2

MODELS FOR CONCURRENT SYSTEMS

The best material model of a cat is another, or preferably the same, cat.
—Arturo Rosenblueth - Philosophy of Science, 1945

A theory has only the alternative of being right or wrong. A model has a third possibility: it may be right, but irrelevant.
—Manfred Eigen - The Physicist's Conception of Nature, 1973

Summary

This chapter introduces the fundamentals of the models used for the specification, synthesis and verification of systems in the subsequent chapters. In particular, *transition systems* and *Petri nets* are introduced as models for untimed systems. Other classes of transition systems, such as *timed transition systems* and *lazy transition systems* are introduced as the models used for timed systems.

Transition systems and Petri nets provide an abstract view of the events and states of a system, without considering any binary encoding. In some cases, such as for the logic synthesis of circuits, or simply in order to achieve efficient implementations of verification algorithms, such encoding is required. The encoding allows the symbolic manipulation of a system using compact representations and efficient algorithms based on BDDs, for example. Chapter 5 provides details on the binary encoding of transition systems.

2.1 Introduction

As it will be seen in Chapter 4, the proposed verification approach uses *timed transition systems* to model the timed systems under verification. The approach, however, does not manipulate the exact timed state space of the system. Instead, the untimed state space, modeled by a *transition system*, is used as the starting point of the verification approach. Then, an incremental refinement of the untimed state space with relative timing information is carried out, thus leading to the use of *lazy transitions systems*. This chapter presents the fundamental concepts and notation related to the different types of transition systems involved in the verification approach.

On the other hand, *Petri nets* and its interpretation as *signal transition graphs* are often used to model asynchronous digital circuits and other concurrent systems. As a consequence, a number of verification methods have been developed under such formalisms (see Section 3.6). Moreover, in Chapter 5 some illustrative examples of the verification of timed systems are presented, which are originally specified with *Petri nets* and *signal transition graphs*. For this reason, we have considered appropriate to introduce the fundamentals of such modeling formalisms at the end of this chapter.

2.2 Transition systems

Transition systems (TS) are a formalism used to describe systems of concurrent processes [Arn94]. The formalism, although mathematically simple, can model most of the properties of such systems, and so can be used to study their semantics. Several theoretical tools based on transition systems have been developed, including equivalence relations with other formalisms (formal languages, Petri nets, etc.). The usefulness of these theoretical tools is supported by the existence of a variety of software tools.

Intuitively, a *transition system* consists of the set of possible states of a system, and a set of transitions that the system can produce in order to change from one state to another. In comparison to event-based models such as Petri nets, transition systems offer a view of a system at a lower level of abstraction.

DEFINITION 2.1 (TRANSITION SYSTEM)

A transition system (TS) *[NRT92] is a quadruple* $A = \langle S, \Sigma, T, \mathsf{s}_0 \rangle$*, where* S *is a non-empty set of* states*,* Σ *is a non-empty alphabet of* events*,* $T \subseteq S \times \Sigma \times S$ *is a* transition relation*, and* s_0 *is the* initial state*.*

The elements of T *are called* transitions *and are indistinctly denoted by* $\mathsf{s} \xrightarrow{\mathsf{e}} \mathsf{s}'$ *or by* $(\mathsf{s}, \mathsf{e}, \mathsf{s}')$*.*

An event e *is* enabled *at state* s *if* $\exists\ \mathsf{s} \xrightarrow{\mathsf{e}} \mathsf{s}' \in T$*. We will denote by* $\mathcal{E}(\mathsf{s})$ *the set of events enabled at state* s*. The* firing region *of event* e *is defined as* $\mathsf{FR}(\mathsf{e}) = \{\mathsf{s} \in S \mid \mathsf{e} \in \mathcal{E}(\mathsf{s})\}$*,* i.e. *the set of states where* e *is enabled.*

■ 2.1

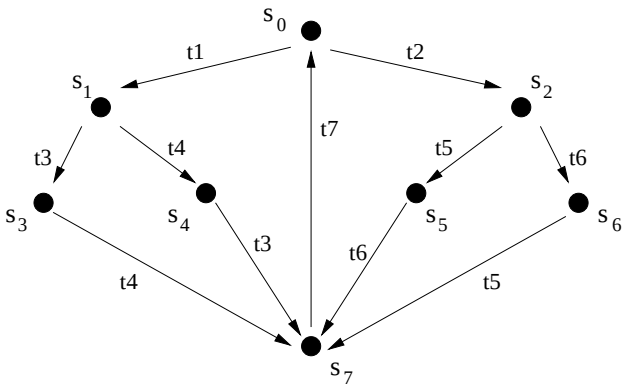


Figure 2.1 An example of transition system.

A TS is *finite* if S and Σ are finite. A TS is called *deterministic* if for each state s and each event e there is at most one state s' such that $\mathsf{s} \xrightarrow{\mathsf{e}} \mathsf{s}'$. In the sequel, only finite transition systems will be considered. Moreover, no multiple arcs should exist between any pair of states, *i.e.* $\mathsf{s} \xrightarrow{\mathsf{e}} \mathsf{s}' \in T \ \wedge \ \mathsf{s} \xrightarrow{\mathsf{e}'} \mathsf{s}' \in T \ \Rightarrow \ \mathsf{e} = \mathsf{e}'$.

EXAMPLE 2.1 *A TS can be represented by an arc-labeled directed graph. A simple example of a TS is shown in Figure 2.1. States are represented as dots, transitions are the directed arcs between the states, and the events are the labels of the transitions. That is,* $S = \{\mathsf{s}_0, \ldots, \mathsf{s}_7\}$, $\Sigma = \{\mathsf{t}_1, \ldots, \mathsf{t}_7\}$ *and* $T = \{\mathsf{s}_0 \xrightarrow{\mathsf{t}_1} \mathsf{s}_1, \mathsf{s}_1 \xrightarrow{\mathsf{t}_4} \mathsf{s}_4, \ldots\}$. *As an example, the firing region of event* t_5 *is* $\mathsf{FR}(\mathsf{t}_5) = \{\mathsf{s}_2, \mathsf{s}_6\}$.

■ 2.1

DEFINITION 2.2 (RUN)

A run *of a transition system* $A = \langle S, \Sigma, T, \mathsf{s}_0 \rangle$ *is a sequence of transitions* $\rho = \mathsf{s}_1 \xrightarrow{\mathsf{e}_1} \mathsf{s}_2 \xrightarrow{\mathsf{e}_2} \cdots$, *such that* $\mathsf{s}_1 = \mathsf{s}_0$ *and* $\forall i \geq 1 : \mathsf{s}_i \xrightarrow{\mathsf{e}_i} \mathsf{s}_{i+1} \in T$. *Event* e_i *is said to* fire *at step* i *of the run.*

■ 2.2

With an abuse of notation, the expressions $\mathsf{s}_i \in \rho$, $\mathsf{s}_i \xrightarrow{\mathsf{e}_i} \mathsf{s}_{i+1} \in \rho$, $\mathsf{s}_i \xrightarrow{\mathsf{e}_i} \in \rho$, $\xrightarrow{\mathsf{e}_i} \mathsf{s}_{i+1} \in \rho$, etc, will be often used to denote the fact that different fragments of a sequence belong to a run.

Several ordering relations between the events of a TS can be defined.

DEFINITION 2.3 (TRIGGERING, CONFLICT, CONCURRENCY)

Given a transition system $A = \langle S, \Sigma, T, \mathsf{s}_0 \rangle$ *and two events* $\mathsf{e}_1, \mathsf{e}_2 \in \Sigma$ *:*

(a) e_1 triggers e_2 *if the firing of* e_1 *enables* e_2, i.e. *if* $\exists \mathsf{s}_1 \xrightarrow{\mathsf{e}_1} \mathsf{s}_2 \in T$, *such that* $\mathsf{s}_1 \notin \mathsf{FR}(\mathsf{e}_2)$ *and* $\mathsf{s}_2 \in \mathsf{FR}(\mathsf{e}_2)$.

(b) e_1 disables e_2 *if* $\exists \mathsf{s}_1 \xrightarrow{\mathsf{e}_1} \mathsf{s}_2 \in T$, *such that* $\mathsf{s}_1 \in \mathsf{FR}(\mathsf{e}_2)$ *and* $\mathsf{s}_2 \notin \mathsf{FR}(\mathsf{e}_2)$. e_1 *and* e_2 *are in* conflict *if either* e_1 disables e_2 *or* e_2 disables e_1.

(c) e_1 *and* e_2 *are* concurrent *if there is a state in which both events are enabled and the firing of one of them does not disable the other:* $\exists\ \mathsf{s} \in \mathsf{FR}(\mathsf{e}_1) \cap \mathsf{FR}(\mathsf{e}_2)\ :\ \mathsf{s} \xrightarrow{\mathsf{e}_1} \mathsf{s}_1 \in T\ \wedge\ \mathsf{s} \xrightarrow{\mathsf{e}_2} \mathsf{s}_2 \in T\ \Rightarrow\ \exists\ \mathsf{s}_3 \in S\ :\ \mathsf{s}_1 \xrightarrow{\mathsf{e}_2} \mathsf{s}_3 \in T\ \wedge\ \mathsf{s}_2 \xrightarrow{\mathsf{e}_1} \mathsf{s}_3 \in T$.

■ 2.3

The following definition captures the notion of enabling of an event at a given state of a run, and the conditions for such event to keep enabled along the run until it fires or is disabled by the firing of another event.

DEFINITION 2.4 (ENABLING INTERVAL)

Let $A = \langle S, \Sigma, T, \mathsf{s}_0 \rangle$ *be a TS and let* $\rho = \mathsf{s}_1 \xrightarrow{\mathsf{e}_1} \mathsf{s}_2 \xrightarrow{\mathsf{e}_2} \cdots$ *be a run of* A. *Given an event* e *and a state* $\mathsf{s}_i \in \rho$ *such that* $\mathsf{s}_i \in \mathsf{FR}(\mathsf{e})$, $\mathsf{FirstEnabled}(\rho, \mathsf{s}_i, \mathsf{e})$ *is defined as the state* s_j, $j \leq i$, *such that:*

- $j \leq k \leq i \ \Rightarrow\ \mathsf{s}_k \in \mathsf{FR}(\mathsf{e})$ (e *is continuously enabled between* s_j *and* s_i)
- $j > 0 \ \Rightarrow\ \mathsf{s}_{j-1} \notin \mathsf{FR}(\mathsf{e})$ (e *is not enabled before* s_j)

The sequence $\mathsf{s}_j \xrightarrow{\mathsf{e}_j} \cdots \xrightarrow{\mathsf{e}_{i-1}} \mathsf{s}_i$ *is called the* enabling interval *of* e *with respect to* s_i.

■ 2.4

Notice that this definition imposes nothing about the necessity of event e to actually fire in ρ. Therefore, given $\mathsf{s}_j \xrightarrow{\mathsf{e}_j} \cdots\ \mathsf{s}_i \xrightarrow{\mathsf{e}_i} \mathsf{s}_{i+1} \in \rho$ and $\mathsf{FirstEnabled}(\rho, \mathsf{s}_i, \mathsf{e}) = \mathsf{s}_j$ such that $\mathsf{s}_{i+1} \notin \mathsf{FR}(\mathsf{e})$, e actually fires if $\mathsf{e} = \mathsf{e}_i$, whereas e is disabled by the firing of e_i otherwise. As an example, consider the run $\rho = \mathsf{s}_0 \xrightarrow{\mathsf{t}_2} \mathsf{s}_2 \xrightarrow{\mathsf{t}_6} \mathsf{s}_6 \xrightarrow{\mathsf{t}_5} \mathsf{s}_7 \cdots$ of the TS in Figure 2.1. In this case we have that $\mathsf{FirstEnabled}(\rho, \mathsf{s}_6, \mathsf{t}_5) = \mathsf{s}_2$.

The transitive closure of the transition relation T is called the *reachability relation* between states and is denoted by T^*. In other words, state s' is reachable from state s if there is a run between both, denoted by $\mathsf{s} \xrightarrow{\rho} \mathsf{s}'$, or simply $\mathsf{s} \xrightarrow{*} \mathsf{s}'$ if the run is not relevant. The set of states reachable through all possible runs of the system is given by the following definition:

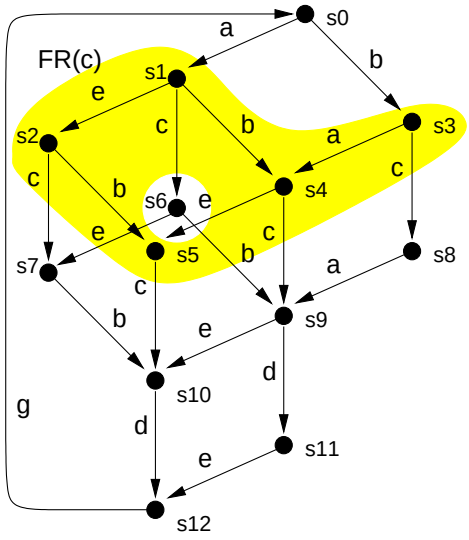
DEFINITION 2.5 (REACHABLE STATES)

Given a TS $A = \langle S, \Sigma, T, \mathsf{s}_0 \rangle$, *the set of reachable states from a state* s *is recursively defined as:*

$$\mathit{Reach}(\mathsf{s}, T) = \{\mathsf{s}\} \cup \bigcup_{\mathsf{s} \longrightarrow \mathsf{s}' \in T} \mathit{Reach}(\mathsf{s}', T)$$

■ 2.5

Henceforth, it is assumed that $S = \mathsf{Reach}(\mathsf{s}_0, T)$ for any TS.



(a)

δ(a) ∈ [1,2]
δ(b) ∈ [1.5,2]
δ(c) ∈ [3,4]
δ(d) ∈ [1,1.5]
δ(e) ∈ [1,1.5]

(b)

Figure 2.2 An example of timed transition system: (a) underlying TS, (b) associated delay intervals. The firing region of c is shadowed.

2.3 Timed transition systems

Timed transition systems (TTS) [HMP92a] allow to model systems in which timing information is particularly relevant for their operation, such as real-time systems. TTSs generalize the basic computational model of transition systems by associating minimum and maximum delays to the transitions. The real line is generally used as timed domain although integer and rational values are also used by some authors.

Verification methods for timed transition systems have been developed for various logical specification languages. The methods include both algorithmic techniques for finite-state systems [AH90, HLP90, Ost90] and deductive techniques based on proof systems [Hen90, Ost90, HMP91].

Time is incorporated to transition systems by assuming that transitions happen instantaneously, while minimum and maximum delay bounds restrict the times at which transitions may occur. The delay bounds ensure that transitions occur neither too early (*i.e.* never before the minimum delay bound) nor too late (*i.e.* never after the maximum delay bound). The absence of a lower delay bound requirement is modeled by a minimum delay bound of 0, whereas the absence of an upper delay bound requirement is modeled by a maximum delay bound of ∞. Formally:

DEFINITION 2.6 (TIMED TRANSITION SYSTEM)

A timed transition system (TTS) *[HMP92a] is a triple* $A = \langle A^-, \delta^l, \delta^u \rangle$, *where* $A^- = \langle S, \Sigma, T, \mathsf{s}_0 \rangle$ *is a* TS *called the* underlying transition system, $\delta^l : \Sigma \to \mathbb{R}^+$ *and* $\delta^u : \Sigma \to \mathbb{R}^+ \cup \{\infty\}$ *respectively associate a minimum and a maximum* delay bound *to each event, such that* $\forall\, \mathsf{e} \in \Sigma : \delta^l(\mathsf{e}) \leq \delta^u(\mathsf{e})$.

Min-max delay ranges are represented by $[d, D]$, *or by* $[d, \infty)$ *if the maximum delay is unbounded.* ■ 2.6

EXAMPLE 2.2 *Figure 2.2 depicts an example of* TTS *described by means of: the underlying* TS , *where the firing region of event* c *is shadowed, and the delays associated to the events of the system. If the delay of an event is omitted we assume it belongs to the less constraining interval* $[0, \infty)$. *This is the case of event* g, *for example.* ■ 2.2

The TS A^- captures the behavior of the system in the untimed domain, *i.e.* without considering the delay information. The behavior of the system in the timed domain is obtained by associating to each state a time stamp that indicates when the state was reached. Thus, the same state can be visited at different time instants depending on its prehistory, and the time stamps must comply with the delay bounds specified for each event (see Definition 2.7). Conversely, a state can never be reached in the timed domain if its incoming transitions are never fired due to their respective delay constraints. Time is always relative to that of the initial state of the system, and must monotonically increase along the firing sequences.

Because of the time dimension, the computation of the exact timed state space of a system has been proved to be a PSPACE-complete problem [AD90] and demonstrated to be a highly complex task in several contexts such as real-time systems [AD94, HMP91] and asynchronous circuits [DKMW92, Bur92, HB94, MP95, SY96, VdJL96].

The following definition characterizes which sequences of states of the system are actually feasible when the specified delay bounds are taken into consideration.

DEFINITION 2.7 (TIMING-CONSISTENT RUN)

Let $A = \langle A^-, \delta^l, \delta^u \rangle$ *be a* TTS *and let* $\rho = \mathsf{s}_1 \xrightarrow{\mathsf{e}_1} \mathsf{s}_2 \xrightarrow{\mathsf{e}_2} \cdots$ *be a run of* A^-. ρ *is* timing-consistent *with* A *if a sequence* $\tau_1 \leq \tau_2 \leq \cdots$ *of monotonically increasing real-valued time stamps can be assigned to the states in* ρ *such that:*
$\forall \mathsf{s}_i \xrightarrow{\mathsf{e}_i} \mathsf{s}_{i+1} \in \rho$ *and* $\forall \mathsf{e} \in \mathcal{E}(\mathsf{s}_i)$ *such that* $\mathsf{FirstEnabled}(\rho, \mathsf{s}_i, \mathsf{e}) = \mathsf{s}_j$:

- $\mathsf{e} = \mathsf{e}_i \;\Rightarrow\; \delta^l(\mathsf{e}) \leq \tau_{i+1} - \tau_j$ i.e. *an event cannot fire before its minimum delay has elapsed, and*
- $\tau_{i+1} - \tau_j \leq \delta^u(\mathsf{e})$ i.e. *an event cannot remain enabled after its maximum delay has elapsed.* ■ 2.7

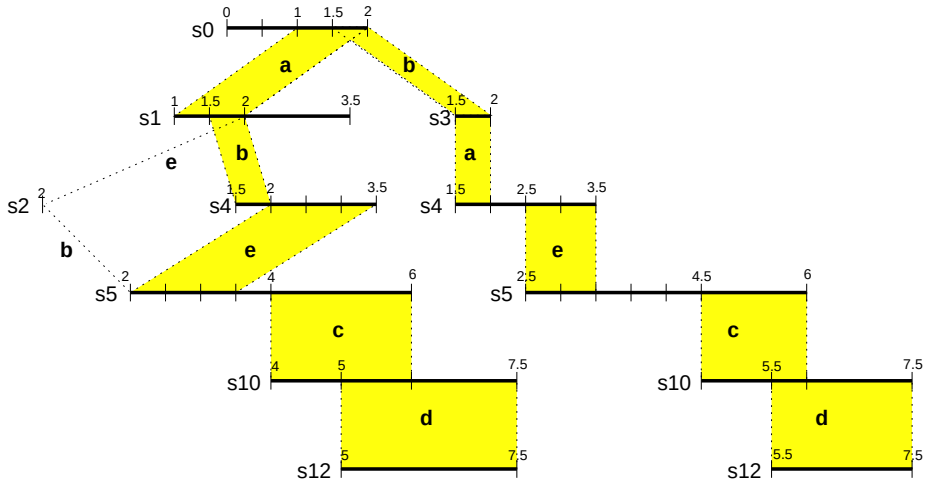


Figure 2.3 Portion of the timed state space of the TTS of Figure 2.2.

The previous definition characterizes those runs which are possible according to the delay bounds associated to the events of the system. The time stamp τ_{i+1} is assigned to state s_{i+1} and corresponds to the *firing time* of event e_i along ρ. Similarly, τ_j corresponds to the *enabling time*. Thus, the firing time of an event only depends on its enabling time plus certain delay amount within the given bounds. Moreover, the disabling of an event must be produced before its maximum delay has elapsed, otherwise it should have fired already.

A timing-consistent run can be represented as a sequence of transitions of the form: $(\mathsf{s}_1, \tau_1) \xrightarrow{\mathsf{e}_1} (\mathsf{s}_2, \tau_2) \xrightarrow{\mathsf{e}_2} \cdots$. Where for all $i > 0$: each pair (s_i, τ_i) is called a *timed state* with $\mathsf{s}_i \in S$ and $\tau_i \in \mathbb{R}^+$; and $\mathsf{s}_i \xrightarrow{\mathsf{e}_i} \mathsf{s}_{i+1} \in T$. Also $\tau_1 = 0$ and $\lim_{i\to\infty} \tau_i = \infty$. We will use this notation to represent the timed runs of a TTS when it eases the reading.

We say that a run is *Zeno* if it contains an infinite number of transitions in a finite amount of time [AH97], *i.e.* $\sum_{1\leq i<\infty}(\tau_{i+1} - \tau_i) < \infty$. For example, that could be the case produced by a cycle of events with 0 delay. Without loss of generality, in the sequel we will only consider systems modeled by *non-zenoess* TTSs, that is they can not produce Zeno runs.

Example 2.2 (cont.) *Figure 2.3 depicts a portion of the timed state space of the TTS of Figure 2.2. The horizontal axis represents the time intervals in which the system remains at each state, assuming that time begins at state* s_0*. Notice that some states can be reached, and also left, at different time instants depending on their prehistory.*

To illustrate the behavior of the system in the timed domain assume the execution starts at the timed state $(\mathsf{s}_0, 0)$*. The system remains at such state until the minimum delay of* a *elapses. Then the system can either fire* a *or wait until* b *becomes firable* 0.5 *time*

units later. Thus, a *can fire between time* $0+\delta^l(\mathsf{a})=1$ *and time* $0+\delta^u(\mathsf{a})=2$*, whereas* b *can fire between time* $0+\delta^l(\mathsf{b})=1.5$ *and time* $0+\delta^u(\mathsf{b})=2$*. Assume* a *fires at time* 1.5 *leading to the timed state* $(\mathsf{s}_1, 1.5)$*. Now* b *is still enabled and can fire, while* e *and* c *become newly enabled.* e *cannot fire until time* $1.5+\delta^l(\mathsf{e})=2.5$*, whereas* c *is much slower and cannot fire until* $1.5+\delta^l(\mathsf{c})=4.5$*. Therefore, if for example* b *fires at time* 1.5 *leading to the timed state* $(\mathsf{s}_4, 1.5)$*, it will be followed by* e *and* c*, leading for example to the timed states* $(\mathsf{s}_5, 2.5)$ *and* $(\mathsf{s}_{10}, 4.5)$ *successively. The execution continues by firing* d*, and so on.*

None of the states s_5*,* s_6*,* s_7*,* s_8*,* s_9 *or* s_{11} *is reachable in the portion of the timed domain shown in the figure. For example, state* s_8 *can not be reached from* s_3 *by firing* c*, since* c *is slower than* a*, even in the case that both events were enabled at the same instant. In fact, only the following three runs are potentially timing-consistent if appropriate time stamps are assigned to each state:* $\mathsf{s}_0 \xrightarrow{\mathsf{a}} \mathsf{s}_1 \xrightarrow{\mathsf{e}} \mathsf{s}_2 \xrightarrow{\mathsf{b}} \mathsf{s}_5 \xrightarrow{\mathsf{c}} \mathsf{s}_{10} \xrightarrow{\mathsf{d}} \mathsf{s}_{12} \cdots$*,* $\mathsf{s}_0 \xrightarrow{\mathsf{a}} \mathsf{s}_1 \xrightarrow{\mathsf{b}} \mathsf{s}_4 \xrightarrow{\mathsf{e}} \mathsf{s}_5 \xrightarrow{\mathsf{c}} \mathsf{s}_{10} \xrightarrow{\mathsf{d}} \mathsf{s}_{12} \cdots$ *and* $\mathsf{s}_0 \xrightarrow{\mathsf{b}} \mathsf{s}_3 \xrightarrow{\mathsf{a}} \mathsf{s}_4 \xrightarrow{\mathsf{e}} \mathsf{s}_5 \xrightarrow{\mathsf{c}} \mathsf{s}_{10} \xrightarrow{\mathsf{d}} \mathsf{s}_{12} \cdots$*. For example, the following timed run is timing-consistent:*

$$(\mathsf{s}_0, 0) \xrightarrow{\mathsf{a}} (\mathsf{s}_1, 1) \xrightarrow{\mathsf{b}} (\mathsf{s}_4, 1.5) \xrightarrow{\mathsf{e}} (\mathsf{s}_5, 2) \xrightarrow{\mathsf{c}} (\mathsf{s}_{10}, 4) \xrightarrow{\mathsf{d}} (\mathsf{s}_{12}, 5) \cdots$$

Conversely, the following timed run is not timing-consistent:

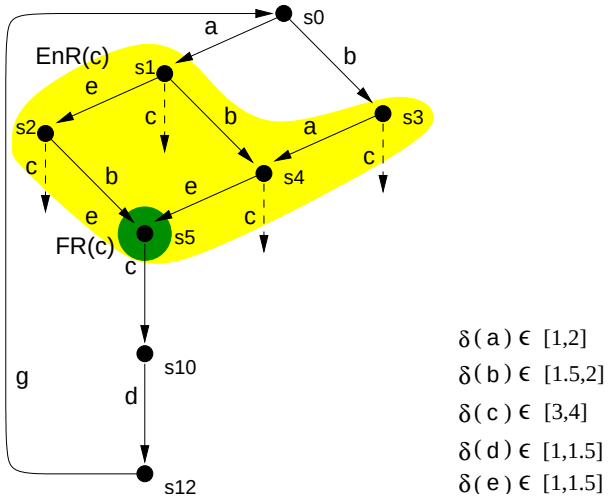
$$(\mathsf{s}_0, 0) \xrightarrow{\mathsf{a}} (\mathsf{s}_1, 1) \xrightarrow{\mathsf{b}} (\mathsf{s}_4, 2) \xrightarrow{\mathsf{e}} (\mathsf{s}_5, 3.5) \xrightarrow{\mathsf{c}} (\mathsf{s}_{10}, 5) \xrightarrow{\mathsf{d}} (\mathsf{s}_{12}, 5) \cdots$$

since the firing of e *happens at time* 3.5*, but according to its delays and the firing of its trigger* a *at time* 1*,* e *must fire between time* 2 *and time* 2.5. ■ 2.2

To conclude this section, we want to remark that different notions of timed transition systems with timing constraints for the discrete-time paradigm have been also proposed by several authors [PH88, Ost90, HMP91]. Similarly, *clocked transition systems* were defined in [MP96] as an alternative model for real-time systems, where time is explicitly represented by means of a set of timers (often called clocks) which increase uniformly whenever time progresses, and can be set to specific values by the firing of transitions. This type of transition systems is inspired by a more commonly used model called *time automata* which is analyzed in detail in Chapter 3.

2.4 Lazy transition systems

When time becomes an essential magnitude in a model for concurrent systems, a complexity dimension is added to the derived analysis and verification methods. In those formalisms and analysis mechanisms where time is an explicit magnitude, such complexity is specially noticeable when computing the timed state space of the system. For example, the problem of reachability in *timed automata* is proved to be PSPACE-hard [AD94], where the exponentiality depends on the number of clocks and on the encoding of the maximum values that can be taken by the clocks (see Section 3.3). This complexity makes the analysis of systems with a moderate amount of untimed states almost impractical.



(a) (b)

Figure 2.4 An example of lazy transition system: (a) LzTS corresponding to the TTS in Figure 2.2, (b) delays associated to the events.

Several approaches have been devised to represent timed states in a succinct form, *e.g.* [BM00]. However, the incorporation of the timed domain in the representation of the states hampers an efficient representation of large state spaces with BDDs [Bry86]. Even the discretization of time [HMP91] poses serious problems when the number of clocks or the constants of the timing constraints are large. An interesting approach to face this complexity problem was proposed in [AK95], where the clocks used during the analysis of the system and their accuracy, are determined dynamically upon demand. In this way, only that timing information relevant to the analysis emerges during the calculation of the reachable states.

In some cases, however, rather than calculating the exact time intervals in which each state can be visited by any valid run of the system, the only information required is to know whether each state is visited by some timing-consistent run and what are the enabling conditions for every visited state. In other words, only the set of reachable states in the timed domain and the transition relations for every event are required. This information can be represented by abstracting absolute time information out of the model. The abstraction leads to the definition of a new computational model called *lazy transition systems* [CKK+98], in which timing information is represented in terms of the notion of *laziness*. This notion explicitly distinguishes among the enabling and the firing of an event, assuming certain implicit delay between them. Formally:

DEFINITION 2.8 (LAZY TRANSITION SYSTEM)
A lazy transition system (LzTS) *[CKK+98] is a five-tuple* $A = \langle S, \Sigma, T, \mathsf{s}_0, EnR \rangle$, *where* $\langle S, \Sigma, T, \mathsf{s}_0 \rangle$ *is a TS, and the function* $\mathsf{EnR} : \Sigma \rightarrow 2^S$ *defines the* enabling region *of each event.*

■ 2.8

Event e is said to be *enabled* at state $\mathsf{s} \in S$ if $\mathsf{s} \in \mathsf{EnR(e)}$. Similarly, event e is said to be *firable* at state $\mathsf{s} \in S$ if $\mathsf{s} \in \mathsf{FR(e)}$. For each event $\mathsf{e} \in \Sigma$ the condition $\mathsf{FR(e)} \subseteq \mathsf{EnR(e)}$ must hold. Event e is said to be *lazy* if $\mathsf{FR(e)} \neq \mathsf{EnR(e)}$. Therefore, any state of a LzTS in the set $\mathsf{EnR(e)} \setminus \mathsf{FR(e)}$ is a state in which the event e is enabled but cannot fire due to the delays associated to the events of the system.

EXAMPLE 2.2 (CONT.) *Figure 2.4 shows the lazy transition system corresponding to the timed transition system in Figure 2.2. Analyzing the delays, it can be proved that event* c *is always slower to fire than events* a, b *and* e. *Therefore,* c *becomes lazy in those states where it is concurrently enabled with those faster events. Thus, although* c *is enabled in states* $\mathsf{EnR(c)} = \{\mathsf{s}_1, \mathsf{s}_2, \mathsf{s}_3, \mathsf{s}_4, \mathsf{s}_5\}$, *it can only fire in* $\mathsf{FR(c)} = \{\mathsf{s}_5\}$, *once* a, b *and* e *have already fired.* ■ 2.2

Notice that a TS is just a particular case of LzTS in which both enabling and firing regions coincide for all the events. Thus, the notion of enabling interval (see Definition 2.4) is naturally extended to lazy transition systems by considering $\mathsf{EnR(e)}$ instead of $\mathsf{FR(e)}$ for the enabledness of event e. Similarly, the ordering relations defined between the events of a TS (see Definition 2.3) can be extended for LzTSs as follows:

DEFINITION 2.9 (TRIGGERING, CONFLICT, CONCURRENCY)
Given a lazy transition system $A = \langle S, \Sigma, T, \mathsf{s}_0, \mathsf{EnR} \rangle$ *and two events* $\mathsf{e}_1, \mathsf{e}_2 \in \Sigma$ *:*

(a) e_1 triggers e_2 *if the firing of* e_1 *enables* e_2, i.e. *if* $\exists \mathsf{s}_1 \xrightarrow{\mathsf{e}_1} \mathsf{s}_2 \in T$, *such that* $\mathsf{s}_1 \notin \mathsf{EnR}(\mathsf{e}_2)$ *and* $\mathsf{s}_2 \in \mathsf{EnR}(\mathsf{e}_2)$.

(b) e_1 disables e_2 *if* $\exists \mathsf{s}_1 \xrightarrow{\mathsf{e}_1} \mathsf{s}_2 \in T$, *such that* $\mathsf{s}_1 \in \mathsf{EnR}(\mathsf{e}_2)$ *and* $\mathsf{s}_2 \notin \mathsf{EnR}(\mathsf{e}_2)$. e_1 *and* e_2 *are in* conflict *if* e_1 disables e_2 *or* e_2 disables e_1.

(c) e_1 *and* e_2 *are* concurrent *if* $\mathsf{EnR}(\mathsf{e}_1) \cap \mathsf{EnR}(\mathsf{e}_2) \neq \emptyset$ *and they are not in conflict, and* $\exists\, \mathsf{s} \in \mathsf{FR}(\mathsf{e}_1) \cap \mathsf{FR}(\mathsf{e}_2) \;:\; \mathsf{s} \xrightarrow{\mathsf{e}_1} \mathsf{s}_1 \in T \;\wedge\; \mathsf{s} \xrightarrow{\mathsf{e}_2} \mathsf{s}_2 \in T \;\Rightarrow\; \exists\, \mathsf{s}_3 \in S \;:\; \mathsf{s}_1 \xrightarrow{\mathsf{e}_1} \mathsf{s}_3 \in T \;\wedge\; \mathsf{s}_2 \xrightarrow{\mathsf{e}_2} \mathsf{s}_3 \in T$.

■ 2.9

Notice that the second condition for concurrency is analogue of the non-conflict requirement, but applies to the FR rather than to the EnR.

The use of LzTSs enables to reason in terms of partial orders of events, *i.e.* in terms of the so-called *relative timing* paradigm [SGR99], which is much more intuitive than

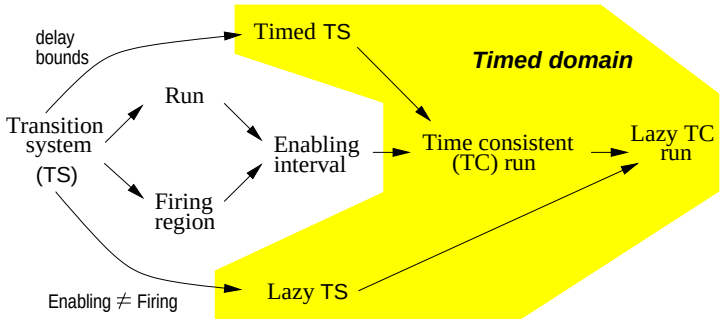


Figure 2.5 Relations among the main notions related to transition systems.

defining absolute time separations between pairs of events (see also Chapter 3). Moreover, whereas absolute timing information requires complex techniques to represent the space of reachable timed regions or states [Alu98] (*e.g.* using difference bound matrices, time polyhedra, etc.), the generation of the reachable state space for relative timing is of the same complexity as for untimed systems. Thus Definition 2.5 can be extended to LzTSs in a straightforward manner.

Figure 2.5 depicts the relationships among the major notions around tansition systems, and their timed and lazy counterparts.

2.5 Petri nets

Petri nets (PNs) were initially proposed in [Pet62] as a graphical and mathematical formalism for describing information processing systems, characterized as being concurrent, asynchronous, distributed, parallel, non deterministic and/or stochastic. Since their introduction, Petri nets have been used in a wide range of areas such as communication networks, computer architecture, distributed systems, manufacturing, digital circuit synthesis and verification, etc.

Even though Petri nets constitute a powerful model, they consist of a few objects, relations and behavior rules. Namely, a Petri net consists of:

- The *net structure*, a bipartite directed graph containing *places*, *transitions* and *arcs*, that represent the static nature of the model.
- The *net marking*, which represents a distributed state of the model, in which a place can be marked by several *tokens*.
- The *execution rules*, which represent the dynamic evolution of the state of the model, *i.e.* describe how the tokens evolve through the places and transitions.

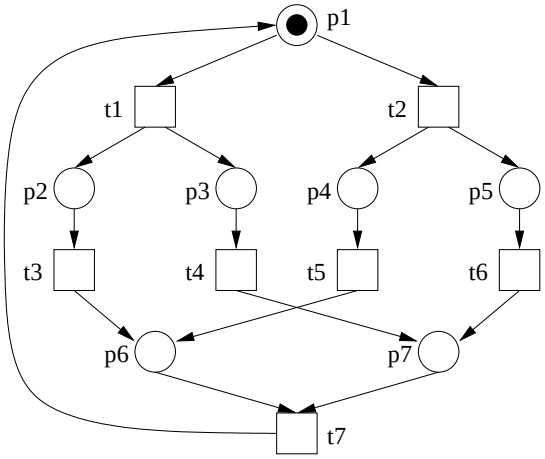


Figure 2.6 An example of Petri net.

Places can be seen as the state variables of the model. Transitions can be seen as the events that transform the state of the model. Arcs determine which are the necessary conditions for an event to occur and the values of the variables once an event has occurred.

A PN may also have additional information in its structure or in the markings, leading to different classes of nets, such as High-level Petri nets [JR91] or Coloured Petri nets [Jen92]. In the sequel, we restrict ourselves to the use of Place/Transition Petri nets, in which the tokens do not carry any particular information. For a good introduction to the different classes of Petri nets, their properties and usage, the reader is referred to [Pet81], [Rei85] and [Mur89], for example.

The remaining of this section introduces the notions related to Petri nets necessary for the work presented in the subsequent chapters.

DEFINITION 2.10 (PETRI NET)

A Petri net (PN) *is a quadruple* $N = \langle P, T, F, M_o \rangle$*, where* P *is a finite set of* places*,* T *is a finite set of* transitions*,* $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ *is the* flow function*, and* $M_o : P \to \mathbb{N}$ *is the initial* marking *(state) of the Petri net.*

■ 2.10

If the flow function is a relation on $P \cup T$, *i.e.* a mapping $F : (P \times T) \cup (T \times P) \to \{0, 1\}$, then the PN is called *ordinary*. In the sequel we will assume all the PNs to be ordinary, and will often talk about the flow relation F instead of the flow function.

The *pre-set* and *post-set* of a node $x \in P \cup T$ are denoted by ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (x, y) \in F\}$, respectively. Informally, the pre-set of a transition (place) corresponds to its input places (transitions), whereas its post-set corresponds to its output places (transitions).

When Petri nets are graphically represented, places are drawn as circles, transitions are drawn as boxes (or bars), the flow relation is represented by directed arcs, and tokens appear as dots circumscribed into the places.

If restrictions are imposed on the structure of the net, several subclasses of PNs can be defined [Mur89]: *state machines*, in which each transition has exactly one input place and one output place; *marked graphs*, in which each place has exactly one input transition and one output transition; *free-choice Petri nets*, in which if $(p,t) \in F$ then ${}^\bullet t \times p^\bullet \subseteq F$ for every place p; etc.

EXAMPLE 2.3 *Figure 2.6 depicts a Petri net, consisting of seven places,* $P = \{p_1, \ldots, p_7\}$ *and seven transitions,* $T = \{t_1, \ldots, t_7\}$. *The initial marking is indicated by the token in place* p_1. *When a place has only one predecessor and only one successor transition, it is often replaced by a simple arc between both transitions. According to this, in the PN of Figure 2.6, places* p_2, p_3, p_4 *and* p_5 *could be omitted.*

This Petri net has only one choice place, p_1, *which has two successor transitions,* t_1 *and* t_2. *Since* p_1 *is the only predecessor of* t_1 *and* t_2, p_1 *is a* free-choice place *and so it is the Petri net.* ■ 2.3

The structure of a Petri net defines the rules that determine its dynamic behavior. That is, what are the conditions that make a transition to become enabled and what happens when such transition actually fires.

DEFINITION 2.11 (ENABLING AND FIRING)

Given a Petri net $N = \langle P, T, F, M_o \rangle$:

(a) *A* marking *is a function* $M : P \to \mathbb{N}$, *which can be represented by a vector in* $\mathbb{N}^{|P|}$, *such that a place* p *is said to be* marked *at* M *by* $M(p)$ *tokens if* $M(p) > 0$.

(b) *A transition* $t \in T$ *is* enabled *in marking* M, *denoted by* $M[t\rangle$, *if all places in* ${}^\bullet t$ *are marked by at least one token. That is* $\forall p \in {}^\bullet t \ , \ M(p) > 0$. *We denote by* $[t\rangle$ *the set of all markings where transition* t *is enabled.*

(c) *When a transition* $t \in T$ *is enabled in marking* M, *it can* fire *reaching a new marking* M', *denoted by* $M[t\rangle M'$, *by removing a token from each place in* ${}^\bullet t$ *and adding a token to each place in* $t^\bullet$. *Formally:*

$$\forall p \in P \ , \ M'(p) = \begin{cases} M(p) - 1 & \text{if } \ p \in {}^\bullet t \setminus t^\bullet \\ M(p) + 1 & \text{if } \ p \in t^\bullet \setminus {}^\bullet t \\ M(p) & \text{otherwise} \end{cases}$$

■ 2.11

With the rules provided by the previous definition, the complete state space determined by the PN can be defined as follows:

DEFINITION 2.12 (REACHABILITY)
Given a Petri net $N = \langle P, T, F, M_o \rangle$*:*

(a) *Marking* M' *is* reachable *from marking* M *if there is a* firing sequence *of transitions* $\sigma = t_1 t_2 \ldots \in T^*$ *that transforms* M *into* M', i.e. $M[\sigma\rangle M'$. *The* reachability set *of* N*, denoted by* $[M_o\rangle$*, contains all the markings reachable from* M_o.

(b) *The* reachability graph (RG) *of* N *contains all the reachable markings and all the possible firing sequences of* N*. It is a directed graph* $RG = \langle [M_o\rangle, E\rangle$ *where* $E \subseteq [M_o\rangle \times T \times [M_o\rangle$ *is the set of arcs such that* $(M, t, M') \in E \Leftrightarrow M[t\rangle M'$*. A formal definition of RG in terms of transition systems is given in Section 2.2.*

■ 2.12

EXAMPLE 2.3 (CONT.) *Figure 2.7 (a) depicts the reachability graph corresponding to the Petri net of Figure 2.6, consisting of eight markings. For example, marking* $M_1 = \{p_2, p_3\}$ *indicates that places* p_2 *and* p_3 *contain one token each, whereas the rest of the places are empty.*

■ 2.3

In marking M_1 of the previous example, transitions t_3 and t_4 are both enabled simultaneously and can fire in any order without disabling each other. Thus, it can be expected that they can potentially fire together at the same instant, as it is shown by the arc between M_1 and M_7 if Figure 2.7 (b). The semantics of PNs that allow this type of *step-transitions* is called a *true concurrency* semantics, see the RG in Figure 2.7 (b). Conversely, the semantics that only allows a single transition to fire at a time is called *interleaving* semantics, see the RG in Figure 2.7 (a). Although true concurrency is more general than interleaving, it is often much more difficult to deal with it. In the sequel, only interleaving semantics is considered since it is general enough for our purposes. In fact, Definition 2.12 for reachability of PNs is already built upon the interleaving semantics.

Transition systems and Petri nets are linked through the notion of reachability graph. In fact the reachability graph of a PN is a transition system, Thus, given a PN, $N = \langle P, T, F, M_o \rangle$, its reachability graph is a transition system, $RG(N) = \langle S, \Sigma, T, \mathsf{s}_0 \rangle$, in which the set of states of the TS corresponds to the reachability set of the PN ($S = [M_o\rangle$), the events of the TS correspond to the transitions of the PN, and a transition (M_1, t, M_2) exists in the TS if and only if $M_1[t\rangle M_2$ in the PN. For example, it can be easily seen that the TS depicted in Figure 2.1 is isomorphic to the reachability graph depicted in Figure 2.7 (a) derived from the PN of Figure 2.6.

Even though the construction of a TS equivalent to a PN is a straightforward task, the opposite is not that simple in general. The interested reader is referred to [NRT92] and [CKLY98] for more details on the problem.

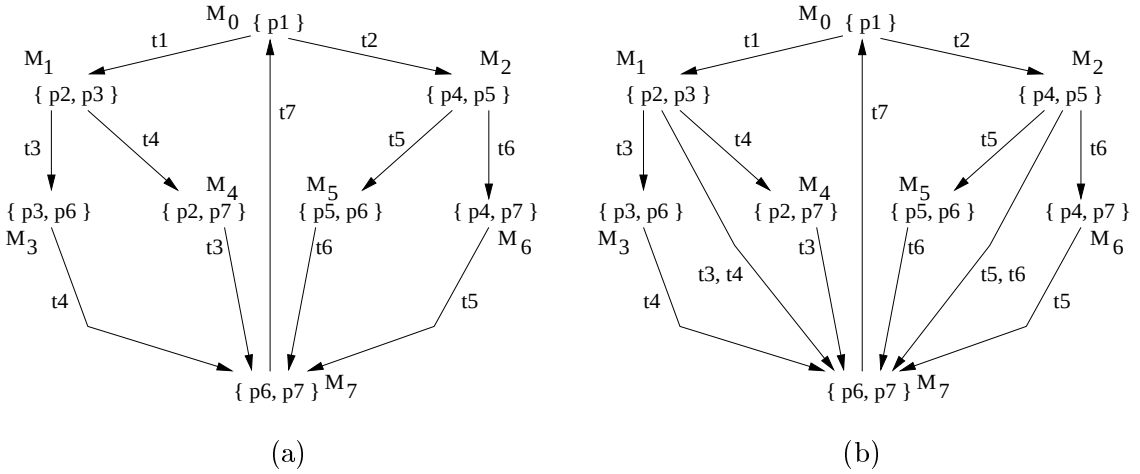


Figure 2.7 Reachability graph of the Petri net of Figure 2.6: (a) assuming interleaving semantics, and (b) assuming true concurrency.

A PN is called *live* iff every transition can be infinitely enabled through some feasible sequence of firings from any marking in $[M_o\rangle$. A PN is called *safe* if no marking in $[M_o\rangle$ assigns more than one token to any place. Safe PNs are widely used in many applications since they have simple analysis algorithms and simple semantics [EN94]. Without loss of generality for our pusposes, in the sequel we assume all the PNs to be safe. For example, the Petri net of Figure 2.6 is safe.

A static characterization of the dynamic behavior of the PN was defined by [Chu87] in terms of the so-called *temporal relations*. The relations define which pairs of transitions are causally related or are concurrent. Conflict relations are also defined, *i.e.* if the firing of one transition prevents the firing of another transition which was already enabled. These relations are often defined using the reachability graph of the PN, however they can be computed efficiently from the structure of the net [KE96]. Since the reachability graph of a PN is actually a transition system and these relations are also naturally defined for them, we omit the definition of the temporal relation at this point and refer the reader to Definition 2.3.

2.5.1 Labeled Petri nets

A *labeled* Petri net is a PN augmented with a labeling function which puts in correspondence every transition of the net with a *symbol* (called label) of an *alphabet*. If no two transitions have the same label (unique labeling), then each transition in the net can be uniquely identified by its label. Formally:

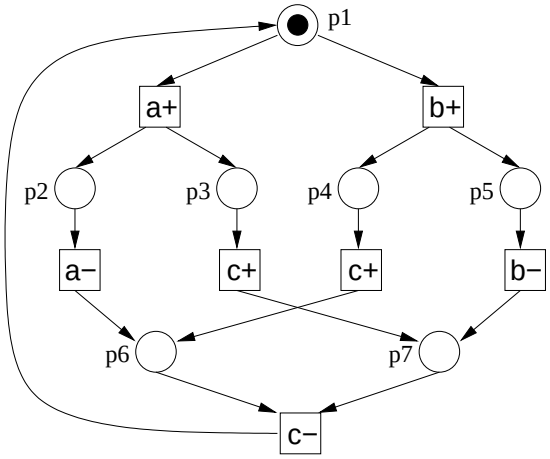


Figure 2.8 Petri net of Figure 2.6 with labeled transitions.

DEFINITION 2.13 (LABELED PETRI NET)

A labeled Petri net *is a triple* $LPN = \langle N, \Sigma, \Lambda \rangle$*, where* N *is a Petri net,* Σ *is a finite* alphabet*, and* $\Lambda : T \to \Sigma \cup \{\epsilon\}$ *is a* labeling function*. The special symbol* ϵ *is used to label those transitions without a particular meaning. Such "*silent*" transitions are often called* dummy *or* sequencing *transitions.*

■ 2.13

Figure 2.8 depicts a labeled PN with $\Sigma = \{\mathsf{a}+, \mathsf{a}-, \mathsf{b}+, \mathsf{b}-, \mathsf{c}+, \mathsf{c}-\}$.

Signal Transition Graphs (STGs), which are a particular class of PNs, are used in Chapter 5 to model asynchronous circuits. STGs were introduced independently in [RY85] and [Chu87] as a formalism for modeling the behavior of asynchronous circuits and their environment. In short, an STG is a labeled Petri net interpreted such that transitions describe value changes at the signals of a circuit. A signal transition can be represented by $\mathsf{a}+$ (or $\mathsf{a}-$) for the transition of signal a from 0 to 1 (or from 1 to 0), while $\mathsf{a}*$ is a generic name for either a rising or a falling transition of signal a. Formally:

DEFINITION 2.14 (SIGNAL TRANSITION GRAPH)

A signal transition graph (STG) *is a labeled Petri net* $LPN = \langle N, \Sigma, \Lambda \rangle$*, where* $\Sigma = \Sigma_I \cup \Sigma_O \cup \Sigma_H$ *is a set of* signal *names formed by the union of three non-intersecting subsets of input, output and internal signals, and* $\Lambda : T \to \Sigma \times \{+, -\}$ *is the* labeling function*.*

■ 2.14

The PN in Figure 2.8 can be interpreted as an STG that specifies the behavior of a circuit with, for example, two input signals ($\Sigma_I = \{\mathsf{a}, \mathsf{b}\}$) and an output signal ($\Sigma_O = \{\mathsf{c}\}$).

2.6 Conclusions

This chapter has presented the fundamentals of the formal models for concurrent systems that will be used in the subsequent chapters.

Transition systems (TS) are introduced as a state-based formalism for untimed systems. A TS is a mathematically simple formalism, however it allows to deal with a wide variety of other formalisms and to reason about them using a common formalism. As an example, the relation between PNs and TSs has been outlined. Fundamental notions for our later developments, such as the notion of enabling interval are presented.

Time can be incorporated on top of TSs in different forms. Timed transition systems (TTS) constitute the most common alternative, which associate time intervals to the events of the system. Therefore time is incorporated as an explicit real-valued exact magnitude. This fact causes that the state explosion problem becomes even less tractable than in the case of TSs. Another fundamental notion, that of timing-consistent run, is presented.

Conversely, lazy transition systems (LzTS) incorporate time using the relative timing paradigm, thus abstracting exact timing away and dealing only with partial orders of the events in the timed domain. LzTSs are the model underlying the development of the formal verification algorithms for timed systems in Chapter 4 and the related.

Petri nets provide a graphical and mathematical formalism for describing concurrent systems in a very intuitive way at the level of events. Apart from the net structure and its graphical representation, some basic properties have been briefly introduced. We have chosen PNs as the model for systems in which time is not a relevant magnitude. In particular, STGs will be used in Chapter 5 to model speed-independent asynchronous circuits.