

$$\begin{aligned} 2M(p_1) + M(p_2) + M(p_3) &= 2M_0(p_1) + M_0(p_2) + M_0(p_3) = K_1(M_0) \\ M(p_1) + M(p_4) + M(p_5) + M(p_6) &= M_0(p_1) + M_0(p_4) + \\ &M_0(p_5) + M_0(p_6) = K_2(M_0) \end{aligned}$$

where M_0 is the initial marking and M any reachable marking. Therefore:

$$\begin{aligned} M(p_1) &\leq \min(K_1(M_0)/2, K_2(M_0)) \\ M(p_i) &\leq K_1(M_0) \quad i = 2, 3 \\ M(p_j) &\leq K_2(M_0) \quad j = 4, 5, 6 \end{aligned}$$

The above inequalities mean that for any M_0 the net system is bounded. This property, stronger than boundedness, is called **structural boundedness** because it holds independently of the initial marking (only finiteness of M_0 is assumed).

Let us now fire t_1 from the marking in Fig. 1.12(a). After that, no transition can be fired: a **total deadlock** has been reached. A net system is said to be **deadlock-free** (i.e. from any reachable marking) if at least one transition can always be fired. A stronger condition than deadlock-freeness is **liveness**. A transition t is potentially firable at a given marking M if there exists a transition firing sequence σ leading to a marking M' in which t is enabled (i.e. $M[\sigma] M' \geq \text{Pre}(t)$). A transition is **live** if it is potentially firable in all reachable markings. In other words, a transition is live if it never loses the possibility of firing (i.e. of performing some activity). A net

For any initial marking we can define on the net structure in Fig. 1.12(a) non-liveness holds (in fact, a total deadlock can always be reached). Non-liveness for arbitrary initial markings reflects a pathology of the net structure: **structural non-liveness**. A net is structurally live if there exists at least one live initial marking.

A paradoxical behavior of concurrent systems is the following: at first glance it may be accepted as intuitive that increasing the initial marking (e.g. increasing the number of resources) of a net system 'helps' in making it live. The live net system in Fig. 1.12(b) shows that increasing the number of resources can lead to deadlock situations: adding a token to p_5 , t_2 can be fired and a deadlock is reached!

Another interesting property is **reversibility**. A net system is reversible if it is always possible to return to the initial marking (i.e. it is reachable from any other reachable marking). The net system in Fig. 1.12(a) is not reversible. In fact if a total deadlock exists at some reachable marking, the net system cannot be reversible; the reverse is not true as is pointed out in Fig. 1.13(6), where the net system is not reversible but live, thus deadlock-free.

Liveness, boundedness and reversibility are just three different 'good' behavior properties that may be interesting to study in a net system. Figure 1.13 shows examples of the eight cases we may have. Therefore boundedness, liveness and reversibility are independent properties.

The last basic property we introduce in this section is **mutual exclusion**. This property captures constraints like the impossibility of a simultaneous access by two robots to a single store. Two places (transitions) are in mutual exclusion if they can never be simultaneously marked (fired). For the net system in Fig. 1.4(g) we can write: $M(p_2) + M(p_4) + M(\mathcal{R}) = 1$. Thus

$$\begin{aligned} M(p_2) = 1 &\Rightarrow M(p_4) = M(\mathcal{R}) = 0 \\ M(p_4) = 1 &\Rightarrow M(p_2) = M(\mathcal{R}) = 0 \end{aligned}$$

and $[M(p_2) = 0]$ or $[M(p_4) = 0]$ is true for every reachable marking (i.e. p_2 and p_4 are in mutual exclusion). Table 1.1 summarizes the definitions of the different properties we introduced in this section.

1.6 QUALITATIVE ANALYSIS OF NET SYSTEM MODELS

Techniques for analyzing net systems can be divided into the following groups:

1. analysis by enumeration
2. analysis by transformation
3. structural analysis

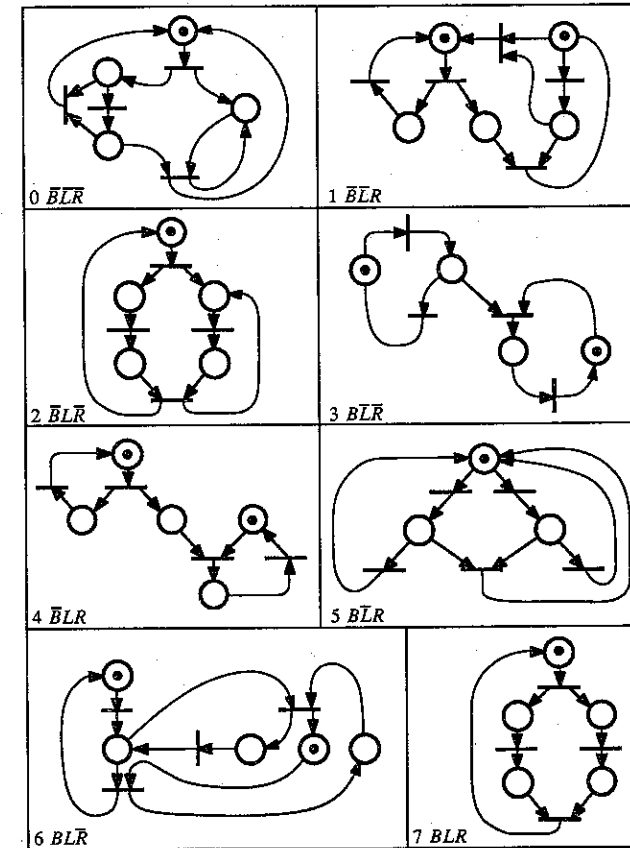


Figure 1.13 Boundedness (B), liveness (L) and reversibility (R) are independent properties.

The first three groups are called **static methods**, and their application to nets systems as abstract models leads to **exact results**. Simulation methods are called **dynamic** and proceed exercising the net system model under certain strategies. In this case some bugs can be detected (e.g. some deadlocks), allowing 'some confidence on the model', if problems are not manifested during the simulation process. However, in general, simulation methods do not allow properties to be proved, even if they might be of great help in understanding the modeled system. In particular, simulation methods are extremely useful when time is associated with the net evolution (**timed** systems), or when we wish to know the response of the system described with a net in an environment which is also defined by simulation. In this section we will only overview some static methods applied to autonomous nets.

Table 1.1 Summarizing some basic qualitative properties

1. Bound of place p in $\langle N, M_0 \rangle$
 $B(p) = \sup\{M(p) | M \in R(N, M_0)\}$
2. p is bounded in $\langle N, M_0 \rangle$ if $B(p) < \infty$
3. $\langle N, M_0 \rangle$ is bounded if all places are bounded
4. $\langle N, M_0 \rangle$ is a deadlock-free system if $\forall M \in R(N, M_0) \exists t \in T$ such that t is fireable at $M, M[t]$
5. t is live in $\langle N, M_0 \rangle$ if $\forall M \in R(N, M_0) \exists \sigma$ such that $M[\sigma, > M'$
6. $\langle N, M_0 \rangle$ is live if all transitions are live
7. $\langle N, M_0 \rangle$ is reversible if $\forall M \in R(N, M_0) \exists \sigma$ such that $M[\sigma > M_0$
8. Mutual exclusion in $\langle N, M_0 \rangle$:
 - p_i and p_j are in marking mutual exclusion if $\nexists M \in R(N, M_0)$ s.t. $[M(p_i) > 0] \wedge [M(p_j) > 0]$
 - t_i and t_j are in firing mutual exclusion if $\nexists M \in R(N, M_0)$ s.t. $M \geq Pre(t_i) + Pre(t_j)$
9. Structural properties (represent abstractions of behavioral properties):
 - N is structurally bounded if $\forall M_0$ (finite) $\langle N, M_0 \rangle$ is bounded
 - N is structurally live if $\exists M_0$ (finite) making $\langle N, M_0 \rangle$ a live system

graph (RG) which represents, **individually**, the net markings and transition firings. If the net system is bounded, the reachability graph is finite and the different qualitative properties can be verified easily. If the net system is unbounded, the above graph is infinite and it is therefore impossible to construct. In this case, finite graphs known as **coverability graphs** can be constructed (see, for example, Finkel (1990)). In spite of its power, enumeration is often difficult to apply, even in nets with few places, because of its computational complexity (it is strongly combinatorial).

The enumeration analysis technique for bounded stochastic Petri net systems consists of the generation of an underlying **Markov chain (MC)**. All performance figures are later computed on the MC. Assuming exponential (i.e. memoryless) firing times, the MC is isomorphic to the RG.

Analysis by transformation is based on the following idea: given a net system $\langle N, M_0 \rangle$ in which we wish to verify the set of properties Π , we transform it into the net system $\langle N', M'_0 \rangle$ such that:

1. $\langle N', M'_0 \rangle$ satisfies the properties Π iff $\langle N, M_0 \rangle$ satisfies them (i.e. the transformation *preserves* the properties Π).
2. It is easier to verify the properties Π in $\langle N', M'_0 \rangle$ than in $\langle N, M_0 \rangle$.

Reduction methods are a special class of transformation methods in which a sequence of net systems preserving the properties to be studied is constructed. The construction is done in such a way that the net system $\langle N^{i+1}, M_n^{i+1} \rangle$ is 'smaller' (i.e. has less markings) than the previous in the sequence,

The applicability of reduction methods is limited by the existence of **irreducible net systems**. Practically speaking, the reductions obtained are normally considerable, and can allow the desired properties to be verified directly. Because of the existence of irreducible systems, this method must be complemented by some other methods.

Finally, **structural analysis** techniques carefully consider the net structure (hence their name), while the initial marking acts basically as a parameter. Structural analysis techniques investigate the relationships between the behavior of a net system and the structure of the net. In this last class of analysis techniques, we can distinguish two subgroups:

1. **Linear algebra/Linear programming-based techniques**, which are based on the net system **state equation**. In certain analysis they permit a fast diagnosis without the necessity of enumeration.
2. **Graph-based techniques**, in which the net is seen as a bipartite directed graph and some *ad hoc* reasonings are applied. These methods are especially effective in analyzing restricted subclasses of ordinary nets.

The three groups of analysis techniques outlined above are by no means exclusive, but rather they are complementary. Normally the designer can use them according to the needs of the ongoing analysis process. Obviously, although we have distinguished between reduction and structural analysis methods, it must be pointed out that most popular reduction techniques act basically on the net structure level and thus can be considered also as structural techniques.

For what concerns the qualitative analysis of **interpreted systems**, it should be pointed out that the properties of the underlying autonomous model can be only sufficient (e.g. for boundedness), necessary (e.g. for reachability) or neither sufficient nor necessary (e.g. for liveness). For particular net subclasses (e.g. simple nets) liveness properties are preserved under 'reasonable assumptions' on the behavior of the environment (e.g. fair progress: i.e. not infinite delay in firing a continuously enabled transition; and local fairness on choices, i.e. all outcomes of a conflict are being chosen repeatedly).

1.6.1 Reachability graph

A marking is said to be reachable in a system $\langle N, M_0 \rangle$ if there exists a sequence σ applicable at M_0 such that $M_0[\sigma] M$. If we were able to compute all reachable markings, $M \in R(N, M_0)$, and their reachability relationships, all qualitative behavioral properties should be analyzable. A major problem arises in systems in which the number of reachable markings is infinite (unbounded systems). Because infinite state (marking) systems cannot easily be represented by enumeration, finite representations have been proposed. But in this case it is possible that relevant information for

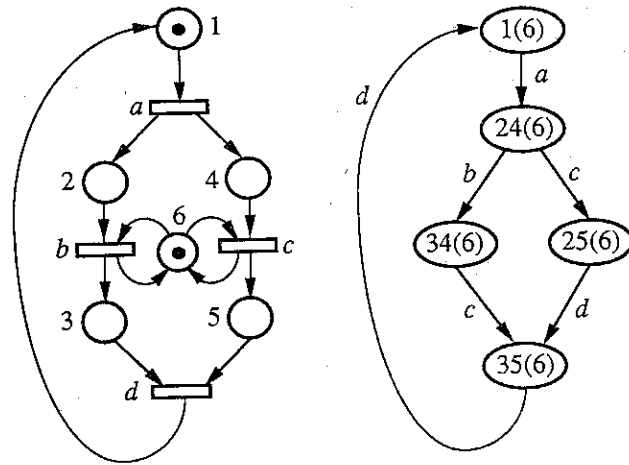


Figure 1.14 Bounded, live and reversible system and its reachability graph.

This presentation being more engineering oriented, let us restrict ourselves to the case of bounded systems.

Reachability analysis approach for bounded systems is based on the **exhaustive sequentialized simulation** of the possible marking evolutions. The main limitation of the approach is its computational complexity, so-called **state explosion problem**: the number of markings can be exponential with respect to the size of the net (measured, for example, by the number of places).

Definition 1.6. The **reachability graph** associated with a system $\langle N, M_0 \rangle$ is a graph $RG(N, M_0)$ in which each node represents a marking reachable from M_0 and each arc represents the firing of a transition. There exists an arc, labeled t_k , which goes from the node representing M_i to that representing M_j iff on firing t_k from M_i we reach $M_j : M_i[t_k] M_j$.

If the marked net is bounded, the graph construction process is straightforward. It finishes when all the possible firings from the reachable markings have been explored.

Let us consider, for example, the system in Fig. 1.14, assuming p_6 is removed. Initially only a is enabled. Firing it, p_2 and p_4 are marked and b and c are enabled. If b is fired before, the marked places are p_3 and p_4 , otherwise if c is fired before, p_2 and p_5 should be marked. From any of the last two markings p_3 and p_5 will be unavoidably marked and d being the only enabled transition, the initial marking should be recovered. Therefore our system has five markings (thus is **bounded**). From a direct inspection of the set of markings (i.e. the state space) it is easy to conclude

p_2 and p_3 (p_1 , p_4 and p_5) are in **mutual exclusion** (i.e. a pair of places is never simultaneously marked).

Moreover, considering the reachable markings and the net structure (the pre-function), **firing concurrency** between transitions b and c appears (in the reachability graph b and c seem to be conflicting!). Observe at this point that introducing p_6 in our system does **not** change the reachability graph, but transitions b and c become in **firing mutual exclusion**. This example shows that concurrency-mutual exclusion on firings cannot be studied on the reachability graph alone, because this gives a sequentialized vision of the concurrent evolutions.

Reversibility (i.e. recoverability of the initial marking) cannot be studied considering only the state space. The reachability relations among states (markings) must be taken into account. Because all markings are reachable from the initial one, a system should be reversible iff in the reachability graph there exists a path from any node to the original one. In other words:

Property 1.1. A net system is reversible iff its reachability graph is strongly connected.

If a system is reversible, any transition can be fired once and again iff that transition is firable at least once from the initial marking. This is true because the initial marking being always reachable, the transition can be fired again. More precisely, we are saying that in a reversible net system a transition is live iff it is firable in a sequence starting in the initial marking. If the system is not reversible but bounded, liveness is also characterizable but the condition is slightly more complex:

Property 1.2. Let $\langle N, M_0 \rangle$ be a bounded system. Transition t is live in $\langle N, M_0 \rangle$ iff t labels at least one arc of every strongly connected component of the reachability graph containing its transitive closure (i.e. such that the set of its successor nodes is included in the component itself).

In general, a system may be unbounded. Therefore the reachability graph construction process would never end. To avoid this, an abandon condition can be taken into account in the construction process:

Property 1.3. Place p is bounded in $\langle N, M_0 \rangle$ iff there exists no M_j , reached from M_i such that $M_j \geq M_i$ and $M_j(p) > M_i(p)$. Therefore, the system $\langle N, M_0 \rangle$ is unbounded (abandon condition) iff there exists M_j reachable from M_i such that $M_j \geq M_i$ and $M_j \neq M_i$.

If $M_i[\sigma]M_j$, $M_j \geq M_i$ and $M_j \neq M_i$, the repetition of σ allows us to conclude on unboundedness. The proof that the condition is also necessary is based on a result from Karp-Miller (Karp and Miller, 1969).

A last and obvious consideration about the construction/analysis of the

Property 1.4. A system is deadlockable iff a marking not enabling any transition is found (i.e. a node without successor points out a deadlock).

Concluding, analysis techniques based on the reachability graph (theoretically possible for bounded systems) are very simple from a conceptual point of view. The problem that makes this approach impractical in many cases is its computational complexity: the **state explosion problem**. Figure 1.15 shows this on a very simple net system: parts from store 1 to stores 2 and 3. The subnet generated by places $\{B, C, D, E\}$ imposes some restrictions on the way parts are distributed to the destination stores (i.e. partially schedule the distribution). The reachability graph is, even if it has been 'structured' for clearer presentation, difficult to understand and manage. The reader can try to check on the reachability graph (!) that the imposed distribution strategy is: parts are sent in a 1 : 1 relation to the destination stores, but allowing sometimes up to four consecutive dispatches to a given store (i.e. locally adjusting the possible demand, but maintaining the overall fair distribution).

Last, but not least, it is important to observe that reachability/coverability graphs are built for a given initial marking. If the number of resources (e.g. number of machines, size of stores, etc. . . .) changes, new (and completely different) graphs should be computed. Otherwise stated: reachability/coverability does not allow **parametric analysis** on the behavior of net systems.

1.6.2 Net system reductions

Even if reachability graph-based analysis techniques are complete for bounded systems, the computational complexity limits their applicability in practice. Net system reduction is a different analysis technique that allows the analysis of net models by producing transformations on its structure and, eventually, on its initial marking.

The approach is based on the definition of a **kit** or **catalog of reduction rules**, each one preserving the subset of properties (liveness, boundedness, reversibility, etc.) to be analyzed. The transformation procedure is **iterative** by nature: given the property (or properties) to be analyzed, the subset of rules that preserve it (them) is applied until the reduced system becomes **irreducible**. The irreducible system can be so simple that the property under study is trivially checked (see later, Fig. 1.17(d)). In other cases, the irreducible net is just 'more simple' to be analyzed, and other analysis techniques should be used: in other words, techniques to analyze net system models are **complementary**, not exclusive.

Reduction rules are transformation rules interesting for net analysis. When considered in the reverse sense they become **expansion** rules, interesting for net synthesis: **stepwise refinements** (or **top-down**) approach.

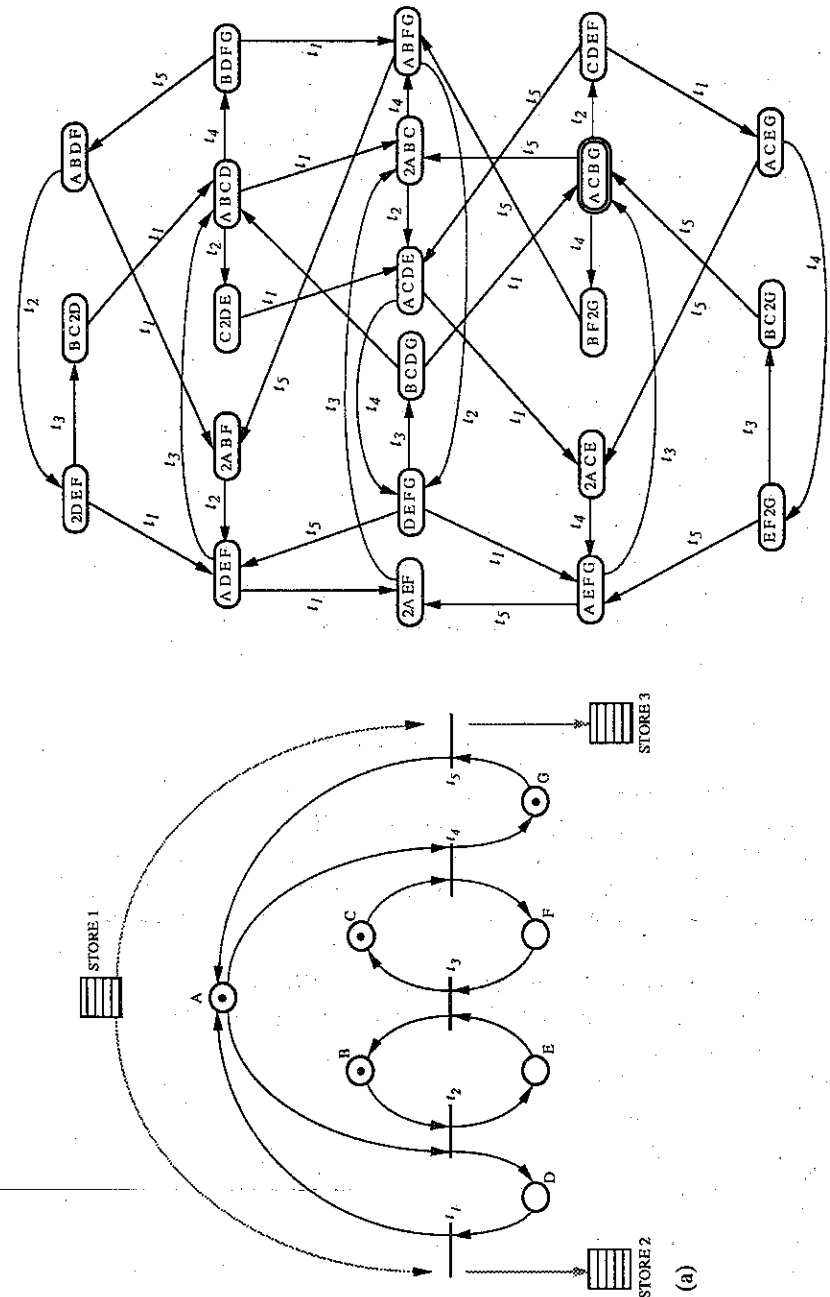


Figure 1.15 Parts of STORE 1 are sent to STORE 2 and STORE 3 according to the strategy defined by the subnet generated by $\{B, C, E, F\}$: (a) the net system; (b) the RG.

the specification by construction. This is interesting when comparing with the more classical approach based on the iteration of description and analysis. The iterative process has two basic disadvantages:

1. the **lack of general criteria** for modifying (correcting) a model which does not meet the requirements made in the validation.
2. the **operational difficulty** inherent to the validation phase. Obviously, this difficulty will be strongly reduced if a computer-aided design (CAD) system is available.

Nevertheless, because there exists no **universal** reduction rules kit (i.e. that fully reduces any system), it is not possible to synthesize all of them by stepwise refinements.

From a practical point of view, the design of the transformation rules catalog represents a **compromise** between **completeness** (i.e. transformation capabilities) and **usefulness**.

Reduction rules have a single pattern:

if an applicability precondition is true **then** reduce the net system.

Behavioral and/or **structural** statements can be done for the applicability precondition. The behavioral statements can be more powerful for a given initial marking, but their computation is usually much more complex. So the applicability preconditions presented here are based on **structural** considerations, the initial marking playing an auxiliary role as a **parameter**. According to this, reduction rules will have the following general pattern:

if structural condition and initial marking condition are true then make structural change and marking change.

A very basic kit of reduction rules is presented. Additional details are given only for the rule of *implicit places*, which are redundancies in the net system model: if an implicit place is removed, then (illusory) synchronizations disappear and other reduction rules can be applied.

(a) A basic kit of reduction rules

Figure 1.16 presents graphically structural and marking conditions of a kit of six particular cases of reduction rules (Silva, 1985). It is not difficult to observe that they preserve such properties as liveness, the bounds of places (thus boundedness) and, if the second place in RA1 has only one input transition, reversibility:

- RA1 is a particular case of the **macroplace rule** (Silva, 1981).
- RA2 is a particular case of the **transition fusion rules** (Berthelot, 1987).
- RB1 and RC1 are particular cases of the **implicit place rule** (Silva, 1985);

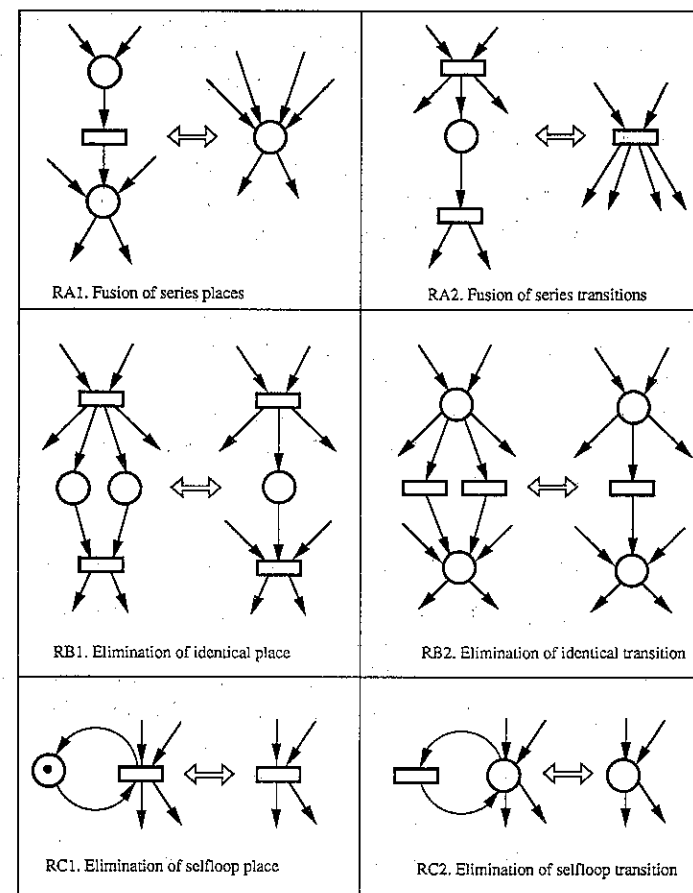


Figure 1.16 A basic reduction kit.

that RC1 can be trivially generalized creating **several self-loops** in which the place always appears. Liveness, the bound of places and reversibility are preserved. Moreover, if the place contains **several tokens**, liveness, boundedness (in general, not the bound of the net system) and reversibility are preserved.

- RB2 and RC2 are particular cases of **identical and identity transition rules** (Berthelot, 1987).

An interesting remark is the analogy between rules at the same level in Fig. 1.17: basically rules RX2 are obtained from rules RX1 by changing the role of places and transitions (**duality**) and **reversing** the arrows (something important only for rules RA). Duality (and reversing) are important concepts for a deep understanding and systematic presentation of many

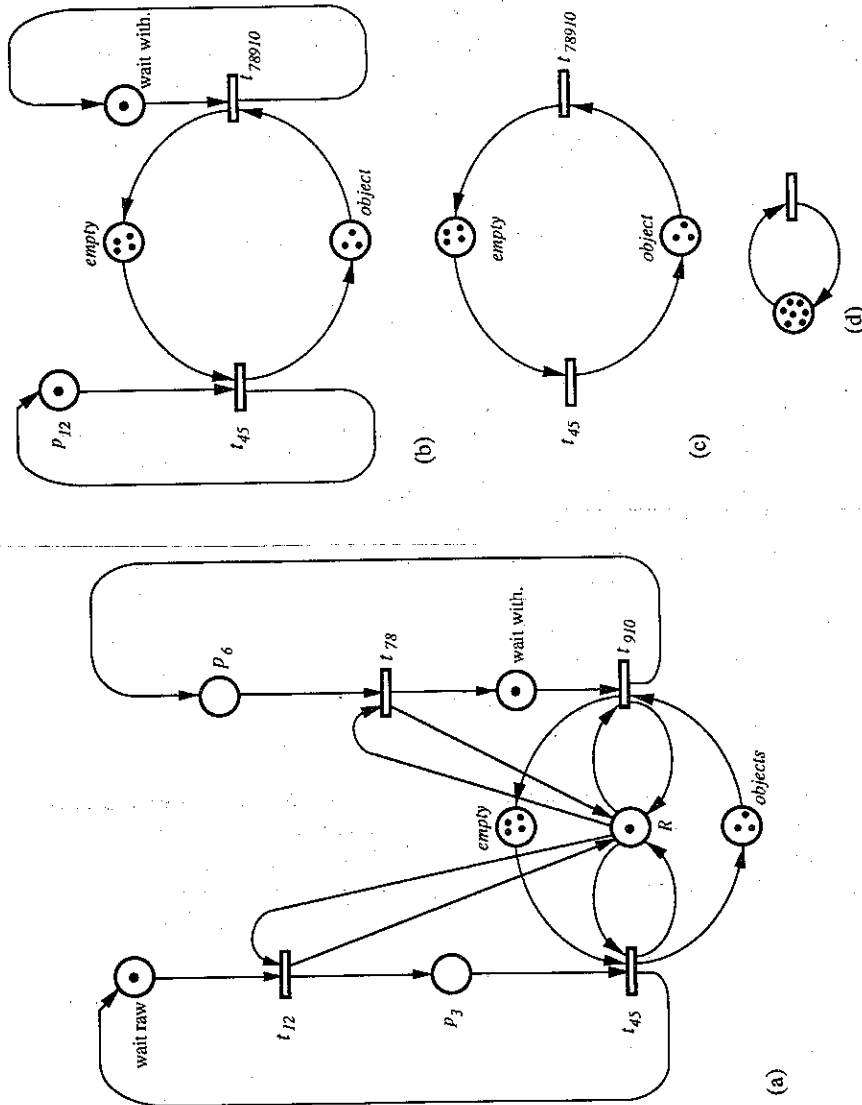


Figure 1.17 The reduction process shows (see (d)) that the net system in Fig. 1.11 is live, 7-bounded and reversible.

Let us now consider the net system in Fig. 1.9. The subnet defined by $op_1-t_3-wait\ dep.$ verifies the precondition of rule RA1. Thus it can be reduced to a place, p_3 (Fig. 1.17(a)). The same holds for $op_2-t_6-wait\ free$ that is reduced to p_6 (Fig. 1.17(a)). The subnets $t_1-load-t_2$, $t_4-deposit-t_5$, $t_7-unload-t_8$ and $t_9-withdraw-t_{10}$ can be reduced according to RA2 (see t_{12} , t_{45} , t_{78} and t_{910} in Fig. 1.17(a)). Place R in Fig. 1.17(a) is implicit (one of the trivial generalizations mentioned for RC1). Thus it can be removed, and $wait\ raw-t_{12}-p_3$ and $t_{910}-p_6-t_{78}$ can be reduced to p_{12} and t_{78910} , respectively (see Fig. 1.17(b)). Places p_{12} and $wait\ with.$ are implicit (RC1) in Fig. 1.17(b), thus the net system in Fig. 1.17(c) is obtained. Playing the token game, a place (e.g. $object$) can become empty in Fig. 1.17(c) and $t_{45}-object-t_{78910}$ can be reduced (RA2) to a single transition (Fig. 1.17(d)). Therefore, the original net system is live, 7-bounded and reversible.

(b) Implicit places

A place in a net system can only constrain the firable sequences. If a place, for an initial marking, never constrains the firable sequences, it can be removed without changing the **sequential observation** of the behavior of the net system (i.e. the set of firable sequences). These behaviorally defined places are called **(firing) implicit places**.

Let N be a net and N_p the net resulting from removing place p from N , Pre and Pre^p are the corresponding pre-incidence functions. If M_0 is an initial marking for N_p , $M_0 = (M_0^p, M_0(p))$ denotes the initial marking of N .

Definition 1.7. Given a system $\langle N, M_0 \rangle$, the place p is implicit (IP) if for any reachable marking in $\langle N_p, M_0^p \rangle$, (i.e. $\forall M^p \in R(N_p, M_0^p)$) and any output transition of p (i.e. $\forall t \in \{p^*\}$) the following holds:

$$M^p \geq Pre^p(t) \Rightarrow M(p) \geq Pre(p, t)$$

The net system in Fig. 1.18 is unbounded (p_4 is the unique unbounded place) and non-reversible (also because of p_4). Place p_4 is implicit. Removing p_4 the system becomes bounded and reversible! Place p_6 in Fig. 1.14 imposes firing mutual exclusion between b and c . Since p_6 is an implicit place, the reduction rule does not preserve firing mutual exclusion. According to the definition, firable sequences are preserved. Thus the following is true:

Property 1.5. The elimination of implicit places:

1. **Preserves:** deadlock-freeness, liveness and marking mutual exclusions.
2. **Does not preserve:** boundedness, reversibility and firing mutual exclusion.

Sometimes it is practical to impose a second condition to the definition of implicit places: marking redundancy (i.e. computable from the other markings). The marking of p_4 in Fig. 1.18 cannot be computed from the

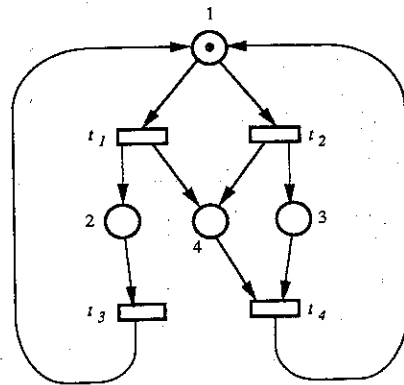


Figure 1.18 Place p_4 is firing implicit but not marking implicit. Removing p_4 the 'false' synchronization in t_4 disappears.

considering now: marking implicit places. Because of the additional redundancy, marking implicit places preserves the state space (i.e. the reachability graphs of the net system with and without p are isomorphic) and therefore preserves two already mentioned properties: **boundedness** and **reversibility**.

How can implicit places be detected? How complex is the process? The property is behavioral, so computationally complex behavior-based algorithms should be used. The next property gives a very simple algorithm, based on the solution of a linear programming problem (LPP1) to detect 'most of the practical cases'. Because, LPPs are of polynomial time complexity (Nemhauser *et al.*, 1989), the technique has this complexity. This approach derives from some relatively complex arguments (see Colom and Silva, 1991b).

Property 1.6. Let $\langle N, M_0 \rangle$ be a net system, and z defined as follows:

$$\begin{aligned} z = \min & \quad Y^T \cdot M_0 + \mu \\ \text{subject to} & \quad Y^T \cdot C \leq C(p) \\ & \quad Y^T \cdot Pre(t) + \mu \geq Pre(p, t) \quad \forall t \in p^* \\ & \quad Y \geq 0, Y(p) = 0 \end{aligned} \quad (\text{LPP1})$$

Assuming $M_0(p) \geq 0$, if $M_0(p) \geq z$ then p is implicit.

Remark if $Y^T C = C(p)$ constraint, then p is a marking implicit place.

The computation of LPP1 for p_9 (Fig. 1.19(a)) gives $z = 0$, for:

1. $[Y^T = (001110100)] \Leftrightarrow [C(p_9) = C(p_3) + C(p_4) + C(p_5) + C(p_7)]$
2. $\mu = -1$

Because $M_0(p_9) > z = 0$, p_9 is implicit and can be removed. Once p_9 is

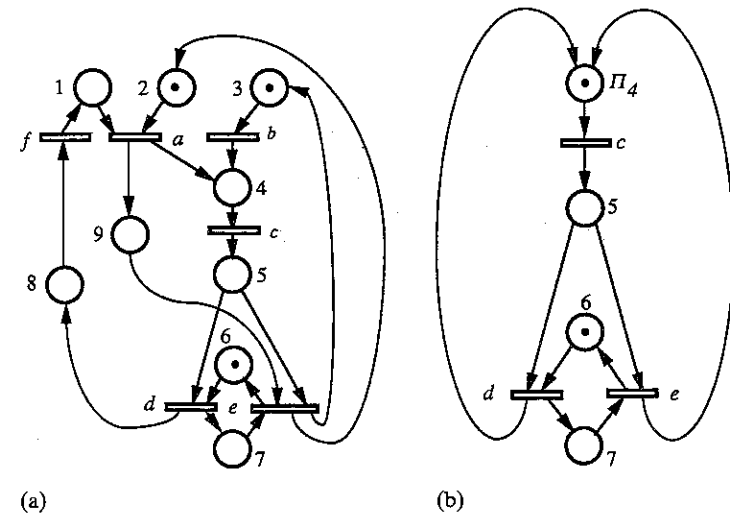


Figure 1.19 Places p_9 and p_2 (or p_2 and p_7) are implicit.

Figure 1.19(b) shows a reduced net system. It can be obtained reducing p_3-b-p_4 into a place (say p_{34}) (RA1) and finally $p_8-f-p_1-a-p_{34}$ into Π_4 . Now RA1 allows us to fuse Π_4 and p_5 . The new place is implicit, so it can be removed. Then a cycle with $p_6-d-p_7-e-p_6$ remains. Finally it can be reduced to a basic net, $p_6-t_1-p_7-e-p_6$, with one token. Therefore the original net system is live and bounded (**Note:** it is also reversible, but we cannot guarantee this because of the fusion of p_3-b-p_4 in p_{34}).

1.6.3 Linear algebraic techniques

The behavior of a net system model is clearly non-linear, nevertheless the so-called state equation, $M = M_0 + C \cdot \bar{\sigma} \geq 0$, $\bar{\sigma} \geq 0$ (see eq. (1.4)), represents a nice **linear relaxation**. Unfortunately the existence of **spurious** solutions (section 1.1.2) leads usually to **semidecision algorithms** (i.e., only necessary or only sufficient conditions) to analyze such behavioral properties as reachability, boundedness, deadlock-freeness, mutual exclusion, liveness or reversibility. For example, $\tilde{M}_1 = (0, 0, 0, 2, 0)^T$ and $\tilde{M}_2 = (0, 2, 0, 0, 0)^T$ are two spurious solutions for the system in Fig. 1.20(a). The first allows us to say that p_4 is 2-bounded, while it is really 1-bounded (check it). \tilde{M}_2 is a deadlock. Then using the state equation we cannot conclude that the system in Fig. 1.20(a) is deadlock-free.

Spurious solutions can be removed using different approaches (Colom and Silva, 1991b). For example, it is clear that adding implicit places, a new system model with identical behavior is obtained. For certain net systems, if the implicit places are chosen carefully, the state equation of

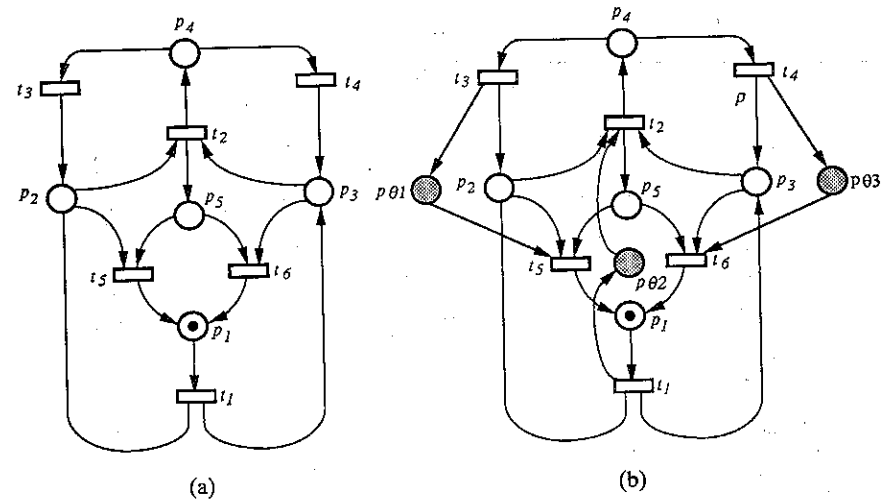


Figure 1.20 Two equal behavior 1-bounded and live net systems: dashed places, $\{p_{01}, p_{02}, p_{03}\}$ are implicit.

system in Fig. 1.20(b) has been obtained by adding implicit places p_{01} , p_{02} and p_{03} to that in Fig. 1.20(a). The above-mentioned spurious solutions, \bar{M}_1 and \bar{M}_2 , are not projections on P of the solutions of the new state equation. Moreover, we can conclude now that the new (and original) net systems were 1-bounded for p_4 and deadlock-free!

Classical reasoning to prove logical properties uses **invariants** on the behavior of a system. The right and left non-negative annullers of the incidence matrix lead to two kinds of structural objects (p - and t -semiflows):

1. $Y \geq 0, Y^T \cdot C = 0 \Rightarrow Y^T \cdot M = Y^T \cdot M_0$ (token conservation law)
2. $X \geq 0, C \cdot X = 0 \Rightarrow \exists M_0$ such that $M_0[\sigma] M_0$ and $\bar{\sigma} = X$ (cyclic marking behavior)

The token conservation laws are **marking invariants** induced by p -semiflows. Usually they are called **p -invariants**.

The computation of minimal p -semiflows (Y) and minimal t -semiflows (X) has been extensively studied. However, an **exponential number** of minimal semiflows may appear. Therefore the time complexity of this computation cannot be polynomial. In Colom and Silva (1991a) a study is carried out merging traditional techniques in convex geometry with those developed within Petri nets. From a conceptual point of view, the consideration of semiflows provides **decomposed views** of the structure of the net model. In Fig. 1.21 the decomposition induced by the minimal p -semiflows of the system in Fig. 1.9 is graphically presented. The induced minimal

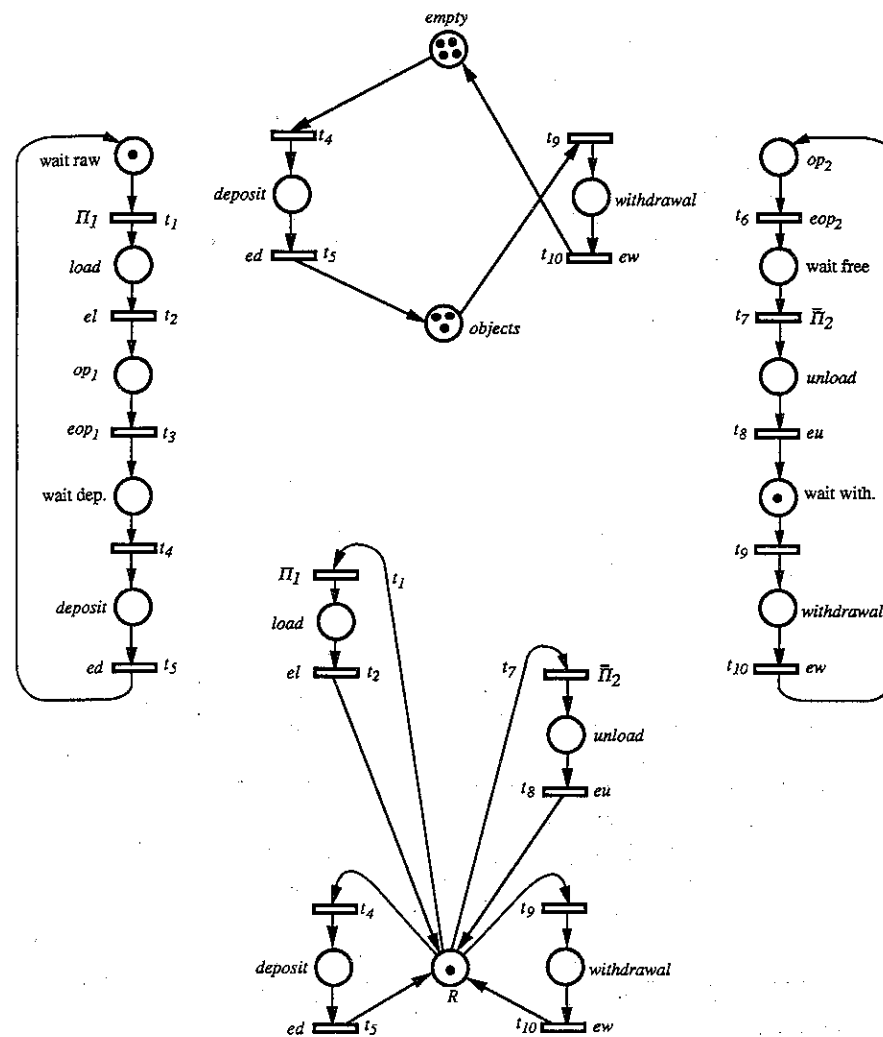


Figure 1.21 A decomposed view of the net system in Fig. 1.9.

$$M(\text{wait raw}) + M(\text{load}) + M(\text{op}_1) + M(\text{wait dep.}) + M(\text{deposit}) = 1 \quad (1.5)$$

$$M(\text{op}_2) + M(\text{wait free}) + M(\text{unload}) + M(\text{wait with.}) + M(\text{withdrawal}) = 1 \quad (1.6)$$

$$M(\text{empty}) + M(\text{deposit}) + M(\text{object}) + M(\text{withdrawal}) = 7 \quad (1.7)$$

$$M(R) + M(\text{load}) + M(\text{unload}) + M(\text{deposit}) + M(\text{withdrawal}) = 1 \quad (1.8)$$

Because markings are non-negative integers (i.e. $\forall p \in P, M(p) \geq 0$), the

1. Bounds: $M(p_i) \leq 1 \quad \forall p_i \in P \setminus \{\text{empty, object}\}$
 $M(\text{empty}) \leq 7, \quad M(\text{object}) \leq 7$
2. Marking mutual exclusions among the following subset of places (i.e. $M(p_i)M(p_j) = 0, i \neq j$):
 - wait raw, load, op_1 , wait dep., deposit
 - op_2 , wait free, unload, wait with., withdrawal
 - R , load, unload, deposit, withdrawal

The decomposed view of a net system is even useful to derive an **implementation**. For example, the net system in Fig. 1.9 can be implemented using two sequential processes (for *Machine1* and *Machine2*) and three semaphores (*object*, *empty* and R), where R is a mutual exclusion semaphore.

Linear algebraic techniques represent *fast to compute* (polynomial time in many cases) *semidecision* (necessary or sufficient conditions) algorithms, easily amenable to initial marking *parametric* analysis (e.g. changing the number of customers, size of resources, initial distribution of customers and/or resources). The following subsections study marking bounds and boundedness (a), deadlock-freeness (b), structural liveness and liveness (c), and reversibility (d).

(a) Marking bounds and boundedness

Relaxing the reachability condition in the definition of the bound of a place p (see Table 1.1), by using the state equation (1.4), the structural bound of p is defined as follows:

$$SB(p) = \sup \{M(p) \mid M = M_0 + C \cdot \bar{\sigma} \geq 0, \bar{\sigma} \geq 0\}$$

Let e_p be the **characteristic vector** of p : $e_p(f) := \text{if } f = p \text{ then } 1 \text{ else } 0$. The structural bound of p , $SB(p)$, can be expressed as a linear programming problem:

$$SB(p) = \max \quad e_p^T \cdot M$$

$$\text{subject to } M = M_0 + C \cdot \bar{\sigma} \geq 0 \quad (LPP2)$$

$$\bar{\sigma} \geq 0$$

Therefore $SB(p)$ can be computed in polynomial time. In sparse-matrix problems (matrix C is usually sparse), good implementations of the classical **simplex method** lead to quasi-linear time complexities.

Because $SB(p)$ has been defined using a linear relaxation of the reachability in the system, then $SB(p) \geq B(p)$. Therefore, if we are investigating the k -boundedness of a place (i.e. $M(p) \leq k$), we have a sufficient condition in polynomial time:

if $SB(p) \leq k$ **then** $B(p) \leq k$ (i.e. p is k -bounded)

theories (see, for example, Murty (1983); Nemhauser *et al.* (1989); otherwise all the needed arguments are compiled and adapted in (Silva and Colom, 1988). The important point in this overview is to convey the idea that other theories are helpful to understand in a deep and general framework many sparse results on net systems' behaviors.

The **dual** linear programming problem of LPP2 is the following (see any text on linear programming to check it):

$$\tilde{SB}(p) = \min \quad Y^T \cdot M_0$$

$$\text{subject to } Y^T \cdot C \leq 0 \quad (LPP3)$$

$$Y \geq e_p$$

LPP2 has always a feasible solution ($M = M_0, \bar{\sigma} = 0$). Using duality and boundedness theorems from linear programming theory, both LPP2 and LPP3 are bounded (thus p is structurally bounded) and $SB(p) = \tilde{SB}(p)$ iff there exists a **feasible solution** for LPP3:

$$\exists Y \geq e_p \quad \text{such that} \quad Y^T \cdot C \leq 0 \quad (1.9)$$

The reader can easily check that LPP3 makes in polynomial time an 'implicit search' for the structural bound of p on a set of structural components including all the p -semiflows ($Y \geq 0, Y^T \cdot C = 0$).

From the above discussion and using the **alternatives theorem** (essentially the Minkowski-Farkas lemma in algebraic form) the following properties can be proved:

Property 1.7. The following three statements are equivalent:

1. p is structurally (i.e. for any M_0) bounded;
2. $\exists Y \geq e_p$ such that $Y^T \cdot C \leq 0$ {place-based characterization};
3. $\forall X \geq 0$ such that $C \cdot X \geq 0, e_p^T \cdot C \cdot X = C(p) \cdot X = 0$ is satisfied {transition-based characterization}.

Property 1.8. The following three statements are equivalent:

1. N is structurally (i.e. $\forall M_0$) bounded;
2. $\exists Y \geq 1$ such that $Y^T \cdot C \leq 0$ {place-based characterization};
3. $\forall X \geq 0$ such that $C \cdot X \geq 0, C \cdot X = 0$ is satisfied (i.e. $\nexists X \geq 0$ such that $C \cdot X \geq 0$ and $C \cdot X \neq 0$) {transition-based characterization}.

(b) Deadlock-freeness (and liveness)

Token conservation laws are invariant properties of the behavior of net systems that may be very useful to prove **deadlock-freeness**. Using the invariants in eqs (1.5–1.8), we shall prove that our net system in Fig. 1.9 is deadlock-free.

If there exists a deadlock, no transition can be fired. Let us try to construct a marking in which no transition is fireable. When a unique input

$M(op_1) = M(deposit) = M(op_2) = M(unload) = M(withdrawal) = 0$, and the token conservation laws in eqs (1.5–1.8) reduce to:

$$M(\text{wait raw}) + M(\text{wait dep.}) = 1 \quad (1.5a)$$

$$M(\text{wait free}) + M(\text{wait with.}) = 1 \quad (1.6a)$$

$$M(\text{empty}) + M(\text{object}) = 7 \quad (1.7a)$$

$$M(R) = 1 \quad (1.8a)$$

Because R should always be marked at the present stage, to prevent the firing of t_1 and t_7 , places 'wait raw' and 'wait free' should be unmarked. The token conservation laws are reduced once more, leading to:

$$M(\text{wait dep.}) = 1 \quad (1.5b)$$

$$M(\text{wait with.}) = 1 \quad (1.6b)$$

$$M(\text{empty}) + M(\text{object}) = 7 \quad (1.7b)$$

$$M(R) = 1 \quad (1.8b)$$

Since $M(\text{wait dep.}) = M(\text{wait with.}) = 1$, to avoid the firing of t_4 and t_9 , $M(\text{empty}) + M(\text{object}) = 0$ is needed. This contradicts eq. (1.7bis), so the net system is deadlock-free.

A more compact, algorithmic presentation of the above deadlock-freeness proof is:

```

if  $M(\text{load}) + M(op_1) + M(\text{deposit}) + M(op_2) + M(\text{unload}) +$ 
 $M(\text{withdrawal}) \geq 1$ 
then one of  $t_2, t_3, t_5, t_6, t_8$ , or  $t_{10}$  is firable
else if  $M(\text{wait raw}) + M(\text{wait free}) \geq 1$ 
then one of  $t_1$  or  $t_7$  is firable
else one of  $t_4$  or  $t_9$  is firable

```

Even if the above is an *ad hoc* proof, it can be fully automated (i.e. automatic proving). To prove deadlock-freeness by means of linear algebra we must express the condition 'transition t is not firable at marking M ' using linear constraints. This can always be done. Nevertheless, it turns out to be very efficient in net systems where for each place the structural bound $SB(p)$, computed through LPP2 or LPP3 is equal to the weight of its output arcs: $SB(p) = W(p, t) \forall p \in P, \forall t \in p^*$. The condition in linear form is:

$$\sum_{p_i \in t^*} M(p_i) \leq \sum_{p \in P} Pre(p, t) - 1$$

In words: the amount of tokens in the input places of t is less than required. Therefore:

Property 1.9. Let $\langle N, M_0 \rangle$ be a net system such that $SB(p) = W(p, t) \forall p \in P, \forall t \in p^*$. A sufficient condition for $\langle N, M_0 \rangle$ to be deadlock-free is that the following linear system has no solution ($\dim(1_q^T) = q$):

For the system in Fig. 1.4(g), the system of inequalities $M^T \cdot Pre \leq 1^T \cdot Pre - 1$ looks as follows ($M(p) \geq 0 \forall p \in P$, thus $M(p) \leq 0$ is equivalent to $M(p) = 0$):

$$M(p_1) + M(R) \leq 1 \quad \{t_1\}$$

$$M(p_2) = 0 \quad \{t_2\}$$

$$M(p_3) + M(R) \leq 1 \quad \{t_3\}$$

$$M(p_4) = 0 \quad \{t_4\}$$

The above system of inequalities and equations together with the state equation form an **inconsistent** linear system. Thus the net system is deadlock-free.

The deadlock-free system in Fig. 1.9 does not verify the precondition of Property 1.9. Nevertheless the approach has been generalized (see Colom *et al.* (1990b)), and that conclusion can be obtained from the inconsistency of a single linear system.

As a final remark, we want to point out that liveness can be proved for the net system in Fig. 1.9. Liveness implies deadlock-freeness, but the reverse is not true in general. Nevertheless, if the incidence matrix, C , has a right-orthogonal space of dimension one (i.e. $C \cdot X = 0 \Rightarrow X = k \cdot A$) and $A \geq 1$ (i.e. the net is consistent), then any infinite behavior must contain all transitions with relative firings given by A (t_i). Thus deadlock-freeness implies, in this case, liveness. For the net in Fig. 1.9 the above property holds with $A = 1$. Therefore the net system being deadlock-free is also live.

(c) Structural liveness and liveness

A general approach to linearly analyze liveness needs the use of **invariants** and some **inductive** reasoning. A **necessary** condition for a transition t to be live in a system $\langle N, M_0 \rangle$ is its eventual infinite firability (i.e. the existence of a **firing repetitive sequence** σ_R containing t).

Using the state equation (1.4) as a linear relaxation of the reachability condition, an **upper bound** of the number of times t can be fired in $\langle N, M_0 \rangle$ is given by the following LPP ($e_t(\theta)$: $>$ if $\theta = t$ then 1 else 0):

$$SR(t) = \max \quad e_t^T \cdot \bar{\sigma} \\ \text{subject to } M = M_0 + C \cdot \bar{\sigma} \geq 0 \quad (LPP4) \\ \bar{\sigma} \geq 0$$

The dual of LPP4 is:

$$SR(t) = \min \quad Y^T \cdot M_0 \\ \text{subject to } Y^T \cdot C \leq -e_t^T \quad (LPP5) \\ Y \geq 0$$

We are interested on characterizing when $SR(t)$ goes to infinity. The problem LPP4 has $M = M_0$ and $\bar{\sigma} = 0$ as a feasible solution. Using first

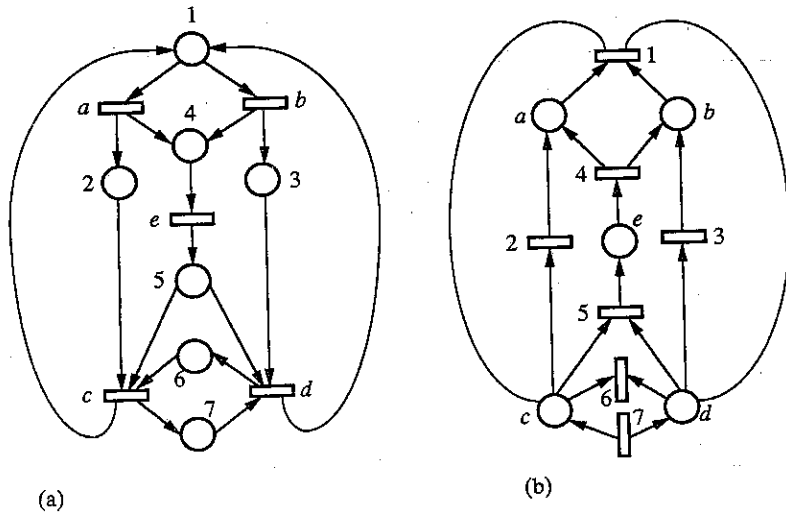


Figure 1.22 Two conservative and consistent, structurally non-live nets:
 (a) $\text{rank}(C) = 4$, $|T| - l - \delta = 5 - 1 - 1 = 3$, thus N is not struct-live;
 (b) $\text{rank}(C) = 4$, $|T| - l - \delta = 7 - 1 - 2 = 4$, thus no answer.

Property 1.10. The following three statements are equivalent:

1. t is structurally repetitive (i.e. there exists a 'large enough' M_0 such that t can be fired infinitely often);
2. $\exists Y \geq 0$ such that $Y^T \cdot C \leq -e_i^T$ {place perspective};
3. $\exists X \geq e_i$ such that $C \cdot X \geq 0$ {transition perspective}.

Property 1.11. The following three statements are equivalent:

1. N is structurally repetitive (i.e. all transitions are structurally repetitive);
2. $\exists Y \geq 0$ such that $Y^T \cdot C \leq 0$ and $Y^T \cdot C \neq 0$;
3. $\exists X \geq 1$ such that $C \cdot X \geq 0$.

Combining Properties 1.8 and 1.11 and considering that structural repetitiveness is a necessary condition for structural liveness, the following classical result can be obtained (Memmi and Roucairol, 1980; Brams, 1983; Silva, 1985):

if N is structurally live and structurally bounded

then $\exists X \geq 1$ such that $C \cdot X = 0$ (i.e. N is consistent)

$\exists Y \geq 1$ such that $Y^T \cdot C = 0$ (i.e. N is conservative)

Net structures in Fig. 1.22 are consistent and conservative, but there exists no live marking for them. A more careful analysis using the above result iteratively allows us to improve it with a **rank condition** on the incidence matrix of N , C . This and other results are summarized in the

Let us say that t_i and t_j are in **equality conflict relation** if $\text{Pre}(t_i) = \text{Pre}(t_j) \neq 0$. Obviously, this is an *equivalence* relation, leading to a *partition* of transitions into *equivalence classes*. Let D_k be an equivalence class. We define the following quantities:

$$\delta_k = |D_k| - 1$$

$$\delta = \sum_k \delta_k$$

Property 1.12. Let C be the incidence matrix of N .

1. if N is structurally live then $\exists X \geq 1$ such that $C \cdot X \geq 0$
2. if N is structurally live and structurally bounded

then $\exists X \geq 1$ such that $C \cdot X = 0$ (i.e. net consistency)

$\exists Y \geq 1$ such that $Y^T \cdot C = 0$ (i.e. net conservativeness)

$$\text{rank}(C) \leq |T| - \delta - 1$$

3. if N is connected, consistent and conservative then it is strongly connected.

The added rank condition allows us to state that the net in Fig. 1.22(a) is structurally non-live. Nevertheless, nothing can be said about structural liveness of the net in Fig. 1.22(b).

Property 1.12 is purely structural (i.e. the initial marking is not considered at all). Nevertheless it is clear that a small enough initial marking can make non-live a net system even if the net structure is well formed. A lower bound for the initial marking to make live a net system is based on p -invariants: if $t \in T$ is firable at least once, for any p -invariant Y , $Y^T \cdot M_0 \geq Y^T \cdot \text{Pre}(t)$. Therefore:

Property 1.13. If $\langle N, M_0 \rangle$ is a live system, then

$$\forall Y \geq 0 \text{ such that } Y^T \cdot C = 0, \quad Y^T \cdot M_0 \geq \max_{t \in T} (Y^T \cdot \text{Pre}(t)) \geq 1$$

Unfortunately no characterization of liveness exists in linear algebraic terms. The net in Fig. 1.12b is structurally live. Adding a token to p_5 , all p -semiflows remain marked, but it is non-live.

(d) Reversibility (and liveness)

Let us now use a **Liapunov-stability-like** technique to prove that the net system in Fig. 1.9 is reversible. It serves to illustrate the use of invariants and some inductive reasonings.

As a preliminary consideration that makes the rest of the proof easier, the following simple property will be used: let $\langle N, M_1 \rangle$ be a reversible system and M_0 reachable from M_1 (i.e. $\exists \sigma$ such that $M_1[\sigma] M_0$). Then $\langle N, M_0 \rangle$ is reversible.

$$\begin{aligned} M_1(\text{wait raw}) &= 0, & M_1(\text{wait dep.}) &= 1 \\ M_1(\text{empty}) &= 0, & M_1(\text{object}) &= 7 \end{aligned}$$

Let us prove first that $\langle N, M_1 \rangle$ is reversible. Let L be a non-negative place weighting such that $L(p_i) = 0$ iff p_i is marked in M_1 . Therefore, $L(\text{wait dep.}) = L(R) = L(\text{object}) = L(\text{wait with.}) = 0$ and $L(p_j) > 0$ for all the other places. The function $V(M) = L^T \cdot M$ has the following properties:

$$V(M) \geq 0 \quad \text{and} \quad V(M_1) = 0$$

For the system in Fig. 1.9 a stronger property holds: $V(M) = 0 \Leftrightarrow M = M_1$. This can be clearly seen because $L^T \cdot M = 0 \Leftrightarrow M(\text{wait raw}) = M(\text{load}) = M(\text{op}_1) = M(\text{deposit}) = M(\text{empty}) = M(\text{op}_2) = M(\text{wait free}) = M(\text{unload}) = M(\text{withdrawal}) = 0$. Even more, it is easy to check the following:

$$M_1 \text{ is the present marking} \Leftrightarrow t_9 \text{ is the unique firable transition}$$

If there exists (warning: in Liapunov-stability criteria the universal quantifier is used!) a finite firing sequence (i.e. a finite trajectory) per reachable marking M_i such that $M_i \xrightarrow{\sigma_k} M_{i+1}$ and $V(M_i) > V(M_{i+1})$, in a finite number of transition firings $V(M) = 0$ is reached. Because $V(M) = 0 \Leftrightarrow M = M_1$, a proof that M_1 is reachable from any marking has been obtained (i.e. $\langle N, M_1 \rangle$ is reversible).

Pre-multiplying the state equation (1.4) by L^T we obtain the following condition:

$$\text{if } \sigma_k = t_j \text{ then } [L^T \cdot M_{i+1} < L^T \cdot M_i] \Leftrightarrow L^T \cdot C(t_j) < 0$$

Now, removing in Fig. 1.9 the places marked at M_1 (i.e. wait dep., R , object, wait with.) and firable transitions (i.e. t_9) an acyclic net is obtained, so there exists an L such that $L^T \cdot C(t_j) < 0 \forall j \neq 9$.

For example, taking as weights the levels in the acyclic graph we have:

$$\begin{aligned} L(\text{op}_1) &= L(\text{unload}) = 1 \\ L(\text{load}) &= L(\text{wait free}) = 2 \\ L(\text{wait raw}) &= L(\text{op}_2) = 3 \\ L(\text{deposit}) &= L(\text{withdrawal}) = 4 \\ L(\text{empty}) &= 5 \end{aligned}$$

and $L^T \cdot C = (-1, -1, -1, -1, -1, -1, -1, -1, +4, -1)$. In other words, the firing of any transition, except t_9 , decreases $V(M) = L^T \cdot M$.

Using the algorithmic deadlock-freeness explanation in section (b), the reversibility of $\langle N, M_1 \rangle$ is proven (observe that the p-invariants in eqs (1.5-1.8) remain for M_1):

$$\text{if } M(\text{load}) + M(\text{op}_1) + M(\text{deposit}) + M(\text{op}_2) + M(\text{unload}) + M(\text{withdrawal}) \geq 1$$

then $V(M)$ can decrease firing t_2, t_3, t_5, t_6, t_8 or t_{10}

also if $M(\text{wait raw}) + M(\text{wait free}) > 1$

else $V(M)$ can decrease firing t_4 or t_9 is the unique firable transition $\{\Leftrightarrow M_1$ is the present marking}

Because M_0 is reachable from M_1 (for example, by means of $\sigma = (t_9 \cdot t_{10} \cdot t_6 \cdot t_7 \cdot t_8)^5 \cdot t_4 \cdot t_5$), $\langle N, M_0 \rangle$ is a reversible system.

Once again liveness of the system in Fig. 1.9 can be proved, because the complete sequence (i.e. containing all transitions) $\sigma = t_1 \cdot t_2 \cdot t_3 \cdot t_4 \cdot t_5 \cdot t_9 \cdot t_{10} \cdot t_6 \cdot t_7 \cdot t_8$ can be fired. Since the system is reversible, no transition loses the possibility of firing (i.e. all transitions are live).

1.7 SOME NET SUBCLASSES AND THEIR ANALYSIS

Net subclasses will be defined exclusively by introducing constraints on the structure of ordinary nets. Therefore it is very easy to recognize if a net model belongs to a subclass (i.e. the **membership problem**). By restricting the generality of the model, it will be easier to study its behavior. In particular, powerful structural results allow us to fully characterize some (otherwise hard to study) properties such as liveness and reversibility.

1.7.1 Definition of four net subclasses

Let us introduce four important subclasses summarized in Fig. 1.23.

Definition 1.8. A state machine (SM) is an ordinary net such that:

$$\forall t \in T \quad |^*t| = 1 \quad \text{and} \quad |t^*| = 1$$

that is, any transition has one input and one output place.

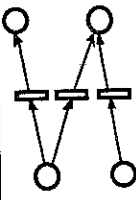

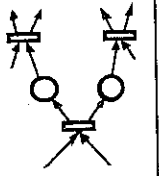

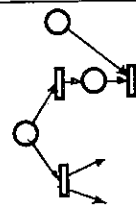
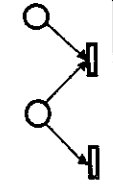
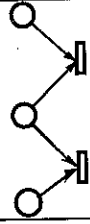
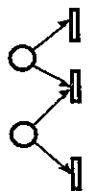
State machines allow the modeling of decisions (conflicts) and re-entrance. It is important to note that the concept of state machine, considered as a subclass of nets, is more general than the classical state diagram or state graph, since it can have more than one token. In any case, state machines do not 'create' tokens; thus can model only finite state systems.

Definition 1.9. A marked graph (MG) (also event or synchronization graph) is an ordinary net such that:

$$\forall p \in P \quad |^*p| = 1 \quad \text{and} \quad |p^*| = 1$$

that is, any place has one input and one output transition.

MGs are a subclass of structurally decision-free nets. They can model systems for ordering activities as PERTs (program evaluation and review techniques) do. They are more general than PERTs in the sense that **recycling** is allowed and places can contain several tokens. Moreover, pro-

	LEGAL	FORBIDDEN
State Machines	 <p>SM</p>	
Marked Graphs	 <p>MG</p>	
Free Choice Nets	 <p>FCN</p>	
Simple Nets	 <p>SN</p>	

• Present neither synchronizations (JOINS) nor structural parallelism (FORKS), but allow reentrancy and concurrency
 • Only model finite state systems
 • Analysis and synthesis theory is well understood

• Allow modeling synchronizations and structural parallelism but cannot model choices
 • Can model non-finite state systems
 • Allow modeling systems for ordering activities similar to PERTs, even integrating repetitive behaviours
 • Analysis and synthesis theory is well understood

• Allow modeling sequence, choice and concurrence relationships
 • Choice and synchronizations cannot coincide on the same transition
 • Do not allow modeling sequential subsystems synchronized through mutual exclusion semaphores
 • Analysis and synthesis theory is well understood

• Each transition has at most one input place shared with other transitions
 • Allow modeling sequence, choice and concurrence relationships
 • Choices are not free in general but can be solved locally
 • Allow modeling sequential subsystems synchronized through mutual exclusion semaphores
 • Analysis theory is partially understood

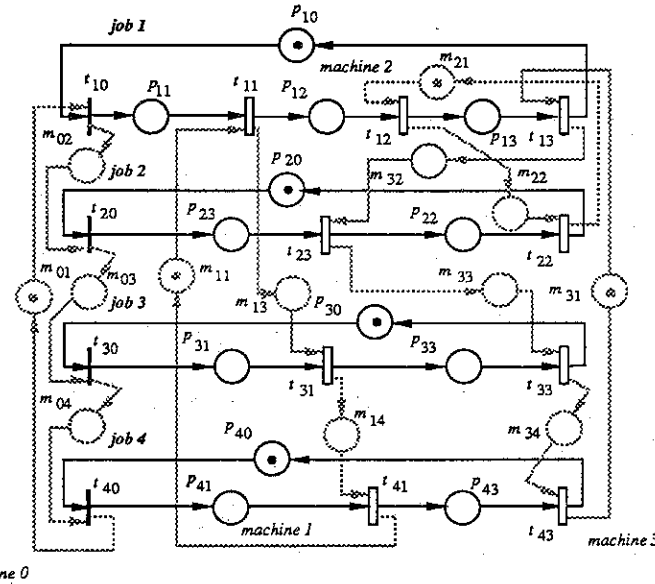


Figure 1.24 A job-shop system modeled with a marked graph (Hillion and Proth, 1989).

Figure 1.23 Four fundamental ordinary net subclasses.

are equivalent (i.e. have the same descriptive power) to **fork/join queuing networks with blockings (FJQN/B)** (Dallery *et al.*, 1990).

Even if MGs are a very restrained class of nets, they may appear in the modeling of job-shop systems, once **production routing** of jobs and the **machine sequencing** are uniquely defined.

Let us assume that we want to model a 4-jobs–3-machines manufacturing system under the following fixed route control strategy:

- (a) Product mix: 25% of job i , $\forall i = 1, 4$;
- (b) Jobs should visit the machines as follows:

job 1: machine 1, machine 2, machine 3
 job 2: machine 3, machine 2
 job 3: machine 1, machine 3
 job 4: machine 1, machine 3

- (c) Machines should sequence the jobs as follows:

M_1 : job 1, job 2, job 3, job 4
 M_2 : job 1, job 2
 M_3 : job 1, job 2, job 3, job 4

The jobs to be performed are modeled with the horizontal circuits:

- job 1: $p_{10} \cdot t_{10} \cdot p_{11} \cdot t_{11} \cdot p_{12} \cdot t_{12} \cdot p_{13} \cdot t_{13} \cdot p_{10}$
- job 2: $p_{20} \cdot t_{20} \cdot p_{23} \cdot t_{23} \cdot p_{22} \cdot t_{22} \cdot p_{20}$
- job 3: $p_{30} \cdot t_{30} \cdot p_{31} \cdot t_{31} \cdot p_{33} \cdot t_{33} \cdot p_{30}$
- job 4: $p_{40} \cdot t_{40} \cdot p_{41} \cdot t_{41} \cdot p_{43} \cdot t_{43} \cdot p_{40}$

Because machines process only one job at a time, vertical circuits marked with a single token are added to determine the sequencing of the jobs on the corresponding machines (i.e. the token represents the availability of the machine to process a job):

- machine 1: $m_{11} \cdot t_{11} \cdot m_{13} \cdot t_{31} \cdot m_{14} \cdot t_{41} \cdot m_{11}$
- machine 2: $m_{21} \cdot t_{12} \cdot m_{22} \cdot t_{22} \cdot m_{21}$
- machine 3: $m_{31} \cdot t_{13} \cdot m_{32} \cdot t_{23} \cdot m_{33} \cdot t_{33} \cdot m_{34} \cdot t_{44} \cdot m_{31}$

Another class of examples where MGs are useful to model manufacturing problems is the following. A **flow line** is a tandem production system (i.e. a series of machines separated by buffers) (Gershwin, 1987). Material flows from the first to the last machine. Figure 1.25(a) shows a flow line with three machines and two buffers. The MG in Fig. 1.25(b) makes explicit the behavior of the model with **reliable** machines, assuming the so-called **blocking-after service mechanism** (i.e. machine processes even if there is no free-space in the output buffer).

Here it is worth noting that the modeling capacities of MGs and SMs are **dual** in the sense that SMs can model choices, but not synchronizations. On the other hand, MGs can model synchronizations, but cannot model choices.

The subclasses we introduce below contain SMs and MGs; therefore, they can model some restricted interleaving between choices and synchronizations, although not in all the cases that ordinary nets allow. Free-choice nets (FCNs) can be considered as an extension of:

- (a) SMs by allowing MGs-type synchronization (i.e. if two places share a common output transition then this is their unique output transition); or
- (b) MGs by allowing SMs-type conflicts (i.e. if two transitions share a common input place, then this is their unique input place).

It is not difficult to realize that both statements represent identical restrictions. FCNs are a common generalization in which choices and synchronizations do not directly interfere with each other.

Definition 1.10. A **free-choice net** (FCN) is an ordinary net such that:

$$\forall p \in P, |p^*| > 1 \Rightarrow \forall t_k \in p^*, |t_k^*| = 1$$

That is, if two transitions, t_i and t_j , have a common input place p , it is the only input place of t_i and t_j .

From a behavioral point of view, FCNs are models in which either all or

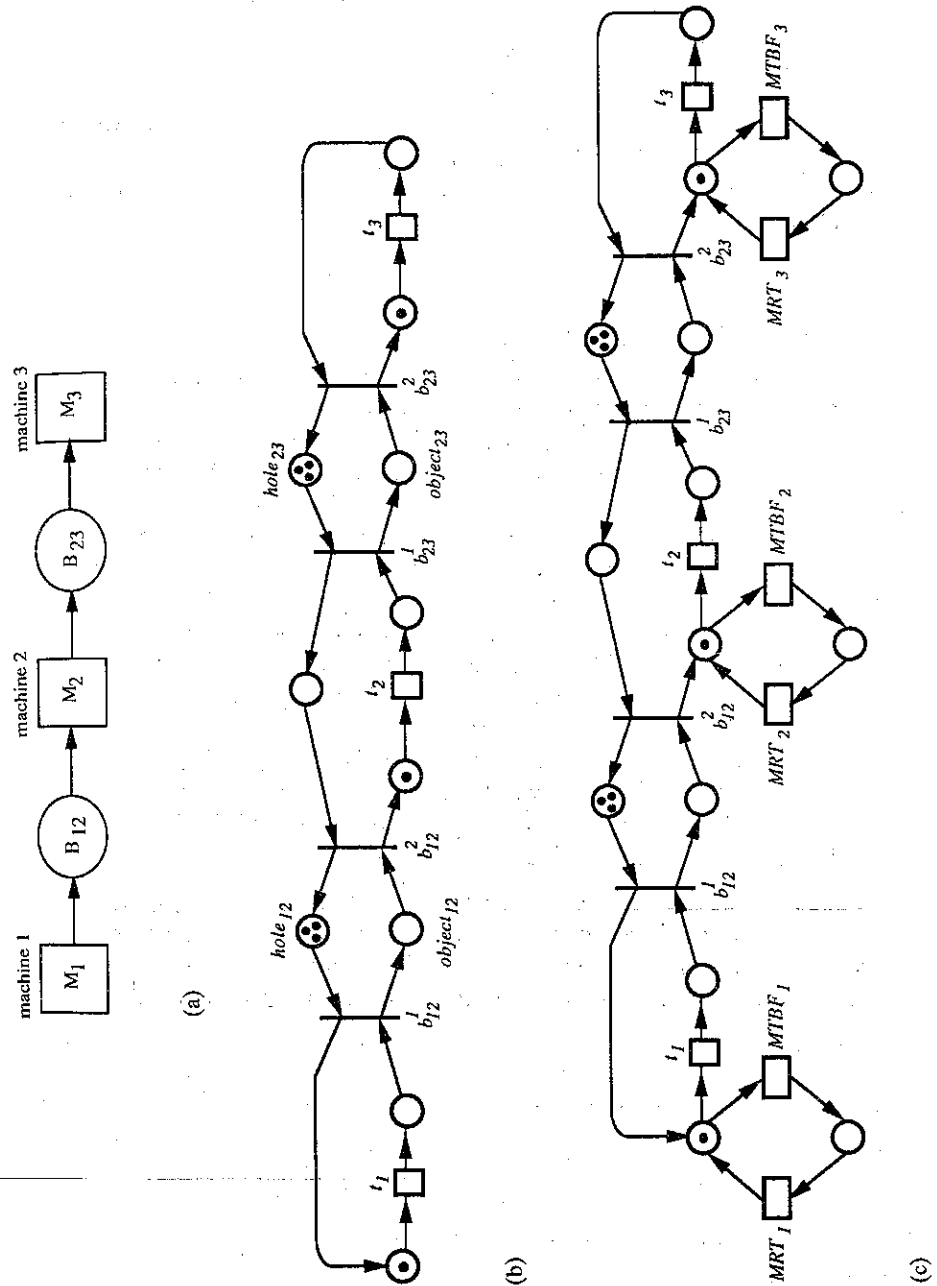


Figure 1.25 A flow line and two net systems representations: (a) a flow line with three machines and two buffers; (b) MG model assuming reliable machines; (c) Macroplace expansion for unreliable machines.

place is marked and has more than one output transition, the transition to be fired can be freely chosen (i.e. independently of the rest of the marking). Hence the name. The behavioral and structural analysis of FCNs is particularly elegant and well understood.

When the machines of the flow line (Fig. 1.25(a)) are considered **unreliable**, the corresponding refinement of the machine-working places leads to the FCN of Figure 1.25(c) (MTBF: mean time between failures; MRT: mean repairing time).

Provided with adequate stochastic interpretation, FCNs extend the FJQN/B model allowing **random routings**. The net in Fig. 1.11(c) is free-choice. The net in Fig. 1.8 is also free-choice. Stochastic free-choice nets can also be viewed as **free-choice synchronized queuing networks** (Campos *et al.*, 1991).

Free-choice nets do not allow the modeling of sequential subsystems synchronized through mutual exclusion semaphores (i.e. shared resources). The next net subclass, simple nets, allows it in some simple case: when no more than one exclusion semaphore is considered in any synchronization. In simple nets, choices are not free in general but they can be **solved locally**, because each choice is centered around a unique shared place.

Definition 1.11. A **simple net (SN)** is an ordinary net in which each transition has at most one input place shared with other transitions. An SN is such that:

$$\forall t \in T \{ |p \in {}^*t \text{ such that } |p'| > 1| \} \leq 1$$

In spite of its relative generality, this subclass has some interesting properties. Nevertheless, its behavior is by far not so well understood as that of FCNs. Many systems are modeled with SNs. The typical basic example of an SN is the model of a system in which a resource is shared by two or more users (Fig. 1.4(g)). The net of our production cell with two machines, one robot and a store (Fig. 1.9) is simple. Other simple non-free-choice nets are in Figs 1.12(b), 1.15, 1.17(a), 1.19 and 1.22(a).

1.7.2 On the analysis of the net systems subclasses

The structure of the considered net subclasses is rich enough to give plenty of information on the net systems we can define by putting an initial marking. This is particularly true for FCN systems and their subclasses, MG and SM systems. For SN systems there exist also some interesting results, but the stronger properties of free-choice systems cannot be extended (e.g. the rank theorem, or the liveness monotonicity with respect to the initial marking).

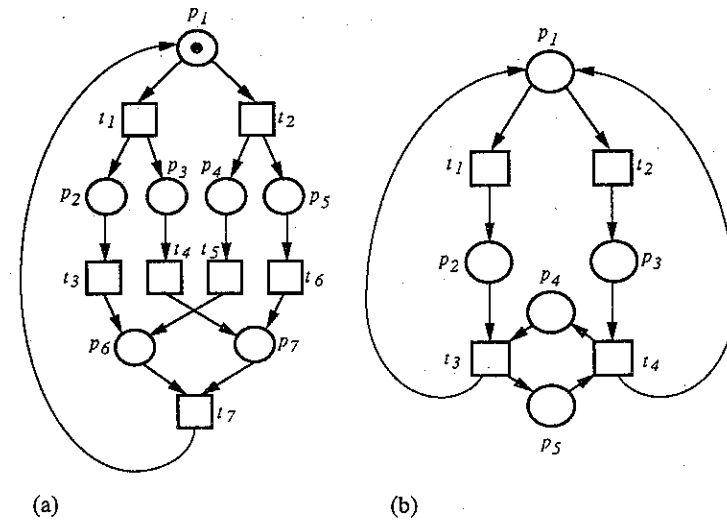


Figure 1.26 Two consistent and conservative free choice nets: (a) Structurally live ($r = \rho$), $r = \text{rank}(C) = 5$, $\rho = m + n - a - 1 = 7 + 7 - 8 - 1 = 5$; (b) Structurally non-live ($r \neq \rho$), $r = \text{rank}(C) = 3$, $\rho = m + n - a - 1 = 4 + 5 - 6 - 1 = 2$.

(a) Siphons, traps, liveness and reversibility

By means of graph-theory based reasoning it is possible to characterize many properties for net subclasses. **Siphons** (also called **structural deadlocks**, or more simply **deadlocks**) and **traps** are easily recognizable subsets of places that generate very particular subnets.

Definition 1.12. Let $N = \langle P, T, F \rangle$ be an ordinary net.

1. A **siphon** is a subset of places such that the set of its input transitions is contained in the set of its output transitions: $\Sigma \subseteq P$ is a siphon $\Leftrightarrow {}^*\Sigma \subseteq \Sigma^*$.
2. A **trap** is a subset of places such that the set of its output transitions is contained in the set of its input transitions: $\theta \subseteq P$ is a trap $\Leftrightarrow \theta^* \subseteq \theta$.

$\Sigma = \{p_1, p_2, p_4, p_5, p_6\}$ is a siphon for the net in Fig. 1.26(a): ${}^*\Sigma = \{t_7, t_1, t_2, t_3, t_5\}$, while $\Sigma^* = {}^*\Sigma \cup \{t_6\}$. Σ contains a trap, $\theta = \Sigma \setminus p_5$. In fact θ is also a siphon (it is minimal: removing any number of places no siphon can be obtained).

Siphons and traps are reverse concepts: a subset of places of a net N is a siphon iff it is a trap on the reverse net, N^{-1} (i.e. that obtained reversing the arcs, its flow relation, F).

The following property 'explains' why 'siphons' (think of 'soda siphons')

Property 1.14

1. Siphons:
 - (i) If M is a behavioral deadlock (i.e. dead-marking), then $D = \{p/M(p) = 0\}$ is an unmarked (empty) siphon.
 - (ii) If a siphon is (or becomes) unmarked, it will remain unmarked for any possible net system evolution. Therefore all its input and output transitions are dead. So the system is not live (but can be deadlock-free).
2. Traps: If a trap is (or becomes) marked, it will remain marked for any possible net system evolution (i.e. at least one token is 'trapped').

If a trap is not marked by M_0 , and the system is live, M_0 will not be recoverable from those markings in which the trap is marked. Thus:

Corollary 1.1. If a live net system is reversible, then M_0 marks all traps.

For live and bounded free-choice systems a stronger property holds: Marking all traps is a necessary and sufficient condition for reversibility (Best *et al.*, 1990). The net system in Fig. 1.26(a) is reversible. Nevertheless, if $M_0^T = (0\ 1\ 0\ 0\ 1\ 0\ 0)$, the new system is live and bounded but non-reversible: The trap $\theta = \{p_1, p_3, p_4, p_6, p_7\}$ is not marked under M_0 .

A siphon which contains a marked trap will never become unmarked. So this more elaborate property can be helpful for some liveness characterizations.

Definition 1.13. Let N be an ordinary net. The system $\langle N, M_0 \rangle$ has the **marked-siphon-trap property**, MST-property, if each siphon contains a marked trap under M_0 .

A siphon (trap) is **minimal** if it is not contained in any other. Thus siphons in the above statement can be constrained to be minimal without any loss of generality.

The MST-property guarantees that all siphons will be marked. Thus no dead marking can be reached, according to property 1.14(1)(i). Therefore:

Property 1.15. If $\langle N, M_0 \rangle$ has the MST-property, the system is deadlock-free.

Figure 1.27 presents some limitations of the MST-property for liveness characterization. Nevertheless, there exist the following interesting results on liveness.

Property 1.16. The MST-property is sufficient for liveness in simple net systems and necessary and sufficient for free-choice net systems.

As a corollary, the following **liveness monotonicity** result is true:

Corollary 1.2. Let $\langle N, M_0 \rangle$ be a live free-choice system. Increasing M_0

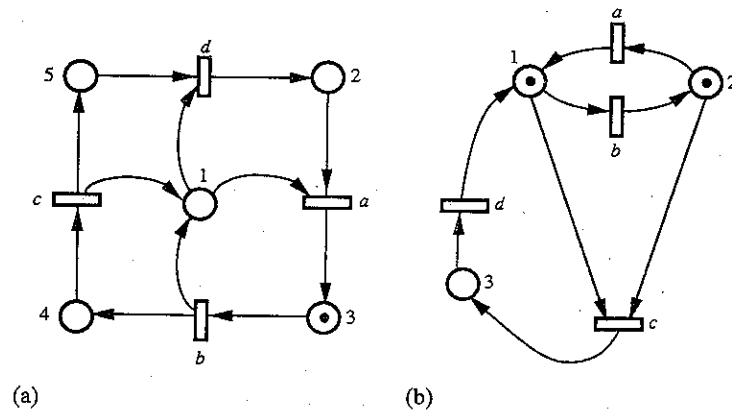


Figure 1.27 For liveness analysis the marked siphon-trap-property is not necessary for simple bounded nets nor sufficient for non-simple bounded nets: (a) the marked siphon-trap-property does not hold and the simple net is live and bounded; (b) the marked siphon-trap-property does not hold but the non-simple net is non-live (although deadlock-free) and bounded.

The above result does not apply to SN systems. The system in Fig. 1.12(b) is simple, $\Sigma = \{p_1, p_2, p_7\}$ is a siphon (${}^*\Sigma = \{t_3, t_4, t_1\}$, $\Sigma^* = {}^*\Sigma \cup \{t_2\}$) that does not contain any trap. If we assume $M_0(p_5) = 1$, t_2 can be fired and Σ becomes empty, leading to non-liveness.

SMs and MGs are FCNs. The set of places of an SM is a minimal siphon and a minimal trap iff it is **strongly connected**. Moreover, any **elementary circuit** of an MG is simultaneously a minimal siphon and a minimal trap. So the following can be stated.

Property 1.17

1. An SM is live for M_0 iff it is strongly connected and marked under M_0 (i.e. there is at least one token).
2. An MG is live for M_0 iff all circuits are marked under M_0 .

Obviously, liveness is a **polynomial complexity** problem for SMs. By conventional logic equivalence the last property can be easily restated as follows: *an MG is live under M_0 iff there is no unmarked circuit*. From this statement it follows immediately that liveness can also be computed in polynomial time for MGs:

1. Remove from N all marked places under M_0 to obtain the net N_0 ;
2. The system $\langle N, M_0 \rangle$ is live iff N_0 is an acyclic graph.

The MG systems in Figs 1.24 and 1.25 are obviously live.

(b) Linear algebra, structural liveness and liveness

Structural boundedness is well characterized for general Petri nets (see

for structurally bounded FCNs and for MGs. Liveness for those structurally live net systems is also fully characterized in linear algebraic terms.

Property 1.18. Let C be the incidence matrix of the free-choice net N . N is structurally live and structurally bounded iff:

$\exists X \geq 1$ such that $C \cdot X = 0$ (i.e. net consistency)
 $\exists Y \geq 1$ such that $Y^T \cdot C = 0$ (i.e. net conservativeness)
 $\text{rank}(C) = |T| + |P| - a - 1$, where a is the number of arcs in the flow relation *Pre*.

With respect to Property 1.12(2), necessary and sufficient conditions are now given. Moreover, the rank condition is now an *equality*, and δ is substituted by its value for FCNs: $\delta = a - |P|$.

The above property allows us to conclude on structural liveness (Fig. 1.26(a)) and structural non-liveness (Fig.1.26(b)) for consistent and conservative FCNs.

The particularization of Property 1.18 for SMs and MGs shows that in both cases the rank condition is redundant: it is satisfied when consistency and conservativeness hold. Even more, the following can be shown:

Property 1.19. A connected SM (MG) is consistent (conservative) if the net is strongly connected.

Connected SMs are conservative (then structurally bounded), thus structurally live iff strongly connected. Connected MGs are consistent, thus structurally bounded and structurally live iff strongly connected. This very last result can be generalized interpreting Property 1.17(2): with 'large enough' M_0 , any MG is live. In other words, **MGs are structurally live nets**.

Property 1.18 has several important consequences (Esparza and Silva, 1991a): (1) Structural liveness can be decided in **polynomial time** (solving the consistency and conservativeness problems, and computing the rank of a matrix); (2) **duality theorem**; (3) a **kit of two reduction rules** (that is **complete**, i.e. is able to reduce all structurally live and structurally bounded FCNs).

Property 1.18 characterizes the lively and boundedly markable FCNs. Once we have one of these nets, we would like to know which are exactly the markings that make it live and bounded.

Property 1.20. Let N be a structurally live and structurally bounded FCN. $\langle N, M_0 \rangle$ is live iff all p -semiflows of N are marked at M_0 (i.e. $\exists Y \geq 0$ such that $Y^T \cdot C = 0$ and $Y^T \cdot M_0 = 0$).

The necessary condition holds in general: if a p -semiflow (that is always a siphon and a trap) is unmarked at M_0 , it remains unmarked forever. Liveness **monotonicity** for live and (structurally) bounded FCN systems is also a consequence of the above property. Moreover, liveness for structurally

(c) *Reversibility, reachability and marking bounds*

Not all live and bounded FCNs are reversible (imagine $M_0^* = (0, 1, 0, 0, 1, 0, 0)^T$ for the net in Fig. 1.26(a)). Nevertheless, it is always possible to reach from M_0 a marking such that a unique reversible system is obtained (i.e. there always exists a so-called **home state**). The following property characterizes the live, bounded and reversible FCN systems.

Property 1.21. Let $\langle N, M_0 \rangle$ be a live and bounded FCN system. It is reversible iff all traps are marked under M_0 .

The reachability problem for the class of systems being considered is also solved (Desel and Esparza, 1990):

Property 1.22. Let $\langle N, M_0 \rangle$ be a live, reversible and bounded FCN system. The three following statements are equivalent:

- (i) M is reachable from M_0 (i.e. $\exists \sigma$ such that $M_0[\sigma \rangle M$);
- (ii) $M = M_0 + C \cdot \bar{\sigma}$, $\bar{\sigma} \geq 0$ where $M \in N^n$ marks all minimal traps;
- (iii) $B^T \cdot M = B^T \cdot M_0$, where B is a basis of left annullers (i.e. $B^T \cdot C = 0$) and $M \in N^n$ marks all minimal traps.

Strongly connected SMs have one minimal trap ($\theta = P$) and $B = 1$. Therefore (Property 1.21) live SMs are reversible and the reachable markings are the non-negative integer solutions of $\Sigma_p M(p) = \Sigma_p M_0(p)$. For MGs minimal traps coincide with elementary circuits. Therefore liveness is equivalent to reversibility for strongly connected MGs. Even more, the above is true for MGs in general:

Property 1.23. Let $\langle N, M_0 \rangle$ be a live (possibly unbounded) MG system. The three following statements are equivalent:

- (i) M is reachable from M_0 (i.e. $\exists \sigma$ such that $M_0[\sigma \rangle M$);
- (ii) $M = M_0 + C \cdot \bar{\sigma}$, $\bar{\sigma} \geq 0$, where $M \in N^n$
- (iii) $B_f^T \cdot M = B_f^T \cdot M_0$, where $M \in N^n$ and B_f is the **fundamental circuit** matrix of the graph.

Therefore it is easy to show that (possibly unbounded) MG systems are reversible iff live. Finally, using reachability characterizations it is not difficult to derive the following last property:

Property 1.24. Let $\langle N, M_0 \rangle$ be a live and bounded FCN system or a live (possibly unbounded) MG system. The behavioral and structural bounds of place p coincide:

$$\begin{aligned} B(p) &= \max\{M(p)/M \in R(N, M_0)\} = SB(p) \\ &= \max\{M(p)/M = M_0 + C \cdot \bar{\sigma} \geq 0, \bar{\sigma} \geq 0\} \end{aligned}$$

As an example, the above means (for the class of systems being considered) that the maximum required size of stores (buffers) can be computed

The net system in Fig. 1.9 is simple but not free-choice (thus it is not a marked graph). As an example of **complementarity** among analysis techniques, let us prove its main global properties combining **reductions** and **structure theory** for net subclasses:

1. Reduction phase:

- Use rule RA2 (Fig. 1.16) to fuse the following series of transitions of Fig. 1.9:

$$\begin{aligned} t_1\text{-load-}t_2 &\Rightarrow t_{12} \\ t_4\text{-deposit-}t_5 &\Rightarrow t_{45} \\ t_7\text{-unload-}t_8 &\Rightarrow t_{78} \\ t_9\text{-withdrawal-}t_{10} &\Rightarrow t_{910} \end{aligned}$$

- Use (marking) implicit place rule (only the basic extension of self-loop place, RC1 in Fig. 1.16) to remove place R .
2. The remaining net is a **marked graph**, therefore:
- The MG is strongly connected, then conservative (Property 1.19). Conservativeness is a particular case of structural boundedness. Therefore the MG system is bounded.
 - Removing marked places (i.e. wait raw, *empty*, *object*, wait with.) an acyclic net is obtained. Then the MG system is live (Property 1.17(2)) and reversible (follows from Property 1.22 or Property 1.23).

Rules RA2 and RC1 preserve boundedness, liveness and reversibility. Therefore, the original manufacturing model is bounded, live and reversible.

1.8 CONCLUDING REMARKS

Petri nets theory and applications is a vast field. In this chapter a selection of **modeling issues** have been presented first. **Autonomous net systems** are abstract models. **Interpreted net systems** are 'natural' extensions of well-known formalisms like **state diagrams** (a formalism to model the functional behavior of sequential switching systems) or **queuing networks** (a formalism to model performance issues of systems where the sharing of resources plays an important role, but there are no synchronizations). **Marking diagrams** and **synchronized queuing networks** are examples of such (particular) net interpreted models.

The GRAFCET (see, for example, David and Alla (1989)), an international standard of the Commission Electrotechnique Internationale (CEI #848, *Preparation of function charts for control systems*) is an interpreted bipartite graph, 1-bounded by definition, closely related in some aspects to marking diagrams. GRAFCETs are difficult to validate in general because too much information is put in the interpretation. They have been exten-

Qualitative (i.e. logical) analysis issues of net system models have been considered in the second part of the chapter. We tried to present a compilation of some practical results. The reader is referred to the literature where many more interesting analytical results are documented. Complementarity of qualitative analysis techniques has been emphasized. Moreover, quantitative (i.e. performance) modeling issues have just been introduced in the framework of net interpretations. A consideration in manufacturing, including quantitative analysis techniques, is delayed to Chapter 4. At the research level a main trend is the in-depth interleaving of qualitative and quantitative analysis techniques.

The behavior of non-autonomous net systems may be constrained by the environment. Therefore the interpreted net system behavior may be strictly included in the behavior of the underlying autonomous net system. Therefore, attention must be paid to the fact that the analysis of the underlying net system may be for the interpreted one **only necessary** (e.g. for marking reachability), **only sufficient** (e.g. for boundedness or mutual exclusion) or **neither necessary nor sufficient** (e.g. for liveness). In other words, the designer should interpret the properties of the underlying net system, given the constraints imposed by the environment: the results of the net system qualitative analysis can be considered as **elaborated warnings**. This generalizes from purely syntactical to a certain semantical level, those warnings in the compilation of programs.

Petri nets as introduced here represent a level in the hierarchy of net level models; namely **place/transition nets**. Subclasses have been considered to increase the practical decision power at the expense of practical modeling power. On the other side, extensions of place/transition net models allow us to increase the practical modeling power (e.g. **capacity** Petri nets: all places are bounded by a given capacity) or even the theoretical modeling power (e.g. **inhibitor arcs** or **transition priority** nets have the same descriptive power as **Turing machines!**). Therefore they are difficult to analyze in general. Nevertheless, in bounded systems inhibitor arcs or priority at transitions do not enlarge the modeling power. They are only practical modeling facilities.

Another kind of extension leads to the so-called **High-Level Nets (HLN)** (see Jensen and Rozenberg (1991) for a broad and recent perspective). In HLN tokens have attributes (e.g. called colors) and arcs receive some inscriptions concerning the flow and transformation of the attributes of tokens. HLN allow us to make much more compact models (see, for example, in manufacturing, Martinez *et al.* (1986) or Silva and Valette (1990)). Their analysis is presently not so mature as for place/transition nets.

Nets potentials for modeling discrete event dynamic systems (a consistent, graphical/analytical family of models covering the main modeling issues), and for its qualitative and quantitative analysis are not the only basic arguments to use nets. **Real-time control systems** can be derived from

(i.e. microprogrammed) and **software** level (see Silva (1985)) for a broad overview of techniques). The generation of real-time software systems received particular attention in the past, putting nets close to **programmable logic controllers** (Colom *et al.*, 1986; Silva and Valette, 1990).

1.9 BIBLIOGRAPHICAL REMARKS

Petri nets were introduced in the Ph.D. of Carl Adam Petri (Petri, 1962). Today it is a very rich but relatively young field having an impact on many different industrial sectors. More than in seminal/historical papers we mainly (but not only) refer to books, tutorials or surveys, where the specialized contributions are explicitly pointed. A bibliography on Petri nets is periodically gathered by the *Gesellschaft für Mathematik und Datenverarbeitung* (GMD). The last issue was compiled in 1986 and published in Rozenberg, 1987, where some 2,634 entries are quoted.

Introductory texts to Petri nets and their applications have been written by Peterson (1981), Brams, (collective name of a group of French researchers) (Brams 1983), Silva (1985) and David and Alla (1989).

The material of two advanced courses on Petri nets is collected in Brauer (1980), Brauer *et al.* (1987a,b). There is a subseries of *Lecture Notes in Computer Science* (LNCS) entitled *Advances in Petri Nets* (Rozenberg, various dates). The *International Conference (European Workshop until 1989) on Application and Theory of Petri Nets* takes place every year, since 1980. The Proceedings are published by the organizers with the support of IBM-Germany. Published by the IEEE Computer Society Press, there are the *Proceedings of the International Workshops on Petri Nets and Performance Models* (PNPM) focusing on the time qualitative and quantitative aspects. Focused on *High-Level Petri Nets* (Jensen and Rozenberg, 1991) is a selection of papers.

There exist several surveys on Petri nets in general. Murata (1989) gives a broad perspective (containing 315 bibliographical entries) and received the IEEE Donald G. Fink Prize Award. Petri nets and manufacturing have also been the subject of several tutorials and surveys. Martinez *et al.* (1986) is a basic tutorial for manufacturing engineers introducing concepts on modeling with Petri nets. More survey-oriented are Valette (1987) and Silva and Valette (1990). The last one assumes knowledge on Petri nets basic concepts. Basically it focuses on some manufacturing features and how Petri nets may be helpful (contains more than 150 bibliographical entries). Invited or regular sessions on Petri nets and manufacturing appear at several conferences (e.g. IEEE International Conference on Robotics and Automation, World Congress of IMACS, and IEEE International Symposium on Circuits and Systems).

Eschler (1990) is a recent paper on the construction of **coverability graphs**.

Berthelot (1987). An overview of seminal works on this topic is Berthelot *et al.* (1980). Silva (1981) studies the macroplace reduction rule, while Silva (1985) and Colom and Silva (1991b) generalize the implicit (or redundant) place concept from Berthelot *et al.* (1980). The reverse process of reduction is **stepwise refinement** for which Valette (1979) and Suzuki and Murata (1983) can be taken into account. A more general/abstract perspective of the topic is summarized in Brauer *et al.* (1991).

A seminal overview on the **state equation** based analysis of net models is Memmi and Roucairol (1980). The bridge between Petri nets and linear programming is covered in Silva and Colom (1988) and Colom and Silva (1991b). The identification of **minimal invariants** as **extreme directions of a cone** allows us to derive fast algorithms to compute the sets of minimal *p*- and *t*-semiflows of a net (Colom and Silva, 1991a). Additional results on linear algebra and Petri nets are in Colom *et al.* (1990b). **Structure theory** for net subclasses is surveyed in several works. Best (1987), Best and Thiagarajan (1987) and Esparza and Silva (1991a) are mainly concerned with the **free choice** subclass. Commoner *et al.* (1971) and Murata (1977) develop the basic theory of **marked graphs**. In Thulasiraman and Comeau (1987) linear programming is considered for MGs.

BIBLIOGRAPHY

- Ajmone, M., Balbo, G. and Conte, G. A class of generalized stochastic Petri nets for the performance analysis of multiprocessor systems. *ACM Transactions on Computer Systems*, 2(2), 93-122, May 1984.
- Ajmone, M., Balbo, G. and Conte, G. *Performance Models of Multiprocessor Systems*. MIT Press, Cambridge, MA, 1987.
- Ajmone, M., Balbo, G., Bobbio, A., Chiola, G., Conte, G. and Cumani, A. The effect of execution policies on the semantic and analysis of stochastic Petri nets. *IEEE Transactions on Software Engineering*, 15(7), 832-846, July 1989.
- Al-Jaar, R. Y. and Desrochers, A. A survey of Petri nets in automated manufacturing systems. In *IMACS World Congress*, vol. 2, pp. 503-510, Paris, June 1988.
- Berthelot, G. Transformations and decompositions of nets. In W. Brauer *et al.* (1987a), pp. 359-376, 1987.
- Berthelot, G., Roucairol, G. and Valk, R. Reductions of nets and parallel programs. In Brauer (1980), pp. 277-290, 1980.
- Best, E. Structure theory of Petri nets: the free choice hiatus. In W. Brauer *et al.* (1987a), pp. 168-205, 1987.
- Best, E. and Thiagarajan, P. S. Some classes of live and save Petri nets. In Voss *et al.* (1987), pp. 71-94, 1987.
- Best, E., Cherskasova, L., Desel, J. and Esparza, J. Characterization of home states in free choice systems. *Hildesheimer Informatik-Berichte*, no. 7/90, July 1990.
- Brams, G. W. *Réseaux de Petri: théorie et pratique* (2 vols). Paris, Masson, 1983.
- Brauer, W. (ed.) *Net Theory and Applications*. LNCS '84, Springer-Verlag, Berlin, 1980.
- Brauer, W., Reisig, W. and Rozenberg, G. *Petri Nets: Central Models and their*

- Brauer, W., Reisig, W. and Rozenberg, G. *Petri Nets: Applications and Relationships to other Models of Concurrency*. LNCS 255, Springer-Verlag, Berlin, 1987b.
- Brauer, W., Gold, R. and Vogler, W. A survey of behaviour and equivalence preserving refinements of Petri nets. *Advances in Petri Nets '90*. (G. Rozenberg, ed.), LNCS 483, pp. 1–46. Springer-Verlag, Berlin, 1991.
- Breeding, K. I. *Digital Design Fundamentals*. Prentice-Hall, Englewood Cliffs, NJ, 1989.
- Bruno, G. and Marchetto, G. Process translatable Petri-nets for the rapid prototyping of control systems. *IEEE Transactions on Software Engineering*, vol. SE-12, no. 2, pp. 346–357, February 1985.
- Campos, J., Colom, J. M. and Silva, M. Performance evaluation of repetitive automated manufacturing systems. In *Proceedings of the 2nd International Conference on Computer Integrated Manufacturing* (IEEE Computer Society Press). Troy, New York, pp. 78–91, May 1990.
- Campos, J., Chiola, G. and Silva, M. Properties and performance bounds for closed free choice synchronized monoclase queueing networks. *IEEE Transactions on Automatic Control*, 36(12), December 1991. [Special issue on *Multidimensional Queueing Networks*.]
- Colom, J. M. and Silva, M. Convex geometry and semiflows in P/T nets. *Advances in Petri Nets '90* (G. Rozenberg, ed.). LNCS 483, Springer-Verlag, Berlin, pp. 79–112, 1991a.
- Colom, J. M. and Silva, M. Improving the linearly based characterization of P/T nets. *Advances in Petri Nets '90* (G. Rozenberg, ed.). LNCS 483, Springer-Verlag, Berlin, pp. 113–145, 1991b.
- Colom, J. M., Silva, M. and Villarreal, J. L. On software implementations of Petri nets and colored Petri nets using high-level concurrent languages. In *Proceedings of the 7th European Workshop on Application and Theory of Petri Nets*. Oxford, pp. 207–241, July 1986.
- Colom, J. M., Campos, J. and Silva, M. On liveness analysis through linear algebraic techniques. *Dpto. Ing. Eléctrica e Informática, Research Report RR-90-11* (16 pp.). Also in *Deliverables of Esprit Basic Research Action Demon*, June 1990.
- Commoner, F., Holt, A. W., Even, S. and Pnueli, A. Marked directed graphs. *Journal of Computer and System Sciences*, vol. 9, no. 2, pp. 72–79, 1971.
- Dallery, Y., Liu, Z. and Towsley, D. Equivalence, reversibility and symmetry properties in fork/join queueing networks with blocking. *Université Pierre et Marie Curie, MASI Technical Report 90-32*, Paris, June 1990.
- David, R. and Alla, H. *Du grafet aux réseaux de Petri*. Hermes, Paris, 1989.
- Deo, N. *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, Englewood Cliffs, NJ, 1974.
- Desel, J. and Esparza, J. Reachability in reversible free-choice systems. *Technical University of Munich, SFB-Bericht*, No. 342/11/90A, June 1990.
- Ercegovac, M. and Lang, T. *Digital Systems and Hardware, Firmware Algorithms*. John Wiley & Sons, New York, 1985.
- Esparza, J. and Silva, M. On the analysis and synthesis of free choice systems. *Advances in Petri Nets '91* (G. Rozenberg, ed.). LNCS 483, Springer-Verlag, Berlin, pp. 243–286, 1991a.
- Esparza, J. and Silva, M. Top-down synthesis of live and bounded free-choice nets. In *Proceedings of the 11th International Conference on Applications and Theory of Petri Nets*, pp. 63–83. Paris, June 1991b.
- Finkel, A. A minimal coverability graph for Petri nets. In *Proceedings of the 11th International Conference on Applications and Theory of Petri Nets*, pp. 1–21.

- Genrich, H. J. and Lautenbach, K. Synchronisationsgraphen. *Acta Informatica* 2, pp. 143–161.
- Gershwin, S. B. Representation and analysis of transfer lines with machines that have different processing rates. *Annals of Operations Research*, 9, 511–530, 1987.
- Gibbons, A. *Algorithmic Graph Theory*. Cambridge University Press, London, 1985.
- Girault, C. and Reisig, W. (eds). *Application and Theory of Petri Nets*. Informatik-Fachberichte 52, Springer-Verlag, Berlin, 1982.
- Hack, M. T. Analysis of production schemata by Petri nets. *MIT, TR-94*. Boston, 1972 (corrected June 1974).
- Hillion, H. P. and Proth, J. M. Performance evaluation of job-shop systems using timed event-graphs. *IEEE Transactions on Automatic Control*, 34(1), 3–9, January 1989.
- Jensen, K. and Rozenberg, G. (eds). *High-Level Petri Nets. Theory and Application*. Springer-Verlag, Berlin, 1991.
- Karp, R. and Miller, R. Parallel program schemata. *Journal of Computer and System Science*, 3(4), 167–195, May 1969.
- Mailles, D. Files d'attente descriptives pour la modélisation de la synchronisation dans les systèmes informatiques. *Université P. et M. Curie. These d'Etat*. September 1987.
- Martínez, J., Alla, H. and Silva, M. Petri nets for the specification of FMSs. In *Modelling and Design of Flexible Manufacturing Systems* (A. Kusiak, ed.). Elsevier, pp. 389–406, 1986.
- Memmi, G. and Roucairol, G. Linear algebra in net theory. In Brauer (1980), pp. 213–223, 1980.
- Molloy, M. K. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, 31(9), 913–917, September 1982.
- Murata, T. Circuit theoretic analysis and synthesis of marked graphs. *IEEE Transactions on Circuits and Systems*, 24(7), 400–405, 1977.
- Murata, T. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, 77(4), 541–580, April 1989.
- Murty, K. G. *Linear Programming*. John Wiley & Sons, New York, 1983.
- Nelson, R. Haiht, L. and Sheridan, P. Casting Petri-nets into programs. *IEEE Transactions on Software Engineering*, 9(5), 590–602, September 1983.
- Nemhauser, G. L., Rinnoy Kan, A. H. G. and Todd, M. J. *Optimization*. Volume I of *Handbook in Operations Research and Management Science*. North-Holland, Amsterdam, 1989.
- Peterson, J. L. *Petri Net Theory and the Modelling of Systems*. Prentice Hall, Englewood Cliffs, NJ, 1981.
- Petri, C. A. *Kommunikation mit Automaten*. Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM No. 2, 1962.
- Proceedings of the European Workshop on Applications and Theory of Petri Nets (EWPN '81: Bad Honnef, Germany; EWPN '82: Varenna, Italy; EWPN '83: Toulouse, France; EWPN '84: Aarhus, Denmark; EWPN '85: Helsinki, Finland; EWPN '86: Oxford, UK; EWPN '87: Zaragoza, Spain; EWPN '88: Venezia, Italy; EWPN '89: Bonn, Germany).*
- Proceedings of the International Conference on Applications and Theory of Petri Nets (ICPN '89: Bonn, Germany; ICPN '90: Paris, France; ICPN '91: Gjern, Denmark).*
- Proceedings of the International Workshop on Petri Nets and Performance Models (PNPM '87: Madison, WI, USA, August 1987; PNPM '89: Kyoto, Japan, December 1989; PNPM '91: Melbourne, Australia, December 1991).* IEEE

- Proceedings of the International Workshop on Timed Petri Nets*. Torino, Italy IEEE-Computer Society Press, July 1985.
- Rozenberg, G. (ed.) *Advances in Petri Nets*. Lecture Notes in Computer Science: 188 (APN '84), 222 (APN '85), 266 (APN '87), 340 (APN '88), 424 (APN '89) and 483 (APN '90). Springer-Verlag, Berlin, various years.
- Silva, M. Sur le concept de macroplace et son utilisation pour l'analyse des reseaux de Petri. *RAIRO-Systems Analysis and Control*, 15(4), 57-67, 1981.
- Silva, M. *Las redes de Petri en la Automática y la Informática*. Editorial AC, Madrid, 1985.
- Silva, M. and Velilla, S. Programmable logic controllers and Petri nets. In *Proceedings of the International Symposium of the IFAC - IFIC on Software for Computer Control*, SOCOCO '82, pp. 29-34 (G. Ferraté and E. A. Puento, eds). Pergamon Press, Oxford, 1982.
- Silva, M. and Colom, J. M. On the computation of structural synchronic invariants in P/T nets. *Advances in Petri Nets '88* (G. Rozenberg, ed.). LNCS 340, pp. 386-417, Springer-Verlag, Berlin, 1988.
- Silva, M. and Valette, R. Petri nets and flexible manufacturing. *Advances in Petri Nets '89* (G. Rozenberg, ed.). LNCS 424, pp. 375-417. Springer-Verlag, Berlin, 1990.
- Suzuki, I. and Murata, T. A method for stepwise refinement and abstraction of Petri nets. *Journal of Computer and Systems Sciences*, 27(1), 51-76, August 1983.
- Thulasiraman, K. and Comeau, M. Maximum-weight marking in marked graphs: algorithms and interpretations based on the simplex method. *IEEE Transactions on Circuits and Systems*, 34(12), 1535-1545, December M1987.
- Valette, R. Analysis of Petri nets by stepwise refinements. *Journal of Computer and Systems Sciences*, vol. 18, pp. 35-46, 1979.
- Valette, R. Nets in production systems. In Brauer (1987b), pp. 191-217, 1987.
- Valette, R., Courvoisier, M., Bigou, J. M. and Alburquerque, J. A. A Petri net based programmable logic controller. *First International Conference on Computer Applications in Production and Engineering*, CAPE '83, Amsterdam, April 1983.
- Voss, K., Genrich, H. J. and Rozenberg, G. (eds). *Concurrency and Nets*. Springer-Verlag, Berlin, 1987.

Principles of system modeling

J. M. Proth

2.1 MANUFACTURING SYSTEM MODELING: BASIC CONCEPTS

A manufacturing system is composed of two main parts: the **physical system** and the **management system**. Hereafter, the latter is also referred to as the **control system** or the **decision-making system** (DMS).

The physical system is the set of resources which operate on the raw material and/or on the work-in-process (WIP). For instance, machines, cells, transportation systems (conveyors, automated guided vehicles (AGVs), cranes, etc.), workers, storage devices, ovens, loading-unloading stations, quality control stations, belong to the physical system.

The DMS allows one to take advantage of the indetermination (i.e. the degree of freedom) of the physical system to make it work in a way which optimizes some criteria like productivity (which has to be maximized), WIP level (which has to be reduced as much as possible), total tardiness (which has to be as close to zero as possible).

The DMS can be divided into two subsystems: the part of the DMS which computes the decisions starting only from the state of the physical system (hereafter referred to as PSS-DMS for Physical System State-based DMS) and the part of the DMS which computes the decision starting from the state of the environment of the physical system and possibly also from the state of the physical system itself (hereafter referred to as ES-DMS for Environment State-based DMS).

Let us give some examples of the DMS subsystems to clarify the previous terminology. Consider a storage resource (the physical system in our example) managed by the following rule: 'Order a quantity Q when the inventory level becomes less than q '. In that case, the DMS is restricted to the PSS-DMS because the decision is made starting from the state of the physical system, which is the inventory level.

Let us consider the same physical system obeying the following management rule: 'Order a quantity Q when the difference (inventory level minus demand) is less than q '. Now, we are in the case where the DMS