

# Proving properties of real-time systems through logical specifications and Petri net models<sup>1</sup>

Miguel Felder, Dino Mandrioli, Angelo Morzenti  
Dipartimento di Elettronica, Politecnico di Milano, Italy

## Abstract

The problem of formally analyzing properties of real-time systems is addressed. A method is proposed that allows specifying system properties in the TRIO language (an extension of temporal logic suitable to deal explicitly with the “time” variable and to measure it) and modeling the system as a timed Petri net. It is argued that such an approach is more general than analyzing program properties. The proof method is based on an axiomatization of timed Petri nets in terms of TRIO so that their properties can be derived as suitable theorems in much the same spirit as classical Hoare’s method allows proving properties of programs coded in a Pascal-like language. The method is then exemplified through two classical “benchmarks” of the literature on concurrent and real-time systems, namely an elevator system and the dining philosophers problem. A thorough review of the related literature and a comparison thereof with the new method is also provided. Possible alternative methods, theoretical extensions, and practical applications are briefly discussed.

## 1. Introduction

In the field of sequential programming there are now several well understood methods suitable to prove program properties that are expressed through some—possibly formal—specification language. A classical example is Hoare’s method, which aims at proving properties of Pascal-like programs stated in terms of a first-order theory. Although the practical application of such methods to real-life cases is still under debate, these are now well-established and are receiving increasing consensus even in the industrial world, at least for the analysis of the most critical parts of the most critical systems [56, 38].

The state of the art is less well-established in the case of the analysis of concurrent systems. In fact such systems are intrinsically more difficult to analyze, what turns out into more complex formalization of their semantics and less satisfactory and less adopted specification languages. The situation is even worse for real-time systems. By “real-time systems” here we mean those systems whose behavior does depend on execution speed, not systems with generic requirements for high performance [95, 1]. In such systems one more difficulty arises from the necessity of modeling explicitly the dependence of system behavior on the time variable, whereas this dependency is usually abstracted away in the modeling of computing systems. On the other hand, real-time systems—which include plant control systems, embedded applications, air traffic control systems, etc.—have quite often

---

«DATA articolo.bibdata»«INCLUDE articolo.refs»

<sup>1</sup> Work partially supported by Piano Finalizzato Sistemi Informatici e Calcolo Parallelo (CNR) and partially supported by ESPRIT project IPTES

highest reliability requirements so that they could strongly benefit from rigorous analysis of their properties.

Rather recently, a number of models has been developed for a formal description and analysis of real-time systems. Among them, we focus here our attention on Petri nets [79, 84] and temporal logic [81]. Both formalisms, in their original formulation, have been proved quite effective for the analysis of concurrent systems but do not deal explicitly and quantitatively with time, what makes them unsuitable to model and specify strict real-time systems. It has been possible, however, to extend them in such a way to overcome this—and other—drawbacks. For instance, Petri nets have been augmented in several ways to allow the description of time dependent phenomena [66] [98] [48] [35]; TRIO [71, 37] and [54, 55, 76] are examples of extensions of pure temporal logic towards the same direction.

As far as it concerns system property analysis, its power and difficulty highly depend on the formalism that is chosen as system model. For instance, finite state machines are quite simple to use as a model, their properties are easy to prove, in general, but they often result far too simple to adequately describe complex systems in detail. Petri nets are certainly deeper to model concurrent systems but most of their properties—say reachability, liveness—are of intractable computational complexity or even undecidable. Things are even worse when the formalism is extended to cope with time or in other ways (in most cases extended Petri nets reach the computational power of Turing machines.) In general, the present methods—whether algorithmic or not—to perform system analysis on the basis of Petri net models are not yet quite satisfactory: either they are ad hoc methods to analyze a specific property in terms of a specialized model (for instance [10, 11] provide an algorithm to build the reachability graph of Timed Petri nets), or they are computationally intractable, or, when the property to be analyzed is undecidable, there is no systematic method to drive the human derivation of the proof.

Recently, a few proposals appeared in the literature suggesting proof methods for concurrent systems that are modelled in terms of Petri nets or other operational formalisms. [91] specifies the properties of a system modeled by a bounded Petri net in terms of temporal logic. Then, such properties are proved by building regular expressions. The method is exemplified through the classical alternating bit protocol. [76] introduces augmented finite state automata with time and predicates on system variables to adequately describe real life systems in an operational way. Then, he defines a language suitable to describe their properties in a temporal logic framework and proposes a method to proof such properties. [33] does a similar job with an operational formalism that is strongly based on Petri nets.

These approaches somewhat complement previous literature that followed more closely Hoare's method for the analysis of sequential programs. Among these, let us mention [47][77][78][8][7] which are all based on proof rules associated with some programming language provided with concurrency constructs (monitors, guarded commands, remote procedure calls, etc.). Haase [43] (see also [32] for complements) performs a temporal analysis of programs based on Dijkstra's guarded commands.

In this paper we present a method for specifying and proving properties of real-time systems. The properties are expressed in terms of the aforementioned language TRIO and the systems are modeled by means of—any type of—timed Petri nets. Thus the method is suitable for a formal analysis of any system—not only a program, provided that it is modeled by a suitable version of a Petri net. For instance, once we defined Ada’s semantics in terms of Petri nets [59] we can apply the method to specify and prove properties of Ada programs. We can apply it as well, however, to the analysis of embedded applications that include both plant, hardware, and software components. Thus, we can claim more generality with respect to methods that are restricted to program analysis and/or to non real-time systems.

The basic idea of the method consists of an axiomatization of the behavior of Petri nets in terms of TRIO axioms and proof rules, in the same style as Hoare’s rules for Pascal-like programs. Then, net properties, such as marking and firing conditions, are expressed as TRIO formulas and their validity is proved by applying the aforementioned axiomatization.

The paper is structured as follows: Section 2 presents an overview of the TRIO language: first its syntax and semantics are briefly and informally summarized; then, a set of axioms is provided to formalize its semantics. Section 3 provides TRIO axioms formalizing the behavior of timed Petri nets. In such a way properties of the systems modeled by the nets are expressed as TRIO formulas that should be proved—or disproved—as theorems in the given axiomatization. A few simplifying assumptions are made to help focus attention on the key aspects of the method. A discussion on how these assumptions can be removed is given partially in this section and partially in the conclusions. Section 4 gives two examples of application of the proposed method by referring to two widely used “benchmarks” of the literature: an elevator system and the dining philosophers problem. Section 5 supplies a survey and a critical analysis of some related works in the literature. Finally, Section 6 contains some concluding remarks. These include an assessment of the proposed proof method and guide-lines for future research. A few, inessential, complements are deferred to Appendices.

A little familiarity with the most classic literature on mathematical logic and its applications to computer science [19, 65, 30, 8, 63] is assumed.

## **2. The TRIO language and its axioms**

TRIO is a first order logical language, augmented with temporal operators that allow the specifier to express properties whose truth value may change over time. Unlike operators of conventional temporal logic, TRIO’s temporal operators provide a metric on time, since they express precisely, and quantitatively, the distance in time between events, and the length of time intervals. The meaning of a TRIO formula is not absolute, but is given with respect to a current time instant which is left implicit in the formula, in much the same way as in temporal logic.

In the following of this section we briefly summarize TRIO’s syntax (Subsection 2.1) and semantics (Subsection 2.3) and we provide a smallest example of use of TRIO for the specification of time dependent systems (Subsection 2.2). This part of the section is very sketchy since it summarizes informations that already appeared in the open literature [37]. In any case a more detailed

formalization of TRIO's semantics is given in Appendix I. Then (Subsection 2.4), we give an axiomatization of the TRIO language and supply a small set of examples of TRIO theorems.

## 2.1 TRIO's syntax: the temporal operators

As we already stated, TRIO is a first order logical language: thus, its syntax includes concepts such as variables, constants, functions, terms, predicates, formulas, etc. which are defined in the usual way. TRIO's only peculiarities are the following ones:

- TRIO is *typed*. Thus, a domain is associated with each variable symbol and domain and ranges are associated with predicates and functions. Among the domains there is a distinguished one, required to be numeric in nature (i.e., arithmetic expressions and relations are defined on it), which is called the *Temporal Domain*.
- Predicates are divided into *time dependent* and *time independent* ones: time independent predicates always represent the same relation, whereas a time dependent predicate corresponds to a possibly distinct relation at every time instant<sup>1</sup>. Accordingly, we will say that *a formula is time independent* if it does not contain any occurrence of a time dependent predicate nor any (basic or derived) temporal operator; it is *time dependent* in the opposite case.
- A fundamental operator  $Dist^2$  is defined in such a way that if  $A$  is a formula and  $t$  is a term of the temporal type, then  $Dist(A, t)$  is a formula. The formula  $Dist(A, t)$  intuitively means that  $A$  holds at an instant laying  $t$  time units in the future (if  $t > 0$ ) or in the past (if  $t < 0$ ) with respect to the current time value, which is left implicit in the formula.

For instance, consider a simple transmission line, that receives messages at one end, to deliver them at the opposite end with a given, fixed delay. The event of arrival of a message at the input end is represented by the atomic formula  $in(m)$ , where  $in$  is a time dependent predicate, and  $m$  is a variable; the event of delivery of the message at the output end is denoted by  $out(m)$ . Then the TRIO formula:

$$(\dagger) \quad in(m) \leftrightarrow Dist(out(m), 5)$$

means that message  $m$  will be output five time units in the future if and only if the same message is arriving at the current moment.

A large number of derived temporal operators may be defined starting from  $Dist$ , using the propositional operators, conditions on the temporal arguments, and first-order quantification. A sample thereof which will be useful subsequently is given in Table 2.1, together with short intuitive explanations, whenever needed.

$Futr(\varphi, t)$	$\stackrel{\text{def}}{=}$	$t \geq 0 \wedge Dist(\varphi, t)$	Future
$Past(\varphi, t)$	$\stackrel{\text{def}}{=}$	$t \geq 0 \wedge Dist(\varphi, -t)$	Past

<sup>1</sup> In the original TRIO definition also variables are divided into time dependent and time independent variables. Here, however, we have introduced this restriction to simplify the notation in the following of the paper. The restriction can be easily removed and does not affect the generality of the language.

<sup>2</sup> We are introducing minor modifications with respect to previous papers [37, 74]

FutrS( $\varphi, t$ )	$\stackrel{\text{def}}{=}$	$t > 0 \wedge \text{Dist}(\varphi, t)$	Strict Future
PastS( $\varphi, t$ )	$\stackrel{\text{def}}{=}$	$t > 0 \wedge \text{Dist}(\varphi, -t)$	Strict Past
Lasts( $\varphi, t$ )	$\stackrel{\text{def}}{=}$	$\forall t'(0 < t' < t \rightarrow \text{Dist}(\varphi, t'))$	
Lasted( $\varphi, t$ )	$\stackrel{\text{def}}{=}$	$\forall t'(0 < t' < t \rightarrow \text{Dist}(\varphi, -t'))$	
SomF( $\varphi$ )	$\stackrel{\text{def}}{=}$	$\exists t (t > 0 \wedge \text{Dist}(\varphi, t))$	$\varphi$ occurs sometimes in the future
SomP( $\varphi$ )	$\stackrel{\text{def}}{=}$	$\exists t (t < 0 \wedge \text{Dist}(\varphi, t))$	$\varphi$ occurs sometimes in the past
AlwF( $\varphi$ )	$\stackrel{\text{def}}{=}$	$\forall t (t > 0 \rightarrow \text{Dist}(\varphi, t))$	$\varphi$ holds always in the future
AlwP( $\varphi$ )	$\stackrel{\text{def}}{=}$	$\forall t (t < 0 \rightarrow \text{Dist}(\varphi, t))$	$\varphi$ holds always in the past
Alw( $\varphi$ )	$\stackrel{\text{def}}{=}$	$\forall t \text{Dist}(\varphi, t)$	
Som( $\varphi$ )	$\stackrel{\text{def}}{=}$	$\exists t \text{Dist}(\varphi, t)$	
WithinF( $\varphi, t$ )	$\stackrel{\text{def}}{=}$	$\exists t'(0 \leq t' \leq t \wedge \text{Dist}(\varphi, t'))$	$\varphi$ will occur within $t$ time units
WithinF <sub>ee</sub> ( $\varphi, t$ )	$\stackrel{\text{def}}{=}$	$\exists t'(0 < t' < t \wedge \text{Dist}(\varphi, t'))$	strict WithinF
Until( $\varphi, \psi$ )	$\stackrel{\text{def}}{=}$	$\exists t (\text{FutrS}(\psi, t) \wedge \text{Lasts}(\varphi, t))$	$\varphi$ will remain true until $\psi$ will eventually become true
Until <sub>w</sub> ( $\varphi, \psi$ )	$\stackrel{\text{def}}{=}$	$\text{Until}(\varphi, \psi) \vee \text{AlwF}(\varphi)$	
NextTime( $\varphi, t$ )	$\stackrel{\text{def}}{=}$	$\text{Futr}(\varphi, t) \wedge \text{Lasts}(\neg\varphi, t)$	the first time in the future when $\varphi$ will hold is $t$ time units apart from now
UpToNow( $\varphi$ )	$\stackrel{\text{def}}{=}$	$\exists d (d > 0 \wedge \text{Past}(\varphi, d) \wedge \text{Lasted}(\varphi, d))$	there is a non-empty time interval ending at the present time, where $\varphi$ has been true

**Table 2.1** A sample of derived temporal operators.

## 2.2 The use of TRIO as a specification language: a smallest example

The following example shows the use of TRIO as a specification language for time dependent systems. Since our attention here is not on the TRIO language by itself but on its use to prove properties of systems modeled as timed Petri nets, we limit ourselves to an oversimplified example. We emphasize however that TRIO—and other languages derived therefrom—have already been successfully applied to several real-life case studies [72]. More comments on this crucial aspect will be given in Section 6.

Consider a hydroelectric pondage power station, where the quantity of water held in the basin is controlled by means of a sluice gate. The gate is controlled via two commands, *up* and *down* which respectively open and close it, and are represented as a TRIO time dependent predicate named *go* with an argument in the range  $\{up, down\}$ . The current state of the gate can have one of the four values: *up* and *down* (with the obvious meaning), and *mvup*, *mvdwn* (meaning, respectively, that the gate is being opened or closed). It is modelled by the time dependent predicate *position* whose domain is the set  $\{up, down, mvup, mvdwn\}$ .

The following formula describes the fact that it takes the sluice gate  $\Delta$  time units to go from the *down* to the *up* position, after receiving a *go(up)* command.

$$\text{position}(\text{down}) \wedge \text{go}(\text{up}) \rightarrow \text{Lasts}(\text{position}(\text{mvup}), \Delta) \wedge \text{Futr}(\text{position}(\text{up}), \Delta)$$

When a *go(up)* command arrives while the gate is not still in the *down* position, but it is moving down because of a preceding *go(down)* command, then the direction of motion of the gate is not reversed

immediately, but the downward movement proceeds until the *down* position has been reached. Only then the gate will start opening according to the received command. This is formalized by the following formula.

$$\left( \text{position}(\text{mvDown}) \wedge \text{go}(\text{up}) \right) \rightarrow \exists t \left( \text{NextTime}(\text{position}(\text{down}), t) \wedge \text{Futr} \left( \left( \begin{array}{l} \text{Lasts}(\text{position}(\text{mvup}), \Delta) \wedge \\ \text{Futr}(\text{position}(\text{up}), \Delta) \end{array} \right), t \right) \right)$$

If the behavior of the sluice gate is symmetrical with respect to its direction of motion, two similar TRIO formulas will describe the commands and their effects in the opposite direction.

### 2.3 TRIO's semantics

TRIO has been given a model theoretic formal semantics [37] in a fairly natural way. In this case too, the essential difference from traditional interpretation schemata of logic languages [65][30] consists of defining an *interpretation structure* that associates a distinguished set, the *temporal domain*, denoted as  $T$ , to each term of temporal type. Any formula  $F$  is then evaluated in a generic element  $i$  of the temporal domain and its truth value is denoted as  $S_i(F)$ . Thus,  $F$  is said to be *temporally satisfiable* for a given interpretation if there exists a time instant  $i \in T$  for which  $S_i(F) = \text{true}$ .  $F$  is said *temporally valid* in the given interpretation if and only if  $S_i(F) = \text{true}$  for every  $i \in T$ ; it is *valid* (formally  $\models F$ ) if it is temporally valid in every syntactically adequate interpretation. An interpretation such that  $F$  is temporally satisfiable for it is called a *model* of  $F$ .

As we mentioned above, a more complete and formal definition of TRIO's semantics is given in Appendix I. Here, it is worth noticing that in [74] a *model-parametric semantics* for TRIO is defined, which may refer to any finite or infinite temporal domain. This is motivated mostly by the wish to achieve *executability* to favor *early prototyping* of system specifications [53][38]. Here we refer, however, to unbounded interpretation domains such that all defined functions and predicates are total. This assumption simplifies the mathematical description in terms of a set of axioms and inference rules, thus allowing shorter and easy-to-understand proofs. More comments on this point will be given in Subsection 2.4.3.

### 2.4 An axiom system for TRIO

In this subsection we provide an axiomatization of TRIO. Since the language allows to use almost any kind of interpretation domains, with particular reference to the temporal domain, any axiom system for it should include a first order theory for all the used domains (say, real numbers for temporal domain, integers for some variables, booleans for other variables, etc.). This part of TRIO axiomatization, however, would become rather lengthy and detailed, without providing any new insight into the proofs. Thus, for the sake of simplicity, and following the same approach as adopted in a variety of different temporal logics [81, 76] and in Hoare-like program logics [8], we will just use in our proofs all valid formulas of the used domains as additional axioms.

TRIO's axioms are given below. For convenience they are partitioned into *general axioms*, which are shared with any first-order theory with equality, and *temporal axioms*, which are peculiar of the language. A universal axiom schema is added at the end of both categories.

Let  $\alpha, \beta, \omega, \dots$  denote any TRIO formula; let  $s, v, u, \dots$  denote any term of a generic domain, whereas  $t, t_1, t_2, \dots$  denote any term of temporal type; let  $x, y, \dots$  denote any variable and  $c$  any constant.

- **General axioms.** These, in turn, are split into first-order predicate axioms and into equality axioms as shown below.
  - *First order axioms*
    1. All instances of propositional calculus tautologies
    2.  $\forall x \alpha \rightarrow \alpha_s^x$  where  $s$  is a term substitutable for  $x$  in  $\alpha$ <sup>1</sup> [30]
    3.  $\forall x(\alpha \rightarrow \beta) \rightarrow (\forall x \alpha \rightarrow \forall x \beta)$
    4.  $\alpha \rightarrow \forall x \alpha$  if  $x$  is not free in  $\alpha$
  - *Equality Axioms*
    5.  $s = s$ , for any term  $s$
    6.  $u = s \rightarrow (\alpha \rightarrow \alpha')$  where  $u$  and  $s$  are terms,  $\alpha'$  is obtained from  $\alpha$  by substituting zero or more occurrences of  $u$  in  $\alpha$  by  $s$ .
- **Temporal axioms**
  7.  $\text{Dist}(\alpha, 0) \leftrightarrow \alpha$
  8.  $\text{Dist}(\alpha, t_1 + t_2) \leftrightarrow \text{Dist}(\text{Dist}(\alpha, t_1), t_2)$
  9.  $\text{Dist}(\alpha \rightarrow \beta, t) \leftrightarrow (\text{Dist}(\alpha, t) \rightarrow \text{Dist}(\beta, t))$
  10.  $\text{Dist}(\neg\alpha, t) \leftrightarrow \neg\text{Dist}(\alpha, t)$
  11.  $\alpha \rightarrow \text{Alw}(\alpha)$  if  $\alpha$  is time independent
- For each formula  $\omega$  in the above list 1 through 11, all formulas of the kind  $\text{Alw}(\omega)$ , and all their applications of universal quantifications (generalizations) are TRIO axioms.

There is a single rule of inference, namely Modus Ponens (MP). By using classical Hoare's notation, it is denoted as

$$\frac{\Gamma \vdash \alpha, \Gamma \vdash \alpha \rightarrow \beta}{\Gamma \vdash \beta}$$

Axioms 7 through 10 describe the essential properties of the basic temporal operator  $\text{Dist}$ : when in the formula  $\text{Dist}(\alpha, t)$  the temporal argument  $t$  is 0 then the other argument,  $\alpha$ , is asserted at the current instant; furthermore, nested applications of the operator compose additively, according to the properties of the temporal domain, and the operator is transparent with respect to propositional operators of its non-temporal argument.

Axiom 11 simply states the time invariance of time independent formulas (those which do not contain any time dependent predicate or temporal operator): if the formula is true *now* then it is true at any time instant of the temporal domain (the converse is trivially true, and in fact the corresponding formula,  $\text{Alw}(\alpha) \rightarrow \alpha$ , is a theorem whose proof is immediate).

---

<sup>1</sup> As usual,  $\alpha_s^x$  denotes the result of substituting any free occurrence of  $x$  in  $\alpha$  by  $s$ .

Next (Subsection 2.4.1), we give some *metatheorems* (properties of the axiomatization, to be distinguished from *theorems* that are derived from deductions *within* the axiom system [65]) that will be useful for the proof of many *theorems*. Then (Subsection 2.4.2), we provide a few sample theorems both to illustrate some relevant properties of the axiom system and to state useful lemmas to be exploited in the proof of Petri nets properties. Finally (Subsection 2.4.3), we shortly discuss the main features of the proposed axiomatization and possible alternatives.

### 2.4.1. Some useful metatheorems

As usual, we distinguish between *general metatheorems* that are well known properties of classical first-order theories and *temporal metatheorems* that are peculiar of the TRIO language.

#### General metatheorems

*Generalization Theorem (GEN)*

if  $\Gamma \vdash \varphi$  and variable  $x$  does not occur free in any formula of  $\Gamma$ , then  $\Gamma \vdash \forall x \varphi$

*Deduction Theorem (DED)*

if  $\Gamma; \varphi \vdash \psi$  then  $\Gamma \vdash \varphi \rightarrow \psi$ <sup>1</sup>

*Existential Instantiation Theorem (EI)*

if  $\Gamma; \varphi_c^x \vdash \psi$  and the constant symbol  $c$  does not occur in  $\varphi, \psi$ , nor  $\Gamma$ , then  $\Gamma; \exists x \varphi \vdash \psi$  and there is a deduction of  $\psi$  from  $\Gamma; \exists x \varphi$  in which  $c$  does not occur<sup>2</sup>.

These results are the TRIO counterpart of well known theorems which hold for the most widely accepted axiomatizations of first-order logic. Their proof is straightforward and runs along the same lines as the corresponding proofs in textbooks such as [30], and hence is omitted. Similarly, since TRIO's axiomatization includes a standard first order part, all the derived inference rules usually employed in proofs for first-order logic [65] [30] are also valid in TRIO. For instance, in the rest of the paper we will use the derived rules of negation, conjunction, disjunction, conditional, and biconditional introduction and elimination, case analysis<sup>3</sup> (CA) and proof by contradiction. Furthermore, we will apply tautologies in the form of an implication, e.g.  $((\alpha \rightarrow \beta) \wedge (\alpha \rightarrow \gamma) \wedge \alpha) \rightarrow \beta \wedge \gamma$ , as derived inference rules, justifying the corresponding derivation step by TAUT, plus the indication of the involved tautology when it will not be obvious from the context.

#### Temporal metatheorems

*Temporal Translation Theorem (TT)*

This metatheorem asserts that if a formula  $\alpha$  can be proved under a given set of assumptions that hold at the present time instant and all these assumptions hold at a different time instant, then it can be proved that  $\alpha$  holds at that time instant too. The metatheorem is formalized as follows.

<sup>1</sup>  $\Gamma; \varphi$  is an abbreviated notation for  $\Gamma \cup \{\varphi\}$ .

<sup>2</sup> Whenever we will make use of the EI theorem in proofs, we will replace the existentially quantified variable by the same symbol in "small cap" format.

<sup>3</sup> The derived rule case analysis (CA) obtains  $\vdash (\alpha \vee \beta) \rightarrow \gamma$  from  $\alpha \vdash \gamma$  and  $\beta \vdash \gamma$  using DED,  $\wedge$ -intro, tautology  $(\alpha \rightarrow \gamma) \wedge (\beta \rightarrow \gamma) \rightarrow ((\alpha \vee \beta) \rightarrow \gamma)$ , and MP.

if  $\Gamma \vdash \alpha$  then  $\{\text{Dist}(\gamma, t) \mid \gamma \in \Gamma\} \vdash \text{Dist}(\alpha, t)$

The proof is by induction on the length of the derivation of  $\alpha$ .

*Basis of the induction.*

- $\alpha \in \Gamma$  implies  $\text{Dist}(\alpha, t) \in \{\text{Dist}(\gamma, t) \mid \gamma \in \Gamma\}$
- if  $\alpha$  is a TRIO axiom then  $\text{Alw}(\alpha)$  is also an axiom, and  $\text{Alw}(\alpha) \vdash \text{Dist}(\alpha, t)$ , by Axiom 2 and MP

*Induction step.*

Assume that, for some  $\beta$ ,  $\Gamma \vdash (\beta \rightarrow \alpha) \wedge \beta$ . This implies, by the inductive hypothesis, that  $\{\text{Dist}(\gamma, t) \mid \gamma \in \Gamma\} \vdash \text{Dist}((\beta \rightarrow \alpha) \wedge \beta, t)$ , from which  $\text{Dist}(\alpha, t)$  can be deduced since  $\text{Dist}$  is transparent with respect to propositional operators (Axioms 9 and 10) ■

*Temporal Generalization Theorem (TG)*

This metatheorem is an extension to the preceding result. It states that if the set of assumptions on which the proof of a property is based is true in every instant of the temporal domain, then the proven formula is also always true. Formally,

if  $\Gamma \vdash \alpha$  and every formula of  $\Gamma$  is of the type  $\text{Alw}(\gamma)$  or is time independent, then  $\Gamma \vdash \text{Alw}(\alpha)$ .

The proof of TG is also by induction on the length of the derivation of  $\alpha$ .

Base case.

- if  $\alpha \in \Gamma$  then if  $\alpha$  is time independent, then  $\alpha \rightarrow \text{Alw}(\alpha)$  (Ax12) and  $\alpha \rightarrow \text{Alw}(\alpha) \vdash \text{Alw}(\alpha)$ ;  
if  $\alpha = \text{Alw}(\phi)$ , then  $\text{Alw}(\phi) \rightarrow \text{Alw}(\text{Alw}(\phi))$  is a TRIO theorem, so  $\alpha \vdash \text{Alw}(\alpha)$ ;
- if  $\alpha$  is a TRIO axiom then  $\text{Alw}(\alpha)$  is also an axiom.

*Induction step.*

Assume that, for some  $\beta$ ,  $\Gamma \vdash \beta$  and  $\Gamma \vdash \beta \rightarrow \alpha$ . This implies, by the inductive hypothesis, that  $\Gamma \vdash \text{Alw}(\beta)$  and  $\Gamma \vdash \text{Alw}(\beta \rightarrow \alpha)$ . From these  $\text{Alw}(\beta \wedge (\beta \rightarrow \alpha))$  trivially follows, and the following derivation holds

1. $\text{Alw}(\beta \wedge (\beta \rightarrow \alpha))$	Ind. Hyp
2. $\text{Dist}(\beta, x) \wedge (\text{Dist}(\beta, x) \rightarrow \text{Dist}(\alpha, x))$	1, Ax2, MP, Ax9+10
3. $\text{Dist}(\alpha, x)$	2, TAUT, MP
4. $\text{Alw}(\alpha)$	3, GEN

■

An important corollary of TG is obtained by taking  $\Gamma = \emptyset$ . In this case TG reduces to: if  $\vdash \alpha$  then  $\vdash \text{Alw}(\alpha)$ . This corresponds to the intuitive fact that if property  $\alpha$  is derived without making any assumption about the current time instant, then  $\alpha$  holds at every time instant. Another consequence of TG is that any theorem  $\tau$  of first-order logic is not only inherited as such in TRIO, but its temporal generalization,  $\text{Alw}(\tau)$  is also a theorem. For instance,  $\text{Alw}(\alpha(t) \rightarrow \exists z \alpha(z))$  holds by the fact that  $\alpha(t) \rightarrow \exists z \alpha(z)$  is a theorem in any first-order logic: in the following we will make use of this theorem in TRIO proofs, referring to it as  $\exists$ -intro.

### 2.4.2. A sample of TRIO theorems

Here we give a short list of theorems that can be deduced from the above TRIO axioms. They will be used in the following sections as lemmas to state properties of Petri nets expressed as TRIO formulas.

In the proofs of all theorems presented in this paper it is assumed, unless otherwise specified, that the temporal domain is a standard model of the real numbers. Thus, we use in the proofs any formula regarding the temporal domain (e.g. formulas expressing density, or linear order) known to be true in such an interpretation without bothering with the derivation of a formal deduction thereof. Such formulas will be labeled as TD (temporal domain). It must be pointed out, however, that, unless the contrary is explicitly stated, all theorems claimed herewith still hold even for integer time domains: all that is needed is just replacing appropriately the used formulas.

**i.**  $\text{Alw}(\alpha) \rightarrow \text{Dist}(\alpha, t)$

*Proof of i.*

- |  |                 |
|--|-----------------|
| 1. $\text{Alw}(\alpha) \equiv \forall x \text{Dist}(\alpha, x)$          | <b>Hyp</b>      |
| 2. $\forall x \text{Dist}(\alpha, x) \rightarrow \text{Dist}(\alpha, t)$ | <b>Ax2</b>      |
| 3. $\text{Dist}(\alpha, t)$  | <b>1, 2, MP</b> |
| 4. $\text{Alw}(\alpha) \vdash \text{Dist}(\alpha, t)$                    | <b>1 ÷ 3</b>    |
| 5. $\text{Alw}(\alpha) \rightarrow \text{Dist}(\alpha, t)$               | <b>4, DED</b>   |

**ii.**  $\text{Alw}(\alpha) \vdash \alpha$

ii is derived much in the same way as i, by just taking  $t = 0$ , and using axiom 7.

**iii.**  $(\text{Alw}(\varphi \rightarrow \psi) \wedge \text{Dist}(\varphi, t)) \rightarrow \text{Dist}(\psi, t)$

iii means that if it is always the case that  $\varphi$  implies  $\psi$ , and  $\varphi$  takes place at a time instant (possibly different from the current one), then  $\psi$  is also true at that same instant.

*Proof of iii.*

- |  |                                    |
|--|------------------------------------|
| 1. $\text{Alw}(\varphi \rightarrow \psi) \wedge \text{Dist}(\varphi, t)$   | <b>Hyp.</b>                        |
| 2. $\text{Alw}(\varphi \rightarrow \psi) \equiv \forall x \text{Dist}(\varphi \rightarrow \psi, x)$              | <b>1, <math>\wedge</math>-elim</b> |
| 3. $\text{Dist}(\varphi \rightarrow \psi, t)$  | <b>2, Ax2, MP</b>                  |
| 4. $\text{Dist}(\varphi, t) \rightarrow \text{Dist}(\psi, t)$  | <b>3, Ax9, MP</b>                  |
| 5. $\text{Dist}(\varphi, t)$   | <b>1, <math>\wedge</math>-elim</b> |
| 6. $\text{Dist}(\psi, t)$  | <b>5, 4, MP</b>                    |
| 7. $\text{Alw}(\varphi \rightarrow \psi) \wedge \text{Dist}(\varphi, t) \vdash \text{Dist}(\psi, t)$             | <b>1 ÷ 6</b>                       |
| 8. $\vdash \text{Alw}(\varphi \rightarrow \psi) \wedge \text{Dist}(\varphi, t) \rightarrow \text{Dist}(\psi, t)$ | <b>7, DED</b>                      |

**iv.**  $\text{Dist}(\varphi, x) \rightarrow \varphi$ , where  $\varphi$  is a time independent formula

iv states that if a time independent formula is true at some instant different than the current one, then it is also true now.

*Proof of iii.*

- |   |             |
|---|-------------|
| 1. $\text{Dist}(\varphi, x)$  | <b>Hyp.</b> |
| 2. $\text{Alw}(\text{Alw}(\varphi) \rightarrow \text{Dist}(\varphi, -x))$ | <b>Ax2</b>  |

3. $\text{Dist}(\text{Alw}(\varphi), x) \rightarrow \text{Dist}(\text{Dist}(\varphi, -x), x)$	<b>2, Ax2, Ax9-10, MP</b>
4. $\text{Alw}(\varphi \rightarrow \text{Alw}(\varphi))$	<b>Ax11</b>
5. $\text{Dist}(\varphi \rightarrow \text{Alw}(\varphi), x)$	<b>4, Ax2, MP</b>
6. $\text{Dist}(\varphi, x) \rightarrow \text{Dist}(\text{Alw}(\varphi), x)$	<b>5, Ax9, MP</b>
7. $\text{Dist}(\text{Alw}(\varphi), x)$	<b>1, 6, MP</b>
8. $\text{Dist}(\text{Dist}(\varphi, -x), x)$	<b>3, 7, MP</b>
9. $\varphi$	<b>8, TD, Ax7, TAUT</b>
10. $\text{Dist}(\varphi, x) \vdash \varphi$	<b>1÷9</b>
11. $\vdash \text{Dist}(\varphi, x) \rightarrow \varphi$	<b>10, DED</b>

Some properties similar to those expressed by axioms 8, 9, and 10 for the basic temporal operator *Dist*, hold as well for the derived operators *Futr* and *Past*. Here are a few of them

**v.**  $\text{Futr}(\alpha \rightarrow \beta, t) \rightarrow (\text{Futr}(\alpha, t) \rightarrow \text{Futr}(\beta, t))$

*Proof of v.*

1. $\text{Futr}(\alpha \rightarrow \beta, t) \equiv t > 0 \wedge \text{Dist}(\alpha \rightarrow \beta, t)$	<b>Hyp.</b>
2. $\text{Futr}(\alpha, t) \equiv t > 0 \wedge \text{Dist}(\alpha, t)$	<b>Hyp.</b>
3. $\text{Dist}(\alpha, t) \rightarrow \text{Dist}(\beta, t)$	<b>1, Ax9, MP</b>
4. $\text{Dist}(\beta, t)$	<b>2, 3, MP</b>
5. $\text{Futr}(\beta, t)$	<b>1, 4, <math>\wedge</math>-intro</b>
6. $\text{Futr}(\alpha \rightarrow \beta, t), \text{Futr}(\alpha, t) \vdash \text{Futr}(\alpha, t)$	<b>1÷5</b>
7. $\vdash \text{Futr}(\alpha \rightarrow \beta, t) \rightarrow (\text{Futr}(\alpha, t) \rightarrow \text{Futr}(\alpha, t))$	<b>6, DED<sup>2</sup></b>

**vi.**  $\text{Futr}(\text{Futr}(\alpha, t1), t2) \rightarrow \text{Futr}(\alpha, t1+t2)$

$\text{Futr}(\text{Past}(\alpha, t1), t2) \wedge t1 > t2 \rightarrow \text{Futr}(\alpha, t1-t2)$

$\text{Futr}(\text{Past}(\alpha, t1), t2) \wedge t1 < t2 \rightarrow \text{Past}(\alpha, t2-t1)$

*Proof of  $\text{Futr}(\text{Futr}(\alpha, t1), t2) \rightarrow \text{Futr}(\alpha, t1+t2)$ .*

1. $\text{Futr}(\text{Futr}(\alpha, t1), t2) \equiv t2 > 0 \wedge \text{Dist}(t1 > 0 \wedge \text{Dist}(\alpha, t1), t2)$	<b>Hyp.</b>
2. $\text{Dist}(t1 > 0, t2)$	<b>1, Ax9-10, MP</b>
3. $t1 > 0$	<b>2, Th. iv, MP</b>
4. $\text{Dist}(\text{Dist}(\alpha, t1), t2)$	<b>1, Ax9-10, MP</b>
5. $\text{Dist}(\alpha, t1+t2)$	<b>4, Ax8, MP</b>
6. $t1 > 0 \wedge t2 > 0$	<b>1, 3, <math>\wedge</math>-intro</b>
7. $t1+t2 > 0$	<b>6, TD</b>
8. $t1+t2 > 0 \wedge \text{Dist}(\alpha, t1+t2) \equiv \text{Futr}(\alpha, t1+t2)$	<b>5, 7, <math>\wedge</math>-intro</b>
9. $\vdash \text{Futr}(\text{Futr}(\alpha, t1), t2) \rightarrow \text{Futr}(\alpha, t1+t2)$	<b>1÷8, DED</b>

The proofs of the other two properties are entirely similar, and are omitted.

**vii.**  $\text{Dist}(\forall x \alpha, t) \leftrightarrow \forall x \text{Dist}(\alpha, t)$ , and  $\text{Dist}(\exists x \alpha, t) \leftrightarrow \exists x \text{Dist}(\alpha, t)$  with  $x$  not occurring in  $t$ .

*Proof of  $\text{Dist}(\forall x \alpha, t) \rightarrow \forall x \text{Dist}(\alpha, t)$ .*

1. $\text{Dist}(\forall x \alpha, y)$	<b>Hyp</b>
2. $\text{Alw}(\forall x \alpha \rightarrow \alpha)$	<b>Ax 2</b>

- |  |                 |
|--|-----------------|
| 3. $\text{Alw}(\forall x \alpha \rightarrow \alpha) \rightarrow \text{Dist}(\forall x \alpha, y)$  | <b>Ax 2</b>     |
| 4. $\text{Dist}(\forall x \alpha, y)$  | <b>2, 3, MP</b> |
| 5. $\text{Dist}(\forall x \alpha \rightarrow \alpha, y) \rightarrow (\text{Dist}(\forall x \alpha, y) \rightarrow \text{Dist}(\alpha, y))$ | <b>Ax9</b>      |
| 6. $\text{Dist}(\forall x \alpha, y) \rightarrow \text{Dist}(\alpha, y)$   | <b>4, 5, MP</b> |
| 7. $\text{Dist}(\alpha, y)$  | <b>1, 6, MP</b> |
| 8. $\text{Dist}(\forall x \alpha, y) \vdash \text{Dist}(\alpha, y)$  | <b>1÷7</b>      |
| 9. $\text{Dist}(\forall x \alpha, y) \vdash \forall x \text{Dist}(\alpha, y)$  | <b>8, GEN</b>   |
| 10. $\vdash \text{Dist}(\forall x \alpha, y) \rightarrow \forall x \text{Dist}(\alpha, y)$   | <b>9, DED</b>   |

*Proof of  $\forall x \text{Dist}(\alpha, t) \rightarrow \text{Dist}(\forall x \alpha, t)$ .*

- |  |  |
|--|--|
| 1. $\forall x \text{Dist}(\alpha, t)$  | <b>Hyp</b>   |
| 2. $\forall x \text{Dist}(\alpha, t) \rightarrow \text{Dist}(\alpha, t)$   | <b>Ax2</b>   |
| 3. $\text{Dist}(\alpha, t)$  | <b>1, 2, MP</b>  |
| 4. $\forall x \text{Dist}(\alpha, t) \vdash \text{Dist}(\alpha, t)$  | <b>1÷3</b>   |
| 5. $\text{Dist}(\forall x \text{Dist}(\alpha, t), -t) \vdash \text{Dist}(\text{Dist}(\alpha, t), -t)$                    | <b>4, TT</b>   |
| 6. $\text{Dist}(\forall x \text{Dist}(\alpha, t), -t) \vdash \alpha$   | <b>5, Ax6, Ax7, 0=t-t</b>  |
| 7. $\text{Dist}(\forall x \text{Dist}(\alpha, t), -t) \vdash \forall x \alpha$   | <b>6, GEN</b>  |
| 8. $\text{Dist}(\text{Dist}(\forall x \text{Dist}(\alpha, t), -t), t) \vdash \text{Dist}(\forall x \alpha, t)$           | <b>7, TT</b>   |
| 9. $\text{Dist}(\text{Dist}(\forall x \text{Dist}(\alpha, t), -t), t) \rightarrow \text{Dist}(\forall x \alpha, t)$      | <b>8, DED</b>  |
| 10. $\text{Dist}(\text{Dist}(\forall x \text{Dist}(\alpha, t), -t), t) \leftrightarrow \forall x \text{Dist}(\alpha, t)$ | <b>Ax6, Ax7, 0=t-t</b>   |
| 11. $\forall x \text{Dist}(\alpha, t) \rightarrow \text{Dist}(\forall x \alpha, t)$                                      | <b>9, 10, TAUT <math>(\alpha \rightarrow \beta) \wedge (\alpha \leftrightarrow \gamma) \rightarrow (\gamma \rightarrow \beta)</math></b> |

The formula  $\text{Dist}(\forall x \alpha, t) \leftrightarrow \forall x \text{Dist}(\alpha, t)$  is a tautological consequence of the two above-proved implications.

*Proof of  $\exists x \text{Dist}(\alpha, y) \rightarrow \text{Dist}(\exists x \alpha, y)$ .*

- |   |                        |
|---|------------------------|
| 1. $\text{Dist}(\forall x \neg \alpha, y)$  | <b>Hyp</b>             |
| 2. $\text{Dist}(\neg \alpha, y)$  | <b>1, Ax2, Ax9, MP</b> |
| 3. $\neg \text{Dist}(\alpha, y)$  | <b>2, Ax8, MP</b>      |
| 4. $\forall x \neg \text{Dist}(\alpha, y)$  | <b>3, GEN</b>          |
| 5. $\vdash \text{Dist}(\forall x \neg \alpha, y) \rightarrow \forall x \neg \text{Dist}(\alpha, y)$           | <b>1÷4, DED</b>        |
| 6. $\vdash \neg \forall x \neg \text{Dist}(\alpha, y) \rightarrow \neg \text{Dist}(\forall x \neg \alpha, y)$ | <b>5, TAUT</b>         |
| 7. $\vdash \exists x \text{Dist}(\alpha, y) \rightarrow \text{Dist}(\exists x \alpha, y)$                     | <b>6, Ax10, TAUT.</b>  |

*Proof of  $\text{Dist}(\exists x \alpha, y) \rightarrow \exists x \text{Dist}(\alpha, y)$ .*

- |   |                             |
|---|-----------------------------|
| 1. $\forall x \neg \text{Dist}(\alpha, y)$  | <b>Hyp</b>                  |
| 2. $\neg \text{Dist}(\alpha, y)$  | <b>1, Ax2, MP</b>           |
| 3. $\text{Dist}(\neg \alpha, y)$  | <b>2, Ax9, MP</b>           |
| 4. $\forall x \text{Dist}(\neg \alpha, y)$  | <b>3, GEN</b>               |
| 5. $\vdash \forall x \neg \text{Dist}(\alpha, y) \rightarrow \text{Dist}(\forall x \neg \alpha, y)$ | <b>1÷4, DED, Ax11, TAUT</b> |
| 6. $\vdash \neg \text{Dist}(\forall x \neg \alpha, y) \rightarrow \exists x \text{Dist}(\alpha, y)$ | <b>5, TAUT</b>              |
| 7. $\vdash \text{Dist}(\exists x \alpha, y) \rightarrow \exists x \text{Dist}(\alpha, y)$           | <b>6, Ax10, TAUT.</b>       |

The formula  $\exists x \text{Dist}(\alpha, y) \leftrightarrow \text{Dist}(\exists x \alpha, y)$  is a tautological consequence of the two above-proved implications.

**viii.**  $\text{Alw}(\alpha) \leftrightarrow \text{Alw}(\text{Dist}(\alpha, a))$

*Proof of  $\text{Alw}(\alpha) \leftrightarrow \text{Alw}(\text{Dist}(\alpha, a))$ .*

1. $\text{Alw}(\alpha)$	<b>Hyp.</b>
2. $\text{Dist}(\alpha, x+a)$	<b>1, Ax2, MP</b>
3. $\text{Dist}(\text{Dist}(\alpha, a), x)$	<b>2, Ax8</b>
4. $\text{Alw}(\text{Dist}(\alpha, a))$	<b>3, GEN</b>
5. $\text{Alw}(\alpha) \vdash \text{Alw}(\text{Dist}(\alpha, a))$	<b>1 ÷ 4</b>
6. $\vdash \text{Alw}(\alpha) \rightarrow \text{Alw}(\text{Dist}(\alpha, a))$	<b>5, DED</b>
7. $\text{Alw}(\text{Dist}(\alpha, a))$	<b>Hyp.</b>
8. $\text{Dist}(\text{Dist}(\alpha, a), x-a)$	<b>7, Ax2, MP</b>
9. $\text{Dist}(\alpha, x)$	<b>8, Ax8</b>
10. $\text{Alw}(\alpha)$	<b>9, GEN</b>
11. $\text{Alw}(\text{Dist}(\alpha, a)) \vdash \text{Alw}(\alpha)$	<b>7 ÷ 10</b>
12. $\vdash \text{Alw}(\text{Dist}(\alpha, a)) \rightarrow \text{Alw}(\alpha)$	<b>11, DED</b>
13. $\text{Alw}(\alpha) \leftrightarrow \text{Alw}(\text{Dist}(\alpha, a))$	<b>6, 12, TAUT</b>

**ix.**  $\varphi \wedge \text{Alw}(\varphi \rightarrow \text{Futr}(\varphi, a)) \rightarrow \forall k \text{Futr}(\varphi, a \cdot k)$ , where  $k$  is a variable ranging over positive natural numbers.

ix states that if a system is periodic and the repeating property takes place at a given instant (the current instant, for simplicity) then the property will actually repeat itself indefinitely in all future times, at a regular pace. The proof of ix is essentially by induction on the natural numbers: at step 4 of the sketchy derivation given below a formula that is an instance of the induction axiom is assumed as a valid formula of the temporal domain.

*Proof of ix*

1. $\varphi, \text{Alw}(\varphi \rightarrow \text{Futr}(\varphi, a))$	<b>Hyp</b>
2. $\varphi \rightarrow \text{Futr}(\varphi, a)$	<b>1, Ax. 2, MP</b>
3. $\text{Futr}(\varphi, a)$	<b>1, 2, MP</b>
4. $\text{Futr}(\varphi, a \cdot 1) \wedge \forall k(\text{Futr}(\varphi, a \cdot k) \rightarrow \text{Futr}(\varphi, a \cdot (k+1))) \rightarrow \forall k \text{Futr}(\varphi, a \cdot k)$	<b>Induction Axiom for TD</b>
5. $\text{Futr}(\varphi, a \cdot k)$	<b>Hyp.</b>
6. $\text{Dist}(\varphi \rightarrow \text{Futr}(\varphi, a), a \cdot k)$	<b>1, Ax. 2, MP</b>
7. $\text{Dist}(\text{Futr}(\varphi, a), a \cdot k)$	<b>5, 6, Axioms 8÷9, MP</b>
8. $\text{Futr}(\varphi, a \cdot (k+1))$	<b>7, Ax. 11, def Futr, Axioms 8÷9</b>
9. $\text{Futr}(\varphi, a \cdot k) \vdash \text{Futr}(\varphi, a \cdot (k+1))$	<b>5 ÷ 8</b>
10. $\vdash \text{Futr}(\varphi, a \cdot k) \rightarrow \text{Futr}(\varphi, a \cdot (k+1))$	<b>9, DED</b>
11. $\vdash \forall k(\text{Futr}(\varphi, a \cdot k) \rightarrow \text{Futr}(\varphi, a \cdot (k+1)))$	<b>10, GEN</b>
12. $\forall k \text{Futr}(\varphi, a \cdot k)$	<b>4, 5, 11, MP</b>
13. $\varphi, \text{Alw}(\varphi \rightarrow \text{Futr}(\varphi, a)) \vdash \forall k \text{Futr}(\varphi, a \cdot k)$	<b>1 ÷ 12</b>
14. $\vdash \varphi \wedge \text{Alw}(\varphi \rightarrow \text{Futr}(\varphi, a)) \rightarrow \forall k \text{Futr}(\varphi, a \cdot k)$	<b>13, DED (applied twice), TAUT</b>

ix can be generalized in several ways. Here are a few examples thereof.

**x .**  $\varphi \wedge \text{Alw}(\varphi \rightarrow \text{Futr}(\varphi, a)) \rightarrow \text{AlwF}(\text{WithinF}(\varphi, a))$

$x$  states that under the same hypotheses as in theorem ix, in all future time instants the condition  $\varphi$  takes place within  $a$  time units. Its derivation is similar to the preceding one. A key step in such a derivation uses the auxiliary result stated in theorem xi below.

**xi.**  $\text{Futr}(\varphi, a) \rightarrow \text{Lasts}(\text{WithinF}(\varphi, a), a)$ .

*Proof of xi*

1. $a > 0 \wedge \text{Dist}(\varphi, a) \wedge 0 < x < a$	<b>Hyp.</b>
2. $\exists y(a = x + y \wedge y \geq 0 \wedge y \leq a)$	<b>1, TD</b>
3. $a = x + Y \wedge 0 \leq Y \leq a$	<b>2, EI</b>
4. $\text{Dist}(\varphi, x + Y)$	<b>1, 3, Ax6, MP</b>
5. $\text{Dist}(\text{Dist}(\varphi, Y), x)$	<b>4, Ax8</b>
6. $\text{Dist}(0 \leq Y \leq a, x)$	<b>3, Ax11, MP</b>
7. $\text{Dist}(0 \leq Y \leq a \wedge \text{Dist}(\varphi, Y), x)$	<b>5, 6, Ax9÷10, MP</b>
8. $\text{Dist}(\text{WithinF}(\varphi, a), x)$	<b>7, TRIO Th. Alw(<math>\alpha(t) \rightarrow \exists z \alpha(z)</math>) (<math>\exists</math>-intro)</b>
9. $\text{Futr}(\varphi, a), 0 < x < a \vdash \text{Dist}(\text{WithinF}(\varphi, a), x)$	<b>1 ÷ 8</b>
10. $\text{Futr}(\varphi, a) \vdash 0 < x < a \rightarrow \text{Dist}(\text{WithinF}(\varphi, a), x)$	<b>9, DED</b>
11. $\text{Futr}(\varphi, a) \vdash \text{Lasts}(\text{WithinF}(\varphi, a), a)$	<b>11, GEN</b>
12. $\vdash \text{Futr}(\varphi, a) \rightarrow \text{Lasts}(\text{WithinF}(\varphi, a), a)$	<b>11, DED</b>

*Proof of x*

1. $\varphi, \text{Alw}(\varphi \rightarrow \text{Futr}(\varphi, a))$	<b>Hyp</b>
2. $\varphi \wedge \text{Alw}(\varphi \rightarrow \text{Futr}(\varphi, a)) \rightarrow \forall i \text{Futr}(\varphi, i \cdot a)$	<b>Th. ix</b>
3. $\forall x (x > 0 \rightarrow \exists k (x = a \cdot k \vee a \cdot k < x < a \cdot (k + 1)))$	<b>TD</b>
4. $x > 0$	<b>Hyp</b>
5. $\exists k (x = a \cdot k \wedge a \cdot k < x < a \cdot (k + 1))$	<b>3, Ax2, 4, MP</b>
6. $x = a \cdot K \vee a \cdot K < x < a \cdot (K + 1)$	<b>5, EI</b>
7. $x = a \cdot K$	<b>Hyp</b>
8. $\text{Futr}(\varphi, a \cdot K)$	<b>1, 2, MP, Ax2, MP</b>
9. $\text{Futr}(0 \leq 0 \leq a \wedge \text{Dist}(\varphi, 0), a \cdot K)$	<b>8, TD, Ax7</b>
10. $\text{Futr}(\text{WithinF}(\varphi, a), x)$	<b>9, <math>\exists</math>-intro, WithinF def, 7, Ax6</b>
11. $x = a \cdot K \rightarrow \text{Futr}(\text{WithinF}(\varphi, a), x)$	<b>7 ÷ 10, DED</b>
12. $a \cdot K < x < a \cdot (K + 1)$	<b>Hyp</b>
13. $x = a \cdot K + y \wedge 0 < y < a$	<b>12, TD</b>
14. $\text{Futr}(\text{Futr}(\varphi, a), K \cdot a)$	<b>1, 8, Ax2, MP</b>
15. $\text{Futr}(\text{Lasts}(\text{WithinF}(\varphi, a), a), a \cdot K) \equiv$ $\equiv \text{Futr}(\forall x (0 < x < 3 \rightarrow \text{Futr}(\text{WithinF}(\varphi, 3), x)), 3 \cdot a)$	<b>Th.xi, 14, TT</b>
16. $\text{Futr}(\text{Futr}(\text{WithinF}(\varphi, a), y), a \cdot K)$	<b>13, 15, Ax2</b>
17. $\text{Futr}(\text{WithinF}(\varphi, a), x)$	<b>16, Th.vi, 13, 17, Ax6</b>
18. $0 < x < a \cdot (K + 1) \rightarrow \text{Futr}(\text{WithinF}(\varphi, a), x)$	<b>12 ÷ 17, DED</b>

19.  $x > 0 \rightarrow \text{Futr}(\text{WithinF}(\phi, a), x)$  **4, 6, 11, 18, TAUT**  
 20.  $\text{AlwF}(\text{WithinF}(\phi, a))$  **19, GEN**  
 21.  $\vdash \phi \wedge \text{Alw}(\phi \rightarrow \text{Futr}(\phi, a)) \rightarrow \text{AlwF}(\text{WithinF}(\phi, a))$  **1÷20, DED**

**xii.**  $\phi \wedge \text{Alw}(\phi \rightarrow \text{Futr}(\text{WithinF}(\phi, b), a)) \rightarrow \text{AlwF}(\text{WithinF}(\phi, a+b))$

xii is useful in the analysis of periodicity properties of nondeterministic systems. For instance, for some systems it can be the case that whenever an event  $\phi$  occurs, after  $a$  time units, a further occurrence of  $\phi$  will take place within  $b$  time units. In this case it can be proved that, if  $\phi$  ever takes place, then from that time on  $\phi$  will always repeat itself within at most  $a+b$  time units. This is what xii states. Its proof makes use of property *xiii* as an intermediate lemma, so that, under the hypotheses expressed by the premise of the implication, the following formula can be proved instead.

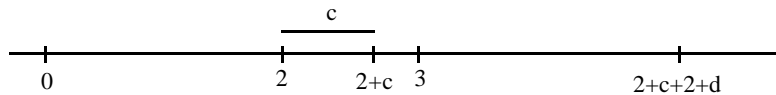
$$\forall k \text{ Futr}(\text{WithinF}(\phi, a+b) \wedge \text{Lasted}(\text{WithinF}(\phi, a+b), a+b), (a+b) \cdot k)$$

In order to make the proof more easily understandable, we will focus on particular values of the involved time intervals, by choosing  $a=2$  and  $b=1$ . As usual, the proof is by induction on the  $k$ .

Base step:  $k=1$ .

- |   |   |
|---|---|
| 1. $\phi, \text{Alw}(\phi \rightarrow \text{Futr}(\text{WithinF}(\phi, 1), 2))$ | <b>Hyp.</b>   |
| 2. $\text{Futr}(\text{WithinF}(\phi, 1), 2)$                                    | <b>1, Ax2, Ax7, MP</b>  |
| 3. $\text{Futr}(0 < c < 1 \wedge \text{Futr}(\phi, c), 2)$                      | <b>2, EI with c new constant</b>  |
| 4. $\text{Futr}(\phi, 2+c) \wedge c+2 < 3$                                      | <b>3, Ax9-10, Ax8, TD</b>   |
| 5. $\text{Lasts}(\text{WithinF}(\phi, 2+c), 2+c)$                               | <b>4, Th xi</b>   |
| 6. $\text{Lasts}(\text{WithinF}(\phi, 3), 2+c)$                                 | <b>5, Th xiv,</b>   |
|   | <b>Th <math>\text{Lasts}(\alpha, a) \wedge \text{Alw}((\alpha \rightarrow \beta)) \rightarrow \text{Lasts}(\beta, a)</math></b> |
| 7. $\text{Futr}(\text{Lasts}(\text{WithinF}(\phi, 3), 2+d), 2+c)$               | <b>1÷6, 4, TT, with d new constant</b>  |

The relative value of the constants 2, 3, c, and d is depicted graphically in the figure below



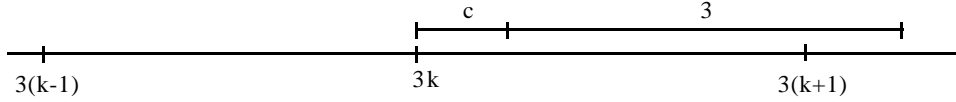
- |  |   |
|--|---|
| 8. $\text{Futr}(\text{Futr}(\text{WithinF}(\phi, 3), 3-2-c), 2+c)$                                       | <b>7, Ax2, MP</b>   |
| 9. $\text{Futr}(\text{WithinF}(\phi, 3), 3)$   | <b>8, Ax8, MP</b>   |
| 10. $0 < x < 3 \rightarrow 0 < x < 2+c \vee x=2+c \vee 2+c < x < 3$                                      | <b>3, TD</b>  |
| 11. $0 < x < 2+c \rightarrow \text{Futr}(\text{WithinF}(\phi, 3), x)$                                    | <b>6, Ax2, MP, DED</b>  |
| 12. $x=2+c \rightarrow \text{Futr}(\phi, x)$   | <b>4, DED</b>   |
| 13. $2+c < x < 3 \rightarrow \text{Futr}(\text{WithinF}(\phi, 3), x)$                                    | <b>7, Ax2, Ax8, MP, DED</b>   |
| 14. $0 < x < 3 \rightarrow \text{Futr}(\text{WithinF}(\phi, 3), x)$                                      | <b>10÷13, TAUT <math>(\alpha \rightarrow (\alpha \vee \beta)) \wedge (\beta \rightarrow (\alpha \vee \beta))</math></b> |
| 15. $\text{Lasts}(\text{WithinF}(\phi, 3), 3)$   | <b>14, GEN</b>  |
| 16. $\text{Futr}(\text{Lasted}(\text{WithinF}(\phi, 3), 3), 3)$  | <b>15, Th <math>\text{Lasts}(\phi, a) \rightarrow \text{Futr}(\text{Lasted}(\phi, a), a)</math></b>                     |
| 17. $\text{Futr}(\text{WithinF}(\phi, 3) \wedge \text{Lasted}(\phi \vee \text{WithinF}(\phi, 3), 3), 3)$ | <b>9, 16, TAUT, MP.</b>   |

Induction step: we prove that if the property holds for the  $k$ -th interval of length 3 then it also holds for the  $(k+1)$ -th interval.

- |   |             |
|---|-------------|
| 1. $\text{Futr}(\text{WithinF}(\phi, 3) \wedge \text{Lasted}(\text{WithinF}(\phi, 3), 3), 3 \cdot k)$ | <b>Hyp.</b> |
|---|-------------|

- |   |                                   |
|---|-----------------------------------|
| 2. $\text{Futr}(\text{WithinF}(\varphi, 3), 3 \cdot k)$   | <b>1, Ax9-10, MP</b>              |
| 3. $\text{Futr}(\varphi, 3 \cdot k)$  | <b>Hyp.</b>                       |
| 4. $\text{Futr}(\text{Futr}(\text{WithinF}(\varphi, 3) \wedge \text{Lasted}(\text{WithinF}(\varphi, 3), 3), 3), 3 \cdot k)$ | <b>proof of the base step, TT</b> |
| 5. $\text{Futr}(\text{WithinF}(\varphi, 3), 3 \cdot k)$   | <b>Hyp.</b>                       |
| 6. $\text{Futr}(0 < c < 3 \wedge \text{Futr}(\varphi, c), 3 \cdot k)$   | <b>5, EI with c new constant</b>  |
| 7. $\text{Futr}(\text{Lasts}(\text{WithinF}(\varphi, 3), c), 3 \cdot k)$  | <b>Th xi, Th xiv, TT</b>          |
| 8. $\text{Futr}(\varphi, 3 \cdot k + c)$  | <b>6, Ax8, MP</b>                 |

Again, for more clarity, the values of the constants involved in the derivation are depicted in the figure below



- |   |                                      |
|---|--------------------------------------|
| 9. $\text{Futr}(\text{Futr}(\text{WithinF}(\varphi, 3) \wedge \text{Lasted}(\text{WithinF}(\varphi, 3), 3), 3), 3 \cdot k + c)$   | <b>8, proof of the base step, TT</b> |
| 10. $\text{Futr}(\text{WithinF}(\varphi, 3), 3 \cdot (k+1))$  | <b>9, 7÷9 in the base step, TT</b>   |
| 11. $\text{Futr}(\text{WithinF}(\varphi, 3) \wedge \text{Lasted}(\text{WithinF}(\varphi, 3), 3), 3 \cdot (k+1))$  | <b>8, 9, 10</b>                      |
| 12. $\text{Futr}(\text{WithinF}(\varphi, 3) \wedge \text{Lasted}(\text{WithinF}(\varphi, 3), 3), 3 \cdot k) \rightarrow \text{Futr}(\text{WithinF}(\varphi, 3) \wedge \text{Lasted}(\text{WithinF}(\varphi, 3), 3), 3 \cdot (k+1))$ | <b>1÷10, DED</b>                     |

This completes the proof by induction; the original formula is then trivially derived by repeated applications of DED and TAUT.

The above proof can be easily adopted to any particular value of constants  $a$  and  $b$ .

**xiii.**  $\forall k \text{ Futr}(\alpha \wedge \text{Lasted}(\alpha, a), a \cdot k) \rightarrow \text{AlwF}(\alpha)$   $k$  being a natural number

Theorem *xiii* follows trivially from the properties of the temporal domain.

Other important properties of the  $\text{WithinF}$  operator, to be used in subsequent sections, are:

**xiv.**  $\text{WithinF}(\varphi, t) \rightarrow \forall s (s > t \rightarrow \text{WithinF}(\varphi, s))$

Intuitively, *xiv* states that if an event occurs in the future within a given timeout then it certainly satisfies any greater timeout

*Proof of xiv.*

- |   |                                      |
|---|--------------------------------------|
| 1. $\text{WithinF}(\varphi, t) \equiv \exists t' (0 \leq t' \leq t \wedge \text{WithinF}(\varphi, t'))$     | <b>Hyp</b>                           |
| 2. $0 \leq a \leq t \wedge \text{Dist}(\varphi, a)$   | <b>1, EI</b>                         |
| 3. $s > t$  | <b>Hyp</b>                           |
| 4. $0 \leq a \leq s \wedge \text{Dist}(\varphi, a)$   | <b>2, 3, TD</b>                      |
| 5. $\text{WithinF}(\varphi, s)$   | <b>4, <math>\exists</math>-intro</b> |
| 6. $s > t \rightarrow \text{WithinF}(\varphi, s)$   | <b>3, 5, DED</b>                     |
| 7. $\forall s (s > t \rightarrow \text{WithinF}(\varphi, s))$   | <b>6, GEN</b>                        |
| 8. $\vdash \text{WithinF}(\varphi, t) \rightarrow \forall s (s > t \rightarrow \text{WithinF}(\varphi, s))$ | <b>1÷7, DED</b>                      |

**xv.**  $\text{Futr}(\text{WithinF}(\varphi, t_1), t_2) \rightarrow \forall t (0 < t < t_2 \rightarrow \text{Futr}(\text{WithinF}(\varphi, t_1 + t), t_2 - t))$

*xv* states that if the start of a timeout period is anticipated then the timeout itself must be extended accordingly.

*Proof of xv.*

1. $\text{Futr}(\text{WithinF}(\varphi, t1), t2) \equiv t2 > 0 \wedge \text{Dist}(\exists s(0 \leq s \leq t1 \wedge \text{Dist}(\varphi, s)), t2)$	<b>Hyp</b>
2. $t2 > 0 \wedge 0 \leq S \leq t1 \wedge \text{Dist}(\varphi, t2+S)$	<b>1, EI, Ax8, Ax9÷10</b>
3. $0 < t < t2$	<b>Hyp</b>
4. $t2-t > 0$	<b>3, TD</b>
5. $t2 = t2-t+t$	<b>TD</b>
6. $\text{Dist}(\varphi, t2-t+t+S)$	<b>2, 5, Ax6</b>
7. $\text{Dist}(\text{Dist}(\varphi, t+S), t2-t)$	<b>6, Ax8</b>
8. $0 \leq t+S \leq t1+t$	<b>2, 3, TD</b>
9. $\text{Dist}(0 \leq t+S \leq t1+t \wedge \text{Dist}(\varphi, t+S), t2-t)$	<b>8, Ax.11, Ax.2, 7</b>
10. $\text{Dist}(\exists v(0 \leq v \leq t1+t \wedge \text{Dist}(\varphi, v)), t2-t) \equiv \text{Dist}(\text{WithinF}(\varphi, t1+t), t2-t)$	<b>9, <math>\exists</math>-intro</b>
11. $\text{Futr}(\text{WithinF}(\varphi, t1+t), t2-t)$	<b>4, 10</b>
12. $0 < t < t2 \rightarrow \text{Futr}(\text{WithinF}(\varphi, t1+t), t2-t)$	<b>3÷11, DED</b>
13. $\forall t(0 < t < t2 \rightarrow \text{Futr}(\text{WithinF}(\varphi, t1+t), t2-t))$	<b>12, GEN</b>
14. $\vdash \text{Futr}(\text{WithinF}(\varphi, t1), t2) \rightarrow \forall t(0 < t < t2 \rightarrow \text{Futr}(\text{WithinF}(\varphi, t1+t), t2-t))$	<b>1÷13, DED</b>

### 2.4.3. Remarks on TRIO's axiomatization and on its soundness

Our axiomatization of the TRIO language follows a fairly standard approach [65, 30, 63] so that it should not need many comments. Perhaps the only point that deserves some special attention is the way we deal with generalization, both with respect to a generic variable and with respect to time in particular.

In general, our proof system differs from that of [65] because it does not include a generalization rule such as  $\frac{\alpha}{\forall x \alpha}$ , where  $x$  is *any* variable. Including this rule into the proof system amounts to interpreting free variables as ranging over all their entire domain and requires special care in the application of the deduction (meta)theorem to avoid inconsistencies. Rather, we prefer to follow an approach similar to that of [30], which considers free variables as having definite, but unknown values.

Similarly, when dealing with temporal aspects, some axiomatizations of classical temporal logic, such as those of [61], and [76], include the following rule, usually called “ $\Box$ -insertion”:

$$\frac{\alpha}{\Box \alpha} \text{ for any formula } \alpha,$$

while others, like that of [92] do not consider it as a valid inference rule.

This inference rule is the temporal counterpart of the above cited generalization rule, since its application corresponds to considering an “open parameter” (the implicit current time instant, in the temporal framework) as ranging over the entire domain (the temporal domain in this case). In the proof system for TRIO, the analogue to this rule would be a rule like  $\frac{\alpha}{\text{Alw}(\alpha)}$  for *any* formula  $\alpha$ . We

chose to omit such a rule in our system, since in our opinion a TRIO formula should intuitively refer

to (and be interpreted at) the current time instant *only*. This is mandatory, for instance, when one wants to describe the initial configuration of a system by means of a formula  $\iota$ , which should assert properties that hold at the current time instant, but by no means maintain their validity in all future time instants. In such a case one typically wants to prove that, if the system ensures at the current time instant the property expressed by formula  $\iota$ , then at all future time instants property  $\pi$  will hold. To this purpose one must derive  $\iota \vdash \text{AlwF}(\pi)$ , while the derivation  $\iota \vdash \text{Alw}(\iota)$  (i.e. if property  $\iota$  holds *now*, then the system always ensures it) is certainly not legitimate<sup>1</sup>.

If the  $\Box$ -introduction rule is included in the axiomatization, the system is not sound, since in that case it is easy to find a formula  $\alpha$  such that  $\alpha \vdash \text{Alw}(\alpha)$  but  $\alpha \not\models \text{Alw}(\alpha)$ . In fact, [61] and [76] state the conditions under which the application of the  $\Box$ -introduction rule is legitimate, and consequently make a distinction among *soundness* and *strong soundness* of a proof system. The latter property corresponds to the notion of soundness as usually meant in mathematical logic, while the former corresponds to its restriction to classes of interpretation structures for which the  $\Box$ -introduction rule is valid. Furthermore, they argue that such a rule allows the derivation of formulas that could not be proved in systems which do not include it, and consequently introduce the notions of *completeness* vs. *strong completeness*, the latter being attainable, in presence of the  $\Box$ -introduction rule, at the price of eschewing the strong notion of soundness in favor of a weaker one.

Instead of following the above approach, we decided to give more relevance to the intuitive evidence of formulas and to the clarity in their derivations. Thus, we followed the line of [30] and [92], and considered open parameters of formulas, including the current interpretation time, as constants of unknown value. As a consequence we easily obtained a proof of soundness of TRIO proof system (in the classical sense, i.e. strong soundness in the terminology of [61] and [76]). This proof is given in Appendix II. We do not deal instead, in this paper, with the issue of completeness. It is clear, however, that in the general case, the axiom system is necessarily incomplete, because of the quantification on variables ranging over the temporal domain which may include full axiomatization of arithmetics. At most, *relative* completeness [8] might be accomplished, by assuming an *oracle* for the arithmetics of the temporal domain.

Before closing this section it is worth comparing the present axiomatization of TRIO with that of MTL in [54]. The main difference among the two systems is that the axiomatization in [54] deals with fundamental properties of any linear temporal structure, in that it embodies every possible metric time structure. The result is a very general and powerful axiom system, which may, however, become rather complex and cumbersome to be effectively used in practice for proving properties of real-life systems. In contrast, the present proof system for TRIO has been kept compact and simple, although admittedly less general, by implicitly assuming, in axioms 7 through 10, properties of completeness and uniformity that can be satisfied by most of the time domains to be considered in practical cases.

---

<sup>1</sup> We will see in Section 3 that this is just the case with Petri nets.

### 3. A TRIO proof system for timed Petri nets

The purpose of the present section is to express the semantics of timed Petri nets through a set of TRIO axioms and, more generally, to use TRIO as a logic framework in which properties of such nets can be expressed and proved. It will turn out that this is not just a trivial exercise, the main reasons for this being the nondeterminism of the Petri net model, the possibility of simultaneous events, and the topological complexity of nets, if one wants to consider the most general case. We present here just *a* possible axiomatization, and we will discuss its limits and possible alternatives (see also the conclusions).

To help focus on the essential aspects of the axiomatization, we make the following assumptions:

1. We restrict our attention to expressing and proving properties that hold for every possible computation of the model. Properties that are referred to as “safety” properties in the literature [81, 76, 58] are of this type. This is consistent with what is done in most other approaches.
2. Among the many timed Petri nets models present in the literature [35] we choose Merlin&Farber’s [66], because it is intuitively simple, and widely known.<sup>1</sup>

The second assumption can easily be removed, thus yielding extensions of our approach to more general time models [35], e.g. we may as well consider general time sets associated with transitions instead of time intervals, or provide a different type of time semantics to the nets, or associate sets of possible values with tokens as it happens in PrT nets [34], in ER nets [39], and in Coloured Petri nets [52]. Instead, dealing with other features of nets usually classified such as liveness properties, or reachability, would require a different kind of axiomatization. A thorough discussion on different types of properties is given in Section 5. Furthermore, possible ways towards both types of generalization will be discussed in Section 6.

The rest of this section is organized as follows. Section 3.1 informally discusses the semantics of timed Petri nets. Section 3.2 presents our axiomatization of timed Petri nets. Section 3.3 provides a first set of simple properties proved by making use of this axiomatization. Section 3.4 discusses some critical issues and Section 3.5 provides simplified axioms that exploit restricting hypotheses such as boundedness.

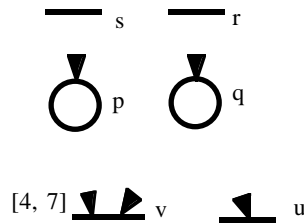
#### 3.1 The informal semantics of timed Petri nets

As we said, we refer mainly to the Merlin and Farber model [66] but we also discuss possible alternative semantic choices and we point out critical spots. We assume some familiarity with the literature on classical–non timed–Petri nets [79]. For the sake of generality, unbounded accumulation of token in places is not prevented. A thorough discussion and comparison of several ways to describe timing aspects in Petri nets is contained in [39]. That paper also provides a generalization of all previous approaches and a full operational formalization thereof.

---

<sup>1</sup> In the following presentation we also assume that there is at most one single arc between a place and a transition. This is not a real restriction, since there are well known systematic equivalence-preserving transformations to eliminate multiple arcs from Petri nets.

In a *timed Petri net* (TPN) every transition  $v$  is associated with a pair of values  $[m_v, M_v]$ , belonging to the temporal domain (with  $0 \leq m_v \leq M_v \leq \infty$ ). They are called, respectively, the *lower and upper bound* of  $v$ , whereas the pair  $[m_v, M_v]$  is called  $v$ 's *time interval*. Intuitively, the meaning of the pair  $[m_v, M_v]$  is that, once  $v$  is enabled by the presence of at least one token in each place of its preset, it *can not fire before a time  $m_v$  elapsed* (in the sequel we will call this property LB, since it originates from the lowerbound of  $v$ ) and it must fire within  $M_v$  unless in the meanwhile it is disabled by the possible firing of another transition that is in conflict with it (this property will be referred to by UB, since it refers to the upperbound of  $v$ ). For instance, consider the net of Figure 3.1



**Figure 3.1** A portion of a timed Petri net.

If transition  $s$  fires at time 0 and  $r$  fires at time 5, then  $v$  cannot fire before time 9. Furthermore, if within time 12 neither  $v$  nor  $u$  fired, then  $v$  must fire (unless  $u$  fires exactly at that time.)

The above rule is the usual interpretation of TPNs. In [39] it is called *strong time semantics* (STS) because it forces a transition to fire after its upperbound elapsed. In [39] it is also noted, however, that this semantics is inconsistent with traditional Petri nets semantics, where a transition is never forced to fire. For this reason, a new semantics is proposed besides STS, that is called *weak time semantics* (WTS). In WTS a transition is never forced to fire, but it is stated that, if it fires, it must fire in a time included in the interval  $[m_v, M_v]$ . For instance, suppose that in the net of Figure 3.1 an upperbound of 3 is associated with  $u$ . Then, in STS  $v$  could never fire; instead, in WTS it could happen that  $v$  fires—after at least 4 time units since its enabling—because it would not be imposed that  $u$  fired before (in that case  $u$  could not fire anymore by using the token that enabled it for more than 3 time units. In [39] it is argued that both semantics may be useful in different contexts and thus both of them are pursued. Here, however, we will mainly focus on STS which is more widely adopted, leaving WTS for a few side remarks.

We point out that tokens are *uniquely* generated and consumed by transition firings. In particular any firing of a transition consumes one and only one distinct token from each place in its preset (we call this property IU, input unicity), and introduces one and only one token into each place of its postset; that token can contribute to no more than a single transition firing (we call this property OU, output unicity).

We now illustrate some consequences of these definitions on a few examples, in order to point out some aspects of the semantics of timed Petri nets that are often left implicit or overlooked. For this discussion we will refer to the portions of Petri nets described in Figure 3.2.

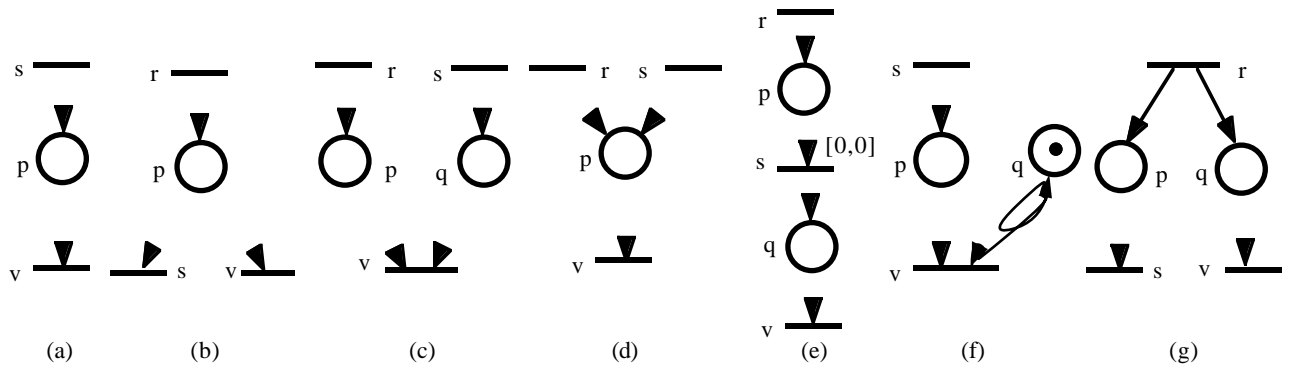


Figure 3.2. A sample of Petri nets fragments.

- Simultaneous firings.** Unlike traditional Petri nets, in the timed version time is treated explicitly. Hence, the notion of simultaneity of events becomes relevant, and any axiomatization of the model should describe this feature in a natural and simple way. Contemporary transition firings may occur in TPNs in two, deeply different, ways:
  - *Simultaneous and concurrent firings.* This case is exemplified in Figure 3.2(g). Assume that both  $s$  and  $v$  have  $m_v = M_v = 3$ . Then, whenever  $r$  fires,  $s$  and  $v$  will *both* fire exactly 3 time units later. It is clear that, in general, they *could fire contemporarily* if and only if the intersection between their associated time intervals is not empty. A special case of simultaneous and concurrent firing will be examined shortly (meaning of the lowerbound).
  - *Simultaneous but logically ordered firings (zero-time transitions).* In general, it is not excluded that, for some transition,  $m_v$ —and possibly even  $M_v$ —are equal to 0. Without going into the philosophical issue of the meaning of an event that takes no time to occur, we simply observe that this abstraction is useful whenever the duration of the described event is negligible with respect to the other time constants of the net and that it is widely adopted in the literature on TPNs. To illustrate the consequences of this assumption, consider the fragment of Figure 3.2(e). In this case, whenever  $r$  fires,  $s$  fires *immediately* too, although the intuition associated with the usual interpretation of Petri nets would lead to say that  $s$  always fires *after*  $r$ . Thus, we are compelled to *clearly distinguish between logical ordering and temporal ordering*, although it is obvious that an event  $s$  that is the logical consequence of an event  $r$  *cannot precede*  $r$ , it is not implied that  $s$  *strictly* follows  $r$  in time.
- Meaning of the lowerbound.** Assume that in the net of Figure 3.2(a)  $m_v = M_v = 3$ . Then, if transition  $s$  fires in two instants that are 1 time units apart, so does transition  $v$ , with the same delay. This semantics could not correspond to an intuitive interpretation of transition firing as executing some action that takes a given time. For instance,  $v$  could be interpreted as executing some algorithm on a datum that is modeled by the token produced in  $p$ . In such a case, if  $m_v$  gives the minimum time that is necessary to execute the algorithm, it is clearly wrong assuming that two consecutive computations may occur at a distance of 1 time unit. This interpretation of transition lowerbound is sometimes called the “recharge time”. Although there is no evidence that one of the

two interpretations is preferable to the other one, we chose the former because it is more general, since the latter can be easily simulated by the former by simply adding a new place—with a single token assigned to it in the initial marking—that is both in the preset and in the postset of the considered transition. For instance, Figure 3.2(f) shows how the fragment of Figure 3.2(a) should be transformed if one wants that two consecutive firings thereof never occur in a time interval that is shorter than  $m_v$ .

As a side remark, notice that this semantics of the lowerbound may introduce another type of *simultaneous and concurrent firing of the same transition*. An example thereof is supplied by the fragment of Figure 3.2(d), assuming that  $r$  and  $s$  fire simultaneously and that  $m_v = M_v = 3$ .

- **Transitions with empty preset.** In non timed Petri nets a transition  $v$  with no input places is conventionally considered as always enabled. This would allow an extension of the semantics to the timed case where, once the lowerbound  $m_v$  elapsed after the initial time,  $v$  may fire an unlimited number of times at any time. This interpretation is clearly of no use in most cases since it reduces the meaning of  $m_v$  to that of an initial delay with no impact in the subsequent behavior of the net. Thus, we assume that such a transition cannot fire twice consecutively in less than  $m_v$  time units. If one wished to make this convention more explicit, it would suffice to add  $v$  a new place that is both input and output thereof in much the same way as in Figure 3.2(f).

If in exceptional cases the other interpretation is preferred, this can be dealt with in several ways that are left to the reader.

- **Infinite upperbounds.** If the upper bound time value  $M_v$  of a transition  $v$  is equal to  $\infty$ , then a behavior in which  $v$  never fires, and tokens entering the places of its preset stay there forever is admissible for the net. This is consistent with the traditional assumption that TPNs reduce to classical Petri nets when, for every  $v$ ,  $m_v = 0$  and  $M_v = \infty$  [58].

If one looks at the possible intuitive meanings of the time interval  $[m_v, \infty]$ , however, this can also be seen as “the limit for  $M_v \rightarrow \infty$  of  $[m_v, M_v]$ ”. Since in STS this means that  $v$  *must* fire in a time  $t$ , such that  $m_v \leq t \leq M_v$  unless previously disabled, we could deduce that the meaning of  $[m_v, \infty]$  is that  $v$  *must* fire in a time  $t < \infty$ , what is different from saying that the semantics of the net is the same as in non timed Petri nets, where an enabled transition could *never* fire. In other words a *fairness* hypothesis could be assumed for the firing of enabled transitions (a transition cannot remain enabled for an infinite time without firing). In our opinion, the new interpretation is more consistent with STS, whereas the traditional one is more consistent with WTS. Since we decided, however, to stay as close as possible to the most widely known literature on TPNs, we will assume the original semantics for the interval  $[m_v, \infty]$ . We propose instead the use of the symbol  $\diamond$  to denote an unbounded finite value, i.e., the fact that the transition cannot remain enabled for an infinite time without firing<sup>1</sup>.

---

<sup>1</sup> From the point of view of the theory of computation it must be noticed that the  $\diamond$  symbol denotes *any* finite value in an infinite domain, which is uncomputable. This should not prevent, however, from using it in *specifications*, by knowing a priori that it can only be *implied* by possible implementations (e.g., an implementation that guarantees the firing within a fixed time satisfies, but is not equivalent to, the requirement of guaranteeing the firing within  $\diamond$ ).

### 3.2 The axiomatization

We now introduce the formal description of timed Petri nets by means of TRIO axioms, according to the above informal report of their semantics. First, we give axioms describing general properties that are independent from the topology of the net; then, we present axioms that describe how the topology of the net and time bounds associated with the transitions determine its behavior.

In the rest of the paper, it is intended that all axioms describing Petri nets semantics are temporally universally quantified by an implicit  $Alw$  operator, and that all their free variables are implicitly universally quantified.

#### 3.2.1 Basic predicates and general axioms

We stated that the semantics of timed Petri nets admits multiple simultaneous firings of a single transition. Accordingly, we define a time dependent predicate  $nFire(v, n)$  whose meaning is that at the current time instant transition  $v$  fires  $n$  times ( $n = 0$  means that  $v$  does not fire).

It will also be useful to refer individually to the  $i$ -th firing of a transition: for this purpose we define a predicate  $fireth(v, i)$ : its meaning is that there is an  $i$ -th firing of transition  $v$  at the current time. Since the intended meaning of  $fireth(v, i)$  is that  $v$  does fire, we will impose that  $i$  is  $> 0$ . Clearly, the predicates  $nFire$  and  $fireth$  are closely related, as expressed by the following axioms.

1.  $fireth(v, i)$  is true for all and only those values of  $i$  that are less than, or equal to, the actual total number of current firings of transition  $v$ .

$$FR(v) \text{ (Firing of } v\text{):} \quad fireth(v, i) \leftrightarrow (\exists n(n \geq i \wedge nFire(v, n)) \wedge (i > 0))$$

2. The number of transition firings in a given time instant is unique.

$$UFN(v) \text{ (Unique firing number of } v\text{):} \quad \neg \exists m \exists n (m \neq n \wedge nFire(v, m) \wedge nFire(v, n))$$

3. There always exists a nonnegative number of firings for each transition.

$$NNF(v) \text{ (Nonnegative firing of } v\text{):} \quad \exists n (n \geq 0 \wedge nFire(v, n))$$

Unlike traditional Petri nets, the instantaneous marking of a TPN does not adequately characterize its overall state: the duration of the token staying in places is significant too. This has been realized in several ways in the literature: for instance, in [39] the temporal aspects of timed Petri nets are described by assigning to each token a value representing the absolute time instant of its production. In TRIO, reference to absolute time is intentionally avoided since it is considered inappropriate for the description of time invariant systems. Thus, in order to uniquely identify tokens, we refer to the events of their production and consumption (i.e., to the firings of transitions), and to the places where they are inserted, or from which they are deleted. The temporal semantics of the nets will be ultimately provided by imposing constraints on the distance in time among transition firings. On the contrary, the notion of *instantaneous marking* will be formalized as a *derived concept* on the basis of transition firing axioms.

The predicate  $tokenF(s, i, p, v, j, d)$  states that the token produced at the current instant by the  $i$ -th firing of transition  $s$  enters place  $p$  and will be consumed by the  $j$ -th firing of transition  $v$  after  $d$  time units. Of course, this implies that place  $p$  is in the postset of transition  $s$  and in the preset of transition

$v$ . We do not rule out the possibility that other transitions may be input to place  $p$  (and may even fire) and  $v$  may be output to other places besides  $p$ .

$tokenF$  is the key predicate in the present axiomatization of TPNs, since each one of its occurrences uniquely identifies a single token produced and consumed in the net.

$tokenF$  is asserted at the time instant of the firing of the first transition argument,  $s$ . For reasons of simplicity and symmetry of the formulas, it is useful to introduce a dual predicate,  $tokenP$ , which has the same meaning but refers to the time of firing of the second transition, i.e., the one that is output to place  $p$ . Thus predicates  $tokenF$  and  $tokenP$  are related by the following axioms.

4 . FP (Future to Past):  $tokenF(r, i, p, s, j, d) \leftrightarrow Futr(tokenP(r, i, p, s, j, d), d)$

5 . PF (Past to Future):  $tokenP(r, i, p, s, j, d) \leftrightarrow Past(tokenF(r, i, p, s, j, d), d)$

$tokenF$  and  $tokenP$  necessarily imply the firing of the involved transitions, as expressed by the following axioms.

6 . FI (Future Implication):  $tokenF(r, i, p, s, j, d) \rightarrow fireth(r, i) \wedge Futr(fireth(s, j), d)$

7 . PI (Past Implication):  $tokenP(r, i, p, s, j, d) \rightarrow fireth(s, j) \wedge Past(fireth(r, i), d)$

The above axioms state the only properties that hold independently from the related transitions and from the net topology. Other properties are specified by means of axioms that differ from case to case.

Because of the presence of six arguments, the  $token$  predicates may be regarded as exceedingly complex; nevertheless, all their arguments are necessary to allow the description of nets in their full generality. Specifically, the indexes  $i$  and  $j$  referring to transition firings are necessary for the description of simultaneous firings when the described net is not 1-bounded, and the name of the place laying between the two transitions is necessary for the description of complex topologies. It will be apparent from the succeeding examples that some of these arguments can be omitted in many relevant particular cases, thus permitting a significant reduction of complexity of the axiomatization. This will be exploited in Section 3.5.

### 3.2.2 Topology-dependent axioms

We now illustrate the axiomatization of the behavior of TPNs as determined by the net topology and the time bounds associated with the transitions. In order to make the presentation more intuitive and understandable, we will refer to the simple net fragments of Figure 3.2, that were used as examples in the foregoing informal account of the semantics. They are general enough, however, to cover most cases of normal use, such as the examples discussed in Section 4. In Appendix III, a more general scheme to derive axioms from the topology and the time intervals associated with the transitions will be presented. It will appear that such a scheme can be made even algorithmic.

#### *Lowerbound axioms*

The axiomatization of the LB property—which is simpler than the UB property—will be exemplified only for the net fragment of Figure 3.2(d). In this case LB prescribes that when transition  $v$  fires, it must consume a token produced by a preceding firing of transition  $r$  or by one of transition  $s$  no less than  $m_v$  time units before.

$$8. \quad \text{LB}(v): \text{ fireth}(v, i) \rightarrow \exists d \left( d \geq m_v \wedge \left( \begin{array}{c} \exists j \text{ tokenP}(r, j, p, v, i, d) \\ \vee \\ \exists j \text{ tokenP}(s, j, p, v, i, d) \end{array} \right) \right)$$

This axiom can be easily generalized to the case when the preset of the transition includes several places.

### *Upperbound axioms*

We now turn to the axiomatization of the UB property: this will be illustrated by referring to three different examples, those of Figure 3.2(c), 3.2(d), and 3.2(b), because it takes different forms (which can however be reduced to a single general scheme), depending on the topology of the described net. We assume here that for any transition  $v$ , the upperbound time has a finite value: the cases when  $M_v$  is equal to ' $\infty$ ' or to the special value ' $\diamond$ ' informally introduced in Section 3.1 deserve a separate discussion, to be provided in the final part of this section.

**9.** For the net fragment of Figure 3.2(c) the UB property may be intuitively formulated as follows: it can never be the case that two tokens stay in places  $p$  and  $q$  for more than  $M_v$  time units. More precisely, it can never happen that a token has been produced by a firing of transition  $r$  and one has been produced by  $s$ , both more than  $M_v$  time units ago, and none of them has been consumed by a firing of transition  $v$ . This is expressed in terms of a TRIO axiom as follows.

UB.c (Upperbound related to Figure 3.2(c)):

$$\neg \left( \begin{array}{c} (d_r \geq M_v \wedge \text{Past}(\text{fireth}(r, i), d_r)) \wedge \neg \exists d_v (d_v \leq d_r \wedge \exists k \text{ Past}(\text{tokenP}(r, i, p, v, k, d_r - d_v), d_v)) \\ \wedge \\ (d_s \geq M_v \wedge \text{Past}(\text{fireth}(s, j), d_s)) \wedge \neg \exists d_v (d_v \leq d_s \wedge \exists k \text{ Past}(\text{tokenP}(s, j, p, v, k, d_s - d_v), d_v)) \end{array} \right)$$

or, equivalently:

UB.c':

$$\left( \begin{array}{c} d_r \geq M_v \wedge \text{Past}(\text{fireth}(r, i), d_r) \\ \wedge \\ d_s \geq M_v \wedge \text{Past}(\text{fireth}(s, j), d_s) \end{array} \right) \rightarrow \left( \begin{array}{c} \exists d_v (d_v \leq d_r \wedge \exists k \text{ Past}(\text{tokenP}(r, i, p, v, k, d_r - d_v), d_v)) \\ \vee \\ \exists d_v (d_v \leq d_s \wedge \exists k \text{ Past}(\text{tokenP}(s, j, p, v, k, d_s - d_v), d_v)) \end{array} \right)$$

**10.** For the fragment of Figure 2(d) the UB axiom is the following, fairly natural, modification of UB.c.

UB.d (Upperbound related to Figure 3.2(d)):

$$\left( \begin{array}{c} (d_r \geq M_v \wedge \text{Past}(\text{fireth}(r, i), d_r)) \rightarrow (\exists d_v (d_v \leq d_r \wedge \exists k \text{ Past}(\text{tokenP}(r, i, p, v, k, d_r - d_v), d_v))) \\ \wedge \\ (d_s \geq M_v \wedge \text{Past}(\text{fireth}(s, j), d_s)) \rightarrow (\exists d_v (d_v \leq d_s \wedge \exists k \text{ Past}(\text{tokenP}(s, j, p, v, k, d_s - d_v), d_v))) \end{array} \right)$$

The two conjuncts of UB.d are called UB.d( $r, v$ ) and UB.d( $s, v$ ) since they relate the firings of transitions  $r$  and  $v$  and  $s$  and  $v$ , respectively. They are asserted at (i.e., they implicitly refer to) a time different from the firing times of both transitions, for similarity with UB.c. They take, however, a

simpler and more intuitive form if asserted at the time of the firing of transitions  $r$  and  $s$ , as shown below.

$$\text{UB.d}(r, v)': \quad \text{fireth}(r, i) \rightarrow \exists d (d \leq M_v \wedge \exists k \text{ tokenF}(r, i, p, v, k, d))$$

$$\text{UB.d}(s, v)': \quad \text{fireth}(s, j) \rightarrow \exists d (d \leq M_v \wedge \exists k \text{ tokenF}(s, j, p, v, k, d))$$

**11.** The last instance of UB axiomatization refers to the net of Figure 3.2(b). There are now two conflicting transitions,  $s$  and  $v$ . Thus, the axiomatization of UB must impose the firing of *either*  $s$  or  $v$ . We give two separate axioms for the two transitions.

$$\text{UB.b}(s): \quad \text{fireth}(r, i) \rightarrow \exists d \left( d \leq M_s \wedge \left( \begin{array}{c} \exists j \text{ tokenF}(r, i, p, s, j, d) \\ \vee \\ \exists j \text{ tokenF}(r, i, p, v, j, d) \end{array} \right) \right)$$

$$\text{UB.b}(v): \quad \text{fireth}(r, i) \rightarrow \exists d \left( d \leq M_v \wedge \left( \begin{array}{c} \exists j \text{ tokenF}(r, i, p, s, j, d) \\ \vee \\ \exists j \text{ tokenF}(r, i, p, v, j, d) \end{array} \right) \right)$$

These axioms have a form that is similar to the previous UB axioms, but they take into account the fact that transitions  $s$  and  $v$  share the input place  $p$ .

The reader has probably noticed that  $\text{UB.b}(s)$  and  $\text{UB.b}(v)$  could be easily unified by substituting both  $M_s$  and  $M_v$  with the minimum between the two. We chose however the formulation above since it is more general and can therefore be extended more easily to other more complex net topologies as it will be illustrated in Appendix III.

#### *Unicity axioms*

The property of output unicity, stating that any transition firing introduces one and only one distinct token in each place of the postset, can be expressed in the typical form of unicity axioms. Similar remarks hold for input unicity.

With reference to the net of Figure 3.2(b) output and input unicity axioms take the following form.

$$\mathbf{12.OU}(r) \text{ (Output Unicity): } \text{tokenF}(r, i, p, t1, j, d) \wedge \text{tokenF}(r, i, p, t2, k, e) \rightarrow t1=t2 \wedge j=k \wedge d=e$$

$\text{OU}(r)$  is useful in avoiding inconsistent derivations in presence of a conflict among transitions sharing an input place, since one given token cannot be consumed by two distinct firings.

$$\mathbf{13.IU}(s) \text{ (Input Unicity for } s\text{): } \text{tokenP}(r, i, p, s, j, d) \wedge \text{tokenP}(r, k, p, s, j, e) \rightarrow i=k \wedge d=e$$

$$\text{IU}(v) \text{ (Input Unicity for } v\text{): } \text{tokenP}(r, i, p, v, j, d) \wedge \text{tokenP}(r, k, p, v, j, e) \rightarrow i=k \wedge d=e$$

#### *Marking axioms*

The predicate  $\text{marked}(p, n)$  describes the fact that place  $p$  currently contains  $n$  tokens. In our approach, the evolution of the net is axiomatized in terms of transition firings and token production/consumption. Hence, the notion of marking is essentially a derived one.

Two fundamental axioms regarding the predicate  $\text{marked}$  state the conditions under which the marking of a given place remains unchanged or is modified. These axioms, together with the description of the initial marking—to be given next—and the axiomatization of the firing behavior of the

net can uniquely determine the number of tokens in all places during the evolution of the net. For simplicity, the marking axioms will be illustrated on the simple net of Figure 3.2(d) but it will be evident that the generalization for an arbitrary number of input and output transitions to the place is straightforward.

**14.** The first axiom, called *Marking Invariance*, states that the marking of place  $p$  does not change except when a suitable combination of firings of transitions  $r, s, v$  occurs. Since we admit the possibility of multiple simultaneous firings for both one given transition and for distinct transitions, the difference between the total number of firings of input and output transitions must be considered.

MI(p) (Marking Invariance of p):

$$\text{marked}(p, n) \rightarrow \text{Until}_w \left( \text{marked}(p, n), \exists i \exists j \exists k \left( i+j \neq k \wedge \left( \begin{array}{c} \text{nFire}(r, i) \\ \wedge \\ \text{nFire}(s, j) \\ \wedge \\ \text{nFire}(v, k) \end{array} \right) \right) \right)$$

**15.** The second axiom, called *Marking Change*, asserts that the marking changes from the moment (included) of the firing of transitions that generate it.

MC(p) (Marking Change):

$$\text{UpToNow}(\text{marked}(p, n)) \wedge \left( \begin{array}{c} \text{nFire}(r, i) \\ \wedge \\ \text{nFire}(s, j) \\ \wedge \\ \text{nFire}(v, k) \end{array} \right) \wedge i+j \neq k \rightarrow \text{marked}(p, n+i+j-k)$$

### *Initial marking axioms*

Axioms MI and MC specify only the variations in the number of tokens included in the different places, not its absolute values. The actual marking may be different for distinct executions of the same net, depending on the *initial marking*, i.e. the state from which the evolution of the net is assumed to start. Consequently the initial marking is described by means of separate, additional axioms that are not asserted for all execution sequences of the net, but only for those starting from the described initial configuration.

When describing the initial state of a net, it is not sufficient to assign the number of tokens that are present at a given time in the places of the net: their production time must also be indicated. We present here a solution that we prefer to others for reasons of simplicity and practical usefulness. It is not the only possible solution but it is general enough to cope with the description of the initial markings of TPNs in the great majority of real-life cases.

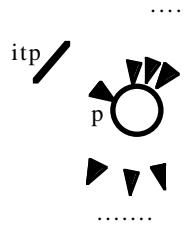
In order to axiomatize the initial marking of the net without invalidating previous axioms and without requiring any preceding firing of transitions in the net we assume, as suggested in Figure 3.3, that, for each place  $p$  in the net, there is one extra transition  $itp$  (*initializing transition for p*) that is input only to  $p$  and is output to no place; we also assume that  $itp$  time lowerbound is 0 so that it may fire any number of times at any instant. Then, we describe the initial marking of  $p$  in terms of a given

set of firings of  $itp$ . We write  $initialMarking(p, k)$  to assert that in the initial state place  $p$  contains  $k$  newly created tokens.

**16.**  $initialMarking(p, k)$  must satisfy the following axiom, where  $v_i, i = 1..k$ , denote any transition other than  $itp$  that is either input or output of  $p$ .

IM(p) (Initial Marking of p):

$$initialMarking(p, k) \leftrightarrow \left( \begin{array}{c} \text{marked}(p, k) \wedge \text{nFire}(itp, k) \wedge \\ \text{AlwP}(\text{nFire}(itp, 0)) \wedge \text{AlwF}(\text{nFire}(itp, 0)) \wedge \bigwedge_{i=1}^k \text{AlwP}(\text{nFire}(v_i, 0)) \end{array} \right)$$



**Figure 3.3.** The dummy transition that “builds” the initial marking.

IM(p) states that  $itp$  fires exactly  $k$  times at the current instant and does not fire in any other instant either in the past or in the future. No other transition of the net must fire before the instant at which IM is asserted.

More complex—but of questionable practical usefulness—cases, such as the presence of tokens of different age in a place, can be easily modelled by suitable variations of IM.

Then, a formula  $\iota$  describing the initial configuration of a net is defined as

$$\iota \stackrel{\text{def}}{=} \bigwedge_j initialMarking(p_j, k_j)$$

In order to prove that the specified initial marking ensures a property  $\pi$  of the net, the theorem  $\iota \vdash \pi$  (or, equivalently  $\vdash \iota \rightarrow \pi$ ) must be derived in the proof system.

### *Infinite and unbounded upperbounds*

Let us now consider the case that  $M_v = \infty$ . For simplicity, we will do this with reference to the net fragment of Figure 3.2(a), since the extension to more general configurations is straightforward.

It is clear that the UB axioms for Figure 3.2(a) can be obtained from that of Figure 3.2(d) by erasing one of the two conjuncts, i.e.:

$$\text{UB.a}(v): \text{fireth}(s, i) \rightarrow \exists d (d \leq M_v \wedge \exists k \text{ tokenF}(s, i, p, v, k, d))$$

If in UB.a we simply replace  $M_v$  by  $\infty$ , we obtain

$$\mathbf{17.} \quad (\bullet) \quad \text{fireth}(s, i) \rightarrow \exists d \exists j \text{ tokenF}(s, i, p, v, j, d).$$

A simple analysis immediately shows, however, that  $(\bullet)$  does not correspond to the traditional meaning of an infinite upperbound in TPNs, since it forces  $v$  to fire, though with no temporal bound. This confirms our previous claim that the traditional semantics of TPNs for infinite time upperbound is more consistent with weak time semantics than with strong time semantics.

Instead,  $(\bullet)$  is perfectly adequate to describe the semantics of the upperbound  $\diamond$  which was proposed informally in Section 3.1. Thus we will refer to  $(\bullet)$  as the  $UB_{\diamond}.a(v)$  axiom.

On the contrary, the conventional semantics of  $M_v = \infty$  does not require any axiom at all for the UB property since this allows, after a firing of  $s$ , both  $v$  firing—at any time—and the fact that  $v$  never fires.

### *Weak time semantics*

The previous discussion on infinite and unbounded upperbounds can also be used to give a few hints on the axiomatization of WTS. Again, let us refer simply to the fragment of Figure 3.2(a). Intuitively, the WTS of the net only states that, if  $v$  fires, then it must consume a token that has been produced by the firing of  $s$  in a past instant that is included within the interval  $[m_v, M_v]$ . This is formalized by the following axiom, which unifies LB and UB properties.

$$\text{WTS.a: } \text{fireth}(v, i) \rightarrow \exists d (m_v \leq d \leq M_v \wedge \exists k \text{ tokenP}(s, k, p, v, i, d))$$

On the other hand the firing of  $s$  has no implication at all, since, after its firing, it may happen both that  $v$  fires (within  $[m_v, M_v]$ ) and that it never fires (to consume the token produced by  $s$ ).

### 3.3 Examples

In this section we provide a first set of simple examples of application of the axioms given in Section 3.2 to the proof of properties of given TPNs. As in Section 2.4.2 the purpose of these examples is both to illustrate the use of the axiom system and to set up a useful set of “lemmas” that will be exploited for the derivation of more complex proofs. Again, we will fully report only the proofs of some more interesting cases, whereas other properties will be simply claimed as theorems. For future use, theorems are listed as TPN.i.

**1.** The first theorem refers to the minimum time elapsing since a token is produced till it is consumed. For any pair of transitions  $s$  and  $v$  connected via a place  $p$  the following property holds.

$$\text{TPN.1: } \text{tokenF}(s, i, p, v, j, d) \rightarrow m_v \leq d$$

We present the proof considering the net of Figure 3.2(d), its generalization using the axioms presented in Appendix III is straightforward. First, we prove the following lemma:

$$\text{LPN.1 : } \text{tokenP}(s, i, p, v, j, d) \rightarrow m_v \leq d$$

*Proof of LPN.1:*

1. $\text{tokenP}(s, i, p, v, j, d)$	<b>Hyp</b>
2. $\text{fireth}(v, j)$	<b>1, PI</b>
3. $\exists e (m_v \leq e \wedge (\exists k (\text{tokenP}(r, k, p, v, j, e)) \vee \exists k (\text{tokenP}(s, k, p, v, j, e))))$	<b>2, LB(v)</b>
4. $\exists e (m_v \leq e \wedge \exists k (r = s \wedge i = k \wedge d = e))$	<b>1, 3, IU(v), MP</b>
5. $m_v \leq d$	<b>4, Ax6</b>
6. $\text{tokenP}(s, i, p, v, j, d) \rightarrow m_v \leq d$	<b>1+5, DED</b>

*Proof of TPN.1:*

1. $\text{tokenF}(s,i,p,v,j,d)$	<b>Hyp</b>
2. $\text{Futr}(\text{tokenP}(s,i,p,v,j,d),d)$	<b>1, F P</b>
3. $\text{Futr}(m_v \leq d, d)$	<b>2, LPN.1, Ax.9, MP</b>
4. $m_v \leq d$	<b>3, Th. iv</b>
5. $\text{tokenF}(s,i,p,v,j,d) \rightarrow m_v \leq d$	<b>1÷4, DED</b>

2. Consider again the net of Figure 3.2(a), where  $v$ 's time intervals is  $[1, 3]$ . Then, the following theorem states that, whenever  $s$  fires,  $v$  will fire in time that is included in the interval  $[1, 3]$ .

**TPN.2:**  $\text{fireth}(s,i) \rightarrow \exists d(1 \leq d \leq 3 \wedge \text{Futr}(\exists j \text{ fireth}(v,j),d))$

*Proof:*

1. $\text{fireth}(s,i)$	<b>Hyp.</b>
2. $\exists d( d \leq 3 \wedge \exists j \text{ tokenF}(s,i,p,v,j,d))$	<b>1, UB(v), MP</b>
3. $D \leq 3 \wedge \exists j \text{ tokenF}(s,i,p,v,j,D)$	<b>2, EI(d)</b>
4. $1 \leq D$	<b>3, TPN.1, MP</b>
5. $1 \leq D \leq 3 \wedge \exists j \text{ Futr}(\text{fireth}(v,j),D)$	<b>3, 4, FI, TD</b>
6. $1 \leq D \leq 3 \wedge \text{Futr}(\exists j \text{ fireth}(v,j),D)$	<b>5, Th. vii</b>
7. $\exists d( 1 \leq d \leq 3 \wedge \text{Futr}(\exists j \text{ fireth}(v,j),d))$	<b>6, <math>\exists</math>-intro</b>
8. $\text{fireth}(s,i) \rightarrow \exists d( 1 \leq d \leq 3 \wedge \text{Futr}(\exists j \text{ fireth}(v,j),d))$	<b>1÷7, DED</b>

3. Consider the net of Figure 3.2(b), with  $[m_s, M_s] = [1, 3]$  and  $[m_v, M_v] = [4, 7]$ . Then, the following theorem states that  $v$  will never fire.

**TPN.3:**  $\text{Alw}(\neg \exists i \text{ fireth}(v,i))$

*Proof.*

By TG, TAUT, and GEN, it suffices to derive  $\neg \text{fireth}(v, i)$ . This will be done by contradiction.

1. $\text{fireth}(v,i)$	<b>Hyp.</b>
2. $\exists d( d \geq 4 \wedge \exists j \text{ tokenP}(r,j,p,v,i,d))$	<b>1, LB(v)</b>
3. $\exists d( d \geq 4 \wedge \text{tokenP}(r,J,p,v,i,d))$	<b>2, EI</b>
4. $\exists d( d \geq 4 \wedge \text{Past}(\text{tokenF}(r,J,p,v,i,d), d) )$	<b>3, PF</b>
5. $\exists d( d \geq 4 \wedge \text{Past}(\text{fireth}(r,J), d) )$	<b>3, PI</b>
6. $\exists d( d \geq 4 \wedge \text{Past}(\exists e(0 \leq e \leq 3 \wedge (\exists k \text{ tokenF}(r, J, p, s, k, e) \vee \exists k \text{ tokenF}(r,J,P,v,k,e))), d))$	<b>5, UB(s)</b>
7. $\exists d( d \geq 4 \wedge \text{Past}(\exists e(0 \leq e \leq 3 \wedge \exists k(\text{tokenF}(r, J, p, v, k, e) \wedge k=i \wedge d=e)), d))$	<b>4, 6, OU(r)</b>
8. $\exists d( d \geq 4 \wedge \exists e(0 \leq e \leq 3 \wedge d=e)$	<b>7, Th vii, Th. iv</b>
9. $\neg \text{fireth}(v, i)$	<b>1÷8, by contradiction, since 8 is false</b>

A direct consequence of TPN.3 and axiom UB(s) is the following corollary, that asserts that the time bounds have resolved the original non-determinism of the net shown in Figure 3.2(b).

**TPN.4:**  $\text{fireth}(r,i) \rightarrow \exists d(1 \leq d \leq 3 \wedge \text{Futr}(\exists j \text{ fireth}(s,j),d))$

### 3.4 Assessment of the proposed axiomatization

Let us now briefly comment on the axiomatization of TPNs that has been proposed in Section 3.2.

A more naive and seemingly simpler axiomatization of timed Petri nets could be based on the predicate  $marked(p, n)$  as a fundamental predicate describing the state of the net in terms of the number  $n$  of tokens currently present in each place  $p$  of the net. Such a predicate could be used to express the enabling conditions for transitions and the consequences of their firings. This kind of axiomatization would be perhaps more appealing from an intuitive point of view, mostly because in traditional Petri nets the behavior of the net can be characterized by referring exclusively to the current marking, independently from any previous transition firings, and considering all tokens residing in a place at a certain step of the computation as indistinguishable.

This kind of axiomatization, however, would suffer from a fundamental flaw: in timed Petri nets the bare marking (intended as the number of tokens included in each place) is not sufficient to characterize the state of the net, and, in particular, the “age” of the tokens, which is the critical aspect in determining the enabling condition for transitions. For instance, consider the net fragment depicted in Figure 3.4(a). A description of the net based on the  $marked$  predicate would include an axiom stating that “if place  $p$  holds a token for 3 time units, then transition  $u$  may fire”. Now, imagine that at time 0, starting with an empty place  $p$ , transition  $r$  fires, and at time 2 both  $r$  and  $v$  simultaneously fire, so that the number of tokens in place  $p$  remains unchanged. Then, at time instant 3 transition  $u$  may fire, according to an axiomatization based solely on the marking of places, because the first token produced by  $r$  could not be distinguished by the second: the axiom would only require the presence in the preset of a sufficient number of tokens for a sufficiently long time interval.

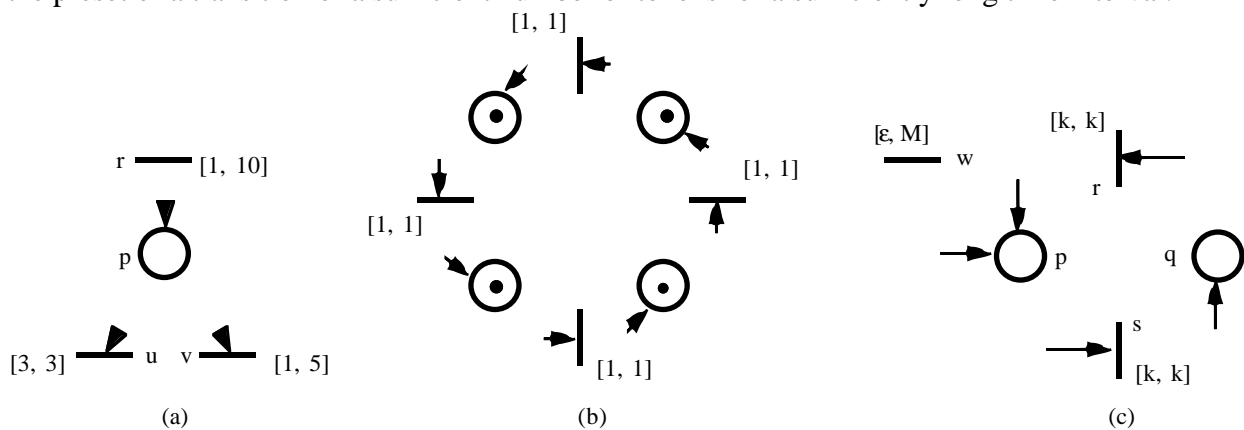


Figure 3.4 Some critical cases of TPNs.

In some cases the age of tokens in a place can be modelled indirectly by considering the successive variations in the markings, but sometimes such a description is not possible or is practically unfeasible. For instance, consider the net of Figure 3.4(b): as far as marking is concerned, the net never changes its state, since every consumed token is instantaneously replaced by another one. Thus any description based exclusively on the marking would be unable to characterize transition firings, which take place according to a precise and well understood pattern.

As a further example, consider Figure 3.4(c), where  $\varepsilon$  and  $M$  represent arbitrarily small and arbitrarily large time constants respectively, and where  $k$  is any constant greater than  $\varepsilon$  and less than  $M$ . During an execution of the net it may happen that places  $p$  and  $q$  contain an arbitrary number of tokens, each of any age between zero and  $k$ . Each token entered in  $p$  by a firing of  $w$  will give rise to an endless sequence of alternate firings of  $r$  and  $s$  with repetition period  $2 \cdot k$ , independently from (but in addition to) the firings caused by other tokens. Describing this behavior purely in terms of marking would be exceedingly complex, whereas in our framework the description is straightforward, as the reader can easily verify.

Another fundamental inadequacy of an axiomatization based on the *marked* predicate is its inability to deal with zero-time transitions. Intuitively, the enabling condition of any transition  $r$  of the net, if stated in terms of the marking, would require a certain marking of  $r$ 's preset to hold for at least  $m_r$  time units in order for  $r$  to be enabled and fire. However, when  $m_r=0$  the above-stated requirement would be trivially satisfied, since at any time any marking held for the at least the last 0 time units. This fact implies that referring only to the marking of the net is not sufficient to axiomatize the firing conditions of zero-time transitions: an explicit reference to other transition firings is indispensable.

On the contrary, in our axiomatization the description of zero-time transitions comes completely for free as a special case of the axioms given in Section 3.2. Simply, when a transition has a zero lowerbound or upperbound it may happen that the corresponding LB and UB axioms are satisfied by assigning extreme values to some of the (quantified) variables. We also emphasize that the axioms remain valid also because *tokenF* and *tokenP* predicates admit a time value greater than or equal to zero, and the temporal operators  $\text{Futr}(\alpha, t)$  and  $\text{Past}(\alpha, t)$  are not strict, i.e., they allow the possibility that their temporal argument  $t$  be null, so that the non-temporal argument  $\alpha$  may take place at the current time instant. As an example, consider the net fragment of Figure 3.2(e). The axioms for the net imply that when transition  $r$  fires, transition  $s$  fires *immediately*, i.e. at the same time. Thus the firings of the two transition are in fact simultaneous, which does not contradict common intuition if we distinguish, as in Section 3.1, between *logical and temporal ordering* among transition firings. Similarly, the fact that—assuming  $m_v>0$ —when transition  $r$  fires, place  $p$  remains empty and the token flows directly into place  $q$  may seem contrived. This is just a convention, however, that is assumed in the axiomatization. It is easy to verify that allowing the token to stay even for one single time instant in place  $p$  would lead to a contradiction.

In conclusion, to deal adequately and in a simple and general way with cases such as simultaneous firings, zero-time transitions, and unbounded nets it is necessary to identify precisely the tokens produced by the firing of a transition. More comments on these subtle points of TPN semantics and on alternative approaches will be briefly given in the Section 6.<sup>1</sup>

---

<sup>1</sup> Yet another comment on the proposed axiomatization could show that some intricate cases, such an unbounded number of firings in an arbitrarily small time interval, are not completely formalized by our marking axioms. We deliberately decided, however, not to go deep into this topic that would be of interest almost exclusively from a mathematical point of view. In fact, in the next Section 3.5 we will introduce a natural hypothesis that rules out *a priori* such intricate situations.

### 3.5 Simplified axioms for particular cases

In this section we show how assuming some restricting hypotheses—which are, however, satisfied in many cases of practical interest—can highly simplify our axiom system, with the result that we must pay the price for high generality only when we use it. We will first exploit 1-bounded nets<sup>1</sup>; then, we will introduce a very simple restriction in the topology of the nets.

#### 1-bounded TPNs

In Section 3.1 we informally discussed the possibility of several simultaneous firings of the same transition. Accordingly, in Section 3.2 we introduced the basic predicates  $nFire(r, n)$  and  $fireth(r, i)$ , to state that transition  $r$  fires  $n$  times and, respectively, to denote the  $i$ -th firing of  $r$ . As a consequence, the predicate  $tokenF(r, i, p, s, j, d)$  has two indices  $i$  and  $j$  denoting which of the (possibly multiple) firings of transitions  $r$  and  $s$  generate or consume the considered token.

The following metatheorem, however, shows that, under very broad hypotheses, in a 1-bounded TPN no simultaneous multiple firings of any transition can occur.

#### Theorem 3.1

Assume that in a timed Petri net no transition has an empty postset, and that there are no cycles for which the sum of the lowerbounds of the included transitions is zero. Then, if the net is 1-bounded, i.e., if at no time any place may contain more than one token, no multiple simultaneous firings of a given transition can occur.

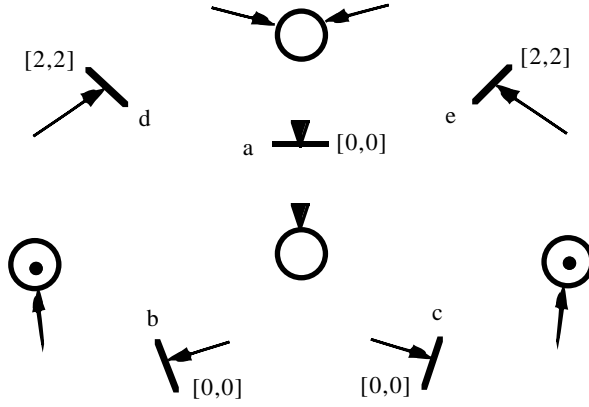
##### *Outline of the proof*

By contradiction, assume that a transition  $v$  fires several times, say, twice, simultaneously. Let us show that it may happen that the two tokens produced by the firings of  $v$  will eventually accumulate in some place, contradicting the 1-boundedness hypothesis. In fact, consider a place  $p$  of the (non-empty, by hypothesis) postset of  $v$ . If  $p$  does not belong to the preset of any other transition, or if all transitions having  $p$  in their preset have strictly positive lowerbounds, then the tokens will accumulate in  $p$ , contradicting the 1-boundedness hypothesis. Otherwise, the same argument can be (inductively) applied to any of the transitions having null lower bounds times: since all transitions have non empty postsets and there are no zero-time cycles, sooner or later the tokens will accumulate in a place that belongs to the preset of a transition with a lowerbound greater than zero.

Notice, in particular, that the contention applies even to the case when place  $p$  is input to several conflicting transitions with zero lower bound, as shown in the example of Figure 3.5, because of the nondeterministic behavior of Petri nets. In the example of Figure 3.5 if transition  $a$  fires twice, the two tokens *could* be consumed one by transition  $b$  and one by  $c$ , but they could also be consumed by two simultaneous firings of  $b$ . Thus, the net is not 1-bounded. ■

---

<sup>1</sup> Notice that our conventions on the marking change (Axiom MC) allows to consider as 1-bounded nets such as that depicted in Figure 3.4(b) that would probably not be considered as 1-bounded on the basis of an intuitive analysis.



**Figure 3.5** A TPN with non zero-time cycles

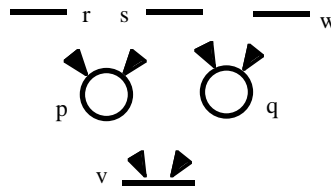
Thanks to Theorem 3.1, whenever its hypotheses are fulfilled, since any transition may fire only once at any time, predicate  $nFire(r, n)$  reduces to  $fire(r)$  (which is true if transition  $r$  fires at the current time, and false otherwise) and predicate  $fireth(r, i)$  becomes superfluous. Similarly, predicates  $tokenF$  and  $tokenP$  can reduce their arguments to the involved transitions, place, and time term. All axioms are therefore simplified accordingly. For instance, axiom  $LB(v)$  is transformed as follows.

$$LB(v): fire(v) \rightarrow \exists d \left( d \geq m_v \wedge \left( \begin{array}{c} tokenP(r, p, v, d) \\ \vee \\ tokenP(s, p, v, d) \end{array} \right) \right)$$

We leave the analysis of other axioms to the reader. Anyway, their use will be illustrated by the following examples.

### Topological restrictions

A different kind of simplification in the axiomatization depends exclusively on the topology of the net. The predicate  $tokenF(r, i, p, s, j, d)$  explicitly refers to the place where the token produced by the firing of  $r$  transits before being consumed by a firing of  $s$ . This explicit reference to the place is needed, in general, when the firing of a transition  $v$  may consume tokens inserted by a unique firing of a transition  $s$  into distinct places of the preset of  $v$ , as shown by Figure 3.6. In this case  $tokenF(s, i, p, v, j, d)$  refers to the token inserted by the firing of transition  $s$  into place  $p$ , while  $tokenF(s, i, q, v, j, d)$  denotes the token inserted (by the same firing of  $s$ ) into place  $q$ .



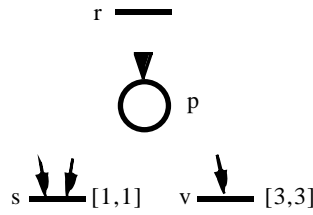
**Figure 3.6** Two places that are both in the postset of a transition and in the preset of another transition.

In many cases, however, for any pair  $r, s$  of transitions of the net there is at most one place that belongs both to the postset of  $r$  and to the preset of  $s$ . So, when we say that a token produced by the  $i$ -th firing of  $r$  is consumed after  $d$  time units by the  $j$ -th firing of  $s$  there is no ambiguity on which place has been holding the token in the time elapsing between the two firings, and we may simply

write  $tokenF(r,i,s,j,d)$ . The way our axioms can be simplified by taking these considerations into account is apparent, so we do not illustrate it further. We will just make use of such a simplified version of predicates  $token$  whenever useful and applicable.

Next, we illustrate the use of simplified predicates and axioms through a few further examples.

1. Consider the portion of TPN depicted in Figure 3.7. It is a slight modification of of the fragment of Figure 3.2(b): in fact, the new arrow shows that transition  $s$  has other places besides  $p$  in its preset. This clearly invalidate the property previously proved as Theorem TPN.4. Instead, the following, weaker, property holds, which will be formalized and proved in the simplified axiomatization that can be used in the hypothesis of Theorem 3.1. We will use also the simplified version of predicates  $token$  since there is a single place joining  $r$  and  $s$ , and  $r$  and  $v$ .



**Figure 3.7** A conflict with multiple-input transitions.

**TPN.5:**  $fireth(r) \rightarrow \exists d (1 \leq d \leq 3 \wedge (Futr(fireth(s),d) \vee Futr(fireth(v),d) )$

*Proof*

1. $fire(r)$	<b>Hyp</b>
2. $Dist(Dist(fire(r),-3),3)$	<b>1, Ax. 7, Ax. 8</b>
3. $Dist(\exists d(0 \leq d \leq 3 \wedge Dist(tokenP(r,s,3-d) \vee tokenP(r,v,3-d),-d)),3)$	<b>2, def. of Past, UB(v), MP</b>
4. $\exists d(Dist(0 \leq d \leq 3 \wedge Dist(tokenP(r,s,3-d) \vee tokenP(r,v,3-d),-d),3))$	<b>3, Th vii</b>
5. $\exists d(Dist(0 \leq d \leq 3, 3) \wedge Dist(Dist(tokenP(r,s,3-d) \vee tokenP(r,v,3-d),-d),3))$	<b>4, Ax.9÷10</b>
6. $\exists d(0 \leq d \leq 3 \wedge Dist(Dist(tokenP(r,s,3-d) \vee tokenP(r,v,3-d),-d),3))$	<b>5,Th iv</b>
7. $\exists d(0 \leq d \leq 3 \wedge Dist(tokenP(r,s,3-d) \vee tokenP(r,v,3-d),3-d))$	<b>6, Ax.8</b>
8. $\exists e(0 \leq e \leq 3 \wedge Dist(tokenP(r,s,e) \vee tokenP(r,v,e),e))$	<b>7, TD</b>
9. $\exists e(0 \leq e \leq 3 \wedge (Dist(tokenP(r,s,e),e) \vee Dist(tokenP(r,v,e),e)))$	<b>8, Ax.9÷10</b>
10. $\exists e(0 \leq e \leq 3 \wedge (tokenF(r,s,e) \vee tokenF(r,v,e)))$	<b>9, PF (twice)</b>
11. $\exists e(0 \leq e \leq 3 \wedge ((tokenF(r,s,e) \wedge e \geq 1 \vee tokenF(r,v,e) \wedge e \geq 3)))$	<b>10, TPN.1 for transitions.(r,s) and (r,v)</b>
12. $\exists e(0 \leq e \leq 3 \wedge (tokenF(r,s,e) \wedge e \geq 1) \vee (tokenF(r,v,e) \wedge e \geq 1))$	<b>11, TD: <math>e \geq 3 \rightarrow e \geq 1</math></b>
13. $\exists e(0 \leq e \leq 3 \wedge (e \geq 1) \wedge (tokenF(r,s,e) \vee tokenF(r,v,e)))$	<b>12, TAUT: <math>((\alpha \wedge \delta) \vee (\beta \wedge \delta)) \rightarrow ((\alpha \vee \beta) \wedge \delta)</math></b>
14. $\exists e(1 \leq e \leq 3 \wedge (Futr(fire(s),e) \vee Futr(fire(v),e)))$	<b>13, FI, FI</b>
15. $fire(r) \rightarrow \exists e(1 \leq e \leq 3 \wedge (Futr(fire(s),e) \vee Futr(fire(v),e)))$	<b>1÷14, DED</b>

2. The next example illustrates the semantics provided for the case of transitions without preset: according to the informal discussion of Section 3.1, a transition  $s$  without preset should be considered as connected to a "dummy" place  $p$ , as shown in Figure 3.8, which in turn has a single input transition  $itp$ . For such a configuration we wish to prove that transition  $s$  never fires more than once in

less than  $m_s$  time units (assuming that it is enabled by a dummy initial marking of  $p$  consisting of a single token). This property is formalized by the formula below.

$$\mathbf{TPN.6.:} \quad n\text{Fire}(itP,1) \rightarrow (\text{AlwF}(\text{fire}(s) \rightarrow \text{Lasts}(\neg\text{fire}(s),m_s)))$$

Notice that the first clause of the statement uses predicate  $n\text{Fire}$  in its general form, whereas the second part uses the simplified predicate  $\text{fire}$ . This formulation emphasizes the method through which the proof is derived. In fact, such a proof consists of two main steps: first, the following lemma is proved.

$$\mathbf{TPN.7:} \quad \text{intitialMarking}(p,1) \rightarrow \text{AlwF}(\text{marked}(p,1))$$

Clearly, TPN.7 states that place  $p$  is always 1-bounded. Once this fact is stated, the rest of the proof—which is by contradiction—can be carried over by using the simplified axioms. Notice that we can exploit simplified axioms even in a “local” way, in the sense that, in order to analyze the firing properties of  $s$ , only  $p$ 's 1-boundedness is relevant, since the firing of  $s$  is certainly not affected by the marking of other places.

*Proof of TPN.7:*

1. $\text{intitialMarking}(P,1)$	<b>Hyp</b>
2. $n\text{Fire}(itP,1) \wedge \text{marked}(P,1) \wedge \text{AlwF}(n\text{Fire}(itP,0))$	<b>1, IM(P)</b>
3. $\text{AlwF}(\forall i \forall k (n\text{Fire}(s,i) \wedge n\text{Fire}(s,k) \rightarrow i=k))$	<b>UFN(s)</b>
4. $\text{AlwF}(\forall i \forall k (n\text{Fire}(s,i) \wedge n\text{Fire}(itP,0) \wedge n\text{Fire}(s,k) \rightarrow i+0=k))$	<b>2, 3, Arit.</b>
5. $\text{AlwF}(\forall i \forall j \forall k (n\text{Fire}(s,i) \wedge n\text{Fire}(itP,j) \wedge n\text{Fire}(s,k) \rightarrow i+j=k))$	<b>4, 1, UFN(itP)</b>
6. $\text{AlwF}(\neg\exists i \exists j \exists k (n\text{Fire}(s,i) \wedge n\text{Fire}(itP,j) \wedge n\text{Fire}(s,k) \wedge i+j \neq k))$	<b>5, <math>\forall \equiv \neg\exists \neg</math>, TAUT<sup>1</sup>.</b>
7. $\text{Until}_w(\text{marked}(P,1), \exists i \exists j \exists k (i+j \neq k \wedge n\text{Fire}(s,i) \wedge n\text{Fire}(itP,j) \wedge n\text{Fire}(s,k)))$	<b>1, MI(P), MP</b>
8. $\text{AlwF}(\text{marked}(P,1)) \vee \exists t (\text{Futr}(\exists i \exists j \exists k (i+j \neq k \wedge n\text{Fire}(s,i) \wedge n\text{Fire}(itP,j) \wedge n\text{Fire}(s,k)),t) \wedge \text{Lasts}(\text{marked}(P,1),t))$	<b>7, Until<sub>w</sub> Def.</b>
9. $\text{AlwF}(\text{marked}(P,1))$	<b>6, 8, TAUT<sup>2</sup></b>
10. $\text{intitialMarking}(P,1) \rightarrow \text{AlwF}(\text{marked}(P,1))$	<b>1÷9, DED</b>

■

*Proof of TPN.6 (By contradiction):*

1. $n\text{Fire}(itP,1) \wedge \neg\text{AlwF}(\text{fire}(s) \rightarrow \text{Lasts}(\neg\text{fire}(s),m_s))$	<b>Hyp.</b>
2. $\text{SomF}(\text{fire}(s) \wedge \text{WithinF}(\text{fire}(s),m_s))$	<b>1, Morgan Laws, SomF and Within<sub>ee</sub> Def.</b>
3. $\text{Futr}(\text{fire}(s) \wedge \text{WithinF}_{ee}(\text{fire}(s),m_s),T)$	<b>2, EI(SomF)</b>
4. $\text{Futr}(\text{fire}(s) \wedge \exists e (\text{Futr}(\text{fire}(s),e) \wedge e < m_s),T)$	<b>3, WhithinF Def.</b>

*For notational simplicity we will omit the outmost temporal operator Futr(a,T)*

5. $\text{fire}(s) \wedge \text{Futr}(\text{fire}(s),E) \wedge E < m_s$	<b>4, EI(e)</b>
6. $\exists d (d \geq m_s \wedge (\text{tokenP}(s,P,s,d) \vee \text{tokenP}(itP,P,s,d)))$	<b>5, LB(s)</b>
7. $\text{Futr}(\exists d (d \geq m_s \wedge (\text{tokenP}(s,P,s,d) \vee \text{tokenP}(itP,P,s,d))),E)$	<b>5, 6, TT(E)</b>

<sup>1</sup> The tautology  $\alpha \rightarrow \beta \leftrightarrow \neg(\alpha \wedge \neg\beta)$  is used

<sup>2</sup> The tautology  $(\alpha \vee \beta) \wedge \neg\beta \rightarrow \alpha$  is used (6 is  $\beta$  while the second disjunct of 8 is  $\neg\beta$ ).

8.  $\exists d( d \geq m_s \wedge \text{Futr}(\text{tokenP}(s,P,s,d) \vee \text{tokenP}(itP,P,s,d),E))$  **7,Th. vii, Th iv.**
9.  $\exists d( d \geq m_s \wedge \text{Futr}(\text{Past}(\text{tokenF}(s,P,s,d),d) \vee \text{Past}(\text{tokenF}(itP,P,s,d),d),E))$  **8, PF, PF**
10.  $\exists d( d \geq m_s \wedge \text{Futr}(\text{Past}(\text{tokenF}(s,P,s,d),d),E) \vee \text{Futr}(\text{Past}(\text{tokenF}(itP,P,s,d),d),E))$  **9, Ax9**
11.  $\exists d( d \geq m_s \wedge (\text{Past}(\text{tokenF}(s,P,s,d),d-E) \vee \text{Past}(\text{tokenF}(itP,P,s,d),d-E)))$  **10, 5, Ax8, TD (E < m\_s \leq d)**
12.  $\exists d( d \geq m_s \wedge (\text{Past}(\text{tokenF}(s,P,s,d),d) \vee \text{Past}(\text{tokenF}(itP,P,s,d),d)))$  **6, PF, PF**

11 and 12 lead to four possibilities:

- $\alpha = \exists d( d \geq m_s \wedge (\text{Past}(\text{tokenF}(s,P,s,d),d)) \wedge \exists e( e \geq m_s \wedge (\text{Past}(\text{tokenF}(s,P,s,e),e-E)))$
- $\beta = \exists d( d \geq m_s \wedge \text{Past}(\text{tokenF}(itP,P,s,d),d)) \wedge \exists e( e \geq m_s \wedge (\text{Past}(\text{tokenF}(s,P,s,e),e-E)))$
- $\gamma = \exists d( d \geq m_s \wedge (\text{Past}(\text{tokenF}(s,P,s,d),d)) \wedge \exists e( e \geq m_s \wedge (\text{Past}(\text{tokenF}(itP,P,s,e),e-E)))$
- $\delta = \exists d( d \geq m_s \wedge \text{Past}(\text{tokenF}(itP,P,s,d),d)) \wedge \exists e( e \geq m_s \wedge (\text{Past}(\text{tokenF}(itP,P,s,e),e-E)))$

13.  $\alpha \vee \beta \vee \gamma \vee \delta$  **11, 12**
- 14..  $\alpha, d=e-E \vdash \exists d( d \geq m_s \wedge \text{Past}(\text{tokenF}(s,P,s,d) \wedge \text{tokenF}(s,P,s,e),d))$  **TD, Ax9**
- 15..  $\alpha, d=e-t \vdash \exists d( d \geq m_s \wedge \text{Past}(n\text{Fire}(s,k) \wedge K \geq 2))$  **14, FI, e \neq d, OU(s), FR(s)**

15 contradicts the assumption of the simplified notation  $Alw(n\text{Fire}(s,k) \wedge K \leq 1)$

16.  $\alpha, d \neq e-E \vdash \exists d( d \geq m_s \wedge \text{Past}(\text{fire}(s),d)) \wedge \exists e( e \geq m_s \wedge \text{Past}(\text{fire}(s),e-E))$  **FI, FI**
17.  $\beta, d < e-E \vdash \text{Past}(\text{Past}(\text{fire}(s),e-E),d)$  **FI, TD, Ax.8**

17 contradicts the initial marking assumptions because it represents a firing of s before the firing of itP..

18.  $\beta, d \geq e-E \vdash \exists d( d \geq m_s \wedge \text{Past}(\text{fire}(itP),d)) \wedge \exists e( e \geq m_s \wedge (\text{Past}(\text{fire}(s),e-E)))$  **FI, FI**
19.  $\beta, d \geq e-E \vdash \exists e( e \geq m_s \wedge \text{Past}(\exists t( t \geq m_s \wedge \text{Past}(\text{fire}(s),t )),e-E))$  **18, LB(s)<sup>1</sup>.**
20.  $\beta, d \geq e-t \vdash AlwP(\text{SomeP}(\text{fire}(s)))$  **16, LB(s) excluding the firing of itP, Th xP<sup>2</sup>,M P**
21.  $\beta, d \geq e-t \vdash \exists d( d \geq m_s \wedge \text{Past}(\text{fire}(itP) \wedge AlwP(\text{SomeP}(\text{fire}(s))),d))$  **18,20**

21 contadicts the assumption that s never fired before the initial marking.

22.  $\gamma \vdash \text{FALSE}$   $\gamma$  is equivalent to  $\beta$ , and then it leads to contradictions like in 17 and 21.
23.  $\delta \vdash \text{FALSE}$   $\delta$  represents two different firings of itP. Hence, 23, due to axiom  $OU(itP)$  and  $e \neq d$ , is in contradiction with the initial marking assumption asserting the uniqueness for the firing of itP.

24.  $\alpha, d \neq e-E$  **15,17,21,22,23, CA**
25.  $\text{Futr}(\exists d( d \geq m_s \wedge \text{Past}(\text{tokenF}(s,P,s,d),d)) \wedge \exists e( e \geq m_s \wedge \text{Past}(\text{tokenF}(s,P,s,e),e-E)), T)$  **24,4<sup>3</sup>**
26.  $\exists d( d \geq m_s \wedge \text{Futr}(\text{tokenF}(s,P,s,d),T-d)) \wedge \exists e( e \geq m_s \wedge \text{Futr}(\text{tokenF}(s,P,s,e),T-(e-E)))$  **25, T>d and T>e-E to not contradict MI.**

Then, there are two other firings of s occurred before the moment we assumed s fired with a firing time difference less than  $m_s(T)$ . They fired at d and e-E time units before T. We assume without loss of generality  $d > e-E$ . In this case there are two possibilities, the firing occurring later is a consequence of the first one or both are consequence of two different and precedent firings. Let us consider the first case. In this case the first must produce a firing that will produce another firing and so on till produce the firing of the seconf one. That is,

<sup>1</sup> The disjoint about the possible firing of itP is excluded, because it contradicts 18 which explicitly refers to another firing of itP.

<sup>2</sup> The theorem used is  $\text{Th } xP: \phi \wedge Alw(\phi \rightarrow \exists a(a > K \wedge \text{Past}(\phi,a))) \rightarrow AlwP(\text{SomeP}(\phi))$  This theorem is the counterpart of Th. x. referring to the past

<sup>3</sup> Including again the temporal operator  $\text{Futr}(\alpha,T)$  omitted after step 4.

27.  $\exists d( d \geq m_s \wedge \text{Futr}(\exists d_1 (\text{tokenF}(s,P,s,d_1) \wedge \text{Futr}(\exists d_2 (\text{tokenF}(s,P,s,d_2) \wedge \dots \wedge \text{Futr}(\exists d_n (\text{tokenF}(s,P,s,d_n) \wedge d_1+d_2+\dots+d_n = d-T-(e-E)),d)))$

but this contradicts 26 and  $OU(s)$  because  $d \neq d1$ .

Then, we must consider the second case where both firings depend on two different and precent firings. That implies, there are again two firings at least  $m_s$  time units in the past. This case is analogous to step 13 with the four possibilities  $\alpha \vee \beta \vee \gamma \vee \delta$  that also lead to step 26 with two other different firings of  $s$ . Continuing in this way, we can prove that always in the past there are two different firings of  $s$ . This leads to a contradiction with the initial marking assumption, because, in particular, there should be firings before the initial marking.

Hence, the hypothesis is false and its negation is valid:

28.  $n\text{Fire}(itP,1) \rightarrow \text{AlwF}(\text{fire}(s) \rightarrow \text{Lasts}(\neg\text{fire}(s),m_s))$  ■



**Figure 3.8** A transition with empty preset, showing the “dummy” preset P.

## 4. Sample Applications

In this section we show how the proposed axiomatization can be applied to prove properties of systems described as timed Petri nets. We use two fairly traditional “benchmarks” for the analysis of concurrent and possibly real-time systems, namely the elevator system [2] [96] [57] [17].and the dining philosophers problem, suitably modified to cope with timing issues [3].

Although the main effort is devoted to the illustration of the basic semantic aspects of the proof system, a few hints are given to provide also a *method* to master the complexity of non-trivial proofs. In particular, it is clear that the proof of complex systems must be modularized in much the same way as their specification and design is modularized too. In other words, complex systems must be decomposed in simpler components whose local properties should be analyzed first; later, global system properties must be derived from the local ones. The issue of scaling up to real-life systems is not the main issue of *this* paper, however. Nevertheless, due to its tremendous relevance for practical applications, we will go back to it at the end of this section and in Section 6.

### An elevator system

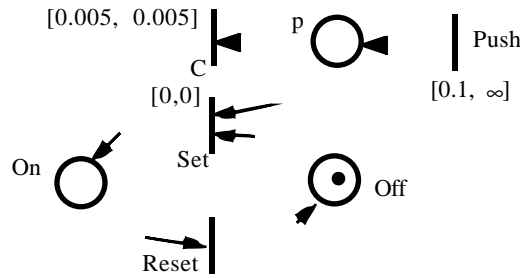
Elevator systems are often considered in the literature, thanks to several distinguishing features such as the concurrency of the movements of several elevators with the requests issued by button pushing and timing constraints in the service of such requests.

Here we focus our attention to button pressing and to door opening-closing. We should be able to derive, however, informations that are general enough to convince the reader that similar reasonings and proof techniques can be applied to the analysis of other components of the system and of global requirements.

## Button illumination

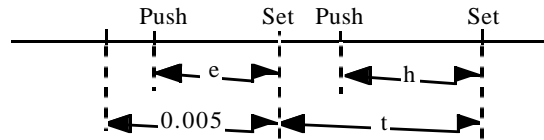
Modules of type `BUTTON`<sup>1</sup> may be described as in Figure 4.1 (which is, however, incomplete as far as it concerns illumination resetting), taken from [38]. Pushing a button is represented by the firing of transition `Push`. If the button is off (a token is in `Off`) and `Push` fires, then `Set` immediately fires and sets the button to on (a token goes in `On`). `C` acts as a token consumer to prevent meaningless accumulation of tokens in `P`. In such a way, an on button can be pushed many times (with a minimum idle time of 0.1 seconds representing the minimum possible time difference between two pushings) without any undesirable consequence. In other words, between any two firings of transition `Set`, there is at least one of `Push`. This informal analysis is formalized by the proof of the following theorem.

**EP.1:**  $\text{fireth}(\text{Set},i) \wedge \text{Futr}(\text{fireth}(\text{Set},j),t) \rightarrow \exists d(d \leq t \wedge \text{Futr}(\exists k \text{ fireth}(\text{Push},k),d))$



**Figure 4.1** A TPN fragment describing button illumination

*Proof:* we first show (steps 1÷11) that a firing of `Set` necessarily requires a preceding firing of `Push` occurring—because of `C`'s upperbound—not earlier than 0.005 time units before. Similarly, a subsequent firing of `Set` occurring after  $t$  time units requires a firing of `Push` preceding it by  $h$  time units, with  $h \leq 0.005$  (step 12). However  $h \leq t$  must hold, as depicted in Figure 4.2, otherwise `Push` would have fired twice in less than 0.005 time units. (steps 14÷17). This allows to conclude (steps 18÷21) that the second firing of `Push` is successive to the first of `Set`.



**Figure 4.2** Relative position on the time axis of the Push and Set events.

- |   |  |
|---|--|
| 1. $\text{fireth}(\text{Set},i) \wedge \text{Futr}(\text{fireth}(\text{Set},j),t)$  | <b>Hyp</b>                                       |
| 2. $\exists d(d \geq 0 \wedge \exists k \text{ tokenP}(\text{Push},k,\text{Set},i,d))$  | <b>1, <math>\wedge</math>-elim, LB(Set), MP,</b> |
| 3. $\text{tokenP}(\text{Push},K,\text{Set},i,D)$  | <b>2, EI, <math>\wedge</math>-elim, EI</b>       |
| 4. $\text{Past}(\text{tokenF}(\text{Push},K,\text{Set},i,D),D)$   | <b>3, PF</b>                                     |
| 5. $\text{Past}(\text{fireth}(\text{Push},K),D)$  | <b>4, PI</b>                                     |
| 6. $\text{Past}(\exists e(e \leq 0.005 \wedge (\exists x \text{ tokenF}(\text{Push},K,C,x,e) \vee \exists y \text{ tokenF}(\text{Push},K,\text{Set},y,e))),D)$                | <b>5, UB(C)</b>                                  |
| 7. $\exists e(e \leq 0.005 \wedge (\text{Past}(\exists x \text{ tokenF}(\text{Push},K,C,x,e),D) \vee \text{Past}(\exists y \text{ tokenF}(\text{Push},K,\text{Set},y,e),D)))$ | <b>6,Th.vii,Ax9÷10,Th.iv</b>                     |

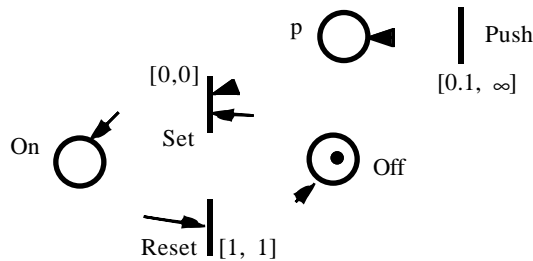
<sup>1</sup> This suggests that an elevator system will contain many instances of such modules. Thus, once we proved any property on the behavior of the TPN fragment given in Figure 4.1, we can rely on that property for all buttons existing in the system.

8.  $(E \leq 0.005 \wedge \text{Past}(\text{tokenF}(\text{Push}, K, C, X, E), D)) \vee (E \leq 0.005 \wedge \text{Past}(\text{tokenF}(\text{Push}, K, \text{Set}, Y, E), D))$  **7, EI, TAUT<sup>1</sup>**
9.  $Y = i \wedge D = E$  **4, 8, OU(Push)**
10.  $\exists e (e \leq 0.005 \wedge \text{Past}(\exists k \text{ fireth}(\text{Push}, k), e))$  **9, 5, Ax.6, 8, TAUT<sup>2</sup>,  $\wedge$ -intro,  $\exists$ -intro**
11.  $\text{fireth}(\text{Set}, i) \rightarrow \exists e (e \leq 0.005 \wedge \text{Past}(\exists k \text{ fireth}(\text{Push}, k), e))$  **1  $\div$  10, DED**
12.  $\text{Futr}(\text{fireth}(\text{Set}, j), t) \rightarrow \text{Futr}(\exists h (h \leq 0.005 \wedge \text{Past}(\exists k \text{ fireth}(\text{Push}, k), h)), t)$  **11, TT**
13.  $\text{Futr}(\exists h (h \leq 0.005 \wedge \text{Past}(\exists k \text{ fireth}(\text{Push}, k), h)), t)$  **1,  $\wedge$ -elim, 12, MP**
14.  $\exists h (h \leq 0.005 \wedge \text{Futr}(\text{Past}(\exists k \text{ fireth}(\text{Push}, k), h), t))$  **13, Th.vii, Ax.9  $\div$  10**
15.  $H \leq 0.005 \wedge \text{Futr}(\text{Past}(\exists k \text{ fireth}(\text{Push}, k), H), t)$  **14, EI**
- By contradiction we assume  $H > t$ :*
16.  $H > t \vdash H \leq 0.005 \wedge \text{Past}(\exists k \text{ fireth}(\text{Push}, k), H - t)$  **15, Th.vi**
17.  $H > t \vdash H - t \leq 0.005 \wedge \text{Past}(\exists k \text{ fireth}(\text{Push}, k), H - t)$  **16, TD**
- 10 and 17 contradict TPN.6 applied to transition Push. Hence
18.  $H \leq t$  **10, 17, TPN.6, by contradiction**
19.  $\text{Futr}(\exists k \text{ fireth}(\text{Push}, k), t - H)$  **15, 18, Th.vi**
20.  $\exists d (d \leq t \wedge \text{Futr}(\exists k \text{ fireth}(\text{Push}, k), d))$  **19, TD,  $\exists$ -intro**
21.  $\text{fireth}(\text{Set}, i) \wedge \text{Futr}(\text{fireth}(\text{Set}, j), t) \rightarrow \exists d (d \leq t \wedge \text{Futr}(\exists k \text{ fireth}(\text{Push}, k), d))$  **1  $\div$  20 DED** ■

It is interesting to contrast the previous analysis of the net of Figure 4.1 with the net of Figure 4.3, which is a slight modification thereof. In authors' experience, the net of Figure 4.3 is a natural *initial* formalization of the illumination mechanism of the elevator system. In such a net, however, the lack of transition C would allow an unbounded accumulation of tokens in place P. Formally, property EP.1 cannot be proved, since in step 6 axiom UB(C) is used. On the contrary, the following property can be proved:

$$\text{EP.2 : } \left( \begin{array}{l} \text{fireth}(\text{Push}, i) \wedge \text{Futr}(\text{fireth}(\text{Push}, j), 0.1) \wedge \\ \text{Futr}(\text{Lasts}(\neg \exists h \text{ fireth}(\text{Push}, h), 1), 0.1) \end{array} \right) \rightarrow \exists k \text{ Futr}(\text{fireth}(\text{Set}, k), 1)$$

Intuitively, EP.2 states that the button is illuminated by a Push operation that preceded its resetting. We must warn the reader that Figure 4.3 deliberately oversimplifies illumination resetting (it is described as a deterministic fact that occurs invariably 1 second after its setting) to help focus attention on the switching *on* of the button.



**Figure 4.3** An incorrect formalization of button illumination.

<sup>1</sup> The tautology  $\alpha \wedge (\beta \vee \gamma) \leftrightarrow ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$  is used.  
<sup>2</sup> The tautology  $(\alpha \wedge \beta) \vee (\alpha \wedge \gamma) \rightarrow \alpha$  is used.

*Proof of EP.2:*

1. $\text{fireth}(\text{Push},i) \wedge \text{Futr}(\text{fireth}(\text{Push},j),0.1)$	<b>Hyp.</b>
2. $\text{fireth}(\text{itOff},n)$	<b>MI(Off)</b>
3. $\text{fireth}(\text{Push},i) \wedge \text{fireth}(\text{itOff},n) \vee \text{fireth}(\text{Reset},m)$	<b>1,2 TAUT<sup>1</sup></b>
4. $\exists k \text{ tokenF}(\text{Push},i,p,\text{Set},k,0) \wedge \exists k \text{ tokenF}(\text{itOff},n,p,\text{Set},k,0) \vee \exists k \text{ tokenF}(\text{Reset},m,\text{Off},\text{Set},k,0)$	<b>3, UB(Set), MP</b>
5. $\exists k \text{ fireth}(\text{Set},k)$	<b>4, FI, MP</b>
6. $\text{Futr}(\exists k \text{ fireth}(\text{Reset},k),1)$	<b>5, TPN.2, MP, TD</b>
7. $\text{Futr}(\text{fireth}(\text{Push},j),0.1) \wedge \text{Futr}(\exists k \text{ fireth}(\text{Reset},k),1)$	<b>1,6</b>
8. $\text{Futr}(\text{Past}(\text{ fireth}(\text{Push},j),0.9) \wedge \exists k \text{ fireth}(\text{Reset},k),1)$	<b>7, Ax8, Ax9</b>
9. $\text{Futr}(\text{Past}(\text{ fireth}(\text{Push},j),0.9) \wedge \text{fireth}(\text{Reset},K),1)$	<b>8, EI(k)</b>
10. $\text{Futr}(\exists d( d \leq 0.9 \wedge \exists x \text{ Past}(\text{tokenP}(\text{Push},j,p,\text{Set},x,0.9-d),d)) \vee \exists x \text{ tokenP}(\text{Reset},K,p,\text{Set},x,0) ,1)$	<b>9, UB(Set), TD</b>
11. $\text{Futr}(\exists d( d \leq 0.9 \wedge \exists x \text{ Past}(\text{tokenP}(\text{Push},j,p,\text{Set},x,0.9-d),d)),1) \vee \text{Futr}(\exists x \text{ tokenP}(\text{Reset},K,p,\text{Set},x,0) ,1)$	<b>10, Ax9</b>
<i>Calling <math>\alpha = \text{Futr}(\exists d( d \leq 0.9 \wedge \exists x \text{ Past}(\text{tokenP}(\text{Push},j,p,\text{Set},x,0.9-d),d)),1)</math> and</i>	
<i><math>\beta = \text{Futr}(\exists x \text{ tokenP}(\text{Reset},k,p,\text{Set},x,0) ,1)</math>:</i>	
12. $\alpha \vdash \text{Futr}( D \leq 0.9 \wedge \exists x \text{ Past}(\text{tokenP}(\text{Push},j,p,\text{Set},x,0.9-D),D),1)$	<b>EI(d)</b>
13. $\alpha, D > 0 \vdash D \leq 0.9 \wedge \exists x \text{ Futr}(\text{ Past}(\text{tokenP}(\text{Push},j,p,\text{Set},x,0.9-D),D),1)$	<b>12, Th. vii, Th. iv.</b>
14. $\alpha, D > 0 \vdash D \leq 0.9 \wedge \exists x \text{ Futr}(\text{ tokenP}(\text{Push},j,p,\text{Set},x,0.9-D),1-D)$	<b>13, Ax9 (<math>0 &lt; D \leq 0.9 &lt; 1</math>)</b>
15. $\alpha, D > 0 \vdash D \leq 0.9 \wedge \exists x \text{ Futr}(\text{fireth}(\text{Set},x),1-D)$	<b>14,PI</b>
16. $\alpha, D > 0 \vdash D \leq 0.9 \wedge \exists x \text{ Futr}(\exists e (e \geq 1 \wedge \exists k \text{ tokenP}(\text{Reset},k,\text{Off},\text{Set},x,e)), ,1-D)$	<b>15, LB(Set)</b>
17. $\alpha, D > 0 \vdash D \leq 0.9 \wedge \exists e (e \geq 1 \wedge \exists k \text{ Futr}(\text{ Past}(\text{fireth}(\text{Reset},k),e),1-D))$	<b>16, Th. vii, Th. iv, Th. vii, PI</b>
18. $\alpha, D > 0 \vdash D \leq 0.9 \wedge \exists e (e \geq 1 \wedge \exists k \text{ Past}(\text{fireth}(\text{Reset},k),e-(1-D)))$	<b>17 Ax8, TD (<math>e \geq 1 \geq 1-D</math>)</b>
<i>17 represents a firing of Reset in the past, i.e. before the initial marking causing a contradiction</i>	
19. $\alpha \vdash D = 0$	<b>13 <math>\div</math> 18, by contradiction and because <math>D \geq 0</math></b>
20. $\alpha \vdash \text{Futr}(\exists x \text{ Past}(\text{tokenP}(\text{Push},j,p,\text{Set},x,0.9),0),1)$	<b>19, Ax6</b>
21. $\alpha \vdash \text{Futr}(\exists x \text{ fireth}(\text{Set},x),1)$	<b>20, PI</b>
22. $\beta \vdash \text{Futr}(\exists x \text{ fireth}(\text{Set},x),1)$	<b>PI</b>
23. $\text{Futr}(\exists x \text{ fireth}(\text{Set},x),1)$	<b>21, 22, CA</b>
24. $\text{fireth}(\text{Push},i) \wedge \text{Futr}(\text{fireth}(\text{Push},j),0.1) \rightarrow \text{Futr}(\exists x \text{ fireth}(\text{Set},x),1)$	<b>1<math>\div</math>23 DED</b>
25. $\text{fireth}(\text{Push},i) \wedge \text{Futr}(\text{fireth}(\text{Push},j),0.1) \wedge \text{Futr}(\text{Lasts}(\neg \exists h \text{ fireth}(\text{Push},h),1),0.1) \rightarrow \text{Futr}(\exists x \text{ fireth}(\text{Set},x),1)$	<b>24, TAUT<sup>2</sup></b>

■

### Door closing

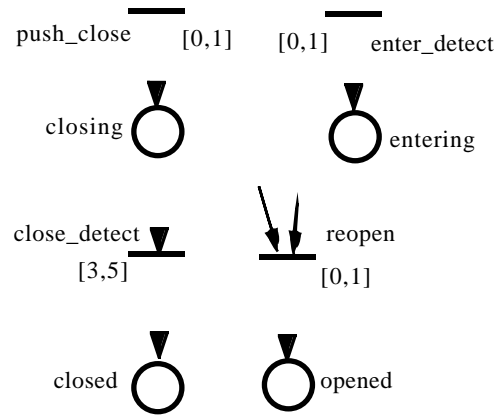
Figure 4.4, taken from [17], is an approximate description of door closing. Transition *push\_close* represents the pushing of a special button—other than the buttons for elevator requests—for closing the doors; transition *enter\_detect* represents the entering of a person in the elevator. The intended meaning of the net is that, if a person enters the elevator while its doors are closing, the doors reopen. This is

<sup>1</sup> The tautology  $\alpha \wedge \beta \rightarrow \alpha \wedge \beta \vee \gamma$  is used.

<sup>2</sup> The tautology  $(\alpha \rightarrow \beta) \rightarrow (\alpha \wedge \gamma \rightarrow \beta)$  is used.

formalized by the following statement, which can be easily proved along the same lines as in the proof of TPN.4.

**EP.3:**  $\text{fireth}(\text{push\_close},i) \wedge \text{WithinF}_{ee}(\text{fireth}(\text{enter\_detect},k), 2) \rightarrow \text{WithinF}_{ee}(\exists j \text{ fireth}(\text{reopen},j), 3)$



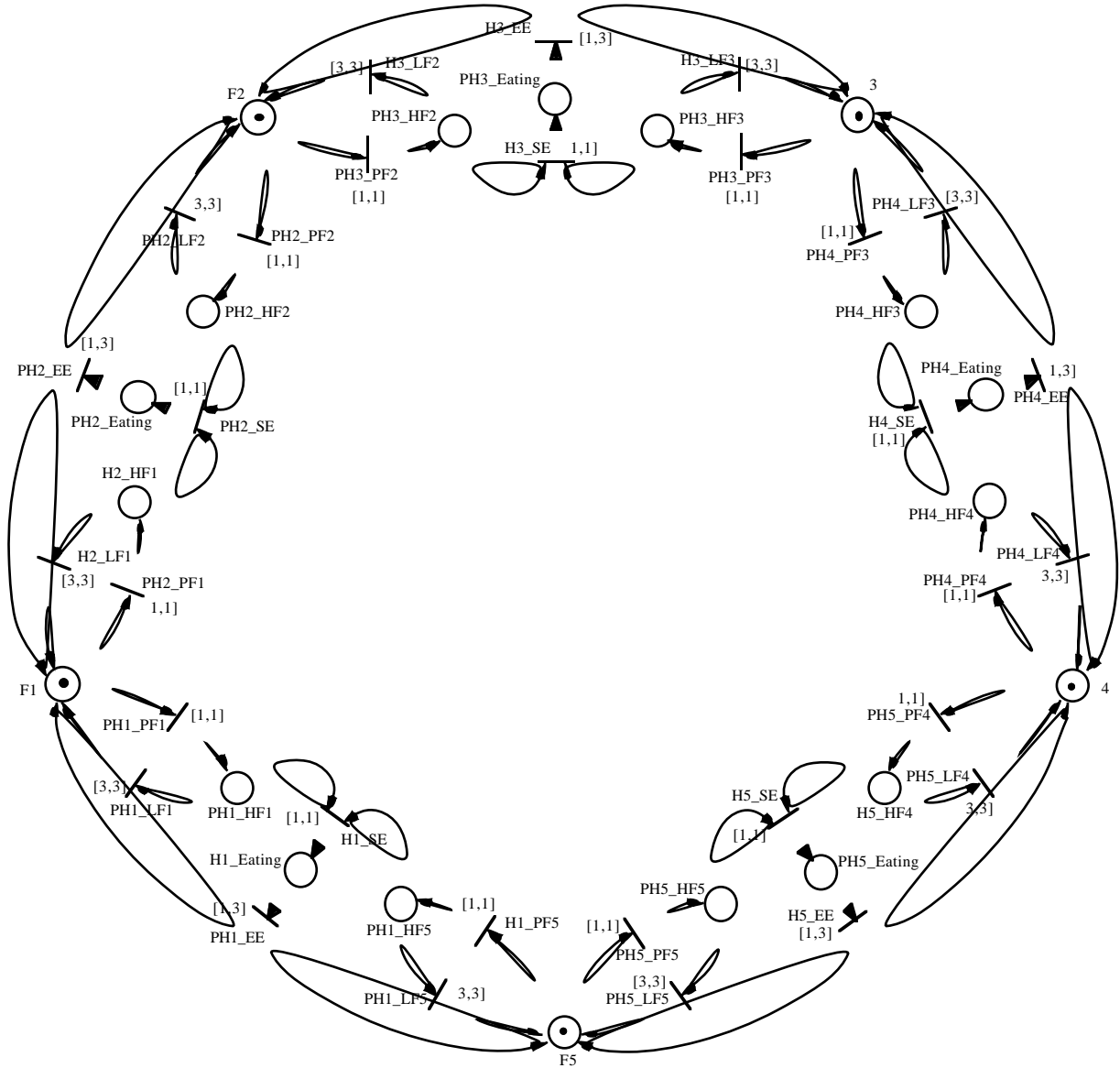
**Figure 4.4** A TPN describing door closing.

As a side remark it is worth pointing out that Figure 4.4 contains a few inadequacies for a precise description of the door behavior. For instance, a similar drawback occurs, to that of Figure 4.3, since an unfortunate token accumulation in place *closing* may cause illegal behaviors. The reader may discover by himself some of these inaccuracies and can prove or disprove desired or undesired system properties.

Furthermore, Figure 4.4 shows that even the adopted TPN model–[66]–cannot completely capture some subtle system aspects. In fact the closing and the reopening of a door are described as events that nondeterministically occur in an instant belonging to given time intervals. In practice, instead, the time needed to reopen a door when a person enters the elevator while door is closing, depends deterministically on the entering time: if one enters immediately after button pushing, reopening takes a very short time; instead, if one enters just before complete closing, it takes more. Such subtle aspects could be described adequately by adopting a more sophisticated TPN model such as TB-nets proposed in [39]. As it will be emphasized in Section 6, it is just a simple exercise extending our axiom system to such a generalized family of TPNs.

### The dining philosophers

Figure 4.5 presents a real-time version of the classical 5 dining philosophers problem [3]. It has been derived from the original, non-timed, version, by assuming that every philosopher has a maximum time to keep a fork without using it. This time-out is represented by transitions  $\text{PHi-LFj}$  (the  $i$ -th philosopher leaves the  $j$ -th fork): if the  $i$ -th philosopher picks up the  $j$ -th fork (transition  $\text{PHi-PFj}$ ) and in the following 3 time units he does not manage to pick up the second fork, he must release the  $j$ -th fork. Instead, if he obtains the second fork, he may start eating one time unit later (transition  $\text{PHi-SE}$ ).



Legend: PH<sub>i</sub>\_PF<sub>j</sub>: philosopher *i* picks up fork *j*, PH<sub>i</sub>\_LF<sub>j</sub>: philosopher *i* leaves fork *j*, PH<sub>i</sub>\_SE: philosopher *i* start to eat, PH<sub>i</sub>\_EE: philosopher *i* finish to eat.; PH<sub>i</sub>\_HF<sub>j</sub>: philosopher *i* has fork *j*, PH<sub>i</sub>\_Eating: philosopher *i* is eating, F<sub>j</sub>: fork *j*.

**Figure 4.5** A real-time version of the dining philosophers problem.

Since the hypotheses of Theorem 3.1 regarding 1-boundedness are clearly satisfied by the net of Figure 4.5, we will analyze it using the simplified predicates and axioms of Section 3.5. The main property of the net is that, unlike the original, non timed, model, it is deadlock-free. This is clearly implied by the following theorem.

**DPh.1:**  $\iota \vdash \text{AlwF}(\text{WithinF}(\text{fire}(\text{PH}_i\text{-PF}_i) \vee \text{fire}(\text{PH}_{i+1}\text{-PF}_i), 7))$

DPh.1 means that always in the future (starting form the initial marking  $\iota$ ) every fork *i* is picked up within 7 time units. Notice that it does not mean that a philosopher will eventually eat: for instance, it could happen that always every philosopher picks up the left fork and never picks up the right one.

Two natural methods are adopted to master the complexity of the proof of DPh.1. First, its formulation is parametric with respect to the number of philosophers and forks. Obviously, a proof

identical to that of DPh.1 can be provided for every  $i$ , so this “position index” may be considered as universally quantified. Clearly, in practice, such a parametrization could be embedded in the language by introducing notions such as the array constructor. Here, however, we are not interested in such issues that can be dealt with in quite a traditional way. Second, the proof is modularized along the following lines:

1. Proof of a periodicity property of the net, stating that if a fork is ever picked up, it will be again picked up not less than 2 and no more than 7 time units later. This property is formalized by the following theorem.

$$\mathbf{DPh.2:} \quad \left( \begin{array}{c} \text{fire}(P_{Hi\_PFi}) \\ \vee \\ \text{fire}(P_{Hi+1\_PFi}) \end{array} \right) \rightarrow \text{Futr} \left( \text{WithinF} \left( \left( \begin{array}{c} \text{fire}(P_{Hi\_PFi}) \\ \vee \\ \text{fire}(P_{Hi+1\_PFi}) \end{array} \right), 5 \right), 2 \right)$$

2. Proof that DPh.2 and the initial marking where all  $F_i$ 's store one token imply DPh.1.

Let us consider separately the two proofs.

1. The proof of DPh.2 is based on the following lemmas.

$$\mathbf{Le.1:} \quad \text{fire}(P_{Hi\_PFi}) \rightarrow \exists d (1 \leq d \leq 3 \wedge (\text{Futr}(\text{fire}(P_{Hi\_SE}), d) \vee \text{Futr}(\text{fire}(P_{Hi\_LFI}), d)))$$

Le.1 represents the fact that if a fork is picked up, it will be either used to eat or released within 3 time units. It is an obvious restating of TPN.5 of Section 3.

$$\mathbf{Le.2:} \quad \text{fire}(P_{Hi\_SE}) \rightarrow \exists d (1 \leq d \leq 3 \wedge \text{Futr}(\text{fire}(P_{Hi\_EE}), d))$$

Le.2 asserts that if the philosopher ever starts to eat, he will finish not earlier than 1 time unit and not after 3 time units. Le.2 is a simple rephrasing of TPN.2.

$$\mathbf{Le.3:} \quad \text{fire}(P_{Hi\_EE}) \rightarrow \text{Futr}(\text{fire}(P_{Hi\_PFi}) \vee \text{fire}(P_{Hi+1\_PFi}), 1)$$

$$\mathbf{Le.4:} \quad \text{fire}(P_{Hi\_LFI}) \rightarrow \text{Futr}(\text{fire}(P_{Hi\_PFi}) \vee \text{fire}(P_{Hi+1\_PFi}), 1)$$

Le.3 and Le.4 establish that, whenever a fork is released, either because the philosopher finished to eat (Le.3) or because the time-out has expired (Le.4), the fork will be again picked up after 1 time unit. Le.3 and Le.4 derive immediately from  $LB(P_{Hi\_PFi})$ ,  $UB(P_{Hi\_PFi})$ ,  $LB(P_{Hi+1\_PFi})$ , and  $UB(P_{Hi+1\_PFi})$ .

The following lemmas Le.5 through Le.8 are just a restatement of Le.1 through Le.4, by referring to philosopher  $P_{Hi+1}$  instead of  $P_{Hi}$ .

$$\mathbf{Le.5:} \quad \text{fire}(P_{Hi+1\_PFi}) \rightarrow \exists d (1 \leq d \leq 3 \wedge (\text{Futr}(\text{fire}(P_{Hi+1\_SE}), d) \vee \text{Futr}(\text{fire}(P_{Hi+1\_LFI}), d)))$$

$$\mathbf{Le.6:} \quad \text{fire}(P_{Hi+1\_SE}) \rightarrow \exists d (1 \leq d \leq 3 \wedge \text{Futr}(\text{fire}(P_{Hi+1\_EE}), d))$$

$$\mathbf{Le.7:} \quad \text{fire}(P_{Hi+1\_EE}) \rightarrow \text{Futr}(\text{fire}(P_{Hi\_PFi}) \vee \text{fire}(P_{Hi+1\_PFi}), 1)$$

$$\mathbf{Le.8:} \quad \text{fire}(P_{Hi+1\_LFI}) \rightarrow \text{Futr}(\text{fire}(P_{Hi\_PFi}) \vee \text{fire}(P_{Hi+1\_PFi}), 1)$$

We now provide the proof of DPh.2, preceded by a brief informal outline.

If philosopher  $i$  picks up the fork he may either start to eat or leave it because the timeout expires (steps 1÷2). In the former case he will return the fork within 7 time units (steps 3÷6). In the latter case the fork will be available again within 4 time units (steps 7÷9). In any case the fork will be available again for philosophers  $i$  and  $i+1$  within 7 time units (steps 10÷13). A similar reasoning applies to the case that fork  $i$  is picked up by philosopher  $i+1$  (step 14), so DPh.2 holds (step 15).

We use the following abbreviations.

$$\alpha = \text{fire}(\text{PHi\_PFi})$$

$$\rho = \text{fire}(\text{PHi\_LFi})$$

$$\beta = \text{fire}(\text{PHi+1\_PFi})$$

$$\lambda = \text{fire}(\text{PHi\_EE})$$

$$\delta = \text{fire}(\text{PHi\_SE})$$

$$\Pi = \text{Futr}(\text{WithinF}(\alpha \vee \beta, 5), 2)$$

- |  |   |
|--|---|
| 1. $\alpha \vdash \exists d (1 \leq d \leq 3 \wedge (\text{Futr}(\delta, d) \vee \text{Futr}(\rho, d)))$   | <b>Le.1</b>   |
| 2. $\alpha \vdash \exists d (1 \leq d \leq 3 \wedge \text{Futr}(\delta, d)) \vee \exists d (1 \leq d \leq 3 \wedge \text{Futr}(\rho, d))$  | <b>1, EI, TAUT<sup>1</sup>, <math>\exists</math>-intro</b>        |
| <i>We reason by Case Analysis: let <math>\Gamma = \exists d (1 \leq d \leq 3 \wedge \text{Futr}(\delta, d))</math> and <math>\Phi = \exists d (1 \leq d \leq 3 \wedge \text{Futr}(\rho, d))</math>. Then</i> |   |
| 3. $\alpha; \Gamma \vdash \exists d (1 \leq d \leq 3 \wedge \text{Futr}(\exists d1 (1 \leq d1 \leq 3 \wedge \text{Futr}(\lambda, d1)), d))$  | <b>Le.2</b>   |
| 4. $\alpha; \Gamma \vdash \exists d (1 \leq d \leq 3 \wedge \text{Futr}(\exists d1 (1 \leq d1 \leq 3 \wedge \text{Futr}(\text{Futr}(\alpha \vee \beta, 1), d1)), d))$  | <b>3, Le.3</b>  |
| 5. $\alpha; \Gamma \vdash \exists d (3 \leq d \leq 7 \wedge \text{Futr}(\alpha \vee \beta, d))$  | <b>4, Th.iv, Th.vii, EI, TD, Ax.8, <math>\exists</math>-intro</b> |
| 6. $\alpha; \Gamma \vdash \text{Futr}(\text{WithinF}(\alpha \vee \beta, 4), 3)$  | <b>5, TD, Ax.8, WithinF definition</b>                            |
| 7. $\alpha; \Phi \vdash \exists d (1 \leq d \leq 3 \wedge \text{Futr}(\text{Futr}(\alpha \vee \beta, 1), d))$  | <b>Le.4</b>   |
| 8. $\alpha; \Phi \vdash \exists d (2 \leq d \leq 4 \wedge \text{Futr}(\text{Dist}(\alpha \vee \beta, d)))$   | <b>7, EI, Ax.8, TD, <math>\exists</math>-intro</b>                |
| 9. $\alpha; \Phi \vdash \text{Futr}(\text{WithinF}(\alpha \vee \beta, 2), 2)$  | <b>8, TD, Ax.8, WithinF deftn</b>                                 |
| 10. $\alpha; \Gamma \vdash \text{Futr}(\text{WithinF}(\alpha \vee \beta, 5), 2)$   | <b>6, Th.xv</b>   |
| 11. $\alpha; \Phi \vdash \text{Futr}(\text{WithinF}(\alpha \vee \beta, 5), 2)$   | <b>9, Th.xiv</b>  |
| 12. $\alpha \vdash (\Gamma \wedge \Phi) \rightarrow \text{Futr}(\text{WithinF}(\alpha \vee \beta, 5), 2)$  | <b>10, 11, CA</b>   |
| 13. $\alpha \vdash \text{Futr}(\text{WithinF}(\alpha \vee \beta, 5), 2)$   | <b>2, 12, MP</b>  |
| 14. $\beta \vdash \text{Futr}(\text{WithinF}(\alpha \vee \beta, 5), 2)$  | <b>derivation similar to 1-13</b>                                 |
| 15. $\vdash (\alpha \vee \beta) \rightarrow \text{Futr}(\text{WithinF}(\alpha \vee \beta, 5), 2)$  | <b>13, 14, CA</b>   |

2. Now the proof of DPh.1 is obtained from DPh.2 through the following steps. Let  $\gamma = \alpha \vee \beta$ : then

- |   |                                 |
|---|---------------------------------|
| 1. $\text{Alw}(\gamma \rightarrow \text{Futr}(\text{WithinF}(\gamma, 5), 2))$ | <b>DPh.2, TG</b>                |
| 2. $\iota \vdash \text{fire}(\text{itFi})$                                    | <b><math>\wedge</math>-elim</b> |
| 3. $\iota \vdash \exists d (d \leq 1 \wedge \text{Futr}(\gamma, d))$          | <b>2, UB(PHi_PFi)</b>           |
| 4. $\iota \vdash \text{Futr}(\gamma, 1)$                                      | <b>2, 3, TPN.1</b>              |
| 5. $\iota \vdash \text{Futr}(\text{AlwF}(\text{WithinF}(\gamma, 7)), 1)$      | <b>1,4, Th.xii, TT</b>          |
| 6. $\iota \vdash \text{Lasts}(\text{WithinF}(\gamma, 1), 1)$                  | <b>4, Th.xi</b>                 |
| 7. $\iota \vdash \text{Lasts}(\text{WithinF}(\gamma, 7), 1)$                  | <b>6, Th.xiv</b>                |
| 8. $\iota \vdash \text{AlwF}(\text{WithinF}(\gamma, 7))$                      | <b>4, 5, 7, TD</b>              |

### Closing remark on the use of the axiom system

The reader who is concerned about the amount of details involved in the derivation of relatively simple statements such as those proved in the previous sections should look at these examples in the same light as traditional mathematical logic: [65, 30, 19] *introduce* the formalization of classical mathematical theories such as group theory, Peano's arithmetics, etc: step by step proofs are given just to *illustrate* axiom systems, but this does by no means imply that this is the way things should be carried out in practice, where many obvious details can be informally accepted on the basis of intuitive

<sup>1</sup> The tautology  $\alpha \wedge (\beta \vee \gamma) \leftrightarrow ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma))$  is used.

evidence and skipped. This remark on the practical use of formal methods will never be over-emphasized [60, 38, 56]. Thus, we will further address it in Section 6.

## 5. Related works

As we already mentioned, several formalisms to support a rigorous analysis of concurrent systems have been proposed in the recent literature. These include:

- Formal languages for the specification of system properties. Among them: algebraic languages as CCS[68], or LOTOS [14]<sup>1</sup>, logic assertions in the Hoare style [47] [77] [78] [53], temporal logic [81, 62], [21], and several variations thereof. Among such variations the most relevant one is probably the introduction of the *next operator* [55, 64].
- Operational models such as those based on finite state automata [44] [97] and those based on Petri nets [79, 84, 34, 52].

Many of them have been or can be extended in such a way to cope explicitly with time measure, so that strict time bounds typical of hard real-time systems can be rigorously expressed and analyzed. Among such extensions let us mention the following ones.

- Extensions of CCS's process algebras [94, 26].
- Extensions of temporal logic. In some cases time is derived on the basis of the next operator, what yields a discrete time domain [55]; in other cases it is added as a more or less explicit new variable or function such as in RTL[51], RTTL[76], TCTL[4]. TRIO[37] and the already mentioned MTL[54] are new logic languages where the classical operators of temporal logic can be derived from basic operators such as *Dist* or *Futr* and *Past*. More comments on the features of some of these languages will be given in the following of this section. Alur and Henzinger [5] define a real-time logic that is based on propositional calculus rather than on first-order predicate calculus, namely Timed Propositional Temporal Logic (TPTL). Since in TPTL quantification over time is not allowed, the language is decidable.
- Extension of finite state machines. These often include not only the possibility of specifying time bounds for transition occurrence, such as Timed Constrained Automata [67] and, more generally, the Timed Graphs [4, 6], but also the possibility of associating variables—possibly with infinite domains—with states and predicates involving such variables with transitions, so that, strictly speaking, the resulting machine is not anymore “finite state”. An instance of this type of machines are the Extended State Machines by [75].
- Extensions of Petri nets. These have been done in several ways with different levels of generality. One of the most widely known and adopted models of timed Petri nets is [66] which has been used in this paper too for this reason. Other examples of temporal extensions of Petri nets are [82, 48, 98], [22, 83, 89, 85, 86]. In [39] a more general model that includes and extends all

---

<sup>1</sup> CCS and the derived language LOTOS are often classified as descriptive languages since their semantics is originally defined in an algebraic style. They are also presented, however, in an operational light on the basis of the notion of process, so that they could also be included in the latter category.

previously studied models is proposed. It also gives hints on combining temporal extensions with extensions that allow tokens to carry values such as in PrT [34]. A comparison between several timed Petri nets can be found in [70]. Rather often, the attention in the analysis of Petri net properties—whether extended with time or not—is restricted to bounded nets, i.e. to nets where the number of tokens that can be stored into a place is bounded a priori by a given constant. This makes the model computationally equivalent to finite state machines, what makes property analysis easier but also decreases their expressive power and generality. [18, 17], [91], [93], [24] are examples of papers referring to this restricted class of Petri nets.

It is worth emphasizing that several of the above formalisms to model and express properties of real-time systems have already been applied to industrial projects. This is the case, for instance, of finite state machines and of Petri nets. In Sections 2 and 6 we mention also some initial application of TRIO to industrial projects. In a few cases the above models have also been used as the basis for commercial tools in this field. For instance, PROTOB [15] and RdP by Verilog are based on Petri nets, Statemate [44], ASA by Verilog and Teamwork by CADRE are based on finite state machines or extensions and/or integrations thereof with other formalisms.

Table 1 summarizes the main features of the above formalisms.

Formalism		Explicit Reference to Time	Executability	Domains Cardinality	Decidability
<b>Descriptive Formalisms</b>	CCS [68]	No	Yes	Infinite	No
	CTL [21]	No	No	Infinite	Yes
	Pure TL [81]	No <sup>1</sup>	No	Infinite	No
<b>Real-Time Descriptive Formalisms</b>	Timed CCS [94]	Yes	No	Infinite	No
	TCTL [4]	Yes	No	Infinite	No
	RTL [51]	Only Discrete	No	Infinite	No
	[33]	Only Discrete	No	Infinite	No
	TL[55]	Only Discrete	No	Infinite	No
	RTTL [76]	Only Discrete	No	Infinite	Only with finite domains
	MTL [54]	Yes	No	Finite and Infinite	No
	TPTL [5]	Only discrete	No	Infinite	Yes
TRIO [37]	Yes	Yes	Finite and Infinite	Only with finite domains	
<b>Operational Formalisms</b>	Statemate [44]	No	Yes	Finite	Yes
	ESM [76]	Yes	No	Infinite	No
	T-Graphs [4] <sup>2</sup>	Yes	No	Infinite	No
<b>Petri nets or similar</b>	HMSM [33]	Only discrete	Yes	Finite	Yes
	TB-Nets [39]	Yes	Yes	Infinite	No
	PRt [34]	No	Yes	Infinite	No
	TPNets [66]	Yes	Yes	Infinite	No

**Legend:** To two columns *Executability* and *Decidability* are defined as follows. A logic language is *decidable* if there exists an algorithm to state the satisfiability of a formula. A logic language is *executable* if a method—not necessarily terminating—is explicitly supplied (at the best of our knowledge) to generate a model of a given formula. For instance, PROLOG is executable but not decidable; PTL is decidable but not–yet–executable. Similarly, an

<sup>1</sup> In the case where the next operator is given, time can be implicitly derived as a discrete variable whose unit is the occurrence of a single next.  
<sup>2</sup> Other models of the same authors related to Timed Graphs are about Timed Buchi Automatas [6] and more general [29]

operational model is *decidable* if there is an algorithm which states whether a state is reachable from another one or not. In principle, all operational models are executable since it is possible to associate them with abstract machines—whether deterministic or not, whether terminating their execution or not—that simulate them. The marks given in the table, however, refer to the existence of implemented simulators at the time being.

**Table 5.1.** A summary of models for the description of concurrent and real-time systems

Furthermore, a strong attention was recently paid to the formal analysis and proof of system properties, which is the main issue of this paper.

In the traditional literature, concurrent systems properties are mainly classified as *liveness* and *safety properties*. They are often intuitively defined as follows [78, 76]:

- $\varphi$  is a *safety formula* if it is of the form “nothing bad will ever happen”
- $\varphi$  is a *liveness formula* if it is of the form “something good will eventually happen”<sup>1</sup>

The above statements, however, are often formalized in different and not always consistent ways. On the one hand a safety formula is often defined as a formula of the type  $A\text{lwF}(\varphi)$ . This can be shown to be equivalent to other formalizations under reasonable assumptions.

Things become more critical, however, when dealing with liveness, mainly with reference to nondeterministic computation models. For instance, in [64] total program correctness is given as an example of liveness property. If we extrapolate this definition to nondeterministic programs—e.g., coded in Dijkstra’s language [28]—we wish to state total correctness as a property that holds *for any* possible execution of the nondeterministic program. This contrasts with the traditional definition of liveness in Petri nets, i.e., the property that, starting from any state reachable from a given initial state, *there exists*, for any given transition, a firing sequence where the transition eventually fires: this does not require that *every* firing sequence starting from that state enables the given transition. Similarly, it is not clear how to distinguish, in the above classification, the *risk* of deadlock in a Petri net (i.e., there is a firing sequence that leads to a deadlock) from the *certainty* of deadlock (i.e., all firing sequences will eventually lead to a deadlock).

For this reason, we propose instead the following classification.

- *Universal properties* are properties claimed *for every possible evolution* of the described system;
- *Existential properties* to denote those properties claimed *for at least one evolution* of the described system;
- *General properties* to denote those properties that can be expressed only by intermixing in more complex ways universal and existential quantifications over execution sequences and their states.

Universal properties include safety in the traditional sense (e.g., “the system will never enter a failure state”, “a PN marking will always be reached within 2 seconds from a given initial marking”, boundedness, etc). They also include, however, properties that are often classified as liveness properties such as program termination—whether we are talking about deterministic or nondeterministic

---

<sup>1</sup> Clearly, the terms “good” and “bad” must be intended in the sense that we are trying to prove that property. Thus, once we proved that “a system will eventually crash, no matter what will happen in its environment” we will be happy to have proved a liveness property.

programs. In fact, for such a property we wish to state that it holds for any possible execution of the program. Unavoidable deadlock is also an example of universal property.

For Petri nets, examples of existential properties are reachability (it is possible to reach a given state (i.e., marking) from a given state) and deadlock *risk* (it *may* happen that a deadlock occurs); etc.

Liveness in the traditional sense of Petri net literature is an example of general property since it states that *for every* firing sequence, *there exists* another firing sequence, starting from the marking reached by the former sequence, that enables a given transition. Another example of general property is Petri net equivalence, as stated, e.g., by the following sentence: "For every firing sequence in net PN1 that leads from state  $s_{1i}$  to state  $s_{1f}$  within time  $t$ , there is a firing sequence in net PN2 that leads from state  $s_{2i}$  to state  $s_{2f}$  within time  $t$ ".

Most part of the present literature, including this paper, deals with the proof of safety properties or at most of universal properties, which, we saw, properly include safety properties. This is due to the fact that axiom systems are well suited to prove *valid formulas*, i.e. properties that hold for every possible system evolution.

Next, we briefly review, at the best of our knowledge, a significant sample of such a literature.

1. [91] proves universal properties of Temporal Petri nets, i.e., Petri nets whose firing sequences are restricted by temporal propositional formulas. The proofs are derived by building regular  $\omega$ -expressions and the reachability graph associated with a given net. However, despite their name, Temporal Petri nets cannot describe explicit time requirements. Moreover, as the author acknowledges, if the reachability graph [79] of the underlying Petri net is not finite (i.e., the PN is not bounded) it cannot be regarded as a Büchi-automaton and the translation into  $\omega$ -expressions is not possible. In a previous work, [90], a simple axiomatization of –again, bounded– Temporal Petri nets is provided, where a given net is analyzed by manipulating formulae directly using "axiom-like" properties<sup>1</sup> and the topology of the underlying reachability graph.

2. [10, 11] presents a temporal analysis of reachability in an algorithmic way. The authors use a "state enumeration approach" to analyze Timed Petri nets [66]. Again, this analysis is possible only when the reachability graph, defined by the authors as an extension of the classical reachability graph, is finite. This happens when the net is bounded and the firing time bounds for each transition are integers or rational numbers. The properties to be analyzed are expressed in an informal way rather than by using a formal specification language.

3.[58] presents procedures to analyze safety, recoverability, and fault-tolerance properties (all of them are universal ) stated for Timed Petri nets [66] augmented with hazard states. The work is based on the reachability graphs defined in [10] and therefore it shares its limitations. Also, it lacks as well a formal specification language.

4.[45, 46] define the Temporal Predicate Transition Nets combining Temporal Petri nets [90] with Predicate Transition Nets [34]. Even in this case, time is not considered explicitly. [46] outlines an algorithm for translating a predicate transition net into a temporal logic proof system, and gives a

---

<sup>1</sup> Authors' words.

refutation proof technique. The proof system consists of an inference rule for each individual transition, and an additional rule for the conflict case. An ad-hoc restriction in the application of this rule is necessary to avoid the risk of contradiction that we pointed out in Section 3 for conflicting transitions.

5.[33] presents the Hierarchical Multi-State (HMS) machines. It is a PN-like formalism with strong emphasis on decompositional description. The operational formalism is paired with an assertion language that allows to state "precondition rules" and "postcondition rules" in the style of temporal logic. The pairing between the HMSs and the assertion language is fairly rigid from the point of view of temporal evolution since every firing of a single transition—or concurrent firings of several transitions—implicitly increments the time variable of 1 unit. From a given execution path of an HMS, and from given pre- and postcondition rules, a set of inequalities is derived that represents the time conditions for the feasibility of the given path. Thus, solving the inequalities determines whether a potential plan can be scheduled. Proving path feasibility is a limited case of property proving, where the considered property is both universal and existential (it is the same as proving that a property holds for a fixed instantiation of a given variable in first-order theories). From an intuitive point of view, however, it seems more appropriate to classify it as an existential property.

6.[21] describes an approach based on the verification of finite-state concurrent systems by model checking their temporal logic specification, while [16] presents a technique for efficiently verifying systems with an extremely large number of states. However, these verification methods are based on the CTL language, a propositional branching temporal logic which does not provide metric on timing operators[20], and is thus unable to describe the quantitative aspects of real-time systems. [4] extends CTL model-checking to the analysis of real-time systems. The authors define Timed CTL by extending the syntax of CTL to allow quantitative temporal operators. Quantifications over the temporal domain are forbidden, however. The operational model introduced is Timed Graphs, i.e., state-transitions graphs with time constants that bound transition occurrence. The important result claimed by the authors is that the model-checker complexity "does not blow up" when dealing with dense time.

7. [18] proposes a simple translation of the reachability graph of a bounded Petri net into a Moore machine and then analyzes net properties by exploiting the model-checker defined in [21]. As usual, the model-checker can be applied only to finite state-machines. A similar approach is taken by [25] for Condition/Events Nets.

A previous work of the first authors [17] applies a similar method to Timed Petri nets. These nets are a slight modification of classical Merlin and Farber's model [66] that allows to deal explicitly with the concurrent firing of several transitions (true concurrency approach [84]) and prevents a transition from firing if a place in its postset has already a number of tokens equal to a predefined bound (thus, the model is finite state). Then, the specification language RTTL (Real-Time Temporal Logic) is given. It enriches the language of [21] by introducing explicit definition of time intervals. A few fairly ad-hoc operators are added to the logical language. In particular the composed formula  $A||B$  is used to denote that A and B occur "in parallel" (they are the logical expression of the contemporary firing of

two concurrent transitions); the formula  $A;B$  is used to denote that  $A$  and  $B$  occur simultaneously but with an ordering relation between the two ( $A$  must *logically* precede  $B$ ). ‘||’ is called the “parallel operator” whereas ‘;’ is called the “domino” operator. These operators allow to avoid the tricky situations that we mentioned in Section 3.1 and 3.4 when dealing with contemporary events but compel the user to give logical formulas that are strongly biased by the underlying operational model so that the logic language can hardly be used to *specify system requirements*, possibly before or independently from its design. A method to prove universal and existential properties of systems is then outlined. Once more, it is based on the construction of a finite reachability graph from the given net and a further analysis thereof against the stated assertions. It is a slight extension of the method presented in [21] where the reachability graph is viewed as a finite machine.

A few technical inaccuracies in the presentation prevent a thorough assessment of the paper.

8. [75, 76] uses Extended State Machines (EMS) as operational model and—yet another—RTTL (Real-Time Temporal Logic) as specification language. EMSs consist of a finite state “kernel” which is augmented by introducing variables with arbitrary domains. Transitions leading from one state to another one are labeled by a *guard* (a first-order formula on state variables) and an *operation* (a value assignment to state variables). Any transition can occur only if its guard evaluates to true. In such a case the global state is updated by executing the operation associated with the transition. RTTL is basically a temporal logic equipped with the next operator. Both EMSs and RTTL are enriched by an explicit time variable that can be used in RTTL formulas and can be tested and updated by guards and operations, respectively. The time variable can have only discrete domains.

It must be emphasized that EMS transitions do not necessarily update the time variable, so that zero-time transitions are allowed, as well as in normal Timed Petri nets. As a consequence, however, the next operator refers to the *next state* in the evolution of the EMS but *not necessarily to the next time instant*. This allows to treat in a consistent way zero-time transitions but, in our opinion, is slightly counterintuitive and generates some risk of confusion in the user who is compelled to pay deep attention to clearly distinguish the logical evolution through a sequence of states—which is specified by the next operator but may occur without any time elapsing—and temporal evolution, which is described by a “normal” variable. RTTL also departs from the original philosophy of temporal logic since time is considered as a component (albeit distinguished) of the modelled system.

An axiom system is then provided to allow the proof of—universal—properties of EMSs stated as RTTL formulas and decision procedures are given for the finite state cases. It is not clear, however, what the axioms should be if the considered EMS were nondeterministic (recall from Section 3 that nondeterminism in the computational model implies a risk of contradiction in an axiom system based on traditional Hoare’s approach)

Other approaches aim at proving *properties of programs*, as opposed to *systems* modeled by finite state machines, Petri nets. etc. The two approaches, however, are clearly related to each other. Among them, let us recall the following ones.

9. [47][77][42][78][8] apply a classical Hoare's approach to the verification of concurrent, Pascal-like, programs.

10. [61] presents the foundations of the use of temporal logic for the verification of programs [64]. In [81] it is shown how the system can be applied to some state-machines as C/E nets [84].

11. [43] and [49] are, at the best of our knowledge, the only cases of formal analysis of concurrent programs that explicitly take time into account. In [43] the used programming language is a concurrent version of Dijkstra's language [28] (see [32] for a few remarks on its formal semantics), and the assertion language is a classical first-order logic language. [49] uses a real-time parallel programming language akin to Occam [50] and an extension to MTL [54] as assertion language.

In general, we claim more generality of the former approaches since the proof of program properties can be easily reduced to the proof of properties of a system modeled by an abstract machine, once rules—and even algorithms—are given to translate a programming language into an operational model (see, e.g., the definition of Ada's task system semantics in term of timed Petri nets as suggested in [59] and [36]).

In summary, it is clear that, after a long time during which formal methods literature almost ignored the field of real-time systems, there is now much attention thereto, although some of the present research results appear in a fairly preliminary stage and often rather unaware of each other. With respect to the several contributions to the formal analysis of real-time systems, we claim, for our approach (we acknowledge, of course, that some of the claims below are based on subjective evaluations):

- *More generality.* In fact, at the best of our knowledge, our approach is the only one that allows to deal with both discrete and continuous time and to analyze even unbounded nets (i.e, infinite state machines). It is also easily extensible to even more general timed Petri nets such as TB nets [35] and their integration with PrT nets [34] (ER nets [39]). See the conclusions for more remarks on this extension.
- *More naturalness.* We believe, in fact, that an asynchronous model such as Petri nets is often more suitable to describe the evolution of several processes with different dynamic behaviors—such as a plant and a set of microprocessors for its control—that synchronize each other only occasionally, maybe through message passing. Similarly, we believe that a language such as TRIO is more suitable to formalize real-time system properties than languages where time is an explicit extra variable or where time is derived by counting the number of occurrences of the *next* operator (which acts as the successor operator in Peano's formalization of natural numbers).
- *More flexibility.* This should stem from the fact that our axiom system is general enough to cover many semantic aspects of timed Petri nets, including their possible extensions, but can be drastically simplified when suitable hypotheses restrict the class of considered nets (e.g. 1-bounded nets, no zero-time firing). Also, the axiom system is clearly undecidable in the general case, but suitable decision algorithms could be designed to cope with several particular cases (see the conclusions for more remarks on this topic).

To help focusing attention on the the main features of the presented literature Table 2 gives a synopsis of most of the above mentioned contributions. Instead, Table 3 gives a few more details by restricting the attention to the works that are more closely related with our own, namely the works based on the so called *dual-language approach* [61][76], where one language is *prescriptive* and algorithmic in nature, and the other one is an *assertional* language, which is descriptive enough to specify system requirements. Among these contributions, more emphasis is given to those using Petri nets as the prescriptive language. For more informations about the current trends on automatic verification of finite state machines, the reader is referred to [87].

Operational Model		Type of Specification Language.			Type of analysed properties		
		No Formal Language	Logic out	with- metric	Logic with metric	Universal	Existential
PN- Based	Berthomieu & [10, 11]	⊗				⊗	
	Suzuki & [90, 91]		⊗		⊗		
	He & [46]		⊗		⊗		
	Chang & [18]		⊗				⊗
	Chang & [17]			⊗			⊗
	Leveson & [58]	⊗			⊗		
	Gabrielian & [33]			⊗ (limited)		⊗	
	TRIO - PN, This paper			⊗	⊗		
Finite State Automata	Ostroff[76]			⊗	⊗		
	Clarke & [21]		⊗				⊗
	Alur & [4]			⊗			⊗
Conc. Lang. Based	Gries, Owicki, Lamport [42, 77, 78]		⊗		⊗		
	Hoare [47]		⊗		⊗		
	Haase [43]			⊗	⊗		
	Manna & [80, 61, 62]		⊗		⊗		

Legend: General properties are restricted to the class expressible by branching temporal logic.

**Table 5.2.** A comprehensive view of the literature on the analysis of concurrent and real-time systems.

Reference	Oper. Language	Assert. Language	Real Time Expr.	Proof Method	Prop- erties <sup>1</sup>	Aut. Verif	Description	Com- ments
Alur & [4]	Timed Graphs (Timed Automatas)	TCTL Timed CTL	Yes With a posi- tive dense domain	Extended Model Checking	General	Yes	Adequate [21] method to deal with metric time	Dense do- mains without computa- tional blow up
Chang & [18]	Bounded Petri nets	Proposi- tional Branching Temporal Logics	No	Model Checking	General	Yes	The reachability Graph is trans- formed into a Moore Machine. Then, [21] algorithm is applied	Applicable only to bounded nets. Also extended to Timed Petri nets

<sup>1</sup> Again, here general properties refer to the class expressible with branching temporal logic

Clarke & [21]	State Machine	CTL (A branching temporal propositional logic)	No	Extended Model Checking	General	Yes	Theorem proving by model-checking the state graph.	Foundational work on model-checking. More efficient extensions [16]
Damm & [24]	AADL-Nets (C/E nets augmented with predicates on places and different kinds of arcs)	MCTL (A modular extension of CTL)	No	An adequate Model Checking algorithm as [21] is given with compositional rules.	Existential	Yes	Model-checking for proving modules correct; proof system for module composition	Only for bounded Nets
Dannelutto & [25]	Finite State Machine or C/E Petri nets	Propositional Branching Temporal Logics	No	Model-Checking as [21]	General	Yes	The same description as [21] holds	Only for bounded nets
Gabrielian & [33]	Hierarchical Multi-State Machine. (Similar to Petri nets)	Pre and Post-condition Laws	Yes Discrete temporal Domain	Inequalities Resolution	Restrictive Existential	No	Determines if there exist a scheduling for a given path (a firing sequence)	Only finite sequences are verifiable. Accent on Top-Down decomposition. Does not permit disproving
He [46, 45]	Temporal Predicate Transition Nets (A mixing of Temporal PN [91] and PRt nets[34].)	Linear Temporal Logic	No	Refutation Proof System derived from the net topology	Universal	No	Every transition defines an inference rule. Special rule for the conflict case.	An ad-hoc restriction is necessary to avoid the risk of contradiction
Manna & [61,62,81,64]	State Machine	Linear Temporal Logic	No	Proof System	Universal	No	Proof system for program verification.	A foundational work on temporal logic program verification.
Ostroff [76]	Extended State Machine	RTTL. Real Time Temporal Logic (Classical Linear TL plus time variable	Yes. Discrete temporal domain. It deals with N	Proof System using pre and post conditions	Universal	Only for a small class of properties	Compositional description proof system as [61]. Time as explicit variable. Proof Diagrams	Two orthogonal concepts of time: States sequences and System time.
Suzuki & [90]	Temporal Petri nets (Firing Sequences restricted with a temporal logic formula.	Linear Temporal Logic	No	Based on axioms-like and reachability graph analysis	Universal	No	Properties proved using the restrictive formula and the given axioms together with the reachability graph topology	Only for bounded nets, i.e, finite reachability graph .

Suzuki [90]	Temporal Petri nets	Linear Temporal Logic	No	Semi-formal proofs based on Regular Expressions and Büchi Automata	Universal	No	Proofs inferred from expressions structure and reachability graph topology	Only for bounded nets, i.e., when the reachability graph is finite.
Tuominen [93]	Elementary Nets (A kind of C/E Nets)	DPDL. Deterministic Propositional Dynamic Logic.	No	A simplified version of the Tableau Method.	Existential	Yes	The net is translated to a DPDL formula and then a tableau method is used .	Only for 1-safe nets
This paper	Timed Petri nets	TRIO	Yes. Dense and Discrete Domains	Proof System	Universal	No	A Timed Petri net axiomatization is given in terms of TRIO. Then, a sound proof system is applied.	

**Table 5.3.** A more detailed view of formal approaches to the analysis of real-time system properties.

## 6. Conclusions

We have presented an approach to proving properties of concurrent and real-time systems. Such properties are expressed in TRIO—an extension of temporal logic—and systems are modeled as timed Petri nets.

The approach is based on an axiomatization of TPNs in terms of TRIO formulas, what turned out to be a non trivial job due to the adoption of a nondeterministic and possibly unbounded computation model.

We have been able to apply the method to the proof of some non-trivial properties of classical benchmarks for the analysis of concurrent and real-time system. A similar success has been obtained for other examples not reported here.

We do acknowledge, however, that the contents of this paper delineate just a few initial steps (the “foundations”) on the way of practical application of the approach to real-life cases (as it is the case with all related literature). Much more work is needed to reach a maturity that is comparable with more established formal methods (yet still under debate) such as VDM, [1, 13] and Z [88] in more traditional application fields. Let us classify such a work into the following categories:

1. Moving from simple cases and hypotheses to more general frameworks suitable to cope with more practical situations. Typically this requires introducing many more details, enriching and generalizing methods, but does not involve solving open scientific problems.
  2. Gaining generality and power by solving problems that are presently still obscure.
  3. Building tools that facilitate practical application of the method.
1. In the first category there are:
    - 1.1. Using a more general TPN model such as TB and/or ER-nets [39].
    - 1.2. Introducing modularization and higher level mechanisms to allow coping with complex systems. Both TPNs and TRIO have already been extended with such mechanisms [73, 40, 41, 23, 69]. We do not expect any major problem in building “proof methods in the large” that allow to decompose the analysis of a modular system into separate proofs of single modules. The examples presented in Section 4 should have shown, however, that a modular attitude can be exploited even without any linguistic mechanism supporting it.
    - 1.3. Applying the method to real-life case studies. Again, this has already been done separately and successfully by using both TRIO and TPNs for the description and analysis of systems developed within industrial environments [72].
    - 1.4. Deriving proof methods for other formalisms by developing suitable translations from a formalism into another one: for instance, a proof method for Ada programs can be obtained on the basis of its translation into augmented PNs as suggested in [59] and [36].
  2. In the second category there are:
    - 2.1. Dealing with properties other than universal properties to achieve more generality. This issue is related with the following ones, which, however, deserve attention by themselves.

- 2.2.** Using alternative logic models. For instance, we saw in Section 5 that branching logic [9, 20, 27] is well suited to deal with universal, existential, and a restricted class of general properties of nondeterministic computations.
- 2.3.** Trying different TRIO axiomatizations of TPNs. In fact, we should emphasize that the axiomatization presented here is just one among several possible alternatives, whose relative advantages and disadvantages have still to be thoroughly investigated. In particular, we plan to look for an axiomatization that is closer to the traditional terminology of Petri nets. For instance, we would like to express the marking of a net in a more immediate way and to relate the firing of transitions directly to such a predicate instead of using the *token* predicate whose number of arguments may compel the user to take care of many details. We have seen in Section 3.4, however, that this goal hides a few subtle problems. A few approaches are presently under investigation. Basically, they consist of excluding truly contemporary events. On the contrary, they assume the possibility only of “almost contemporary events”. The approach has some intuitive appeal and some mathematical difficulties. It implies that the underlying *essential time domain* be dense in nature, but allows, if needed, to derive a discrete time domain as an approximation and abstraction of the dense domain which can be completely hidden to the user.
- 2.4.** Extending TRIO to a higher order logic. This would produce benefits both in the use of TRIO itself as a specification language and in the axiomatization and property expression of TPNs. In fact, it would be useful to quantify concepts such as firing sequence, reachable marking, etc. that are presently either predicates or formulas. Incidentally, this possibility would also allow to express and to prove properties of existential and of general type. Several more or less traditional approaches to express such higher order concepts may be followed.
- 3.** Here, again, we plan to build on top of tools that already exist both for TRIO and for TPNs. Existing–prototype–environments for TRIO and TPNs include editors, interpreters, and verifiers for the writing and prototyping of specifications. By following, for instance, the lines of the ASLAN specification environment [53], tools supporting the use of the proposed method should allow:
- Building system descriptions as TPNs and “annotating” them with assertions written in TRIO;
  - Deriving, in a semiautomatic way, intermediate assertions that help the construction of a complete correctness proof;
  - Verifying the correctness of the proofs that are built in such an interactive way.
- In the construction of these tools, we expect major benefits from the integration of the axiomatic–deductive–approach and of the model-theoretic approach, which is based on the interpretation of formulas on given domains to check their validity<sup>1</sup>. Thus, a complete proof of

---

<sup>1</sup> We have already developed a tool that decides the validity of TRIO formulas on finite discrete domains. It is based on the semantic approach described in [31]. It has, however, exponential complexity with respect to the number of quantifications in the formula (the problem is NP-complete).

a given property could be semiautomatically derived by alternating formulas whose validity is verified by a TRIO interpreter and deductions based on the axioms given in Sections 2 and 3.

The above points 1, 2, and 3 should have shown that our approach to the exploitation of formal methods in practical cases is strongly based on the belief that *formalisms are a useful tool to increase confidence on intuitive reasoning in critical cases, not an alternative thereof*. As it happens even in pure mathematics, most truths can be derived on the basis of evidence and of informal deductions, but tricky situations can be avoided by spending some more effort in the application of formal details when dealing with intricate cases. In our experience with several real-life case studies, the “purely formal part” of any design document amounts in general to a small part thereof, but quite often it replaces or complements a former, less formal, version that exhibited some critical troublespots.

The benefits of formal methods can be further enhanced by the application of (semi)automatic tools which, often, nicely complement—but, again, do not substitute—human reasoning in that they can easily take care of many clerical and error prone details, thus helping focusing attention on the creative aspects of the analysis and design process.

## Acknowledgments

This research has been developed in the context of a long term project on real time systems which involves people from several industries. In particular, ENEL and CISE cooperate to the development of the specification language TRIO, of its environment, and to its application to real-life projects. The present work highly benefited from discussions with several colleagues, in particular Pier Luigi San Pietro. Special thanks are due to Sandro Morasca, Mauro Pezzè and Daniel Yankelevich.

## References

1. Real-Time Systems, Special Issue of IEEE Transactions on Communications, May 1991.
2. 4th International Workshop on Software Specification and Design, IEEE, Monterey, USA, April 1987.
3. 6th International Workshop on Software Specification and Design, IEEE, Como, Italy, October 1991.
4. R.Alur, C.Coucubetis, D.Dill, Model Checking for Real-Time Systems, in 5th IEEE LICS 90, 1990, pp.414-425.
5. R.Alur, T.A.Henzinger, Real-Time Logics: complexity and expressiveness, Tech. Rep. STANCS901307, Department of Computer Science and Medicine, Stanford, CA, Appeared in 5th IEEE LICS'90, 1990.
6. R.Alur, D.Dill, Automata for modeling real-time systems, ICALP'91 (1991).
7. A.Ambler, D.Good, W.Browne, R.Cohen, C.Hoch, R.Wells, Gypsy: A language for specification and implementation of verifiable programs, Proc. of ACM Conference on Language Design for Reliable Software-SIGPLAN Notices 12, 3 (march 1977), 1-10.
8. K.Apt, Ten years of Hoare's Logic: A survey - Part I, ACM-TOPLAS 3, 4 (Oct 1981), 431-483.

9. M.Ben-Ari, A.Pnueli, Z.Manna, The temporal Logic of Branching Time. *Acta Informatica*, 20 (1983).
10. B.Berthomieu, M.Menasche, An enumerative approach for Analyzing Time Petri Nets, In *Proc. of IFIP Congress*, Sept 1983.
11. B.Barthomieu, M.Diaz, Modeling and verificaiton of time dependent systems using time Petri nets, *IEEE-TSE* 17, 3 (March 1991), 259-273.
12. D.Bjorner, C.Jones, *Formal specification and software development*, Prentice Hall International, London, 1982.
13. D.Bjorner, S.Prehen, Software Engineering aspects of VDM, the Vienna Development Method, *Theory and Practice of Software Technology* (1983), 85-134.
14. T.Bolognesi, E.Brinskma, Introduction to the ISO specification Language LOTOS, *Computer Networks and ISDN Systems* 14, 1 (1987), 25-29.
15. G.Bruno, M.Marchetto, Process-translatable Petri Nets for the rapid prototyping of process control systems, *IEEE-TSE* 12, 2 (Feb 1986).
16. J.Burch, E.Clarke, K.L.M.Millan, D.Dill, H.Hwang, Symbolic Model Checking:  $10^{20}$  states and beyond, in *5th symp. on Logic In Computer Science*, IEEE, July 1990, pp.428-439.
17. C.Chang, H.Huang, C.C.Song, An approach to verifying concurrency behavior of real-time systems based on time Petri nets and temporal logic, in *InfoJapan 90*, IPSJ, 1990, pp.307-314.
18. C.Chang, H.Huang, On transforming Petri net model to More Machine, In *COMPSAC 90*, IEEE, 1990, pp.267-272.
19. A.Church, *Introduction to Mathematical Logic*, Princeton University Press (1956).
20. E.M.Clarke, A.E.Emerson, Design and Synthesis of Synchronization skeletons using branching temporal logic. in *IBM Logics of programs workshop*, Springer Verlag, 1981.
21. E.Clarke, E.Emerson, A.Sistla, Automatic verification of finite-state concurrent systems using temporal logic specifications, *ACM-TOPLAS* 8, 2 (April 1986), 244-263.
22. J.E.Coolahan, N.Roussopoulos, Timing requirements for time-driven systems using augmented Petri nets, *IEEE Transactions on Software Engineering* (September 1983).
23. E.Corsetti, A.Montanari, E.Ratto, Dealing with different time granularities in formal specifications of real-time systems, *The journal of Real-Time systems*, 3 (1991) 191-215.
24. W.Damm, G.Dohmen, V.Gerstner, B.Josko, Modular verification of Petri nets: the temporal logic approach. In *LNCS 430*, Springer Verlag, 1990.
25. M.Danelutto, A.Massini, A temporal Logic approach to specify and to prove properties of finite state concurrent systems, In *LNCS 385*, Springer Verlag, 1989.
26. M.Daniels, T.Dillon, Representing real-time, in *Proc. 1990*.
27. D.M.Dermott, A temporal logic for reasoning about processes and plans, *Cognitive Science* 6 (1982).
28. E.W.Dijkstra, *A discipline of programming*, Prentice Hall, Englewood Cliffs, N.J. (1976).

29. D.Dill, Timing assumptions and verification of finite state concurrent systems, LNCS 407 (1989), 197-211, Springer Verlag.
30. H.B.Henderton, A mathematical introduction to logic, Academic Press, New York (1972).
31. M.Felder, A.Morzenti, Validating real time systems by executing logic specifications in TRIO, in IEEE/ACM ICSE 14, 1992.
32. A.Fuggetta, C.Ghezzi, D.Mandrioli, Some considerations on real-time behavior of programs, IEEE-TSE 15, 3, (March 1989), 356-359.
33. A.Gabrielian, M.Franklin, Multilevel specification of real-time systems, CACM 34, 5 (May 1991), 51-60.
34. H.Genrich, Predicate/transition nets, in Advances in Petri Nets, Springer Verlag, W.Reisig and G.Rozenberg, Berlin-New York, 1987.
35. C.Ghezzi, D.Mandrioli, S.Morasca, M.Pezzè, A general way to put time in Petri nets, in 5th International Workshop on Software Specification and Design, IEEE, May 1989.
36. C.Ghezzi, D.Mandrioli, S.Morasca, M.Pezzè, Symbolic Execution of Concurrent Programs Using Petri Nets, Computer Languages (April 1989).
37. C.Ghezzi, D.Mandrioli, A.Morzenti, TRIO, a logic language for executable specifications of real-time systems, Journal of Systems and software 12, 2 (May 1990), 107-123.
38. C.Ghezzi, M.Jazayeri, D.Mandrioli, Fundamentals of software Engineering, Prentice Hall International Editors, Englewood Cliffs, N.J. (1991).
39. C.Ghezzi, D.Mandrioli, S.Morasca, M.Pezzè, A unified high-level Petri net model for time-critical systems, IEEE Transactions on software Engineering 17, 2 (Feb 1991).
40. C.Ghezzi, R.Kemmerer, Executing formal specifications: the ASTRAL to TRIO translation approach, in TAV'91, symposium on testing, analysis, and verification, Victoria, Canada, October 1991.
41. C.Ghezzi, R.Kemmerer, ASTRAL: an assertion language for specifying real-time systems, In proc of the 3rd European Software Engineering Conference, Milano, Italy, October 1991.
42. D.Gries, An illustration of current ideas on the derivation of correctness proofs and correct programs, IEEE-TSE 2, 4 (Dec 1976), 238-244.
43. V.Haase, Real-time behavior of programs, IEEE-Transactions on Software Engineering, 7, 5, (September 1981), 594-601.
44. D.Harel, Statecharts: a visual formalism for complex systems, Science of Computer Programming, North Holland, 8, (1987) 231-274.
45. X.He, Temporal predicate transition nets and their applications, in COMPSAC 1990, IEEE, 1990, 261-266.
46. X.He, J.Lee, Integrating predicate transition nets with first order temporal logic in the specification and verification of concurrent systems, Formal Aspects of computing, 2, (1990), 226-246.
47. C.A.R.Hoare, Monitors: an operating system structuring concept, CACM 17, 10, (Oct 1974), 549-557, See also Erratum CACM 18(2):95, Feb.1975.

48. M.A.Holliday, M.K.Vernon, A generalized timed Petri net model for performance analysis, IEEE Transactions on software Engineering (december 1987).
49. J.Hooman, Specification and compositional verification of real-time systems, Ph.D. dissertation, Eindhoven University, 1991.
50. Occam 2 Reference Manual, INMOS Limited, 1988.
51. F.Jahanian, A.K.Mok, Safety analysis of timing properties in real-time systems, IEEE-TSE 12, 9, (september 1986), 890-904.
52. K.Jensen, Coloured Petri nets, in Advances in Petri nets, Springer Verlag, W.Reisig and G.Rozeberg, Berlin, 1987.
53. R.A.Kemmerer, Testing formal specifications to detect design errors, IEEE Transactions on Software Engineering, (january 1985).
54. R.Koymans, Specifying message passing and time critical systems with temporal logic, Ph.D. dissertation, Eindhoven University of Technology, 1989.
55. F.Kröger, Temporal logic of programs, EATCS monographs on theoretical computer science, Springer Verlag, 1987.
56. J.C. Laprie, N.B.Levenson, E.Pilaud, M.Thomas, Panel on real-life safety critical software, 12th international conference on software engineering, See also Panel on Industrial experience with formal methods organized by D.Bjorner and L.Druffel, 1990, pp.222-229.
57. G.Leòn, Proposal of a case study development: the elevator control system, Tech. Report IPTES-UPM-22V1.0 ESPRIT - IPTES Project, 1991.
58. N.Leveson and J.Stolzy, Safety analysis using Petri nets, IEEE Transactions on Software Engineering, SE-13, 3(March 1987), 386-397.
59. D.Mandrioli, R.Zicari, C.Ghezzi, F.Tisato, Modeling the Ada task system by Petri nets, Computer Languages 10, 1, (1985), 43-61.
60. D.Mandrioli, C.Ghezzi, Theoretical foundations of computer science, John Wiley & sons, 1987.
61. Z.Manna, A.Pnueli, Verification of concurrent programs: a temporal proof system, Tech Rep. STAN-CS-83, Dept. of Computer Science, Stanford University, See also Foundations of computer science IV, Amsterdam, Mathematical Center tracts, June 1983.
62. Z.Manna, P.Wolper, Synthesis of communication processes from temporal logic specifications, ACM Transactions on Programming Languages and Systems 6, 1 (Jan 1984).
63. Z.Manna, The logical basis for computer programming, Addison Wesley, Reading MA, 1985.
64. Z.Manna, A.Pnueli, temporal logic of reactive and concurrent systems (chapter 4), Springer Verlag (1991),
65. E.Mendelson, Introduction of mathematical logic, Van Nostrand Reinold Company, New York (1963).
66. P.M.Merlin, D.J.Farber, Recoverability of communication protocols - Implications of a theoretical study, IEEE Transactions on Communications (September 1976).
67. M.Meritt, F.Modugno, M.Tuttle, Tim constrained automata, August 1990, Draft version.

68. R.Milner, A calculus of Communicating systems, LNCS 92, (1980).
69. S.Morasca, M.Pezzè, Validation of concurrent Ada programs using symbolic execution, in ESEC '89: 2nd European Software Engineering Conference, Springer Verlag, 1989, 469-486.
70. S.Morasca, M.Pezzè, M.Trubian, Timed High level nets, The journal of real-time systems (1991), 165-189.
71. A.Morzenti, The specificaiton of real-time systems: proposal of a logic formalism, Ph.D.dissertation, in Italian, Politecnico di Milano, Dipartimento di Elettronica, 1989.
72. A.Morzenti, P.San Pietro, TRIO+, an object oriented logic specification language , Research report, ENEL-CRA, in Italian, January 1990.
73. A.Morzenti, P.San Pietro, An object oriented logic language for modular system specification, in ECOOP '91, Springer Verlag, Geneva, July 1991.
74. A.Morzenti, D.Mandrioli, C.Ghezzi, A model parametric real-time logic, ACM Transactions on Programming Language and Systems, 1992.
75. J.Ostroff, W.Wohnan, Modelling specifying and verifying real-time embedded computer systems, in Proc. of the real-time systems symposium, December 1987, pp.124-132.
76. J.Ostroff, Temporal Logic for real-time systems, Research Studies Press Ltd, Taunton, Somerset, England, Advanced Software Development Series, 1 (1989).
77. A.Owicki, D.Gries, An axiomatic proof technique for parallel programs I, Acta Informatica, 6, (1975), Springer Verlag.
78. S.Owicki, L.Lamport, Proving properties of concurrent programs, ACM Transactions, on Programming Languages and Systems, 4, 3, (July 1982), 455-495.
79. J.L.Peterson, Petri net theory and the modelling of systems, Prentice Hall, Englewood Cliffs, NJ (1981).
80. A.Pnueli, The temporal semantics of concurrent Programs, Theoretical Computer Science, 13 (1981), North Holland Publishing Company.
81. A.Pnueli, Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends, LNCS 224 (1986), Springer Verlag.
82. C.V.Ramamoorthy, G.S.Ho, Peroformance evaluation of asynchronous concurrent systems using Petri nets. IEEE Transactions on Software Engineering (September 1980).
83. C.Ramchandani, Analysis of asynchronous concurrent systems using Petri nets, Tech Report, MIT, February 1974.
84. W.Reisig, Petri nets: an introduction, Springer Verlag, Berlin, 1985.
85. G.Richter, Clocks and their use for time modeling, Proc TFAIS 85, 1985.
86. J.Sifakis, Performance evauation of systems using nets, in Proc of the advanced course on general net theory, 1980.
87. J.Sifakis (Ed) Automatic verification methods for finite state systems, Springer Verlag, 1989.
88. J.Spivey, The Z notation - A reference manual, Prentice Hall, Englewood Cliffs, NJ, 1989.
89. T.Stotts, T.Pratt, Hierarchical modelling of software systems with timespetri nets, in Proc. of the 1st international workshop on timed Petri nets, July 1985.

90. I.Suzuki, H.Lu, Temporal Petri nets and their application to modeling and analysis of handshake daisy chain arbiter, IEEE TCO 38, 5 (may 1989), 696-704.
91. I.Suzuki, Formal Analysis of alternating bit protocol by temporal Petri nets, IEEE- TSE 16, 11, Nov. 1990, 1273-1281.
92. J.Thistle, W.M.Wonham, Control Problems in a temporal logic framework, International Journal of Control 44, 4, (1986).
93. H.Tuominen, Proving properties of elementary net systems with a special purpose theorem prover, in Proc. Workshop on Automatic Verification methods for finite state systems, 1989, pp.97-104.
94. Y.Wang, Real-time behavior of asynchronous agents, LNCS 458 (1990), 502-520, Springer Verlag.
95. N.Wirth, Toward a discipline of real-time programming, Communications of the ACM 20, 8, (August 77).
96. S.Yadav, R.Bravoco, A.Chatfield, T.Rajkumar, Comparison of Analysis techniques for information requirement determination, Communications of the ACM 31, 9 (Sept. 1988), 1090-1115.
97. P.Zave, W.Schell, Salient features of an executable specification language, IEEE TSE SE-12, 2 (Feb 1986), See also P.Zave, An operational approach to requirements specifications for embedded systems, IEEE TSE 8(3), May 1982.
98. W.Zuberek, Timed Petri nets and performance evaluation, In Proc. of the 7th annual Symposium on computer architecture, May 1980.

## Appendix I TRIO's model theoretic semantics

A straightforward way of defining the semantics of TRIO is based on the concept of a temporal structure, from which one can derive the notions of state—an assignment of values to time dependent predicates—and of evaluation function, which assigns to every TRIO formula a distinct truth value for every time instant in the temporal domain.

A *temporal structure* is denoted as  $S = (D, T, G, L)$ , where

- $D$  is a set of types, that is, of variable valuation domains:  $D = \{D_1, \dots, D_n\}$ . In the following, we will use the notation  $D(x)$  to indicate the evaluation domain associated with variable  $x$ .
- $T$  is the Temporal Domain: it is supposed to be a linearly ordered abelian group, thus possessing a total addition operator '+' with a neutral element '0' and an ordering relation '≤'. The subtraction operation '-' and the relational operator '<' can be trivially derived from '+' and '≤', and will be used in the sequel.  $T$  can be, for instance,  $\mathbb{Z}$ —the set of integer numbers—or  $\mathbb{Q}$ —the set of the rational numbers—or  $\mathbb{R}$ —the set of real numbers.
- $G$  is the time independent, or time invariant, part of the structure.  $G = (\xi, \Pi, \Phi)$ , where
  - $\xi$  is a valuation function defined on a set of time independent variable names, such that, for every variable name  $x$ ,  $\xi(x) \in D_j$ , for some  $D_j \in D$ ;

- $\Pi$  is a valuation function defined on a set of time independent predicate names. If  $p$  is the name of an  $n$ -ary time independent predicate,  $\Pi$  assigns a relation to it, that is,  $\Pi(p) \subseteq D_{p1} \times \dots \times D_{pn}$ , where  $D_{pj} \in D$ , for each  $j \in [1..n]$ ;
- $\Phi$  is a function defined on a set of function names. If  $f_j$  is an  $n$ -ary function name, then  $\Phi(f_j)$  is a total function of type  $D_{j1} \times \dots \times D_{jn} \rightarrow D_{j0}$ , with  $D_{jk} \in D$  for each  $k \in [1..n]$ , and  $D_{j0} \in D$ .
- $L$  is the time dependent part of the structure.  $L = \{\Pi_i \mid i \in T\}$ . Every  $\Pi_i$  defines a *state* of the temporal structure, i.e. a function defined on a set of time dependent predicate names. For every name  $p$  of an  $n$ -ary time dependent predicate,  $\Pi_i$  assigns a relation to it, that is,  $\Pi_i(p) \subseteq D_{p1} \times \dots \times D_{pn}$ , where  $D_{pj} \in D$  for each  $j \in [1..n]$ .

A temporal structure  $S = (D, T, G, L)$  is said to be *adequate* to a TRIO formula if  $D$  contains the valuation domains for all the variables occurring in it, and if functions  $\xi$ ,  $\Pi$ ,  $\Pi_i$ , and  $\Phi$ , assign values of the appropriate types to all time independent and time dependent variables and predicate names, and to all functions. If  $S = (D, T, G, L)$  is a structure adequate to a given TRIO formula  $F$ , then it is possible to define an evaluation function assigning a value, at any time instant  $i \in T$ , to all terms and formulas that can be constructed by using variables, functions, predicates and operators occurring in  $F$ . We call such function  $S_i$ ; its definition for terms is as follows.

- i)**  $S_i(x) = \xi(x)$  for every variable  $x$ ;
- ii)**  $S_i(f(t_1, \dots, t_n)) = \Phi(f)(S_i(t_1), \dots, S_i(t_n))$  for every application of an  $n$ -ary function  $f$ .

For TRIO formulas, the function  $S_i$  yields values belonging to the set  $\{\text{true}, \text{false}\}$ , and is defined as follows:

- iii)**  $S_i(p(t_1, \dots, t_n)) = \text{true}$  iff  $p$  is a time independent  $n$ -ary predicate and  $(S_i(t_1), \dots, S_i(t_n)) \in \Pi(p)$ ,  
or  $p$  is a time dependent  $n$ -ary predicate and  $(S_i(t_1), \dots, S_i(t_n)) \in \Pi_i(p)$ .

If  $A$  and  $B$  are formulas, then

- iv)**  $S_i(\neg A) = \text{true}$                       iff  $S_i(A) = \text{false}$ ;
- v)**  $S_i(A \rightarrow B) = \text{true}$             iff  $S_i(A) = \text{false}$  or  $S_i(B) = \text{true}$ ;
- vi)**  $S_i(\forall x A) = \text{true}$                 iff  $S_i(A_a^x) = \text{true}$  for every  $a \in D(x)$ .

Finally, the truth value of a temporal formula is defined according to the following clause.

- vii)**  $S_i(\text{Dist}(\varphi, t)) = S_{i+S_i(t)}(\varphi)$

A TRIO formula  $F$  is *temporally satisfiable* in a temporal structure  $S$  iff  $S$  is adequate to it, and there exists a time instant  $i \in T$  for which  $S_i(F) = \text{true}$ . In such a case, we say that the temporal structure constitutes a *model* for  $F$ . A formula  $F$  is *temporally valid* in a structure  $S$  iff  $S_i(F) = \text{true}$  for every  $i \in T$ ; it is *valid* iff it is temporally valid in every adequate structure.

## Appendix II The soundness of TRIO's axiom system

The soundness property of a deductive calculus claims that  $\Gamma \vdash \phi$  implies  $\Gamma \models \phi$ , i.e. only valid formulas may be derived. The soundness proof for TRIO's axiom system runs on induction on the length of the derivation, and follows the lines of similar proofs provided in traditional textbooks of mathematical logic [30].

Since the only inference rule of the calculus is Modus Ponens, which is trivially sound, our proof reduces to showing that all the axioms of Section 2.4 are valid. First, we notice that (as can be easily shown) all generalizations of valid formulas are also valid and that, by the very definition of validity (see Appendix I), if  $\alpha$  is valid then its temporal generalization,  $Alw(\alpha)$ , is valid too. Hence it suffices to consider only axioms which are not themselves generalizations or temporal closures of other axioms. For brevity we omit the parts of the proof regarding first order and equality axioms, i.e. axiom groups 1÷6, since they do not differ significantly from standard proofs such as that of [30]. Hence it only remains to show the validity of the temporal axioms.

*Axiom group 7.* From clause vii) of the semantics, and since zero is the neutral element of the addition operation on the temporal domain (assumed to be a linearly ordered abelian group) it follows that  $S_i(\text{Dist}(\alpha), 0) = S_i(\alpha)$ , so  $\alpha$  is true if and only if  $\text{Dist}(\alpha, 0)$  is true.

*Axiom group 8.* The validity of these axioms follows from the fact that '+' is a total operation on the temporal domain, and that the values of terms  $t_1$  and  $t_2$  do not depend on the evaluation time, since they contain only time independent elements of TRIO's alphabet, like variables, constants, and functions.

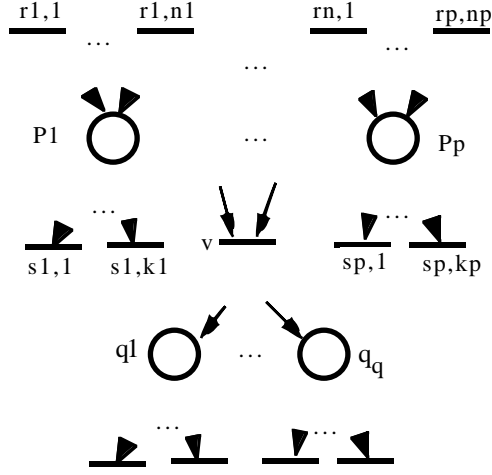
*Axiom groups 9 and 10.* Their validity follows trivially from clauses iv) and v) of the semantics and from the totality of the '+' operation.

*Axiom group 11.* If formula  $\alpha$  is time independent and currently true then  $S_i(\alpha) = \text{true}$  for any  $i \in T$ , by clauses i), ii), iv), v), and vi) of the semantics. Hence  $\text{Dist}(\alpha, t)$  is true for any value of  $t$ . ■

## Appendix III Generalized topological axioms

In this appendix we briefly show how topology dependent axioms can be derived in a systematic way. We will restrict our attention to the LB, UB, IU, OU axioms, leaving the remaining—simpler—ones to the reader.

With reference to Figure A.III.1, let us assume that a generic transition  $v$  is output to  $p$  distinct places  $p_i$ , each one having  $n_i$  input transitions and  $k_i$  output transitions (among which, possibly,  $v$  itself), and is input to  $q$  distinct places  $q_i$ . Also  $v$  may coincide with some of the other transitions, when it is both input and output of some place.



**Figure A.III.1** A general fragment of TPN.

- Axiom LB( $v$ ) states that every token consumed by a firing of transition  $v$  must have been produced at least  $m_v$  time units before.

$$LB(v): \text{fireth}(v, i) \rightarrow \bigwedge_{j=1}^p \bigvee_{k=1}^{n_j} \exists d (d \geq m_v \wedge \exists x \text{ tokenP}(r_{j,k}, x, p_j, v, i, d))$$

- Axiom UB( $v$ ) specifies that if a tuple of  $p$  tokens was formed more than  $M_v$  time units ago, at least one of its tokens must have been consumed by a firing of  $v$  or of one of the transitions in conflict with it.

$$UB(v): \bigwedge_{\substack{j_1=1..n_1 \\ \dots \\ j_p=1..n_p}} \left( \bigwedge_{i=1}^p (d_i \geq M_v \wedge \text{Past}(\text{fireth}(r_{i,j_i}, x_i), d_i)) \rightarrow \bigvee_{i=1}^p \bigvee_{h=1}^{k_i} \exists d (d \leq d_i \wedge \exists y \text{ Past}(\text{tokenP}(r_{i,j_i}, x_i, p_i, s_{i,h}, y, d_i - d), d)) \right)$$

As noticed in Section 3.2 with reference to the net fragment of Figure 3.2(b) some of the transitions in conflict with  $v$  may have a stricter upperbound: in this case their axiom UB would logically imply UB( $v$ ).

Axioms IU( $v$ ) and OU( $v$ ) need not explicitly mention the transitions whose firings are related to those of  $v$ . Hence in the following formulas variables  $x$  and  $y$  represent transitions and—as usual—are to be interpreted as implicitly universally quantified.

- IU( $v$ ) asserts that any token extracted by a firing of transition  $v$  from each of places  $p_i$ ,  $i=1..p$ , was generated by a single firing of a unique transition.

$$IU(v): \bigwedge_{h=1}^p (\text{tokenP}(x, i, p_h, v, j, d) \wedge \text{tokenP}(y, k, p_h, v, j, e) \rightarrow x=y \wedge i=k \wedge d=e)$$

- OU( $v$ ) indicates that every token inserted by a firing of transition  $v$  into each of places  $q_h$ ,  $h=1..q$ , is consumed by only one firing of a unique transition.

$$\text{OU}(v): \bigwedge_{h=1}^q (\text{tokenF}(v, i, p_h, x, j, d) \wedge \text{tokenF}(v, i, p_h, y, k, e) \rightarrow x=y \wedge j=k \wedge d=e)$$

Based on these axiom schemata, an algorithmic procedure can be defined to translate a TPN into a set of TRIO axioms characterizing its temporal behavior.