Correctness Verification and Performance Analysis of Real-Time Systems Using Stochastic Preemptive Time Petri Nets

Giacomo Bucci, *Member*, *IEEE*, Luigi Sassoli, and Enrico Vicario, *Member*, *IEEE*

Abstract—Time Petri Nets describe the state of a timed system through a marking and a set of clocks. If clocks take values in a dense domain, state space analysis must rely on equivalence classes. These support verification of logical sequencing and quantitative timing of events, but they are hard to be enriched with a stochastic characterization of nondeterminism necessary for performance and dependability evaluation. Casting clocks into a discrete domain overcomes the limitation, but raises a number of problems deriving from the intertwined effects of concurrency and timing. We present a discrete-time variant of Time Petri Nets, called stochastic preemptive Time Petri Nets, which provides a unified solution for the above problems through the adoption of a maximal step semantics in which the logical location evolves through the concurrent firing of transition sets. We propose an analysis technique, which integrates the enumeration of a succession relation among sets of timed states with the calculus of their probability distribution. This enables a joint approach to the evaluation of performance and dependability indexes as well as to the verification of sequencing and timeliness correctness. Expressive and analysis capabilities of the model are demonstrated with reference to a real-time digital control system.

Index Terms—Real-time reactive systems, preemptive scheduling, correctness verification, performance and dependability evaluation, discrete time, maximal step semantics, confusion, well definedness, stochastic preemptive Time Petri nets.

1 Introduction

In the construction of real-time reactive systems, development complexity is largely biased by nonfunctional requirements addressing ordered sequencing of events, bounded stimulus-response delay, efficient utilization of resources. Due to the criticality of applications involved, this complexity is often exacerbated by the need to guarantee correctness and quality of service with a high or even complete degree of dependability [40], [52], [39].

In the classical literature of real-time systems, restrictive assumptions on the structure of processes circumvent the problem of logical sequencing and enable efficient analytical techniques for the estimate of the worst case response time [24]. Unfortunately, this kind of analysis does not cover complex (but often realistic) systems with such characteristics as: nondeterministic computation times; sporadic tasks and/or tasks with mutual dependencies in the time of release; intertasks dependencies due to mutual exclusion on shared resources or to dataflow precedence relations; internal sequencing of tasks; multiple processors. Under such conditions, the need for predictability can motivate the complexity of state-space analysis techniques based on models such as Timed Automata [2], [6], [36], Time Petri Nets [9], [56], Process algebras [5], [48], [11]. As a common trait, in all these models, the state is comprised of a logical location and a set of clocks.

If clocks take values in a dense domain, the state space must be covered using state classes, each associating a logical location with a dense variety of clock valuations. Properties pertaining to the reachability of individual states, to the firability of execution sequences, and to the variety of timings that can be associated with a firing sequence can then be derived from the relation of reachability among classes. The theory has been largely developed both for Timed Automata [2], [6], [36] and for Time Petri Nets [10], [9], [17], [56]. However, the use of state classes poses severe limitations for model expressivity. In particular, analysis techniques based on state classes are difficult to be efficiently extended to deal with systems where clocks may be suspended and then resumed, as occurring in preemptive behavior [18], [20], [46], [26], [38]. More importantly for the focus of this paper, enumeration methods based on state classes for dense time have not been sufficiently developed yet to deal with a stochastic characterization of nondeterministic time parameters [22].

The assumption of a discrete time model overcomes the need to rely on state classes, thus smoothing the contrast between expressivity and analyzability. In particular, it largely eases the introduction of a stochastic characterization of time parameters and logical choices, which permits integration of correctness verification with dependability and performance evaluation [5], [31], [12], [15], [41], [19].

In [5], [43], a discrete time process algebra is used to model a set of recurring real-time tasks scheduled on a set of resources under various disciplines of practical interest (e.g., fixed priority and earliest deadline first) and to

• *The authors are with the Dipartimento Sistemi e Informatica, Università di Firenze, via S. Marta, 3 50139, Firenze, Italy.*
E-mail: {bucci, sassoli, vicario}@dsi.unifi.it.

Manuscript received 23 Dec. 2004; revised 9 July 2005; accepted 4 Oct. 2005; published online 1 Dec. 2005.
Recommended for acceptance by G. Ciardo.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-0287-1204.

support schedulability analysis and verification of safety properties [50]. The approach is extended in [44] to represent a failure rate for resources and to evaluate the probability that a deadline is missed due to this kind of failure. However, stochastic characterization does not cover the variability of temporal parameters such as task release times and computation times.

In [31], [12], [15], discrete phase types are used in a Petri Net formalism to provide a stochastic characterization for temporal parameters constrained within a finite support. This enables Markovian analysis in models where the mutual timing of events may yield implicit constraints on the logical sequencing. The use of a Kronecker algebra avoids the need for explicit enumeration of timed states, but requires that the untimed model have a finite number of markings. This rules out those systems where finiteness of behavior depends on sequencing constraints induced by timing (e.g., systems with bounded arrival rate or with timeout protocols).

The limitation is avoided in [59] by affording an explicit enumeration of individual states of a discrete time stochastic Petri Net so as to support a combined approach to schedulability analysis and performance analysis in complex tasking sets running under Rate Monotonic scheduling. However, the explicit enumeration cannot be applied in the presence of stochastic temporal parameters taking values over an infinite support, even in the case of a memoryless (geometric) distribution.

In [41], an analysis technique is proposed which aims at producing a discrete time approximation of the stochastic behavior of a model with non-Markovian dense time parameters. To this end, a discrete time Markov chain is derived by enumerating states comprised of a marking and a phase within a discrete phase type. In the implementation, this is obtained by first enumerating the markings of the untimed model and then expanding them so as to distinguish the different phases in which each marking can be reached [13].

In this paper, we present a Petri Net modeling formalism and an analysis technique which support both the verification of correctness and the evaluation of performance and dependability in real-time systems. The approach assumes a discrete model of time, which permits to manage a stochastic characterization of nondeterministic timed behavior and to encompass the expressivity requested by the application context of real-time systems, including preemptive behavior and flexible computations [25], [42].

The modeling formalism, that we call *stochastic preemptive Time Petri Nets* (spTPN), is characterized by the use of a step semantics. According to this, the logical location evolves through the concurrent execution of transition sets rather than through the interleaved firing of individual transitions assumed in most timed extensions of Petri Nets [31], [12], [59]. The use of a step semantics provides a unified solution for a number of complex effects deriving from the combination of logical concurrency and quantitative discrete timing, and relieves the designer from the burden of coping with stochastic underspecification of the model [51], [32], [54].

The analysis technique relies on the concept of stochastic state, which integrates the enumeration of a succession relation among sets of timed states with the calculus of their probability distribution. This permits the construction of a transition system labeled with durations and probabilities of events which opens the way to Markovian evaluation of performance and dependability indexes as well as to the verification of correctness in the sequencing of events and in their durational properties.

The rest of the paper is organized in five sections: The modeling formalism is introduced in Section 2 and the relevant characteristics of its semantics are discussed in Section 3 with reference to other timed variants in the literature of Petri Nets. The analysis technique is presented in Section 4. In Section 5, a case study in the context of real-time control systems is reported to demonstrate the expressive capabilities of the model, the complexity of the resulting analysis process, and the kind of results that this can provide. Conclusions are drawn in Section 6.

2 Stochastic Preemptive Time Petri Nets

Stochastic preemptive Time Petri Nets (spTPN) develop Time Petri Nets [47], [9] and preemptive Time Petri Nets (pTPN) [20], [18] to obtain a discrete time formalism enabling stochastic modeling of real-time systems: transition delays are characterized through a discrete probability mass function; nondeterministic choices can be conditioned through stochastic switches; progress of timed transitions depends on preemptable resources, assigned according to priorities that can change with the marking; the model evolves through the concurrent firing of transition sets (maximal step semantics).

2.1 Syntax

A stochastic preemptive Time Petri Net is a tuple:

$$spTPN = \langle P; T; A^-; A^+; A^\bullet; M; Res; Req; Prio; \mathcal{D}; \mathcal{C} \rangle. \quad (1)$$

- P and T are disjoint sets of places and transitions, respectively. A^-, A^+, and $A^\bullet$ are sets of precondition, postcondition and inhibitor arcs connecting places and transitions ($A^- \subseteq (P \times T)$, $A^+ \subseteq (T \times P)$, and $A^\bullet \subseteq (P \times T)$). A place p is said to be an *input*, an *output* or an *inhibiting* place for a transition t if there exists a precondition, a postcondition or an inhibitor arc connecting p and t, respectively. M is a *marking* associating each place with a natural number ($M : P \to \mathbb{N}$).
- Res is a set of *resources* disjoint from T and P. Req associates each transition with a set of requested resources ($Req : T \to 2^{Res}$) called *resource request*. $Prio$ associates each transition with a priority level which may depend on the marking ($Prio : T \times \mathbb{N}^{|P|} \to \mathbb{N}$). For simplicity, we assume here that two transitions cannot have the same priority level under any marking

$$\forall M \in \mathbb{N}^{|P|},\ t_1 \neq t_2 \to Prio(t_1, M) \neq Prio(t_2, M)).$$

- $\mathcal{D}$ associates each transition t with a *static* probability mass function $\mathcal{D}_t$, possibly defined over a nonfinite support. The extrema of the support of $\mathcal{D}_t$ are often referred to as static earliest and latest firing time, denoted as $EFT^s(t)$ and $LFT^s(t)$, respectively.
- $\mathcal{C}$ is a *competitiveness* function associating each transition with a natural number ($\mathcal{C} : T \to \mathbb{N}$).

2.2 Semantics

The state of an spTPN is a pair $s = \langle M, \vec{ttf} \rangle$, where M is a *marking* and $\vec{ttf}$ is a vector associating each transition t with a discrete *time to fire* $ttf_t \in \mathbb{N}$. The state evolves according to a state transition rule defined through three clauses of *firability*, *step selection*, and *progress*.

2.2.1 Firability

A transition is *enabled* if each of its input places contains at least one token and none of its inhibiting places contains any token. An enabled transition t_o is *progressing* iff every resource in $Req(t_o)$ is not in the resource request $Req(t_1)$ of any other enabled transition t_1 with higher priority

$$(\forall r \in Req(t_o), \forall\ enabled\ t_1,\\ Prio(t_1, M) > Prio(t_o, M) \to r \notin Req(t_1).)$$

Transitions that are enabled but not progressing are said to be *suspended*. A progressing transition t is *firable* iff its time to fire ttf_t is equal to 0. The set of firable transitions is called *attempting set*.

2.2.2 Step Selection

If the attempting set is empty, time advances by one *tick*.

If the attempting set is not empty, the set of transitions that will actually come to fire, that we call *firing set*, is derived through a *repetitive stochastic selection*: Until the attempting set is not empty, one of its transitions is randomly selected, it is removed from the attempting set, and it is added to the firing set iff none of its input places is an input place for any other transition already added to the firing set (what we call a *conflict*). At each repetition, the probability to select a transition t_* is made equal to the competitiveness of t_* divided by the sum of competitiveness values of all the transitions still contained in the attempting set:

$$Prob\{t_*\} = \frac{\mathcal{C}(t_*)}{\sum_{t \in Attempting\ set} \mathcal{C}(t)}.$$

2.2.3 Progress

If time advances, the marking and the time to fire of every suspended transition remain unchanged, while the time to fire of every progressing transition is reduced by one time unit.

If a set $\{t_n\}_{n=o}^{N-1}$ fires, the marking M is replaced by a new marking M' which is obtained by removing a token from each input place of each transition in the firing set, and by adding a token to each output place of each transition in the firing set. In the transformation, an intermediate marking M_{tmp} is also derived to capture the condition subsequent to the removal of tokens from input places but prior to the addition of tokens in output places:

$$\begin{aligned} M_{tmp}(p) &= M(p) - 1 \quad \forall p.\langle p, t_n \rangle \in A^- \\ M'(p) &= M_{tmp}(p) + 1 \quad \forall p.\langle t_n, p \rangle \in A^+. \end{aligned} \tag{2}$$

Transitions that are enabled both by the intermediate marking M_{tmp} and by the final marking M' are called *persistent*, while those that are enabled by M' but not by M_{tmp} are said *newly enabled*. Any transition belonging to the firing set which is still enabled after its own firing is always regarded as newly enabled. The time to fire of persistent transitions remains unchanged, while the time to fire of each newly enabled transition t takes a nondeterministic value sampled according to the static probability mass function $\mathcal{D}_t$. The time to fire ttf_t of any transition t that is not enabled by the new marking is not relevant for the state of the net as it will be reset to a new value as soon as t will be enabled again.

The state transition rule defines a *direct reachability* relation $\overset{ev,\mu}{\to}$ between states, where μ is a measure of probability and ev is either the advancement of time (*tick* event) or the firing of a set of transitions (*firing* event). Given two states s and s', we write $s \overset{ev,\mu}{\to} s'$ when ev can occur in s and lead to s' with probability μ. Note that the measure of probability μ can be different than 1 only when ev is a firing. In this case, its value depends on the resolution of possible conflicts within the attempting set as well as on the determination of the times to fire assumed by newly enabled transitions.

2.3 Remarks

A few remarks emphasize the salient traits that distinguish spTPNs with respect to other timed extensions of Petri Nets.

2.3.1 Maximal Step Semantics

The state of an spTPN evolves through steps each consisting of the concurrent execution of a set of progressing transitions (the firing set). Steps are maximal as a progressing transition cannot avoid the firing unless its time to fire is not null or it has a conflict with some transition in the firing set. This marks a difference with respect to most timed extensions of Petri Nets where the firing of multiple transitions sharing a common time to fire occurs through the interleaved firing of individual transitions.

The set of transitions that attempt the firing is univocally determined by the marking and the vector of times to fire. If the attempting set does not contain any conflict, all attempting transitions come to the firing in a single step. Only in the presence of conflicts, determination of the firing set is conditioned by competitiveness values of attempting transitions.

The definition of a maximal step semantics for a model with nonMarkovian timed transitions is an original aspect of spTPNs. Previous works [58], [23] have addressed the case of transitions firing either in zero time or after a geometrically distributed delay; this setting drastically simplifies the analysis as the state does not depend on time, but prevents the representation of nondeterministic delays comprised within a finite interval, which are crucial in modeling precedence relations induced by timed

behavior. The case of multiple transitions attempting the firing within the same time slot in a discrete timed model is explicitly addressed in [41]. However, since the concurrency of the model is not explicitly formulated as a step semantics, for each firing set, the feasibility with respect to conflicts, the probability of occurrence, and the resulting state are determined through the analysis of intermediate states reached according to an interleaving semantics [15]. This yields a different behavior with respect to that of spTPNs, as discussed in Section 3.3. Moreover, in [41], any subset of the attempting set is considered firable, thus admitting the case of a transition which does not attempt the firing though its time to fire has a null probability to be higher than 0. This results in nonmaximal steps impairing the property of *nonnull defer*, later introduced in Section 3.1, that relieves spTPNs from the effects of confusion behaviors.

2.3.2 Collective Token Philosophy, No Multiple Enabledness, and Single Service

spTPNs follow a so-called Collective Token Philosophy in which any two tokens in the same place are computationally indistinguishable [16]. In fact, the timing semantics is expressed with reference to clocks (the times to fire) associated with enabled transitions. Tokens determine the conditions of continuity in transition enabling, but they do not carry information about their individual time of permanence in a place.

This limitation prevents considering the concurrent progress of multiple instances of the same transition (multiple service) even when tokens in its input places are sufficient for multiple firings (multiple enabledness). This motivates the assumption that a transition which is still enabled after its own firing must always be considered as newly enabled, as usual in Time Petri Nets [9], [56]. In fact, in order to combine a multiple server semantics with nonexponential delays, each concurrent instance of the same transition should be associated with an individual time to fire depending on the age of the binding of tokens which enable the transition instance. We avoid this approach which would drastically increase the complexity of timed analysis.

In the lack of a multiple server semantics, multiple tokens in a place can still serve to account for a queue of jobs waiting to be started or for a multiplicity of stand-by resources which can give continuity to the flow of a service.

2.3.3 Resources, Priorities, and Preemptive Behavior

resource requests and *priorities* permit the representation of computations whose advancement is suspended and then resumed, as occurring in systems running under preemptive scheduling. In this aspect, spTPNs basically follow the approach proposed in preemptive Time Petri Nets [20].

A basically equivalent expressivity is attained in dense time domain by Scheduling Time Petri Nets [49]. In that model, the conditions of enabledness and progress are distinguished through the introduction of the concept of *active marking*: Places can be associated with resources and priorities, sets of resources are assigned to places by priority, and an enabled transition is progressing (there called active) iff all its input places are assigned requested resources.

Modeling of preemptive behavior was also proposed for Stochastic Petri Nets by associating each transition with one out of three execution policies differing in the way the time-to-fire (there called *age*) is maintained or reset when the transition is disabled [14]. As a main difference with respect to that formulation, spTPNs distinguish two separate mechanisms which drive the conditions of enabling and progress, keeping separate the logical sequencing of processes from the policy for resource scheduling. While the first captures constraints that are expressed within user processes, the latter encodes policies which are implemented in the kernel of the operating system and are usually not accessible to the designer. This separation largely simplifies the construction of a model and also enables its analysis under different policies for resource assignment.

The policy of resource assignment of spTPNs can be extended to deal with transitions with equal priority according to the characteristics of different application contexts. This has no impact on the analysis technique reported in this paper, provided that the assignment can be decided in deterministic manner on the basis of the current marking, of transition priorities and resource requests. The Oris tool later mentioned in Section 5 implements a *priority queue* protocol [53], which sorts transitions with the same priority level so as to give precedence to transitions that can receive a higher number of resources. Conditions of parity that can still occur after this sorting are resolved through a secondary priority level. A nondeterministic choice could also be encompassed in the analysis, but this would require some additional stochastic switch to give each possible determination a probability in order to preserve the property of well definedness discussed in Section 3.2.

2.3.4 Timed and Immediate Transitions

In *dense* time models, the probability that two timed transitions fire at the same time instant has a null measure unless probability density functions of transition delays are dependent or they can yield a nonnull integral over an integration domain of null measure. In most practical applications, this confines the problem of contemporary firings to immediate transitions that are used to model causal connections (possibly with a deterministic delay) between events. Distinction between immediate and timed transitions in the semantics of the model enables ad hoc algorithms which simplify the analysis and separate the problems of concurrency and timing [45].

In *discrete* time models [31], [33], [15], [59], the firing of transitions is concentrated at discrete instants, and the problem of contemporary firings is extended to timed transitions. This basically reduces the advantage in distinguishing timed and untimed transitions. SpTPNs abolish this distinction permitting a direct modelling of events for which the probability to fire at time 0 after the enabling is strictly higher than 0 but strictly lower than 1. This is an expressive capability that may be relevant to smooth the limits of a discrete formalism in the modeling of asynchrouns systems, especially when the analysis is oriented to verify properties of the logical sequencing.

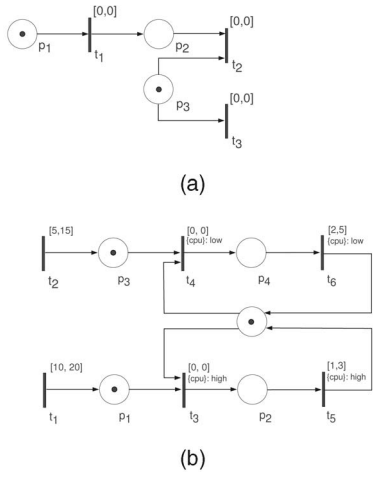


Fig. 1. (a) Under interleaving semantics, transitions t_1 and t_3 cannot be fired concurrently even if their times to fire become both equal to 0; rather, they must be given a fictitious order, which affects the subsequent behavior: a conflict between t_2 and t_3 may arise only in the case that t_1 fires first. (b) If t_1 and t_2 become ready to fire at the same time, their fictitious order of execution determines the result of the conflict between t_3 and t_4, independently from priorities that these transitions are given. According to this, in order to control the precedence between t_3 and t_4, the designer must detect the dependency and give relative priorities to t_1 and t_2 (what in [54] is called *upstream propagation*).

3 Step Semantics and Discrete Time

With the assumption of a step semantics, concurrency does not give rise to nondeterminism. This characteristics, whose convenience has been often advocated in the area of synchronous languages [8], [7], avoids the need of giving an order to the concurrent firing of multiple noninterfering transitions that execute within a null lapse of time. This circumvents a number of problems which arise when discrete timing constraints are combined with an interleaved semantics of concurrency.

In this section, these problems are analyzed to comparatively illustrate the advantages of spTPNs from the perspective of modeling convenience.

3.1 Confusion

Under an interleaving semantics of concurrency, transitions that are ready to fire at the same time must be given an order and executed one-by-one. A *confusion* behavior occurs when the choice on this order affects the resolution of conflicts arising in markings visited along a subsequent immediate firing sequence [51], as exemplified in Fig. 1.

In dense time, this problem is basically confined to immediate transitions as, under commonly accepted conditions on the form of distributions, the probability that two timed transitions take the same time to fire has a null measure and does not affect the evaluation of reward indices. However, this is not true in discrete time where multiple timed transitions normally become ready to fire at the same time with nonnull probability. In this case, confusion may result in complex behaviors combining structural conflicts and timing constraints which are difficult to be detected and intentionally controlled by the designer. Suspension of transition progress in preemptive behaviors exacerbates the problem.

Under the maximal step semantics of spTPNs, a progressing transition with null time to fire necessarily attempts the firing. In the attempt, it will either fire or be overtaken by some conflicting transition. But, in neither of the two cases, the transition can remain persistent and attempt the firing before a finite lapse of time has passed. This mechanism, that we call *nonnull defer*, rules out the case of an immediate firing sequence which includes choices between transitions that have been persistent since the beginning of the sequence and transitions that have become progressing with a null time to fire along the sequence itself.

Nonnull defer improves modeling convenience as it structurally overcomes the need for the designer to detect and rule implicit conflicts occurring within confusion behaviors. At the same time, it does not practically reduce expressive power as the model language still accepts behaviors where multiple events are serialized along a variety of interleaving sequences within a null lapse of time: this can be obtained with self-loop patterns which explicitly represent a constraint of serialization, as occurring for instance in the net of Fig. 4.

3.2 Well Definedness

Quantitative analysis for performance and dependability evaluation relies on a stochastic characterization of choices in the behavior of a model. In principle, this characterization can be partial, combining probabilistic and nondeterministic choices. This supports abstraction about decisions that cannot be stochastically characterized and still permits the derivation of some relevant measures, as for instance in the verification of probabilistic concurrent systems [55], [35].

However, in order to enable derivation of global properties through Markovian analysis, each choice must be characterized through a measure of probability for each feasible determination [32]. To accomplish this condition of *well definedness*, a model often needs to be extended, either with priorities which force the behavior and avoid choices, or with random switches which characterize them [30], [45], [32].

Under interleaving semantics, the effort of this extension is largely increased by the the need to give a fictitious order to the execution of multiple noninterfering transitions. To confine the problem to those choices which actually impact on some relevant reward process, [30] proposes that the set of transitions that may be fired at the same time be partitioned in a number of noninterfering extended conflict sets (ECS). In so doing, choices that must be ruled are only those occurring within the same ECS. However, identification of ECS through structural analysis provides only a necessary condition, and management of false alarms requires additional specification by the designer. The difficulty is largely exacerbated by the fact that an ECS may include transitions that are enabled in different markings visited along a confusion behavior.

In [32], well definedness of a model is automatically verified during the generation of the underlying stochastic process of the net. In [54], sufficient conditions are derived using a structural analysis implemented within an interactive tool which prompts on potential choices that are

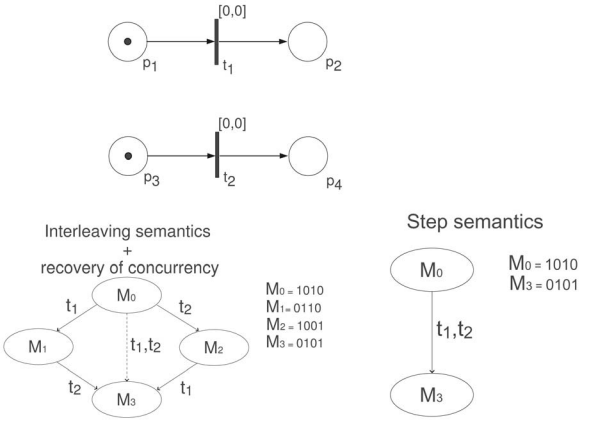


Fig. 2. In an interleaving approach, the existence of both the instantaneous interleaving sequences $\{t_1\} \rightarrow \{t_2\}$ and $\{t_2\} \rightarrow \{t_1\}$ is interpreted so as to recover the concurrent behavior in which t_1 and t_2 are fired simultaneously, leading from state M_o to M_3 in a single step.

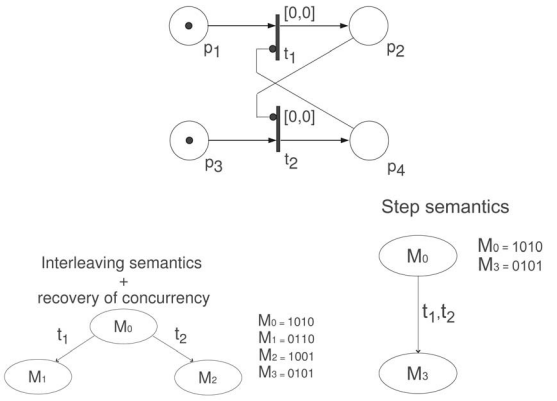


Fig. 3. Under a step semantics, transitions t_1 and t_2 fire in a single step leading to marking M_3. Whereas, under an interleaving semantics, marking M_3 cannot be reached by any interleaved sequence of events as the firing of t_1 disables t_2 and viceversa.

missing a probabilistic characterization and which guides the designer through an incremental extension of the model. However, in so doing, conflicts that correspond to an explicit modeling intention of the designer tend to be mixed with fictitious or unexpected conflicts arising in the interleaved sequencing of events. For the latter kind of choices, the designer is requested to express priorities or random switches which do not carry an apparent physical and intentional meaning (see Fig. 1b).

In the step semantics approach of spTPNs, choices occur only in the selection of a time to fire for a newly enabled transition or in the derivation of a firing set from an attempting set with conflicts. The former kind of choice is fully characterized by the probability function of the newly enabled transition. The latter is ruled by the stochastic selection process with the possible conditioning of competitiveness values. This separates the effects of explicit dependencies of mutual exclusion expressed by conflicts on input places from implicit dependencies induced by timing constraints of transitions.

3.3 Conflict and Concurrency

In analysis techniques based on an interleaving approach, concurrency can be partially recovered by classifying as concurrent a set of events which can be performed in any order, within a null lapse of time, and leading to the same final condition. This approach is applied to a timed model in [15], where a preanalysis of the untimed reachability graph is performed to associate each marking with the sets of concurrent firings (transitions that can fire simultaneously with a nonnull probability): Each of these sets is replaced through a *virtual* immediate transition that fires in zero time and the analysis is completed with the usual algorithm enumerating a new state for each individual transition fired. Concurrent firings are then recovered through the addition of state transitions covering multiple sequential firings (see Fig. 2).

The assumption of a step semantics not only avoids enumeration of fictitious states visited along interleaved sequences, but, more importantly, it eliminates a number of subtle side effects deriving from the attempt of concealing the different orders of concurrent events depending on possible conflicts and nonmonotonic enabling conditions related to priorities and inhibitor arcs. These effects not only prevent complete reconstruction of all the states that are actually reachable in a step semantics (see Fig. 3), but also may detect concurrency between events that are mutually exclusive (see Fig. 4) [16].

4 State Space Enumeration and Analysis

We regard the evolution of an spTPN as a discrete time stochastic process.

The states of the process, that we call *stochastic states* to distinguish them from the states in the semantics of Section 2.2, are pairs $S = \langle M, \vec{\mathcal{P}} \rangle$, where M is a marking and $\vec{\mathcal{P}}$ a vector of probability mass functions for the times to fire of transitions enabled by M.

The evolution across stochastic states is described through a *succession* relation $\overset{ev,\mu}{\Longrightarrow}$, where ev is an event and μ a measure of probability. Given two stochastic states $S = \langle M, \vec{\mathcal{P}} \rangle$ and $S' = \langle M', \vec{\mathcal{P}}' \rangle$, we write $S \overset{ev,\mu}{\Longrightarrow} S'$ iff the following property holds: If the marking of the net is M and the vector of times to fire is a random variable distributed according to

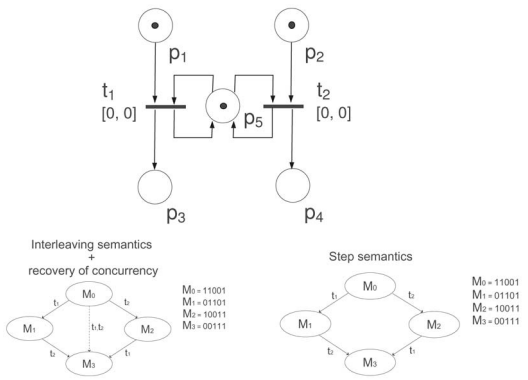


Fig. 4. Transitions t_1 and t_2 have a structural conflict, but they can be executed in any order, within a null lapse of time, leading to the same marking. Following an interleaving approach and trying to recover concurrency from nondeterminism, a false concurrent behavior would be added leading directly from M_o to M_3.

$\vec{\mathcal{P}}$, then ev is a possible next event for the net, which occurs with probability μ, and which leads to a new marking M' and a new vector of times to fire distributed according to $\vec{\mathcal{P}}'$.

While the state of an spTPN evolves through *firing* and *tick* events, in the enumeration of the succession relation among stochastic states, we also consider a third fictitious event that we call *defer*. This is an outcoming event for stochastic states where both the firing of a set of transitions and the advancement of time are possible, and accounts for the instantaneous choice that no firable transition will fire before the advancement of time. With the introduction of *defer*, the *tick* event is considered possible only in stochastic states where no transition is firable, thus avoiding stochastic states where both a *firing* and a *tick* are outcoming events. This partitionment associates each stochastic state with a sojourn time, equal to 0 if *defer* or *firing* are outcoming events, and equal to 1 if *tick* is the (unique) outcoming event.

4.1 Enumeration

Given an initial stochastic state S_o, the succession relation $\overset{ev,\mu}{\Longrightarrow}$ identifies a set of reachable stochastic states $\mathcal{S}$ and a timed stochastic step transition system $STS = \langle S_o, \mathcal{S}, \overset{ev,\mu}{\Longrightarrow}\rangle$. Enumeration of the STS requires algorithms for the detection of the outcoming events from a stochastic state, for the calculus of their probability, and for the derivation of successor stochastic states. These algorithms are different for each kind of event.

4.1.1 Tick Event

Tick is an outcoming event from $S = \langle M, \vec{\mathcal{P}}\rangle$ iff $\mathcal{P}_t(0) = 0$ for every transition t progressing under marking M. If this is the case, *tick* is the unique outcoming event from S and its probability $P_{tick}(S)$ is equal to 1. In the transition from S to $S' = \langle M', \vec{\mathcal{P}}'\rangle$ through a tick event, the marking is not changed. Besides, the time to fire is maintained for each suspended transition and it is reduced by 1 for each progressing transition:

$$\mathcal{P}'_t(n) = \begin{cases} \mathcal{P}_t(n) & \textit{if } t \textit{ is suspended in } S \\ \mathcal{P}_t(n+1) & \textit{if } t \textit{ is progressing in } S. \end{cases} \tag{3}$$

4.1.2 Defer Event

Defer is an outcoming event for S iff $\mathcal{P}_t(0) < 1$ for every progressing transition and $\mathcal{P}_{t_o}(0) > 0$ for at least one progressing transition t_o (which means that every progressing transition t can have a time to fire higher than 0, and at least one progressing transition t_o can have time to fire equal to 0). If defer is an outcoming event, its probability is equal to:

$$P_{defer}(S) = \prod_{t \,\in\, firable(S)} (1 - \mathcal{P}_t(0)), \tag{4}$$

where $firable(S)$ denotes the set of transitions in S that are progressing and have a nonnull probability to fire at time 0.

In the stochastic state S' reached through a defer event, the marking remains unchanged while probability functions in the vector $\vec{\mathcal{P}}$ are updated differently for progressing and for suspended transitions. For each transition that was suspended in S, the time to fire remains unchanged and so is its probability function; whereas, for each transitions that was progressing in S, the time to fire is replaced through a new stochastic variable conditioned to be higher than 0:

$$\mathcal{P}'_t(n) = \begin{cases} \mathcal{P}_t(n) & \textit{if } t \textit{ is suspended in } S \\ 0 & \textit{if } t \textit{ is progressing in } S \\ & \textit{and } n = 0 \\ \mathcal{P}_t(n) * \frac{1}{1-\mathcal{P}_t(0)} & \textit{if } t \textit{ is progressing in } S \\ & \textit{and } n \neq 0. \end{cases} \tag{5}$$

According to (5), in every stochastic state reached through a $defer$ event, $tick$ is the only possible outcoming event.

4.1.3 Firing Event

The firing of a set $\phi = \{t_n\}_{n=o}^{N-1}$ is an outcoming event for S iff: ϕ contains only progressing and nonconflicting transitions, every transition $t \in \phi$ can fire before time advances, and every progressing transition $t_{nc} \notin \phi$ which is not conflicting with ϕ can be delayed, i.e.,

$$\begin{array}{ll} \mathcal{P}_t(0) > 0 & \forall t \in \phi \\ \mathcal{P}_{t_{nc}}(0) < 1 & \forall t_{nc} \notin \phi, t_{nc} \textit{ progressing} \\ & \textit{and not conflicting with } \phi. \end{array} \tag{6}$$

The probability $P_\phi(S)$ that the firing set ϕ is the outcoming event from S jointly depends on the probability that each and every transition in ϕ attempts the firing and on the probability that the attempt is not impaired by conflicts with other transitions. For the derivation, $P_\phi(S)$ can be decomposed as:

$$P_\phi(S) = \sum_{\sigma \in Att(S)} P_{\phi|\sigma} \cdot P_\sigma(S), \tag{7}$$

where $Att(S)$ denotes the set of attempting sets that are feasible in S, $P_\sigma(S)$ is the probability that σ is the attempting set for a $\vec{ttf}$ distributed according to $\vec{\mathcal{P}}$, and $P_{\phi|\sigma}$ is the probability that ϕ is the firing set under the condition that σ is the attempting set.

Since times-to-fire of different transitions are independent variables, $P_\sigma(S)$ can be expressed as the product:

$$P_\sigma(S) = \prod_{t_i \in \sigma} \mathcal{P}_{t_i}(0) \prod_{t_j \in firable(S)\setminus\sigma} \big(1 - \mathcal{P}_{t_j}(0)\big), \tag{8}$$

where $firable(S)\setminus\sigma$ denotes the set of firable transitions that are not included in the attempting set.

Besides, $P_{\phi|\sigma}$ can be decomposed so as to distinguish the different orderings in which transitions of the *attempting set* σ are sampled in the repetitive stochastic selection defined in the clause of firability:

$$P_{\phi|\sigma} = \sum_{seq \in Perm(\sigma)} P_{\phi|seq} \cdot P_{seq|\sigma}, \tag{9}$$

where $Perm(\sigma)$ is the set of permutations of the elements of σ and seq denotes any of such permutations. $P_{seq|\sigma}$ is the probability that the transitions in the attempting set σ are selected in the order defined by seq (i.e., transition t_{seq_i} is selected before transition $t_{seq_{i+1}}$), which can be expressed as:

$$P_{seq|\sigma} = \prod_{i=1}^{||seq||} \frac{\mathcal{C}(t_{seq_i})}{\sum_{h=i}^{||seq||} \mathcal{C}(t_{seq_h})}. \quad (10)$$

Finally, $P_{\phi|seq}$ is equal to 1 iff every transition $t \in \phi$ appears in seq before any other transition which has a conflict with t itself and it is equal to 0 otherwise.

For the sake of an efficient implementation, it is useful to note that $P_{\phi|\sigma}$ appearing in (7) does not depend on the stochastic state and can be computed only once along the enumeration of the entire state space. In fact, $P_{seq|\sigma}$ is determined by the competitiveness factor $\mathcal{C}$, which is not affected by the stochastic state, and $P_{\phi|seq}$ is determined by structural properties of the net.

In the stochastic state reached through the firing of a transition set, the marking is changed according to the firing clause of spTPNs. The components of the vector $\vec{\mathcal{P}}$ are updated differently for transitions that are either newly enabled or persistent after the firing of ϕ. For each newly enabled transition t_a, the time to fire is a new stochastic variable distributed according to the static probability function $\mathcal{D}(t_a)$. For each persistent transition t_w that was suspended in S, the time to fire remains unchanged. Finally, for each persistent transition t_i that was progressing in S, the time to fire becomes conditioned to be higher than 0. In fact, under a *maximal* step semantics, the observation of a progressing transition t_i with a random time to fire ttf_{t_i} which does not attempt the firing is equivalent to the assumption that ttf_{t_i} is higher than 0. Basically, this is the consequence of the *nonnull defer* property mentioned in Section 3.1.

According to this, the vector of times to fire of enabled transitions after the firing of ϕ is distributed according to $\mathcal{P}'$:

$$\mathcal{P}'_t(n) = \begin{cases} \mathcal{D}_t(n) & \text{if } t \text{ is newly enabled} \\ \mathcal{P}_t(n) & \text{if } t \text{ is persistent in } S' \\ & \text{and suspended in } S \\ 0 & \text{if } t \text{ is persistent in } S' \\ & \text{and progressing in } S \\ & \text{and } n = 0 \\ \mathcal{P}_t(n) * \frac{1}{1-\mathcal{P}_t(0)} & \text{if } t \text{ is persistent in } S' \\ & \text{and progressing in } S \\ & \text{and } n > 0. \end{cases} \quad (11)$$

4.2 Boundedness

In general, enumeration of stochastic states is a semi-algorithm as termination is not guaranteed. However, under quite general conditions on the form of static probability mass functions, unboundedness of the stochastic step transition system STS can only derive from an unbounded number of reachable markings:

Theorem 1. *Assume that each transition t has a* static memory scope $\bar{k}_t$ *after which the static probability mass function $\mathcal{D}_t()$ is either null or geometrically distributed:*

$$\forall t \in T, \exists \bar{k}_t \in \mathbb{N}.(\forall k \geq \bar{k}_t, \mathcal{D}_t(k) = 0) \vee (\exists \lambda_t \in (0,1). \forall k \geq \bar{k}_t, \mathcal{D}_t(k) = \lambda_t(1-\lambda_t)^k).$$

If the STS is unbounded, then it includes an infinite number of different markings.

Proof. Ab absurdo, assume that the STS includes an unbounded number of stochastic states with a finite number of markings. This implies the existence of a transition t which is enabled with different *dynamic* probability mass functions in an unbounded number of stochastic states. In any stochastic state in which t is newly enabled, $\mathcal{P}_t()$ is equal to $\mathcal{D}_t()$, and thus exists a *dynamic memory scope* $\bar{k}^d_t = \bar{k}_t$ such that after $\bar{k}^d_t$, $\mathcal{P}_t()$ is either null or geometrically distributed.

Starting from any stochastic state where t is newly enabled, the form of $\mathcal{P}_t()$ can evolve through the repeated application of the transformations corresponding to the events of *firing*, *tick*, and *defer*. However, in these transformations only a finite number of different forms for $\mathcal{P}_t()$ can be generated. In fact: at every *firing* or *defer*, $\mathcal{P}_t()$ is either reset (if t takes part in the firing or if it is disabled by some fired transition) or it is conditioned so that $\mathcal{P}_t(0) = 0$, in which case the form of $\mathcal{P}_t()$ cannot change again until the next tick event; at every *tick*, the dynamic memory scope $\bar{k}^d_t$ is decreased towards 0, so that only a finite number of ticks can occur before $\mathcal{P}_t()$ becomes entirely concentrated in 0 (i.e., $\mathcal{P}_t(0) = 1$) or it assumes a geometric form. □

4.3 Correctness Verification

Properties pertaining to states and behaviors in the semantics of an spTPN model can be recovered from the analysis of stochastic states and paths in the STS. To this end, a stochastic state S can be regarded as a collection of states sharing a common marking but having different times to fire for enabled transitions. In this perspective, we say that a state $s = \langle m, \vec{\tau} \rangle$ *is contained* in a stochastic state $S = \langle M, \vec{\mathcal{P}} \rangle$, and we write $s \in S$, when $m = M$ and $\mathcal{P}_t(\tau_t) > 0$ for every enabled transition t.

The relation between states and stochastic states is made explicit in a *contracted step transition system* $\bar{STS} = \langle \bar{\mathcal{S}}, S_{root}, \overset{ev}{\Longrightarrow} \rangle$ which is derived from $STS = \langle \mathcal{S}, S_{root}, \overset{ev,\mu}{\Longrightarrow} \rangle$ by neglecting the measure of probability μ and by concealing the *defer* event so as to observe only those stochastic states that are reached through a *tick* or a *firing*:

$$\begin{aligned} S \in \bar{\mathcal{S}} \quad & iff \quad S \in \mathcal{S} \text{ and } \exists S_o \in \mathcal{S} \text{ such that} \\ & \qquad S_o \overset{tick,\mu}{\Longrightarrow} S \text{ or } S_o \overset{\phi,\mu}{\Longrightarrow} S \\ S \overset{\phi}{\Longrightarrow} \bar{S} \quad & iff \quad S \overset{\phi,\mu}{\Longrightarrow} \bar{S} \\ S \overset{tick}{\Longrightarrow} \bar{S} \quad & iff \quad S \overset{tick,\mu}{\Longrightarrow} \bar{S} \text{ or } \exists \bar{S}' \in \mathcal{S} \text{ such that} \\ & \qquad S \overset{defer,\mu'}{\Longrightarrow} \bar{S}' \text{ and } \bar{S}' \overset{tick,\mu}{\Longrightarrow} \bar{S}. \end{aligned} \quad (12)$$

The following two lemmas establish a bidirectional correspondence between the successor relation $\overset{ev}{\Longrightarrow}$ between stochastic states in the contracted STS and the reachability relation $\overset{ev,\mu}{\rightarrow}$ between states in the semantics of an spTPN model:

Lemma 1. *If $s \in \bar{S}$ and $s \overset{ev,\mu}{\rightarrow} s_*$, then $\bar{S} \overset{ev}{\Longrightarrow} \bar{S}_*$ with $s_* \in \bar{S}_*$.*

Proof. If $s \overset{tick,\mu}{\rightarrow} s_*$, then, for every transition t_p progressing in s, $ttf_{t_p} > 0$. Since $s \in \bar{S}$, this implies $\mathcal{P}^{\bar{S}}_{t_p}(0) < 1$, which guarantees that $\bar{S}$ accepts a tick as outcoming event.

If $s \xrightarrow{\phi,\mu} s_*$, then ϕ includes only progressing and nonconflicting transitions, for every $t_i \in \phi$, $ttf_{t_i} = 0$, and, for every $t_a \notin \phi$ which is progressing and not in conflict with ϕ, $ttf_{t_a} > 0$. The assumption $s \in \bar{S}$ thus implies $\mathcal{P}^{\bar{S}}_{t_i}(0) > 0$ and $\mathcal{P}^{\bar{S}}_{t_a}(0) < 1$, which guarantees that $\bar{S}$ accepts ϕ as an outcoming event.

By comparing the derivation of the marking and the times to fire of s_* from s against the derivation of the marking and the probability mass function of $\bar{S}_*$ from $\bar{S}$, respectively, it can be easily shown that the marking of s_* is equal to that of $\bar{S}_*$ and that every transition t enabled in s_* has a time to fire within the support of the probability mass function $\mathcal{P}^{\bar{S}_*}_t()$, which concludes the proof. □

Lemma 2. *If* $\bar{S} \overset{ev}{\Longrightarrow} \bar{S}_*$, *then* $\forall s_* \in \bar{S}_*$, $\exists s \in \bar{S}$, *and* $\mu \in (0,1]$ *such that* $s \xrightarrow{ev,\mu} s_*$.

Proof. Let $s_* = \langle M_*, \vec{\tau}^* \rangle$ be a state contained in $\bar{S}_*$.

If $\bar{S} \overset{tick}{\Longrightarrow} \bar{S}_*$, consider the state $s = \langle M, \vec{\tau} \rangle$, where M is the marking of $\bar{S}$ and $\bar{S}_*$ and $\vec{\tau}$ is a vector of times to fire for the transitions enabled by M defined as follows:

$$\tau_t = \begin{cases} \tau^*_t + 1 & \text{if } t \text{ is progressing in } \bar{S} \\ \tau^*_t & \text{if } t \text{ is suspended in } \bar{S}. \end{cases} \quad (13)$$

Since in $\vec{\tau}$ the time to fire of any progressing transition is higher than 0, tick is the next event from s and, by construction, it leads to s_* (i.e., $s \xrightarrow{tick,1} s_*$). In addition, since $s_* \in \bar{S}_*$, for any t enabled in $\bar{S}_*$, $\mathcal{P}^{\bar{S}_*}_t(\tau^*_t) > 0$. The assumption $\bar{S} \overset{tick}{\Longrightarrow} \bar{S}_*$ thus implies that, for any t progressing in $\bar{S}$, $\mathcal{P}^{\bar{S}}_t(\tau^*_t + 1) > 0$ and, for any t suspended in $\bar{S}$, $\mathcal{P}^{\bar{S}}_t(\tau^*_t) > 0$, which is sufficient to demonstrate that $s \in \bar{S}$.

If $\bar{S} \overset{\phi}{\Longrightarrow} \bar{S}_*$, we construct a state $s = \langle M, \vec{\tau} \rangle$, where the time to fire of each transition enabled by M is defined as follows: For each $t_i \in \phi$, let $\tau_{t_i} = 0$; note that, with this choice, $\mathcal{P}^{\bar{S}}_{t_i}(\tau_{t_i}) > 0$ as a consequence of the assumption that ϕ is an outcoming event from $\bar{S}$. For each t_p persistent at the firing of ϕ, let $\tau_{t_p} = \tau^*_{t_p}$. Also in this case $\mathcal{P}^{\bar{S}}_{t_p}(\tau_{t_p}) > 0$; in fact, since $s_* \in \bar{S}_*$ and t_p is enabled in $\bar{S}_*$, $\mathcal{P}^{\bar{S}_*}_{t_p}(\tau^*_{t_p}) > 0$, which, by the assumption $\bar{S} \overset{\phi}{\Longrightarrow} \bar{S}_*$, implies that $\mathcal{P}^{\bar{S}}_{t_p}(\tau^*_{t_p}) > 0$. For each $t_{nc} \notin \phi$ which is progressing under M and is not persistent at the firing of ϕ and is not in conflict with ϕ, let $\tau_{t_{nc}}$ be any value higher than 0 such that $\mathcal{P}^{\bar{S}}_{t_{nc}}(\tau_{t_{nc}}) > 0$; such value exists as a consequence of the assumption that ϕ is an outcoming event from $\bar{S}$. For each other transition t_x, let τ_{t_x} be any value such that $\mathcal{P}^{\bar{S}}_{t_x}(\tau_{t_x}) > 0$.

By construction, $s \in S$. In addition, according to the semantics of spTPNs, ϕ is the next step in s under a feasible result of the competitive selection, and it leads to s_* under a feasible sampling of times to fire of transitions that are newly enabled at the firing of ϕ. This guarantees that $\exists \mu > 0$ such that $s \xrightarrow{\phi,\mu} s_*$. □

The correspondence between the succession relation $\overset{ev}{\Longrightarrow}$ in the $\bar{STS}$ and the direct reachability relation $\xrightarrow{ev,\mu}$ in the semantics of spTPNs can be inductively extended to deal with a step sequence ρ. On the one hand, if a state s^* can be reached from a state $s \in \bar{S}$ through the step sequence ρ, then the $\bar{STS}$ contains a path which originates in $\bar{S}$, follows the same steps of ρ, and leads to a stochastic state $\bar{S}^*$ such that $s^* \in \bar{S}^*$. On the other hand, if the $\bar{STS}$ contains a path ρ which originates in $\bar{S}$ and leads to a stochastic state $\bar{S}^*$, then, for every state $s^* \in \bar{S}^*$, there exists a state $s \in \bar{S}$ from which it is possible to reach s^* through the steps of the sequence ρ.

This supports the verification of correctness requirements pertaining to ordered sequencing and time bounded response through real-time model checking techniques [34], [37], [3] or shortest and longest path algorithms [1]. In so doing, the analysis of spTPNs supports the designer with results that are similar to those that could be obtained through (dense time) formalisms accounting for preemption and finite delays, such as preemptive-TPNs [20], scheduling-TPNs [49], Timed Automata with Tasks [4].

In this perspective, it is worth noting that the $\bar{STS}$ directly represents the step semantics of spTPNs without requiring that the concurrency of the model be recovered from interleaving sequences as occurring in analysis techniques for most (dense or discrete) timed extensions of Petri Nets.

4.4 Performance and Dependability Evaluation

The step transition system $STS = \langle S_o, \mathcal{S}, \overset{ev,\mu}{\Longrightarrow} \rangle$ can be regarded as a discrete time Markov chain $\{\Sigma_n, n \in \mathbb{N}\}$, where Σ_n is the stochastic state reached after n steps and takes values in $\mathcal{S}$, and transition probabilities are encoded by the succession relation $\overset{ev,\mu}{\Longrightarrow}$:

$$Prob\{\Sigma_{n+1} = S^* | \Sigma_n = S\} = \begin{cases} \mu & \text{if } \exists\, ev \text{ such that} \\ & S \overset{ev,\mu}{\Longrightarrow} S^* \\ 0 & \text{otherwise.} \end{cases} \quad (14)$$

As already mentioned in Section 4, the STS can be partitioned in stochastic states with sojourn time equal to 1 (those accepting tick as the unique outcoming event) and stochastic states with null sojourn time (those accepting firing and defer events).

The two cases closely correspond to the concepts of *tangible* and *vanishing* states commonly used in the literature of stochastic Petri Nets [45]. As a difference, while the term vanishing usually denotes a state enabling any immediate transition, in our treatment, we use it to denote a stochastic state where the *dynamic* time to fire of some progressing transition has reached a nonnull probability to be equal to 0. Vanishing states can be eliminated from the STS yielding a *reduced* step transition system $\hat{STS} = \langle \hat{\mathcal{S}}, S_{root}, \hat{\Longrightarrow} \rangle$, where ρ denotes a sequence of steps connecting two stochastic states with sojourn time equal to 1 (by construction, any such sequence is made up of an initial *tick* followed by any number of *firing* events and at most one single *defer* event in the terminal position) and μ is the product of probabilities associated with the steps along ρ.

$\hat{STS}$ encodes a discrete time Markov chain $\{\hat{\Sigma}_m, m \in \mathbb{N}\}$, where $\hat{\Sigma}_m$ is the stochastic state reached after m time units

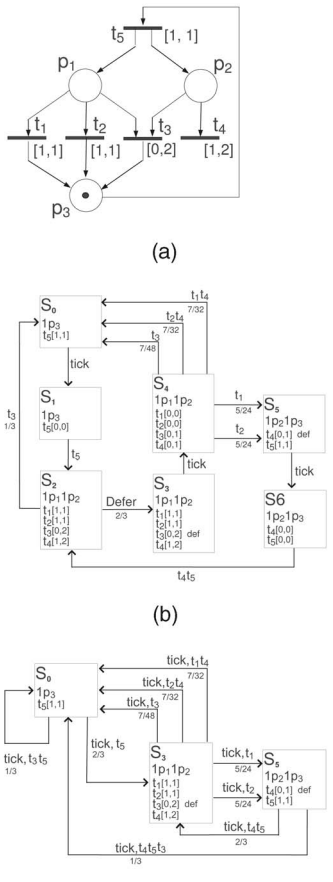


Fig. 5. (a) a simple example spTPN: static times to fire of transitions t_3 and t_4 are assumed to be uniformly distributed; competitiveness factors are equal to 1 for all transitions. (b) The STS and (c) its reduction $\hat{STS}$ concealing vanishing states.

since the beginning of the execution in the initial stochastic state. This supports derivation of transient and steady state probabilities through techniques that are commonly employed on a variety of stochastic timed extension of Petri Nets based on dense [45], [32] or discrete time [15], [33], which permit the derivation of cumulative and instantaneous indexes of dependability and performance [39].

4.5 An Example

We illustrate semantics and analysis of spTPNs with reference to the net of Fig. 5a.

In the STS reported in Fig. 5b, in the initial stochastic state S_0, only t_5 is enabled with $\mathcal{P}^{S_0}_{t_1}(1) = 1$ (i.e., the time to fire is determined and equal to 1): *tick* is thus the unique outcoming event and leads to S_1, where t_5 becomes compelled to attempt the firing (i.e., $\mathcal{P}^{S_1}_{t_5}(0) = 1$). In S_2, t_1, t_2, and t_4 are enabled but have a null probability to fire before time advances. Transition t_3 attempts the firing with probability $\frac{1}{3}$, or defers with probability $\frac{2}{3}$.

In S_4, t_3, and t_4 can attempt the firing or defer with equal probabilities, while t_1 and t_2 are compelled to attempt the firing. Four different attempting sets are thus possible: ($\{t_1, t_2\}$, $\{t_1, t_2, t_3\}$, $\{t_1, t_2, t_4\}$, and $\{t_1, t_2, t_3, t_4\}$). The probability of each case depends on the probability mass functions of times to fire and is computed according to (8).

Due to conflicts, each attempting set may result in different firing sets, with probabilities expressed according to (9). For instance, in the most complex case in which all four transitions take part in the attempt, three firing sets are possible: $\{t_1, t_4\}$, $\{t_2, t_4\}$, and $\{t_3\}$ with probabilities equal to $\frac{3}{8}$, $\frac{3}{8}$, and $\frac{1}{4}$, respectively.

In particular, the attempting set $\{t_1, t_2, t_3, t_4\}$ yields the firing set $\{t_1, t_4\}$ if and only if t_1 is selected from the attempting set before t_2 and t_3 *or* t_4 is selected before t_3 *and* t_1 is selected before t_2. The probability of each case is derived according to (10). Note that, according to (9), even when competitiveness factors $\mathcal{C}$ are equal, transitions involved in multiple conflicts receive a lower probability to be included in the firing set. In the example, this is the case of transition t_3, whose firing requires that t_3 prevails over t_1, t_2, and t_4; conversely, t_4 is advantaged by the same mechanism as its firing will occur either because it prevails over t_3 or because t_3 has been overtaken by t_1 or t_2. For this reason, the probabilities of firing sets in S_4 are biased to include t_4 rather than t_3, though t_4 and t_3 have equal probabilities to attempt the firing and equal competitiveness factors.

Also, note that the same firing set can result from different attempting sets. For instance, in S_4, the firing set $\{t_1, t_4\}$ can also result from the attempting set $\{t_1, t_2, t_4\}$ in the case that t_1 prevails over t_2. This combines the effects of time probabilities (which define the choice among attempting sets) and logical concurrency of *direct* conflicts and competitiveness factors (which define the choice among firing sets that may result from each attempting set).

5 A Case Study

In this section, we show how spTPNs can be used to deal with both the problems of qualitative verification and quantitative evaluation, with reference to the case of a digital control system.

In digital control systems, stability and quality of control depend on the frequency and the time-invariance of control actions. In a worst-case approach, this is achieved by assuming a sampling period longer than the maximum possible delay between the acquisition of a sample and the application of its consequent control action. However, this becomes nonviable or ineffective when limited resources make computation times and scheduling delays significantly variant. In this condition, a control system which assumes a shorter sampling period and applies adequate scheduling strategies to manage variability and occasional overruns can exhibit a better performance and still maintain a correct (stable) behavior [57], [28]. This raises a design trade-off between the nominal sampling period and the probability of a *jitter* in the timing of control actions. While the maximum jitter is a matter of correctness determining stability, the average jitter and the nominal sampling period are a matter of performance affecting the quality of control.

Fig. 6a presents the basic spTPN model of a digital control thread. Output variables of the system are sampled at a constant rate (transition t_S: *sample*). For each sample, the controller calculates the control action to be applied to the system (transition t_{CO}: *compute output*) and updates the state of the controller (transition t_{US}: *update state*). Both *compute*

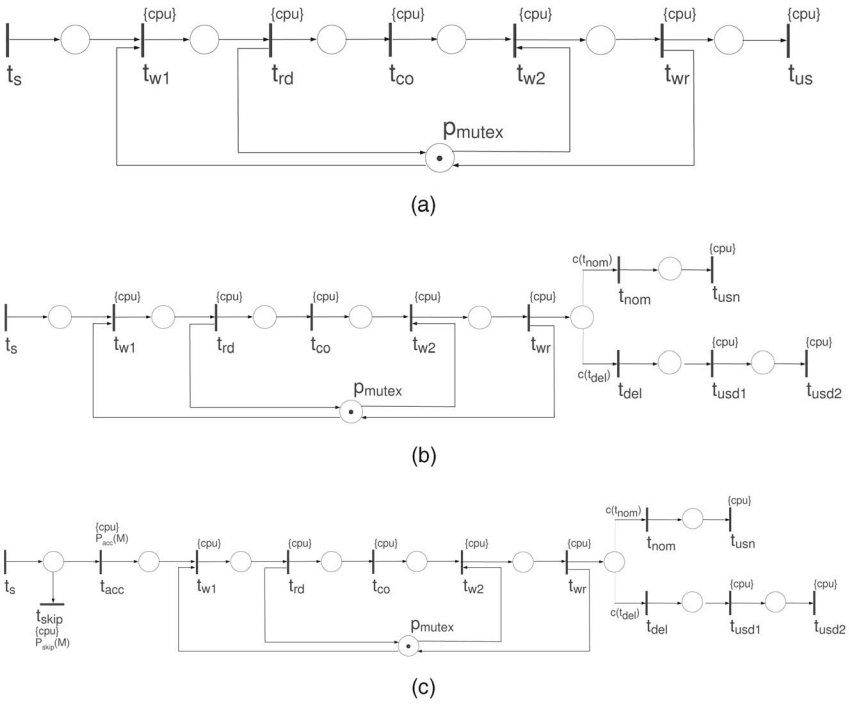


Fig. 6. (a) The basic spTPN model of a digital control thread. (b) A refined model accounting for the choice between the nominal or the delayed duration of the *update state* computation. (c) An extended model representing the implementation of a *skip strategy* for overrun handling.

output and *update state* are allocated to a resource (*cpu*) which is usually shared with other similar threads. In addition, the acquisition of the sample (transition t_{rd}, *read*) and the application of the control action (transition t_{wr}, *write*) are protected within critical sections preventing interference with other threads in the access to a card for AD and DA conversions (place p_{mutex}). With reference to this model, control jitter is measured as the variation in the time elapsed between subsequent firings of transition t_{wr} (i.e., between the application of subsequent control actions).

In [27] and [28], two complementary scheduling strategies for the reduction of the control jitter are devised and assessed through experimental evaluation of two similar testbenches. According to [27], in each control thread the variant part of the computation represented by t_{CO} can be moved into the code segment represented by t_{US}. The resulting timing of t_{US} is well represented as a nominal value plus an additional occasional delay which is uniformly distributed [28]. To account for this special distribution, the spTPN model of each control thread is slightly refined as reported in Fig. 6b: Transitions t_{nom} and t_{del} account for the choice between the case of a nominal or a delayed duration of the *state update*; their relative probability is set through the respective competitiveness factors $c(t_{nom})$ and $c(t_{del})$; transition t_{USn} is the nominal case, while the sequence of t_{USd1} plus t_{USd2} accounts for the delayed case.

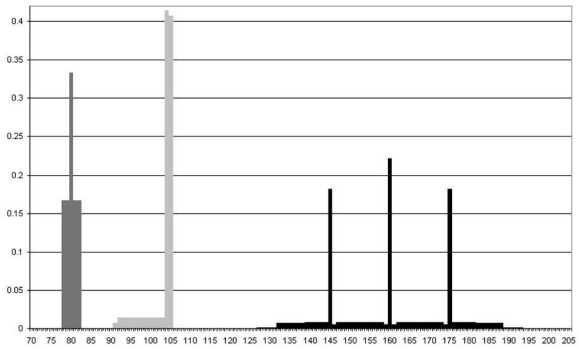
In the testbench of [27], three inverted pendula are controlled by three concurrent threads with equal computations but different sampling periods, running on the same cpu and sharing an exclusive AD/DA converter through a semaphore. This results in a tasking set that can be modeled through three instances of the pattern of Fig. 6b composed so as to share place p_{mutex}. Temporal parameters and priorities of the tasking set are reported in Fig. 7a. Following the Theory of Rate Monotonic, higher priorities are assigned to threads with shorter period, and computations in the critical sections are boosted according to a Highest Priority Locking protocol. State space enumeration produces 7,251 stochastic states with 73 different markings. Fig. 7b reports the resulting minimum/maximum bounds and the distribution of probability for the jitter of control actions for each thread. Note that, due to the possible latency in the acquisition of the exclusive resource, the high priority thread also suffers of a timing variability. For both the mid and low-priority threads, variability also depends on *update state* computations of higher priority threads.

In order to reduce the jitter, [27] proposes that *calculate output* computations are given precedence with respect to *state update* computations. This changes the parameters of the model of the tasking set as described in Fig. 8a. In this case, the analysis produces 7,063 stochastic states with 103 different markings, which yield the results of Fig. 8b. Timing variability is largely reduced for all the threads, though the gap between the minimum and the maximum time between subsequent control actions increases for the mid priority thread.

According to [28], the quality of control can be further improved by reducing the sampling period at the expense of occasional overruns, provided that the maximum jitter is maintained within the limits required for the stability of control [29]. To this end, in [28], various overrun management policies are devised: In the *queue strategy*, when a sample is late, the derivation of its successor is delayed; in the *abort strategy*, a late sample is aborted and the derivation of its successor is started on time; in the *skip* strategy, late samples are completed but their successors are skipped. We

	High priority			Mid Priority			Low Priority		
Trans.	Time	Prio	c.f.	Time	Prio	c.f.	Time	Prio	c.f.
t_s	80	-	-	120	-	-	160	-	-
t_{w1}	0	17	-	0	15	-	0	13	-
t_{rd}	1	23	-	1	21	-	1	19	-
t_{CO}	8	9	-	8	5	-	8	1	-
t_{w2}	0	18	-	0	16	-	0	14	-
t_{wr}	1	24	-	1	22	-	1	20	-
t_{USn}	5	10	-	5	6	-	5	2	-
t_{nom}	0	-	0.8	0	-	0.8	0	-	0.8
t_{del}	0	-	0.2	0	-	0.2	0	-	0.2
t_{USd1}	5	11	-	5	7	-	5	3	-
t_{USd2}	[0,13]	12	-	[0,13]	8	-	[0,13]	4	-

(a)

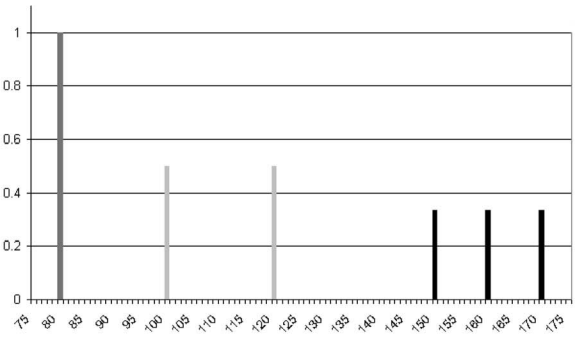


(b)

Fig. 7. (a) Transition temporal parameters, competitiveness factors, and priorities for each thread. Task priorities are assigned according to Rate Monotonic. (b) Minimum/maximum bounds and probability distribution for the jitter of control actions. Shades distinguish results obtained for the different threads (dark gray for the high priority thread, light gray for mid priority and black for low priority).

	High priority			Mid Priority			Low Priority		
Trans.	Time	Prio	c.f.	Time	Prio	c.f.	Time	Prio	c.f.
t_s	80	-	-	120	-	-	160	-	-
t_{w1}	0	16	-	0	13	-	0	10	-
t_{rd}	1	23	-	1	21	-	1	19	-
t_{CO}	8	18	-	8	15	-	8	12	-
t_{w2}	0	17	-	0	14	-	0	11	-
t_{wr}	1	24	-	1	22	-	1	20	-
t_{USn}	5	7	-	5	4	-	5	1	-
t_{nom}	0	-	0.8	0	-	0.8	0	-	0.8
t_{del}	0	-	0.2	0	-	0.2	0	-	0.2
t_{USd1}	5	8	-	5	5	-	5	2	-
t_{USd2}	[0,13]	9	-	[0,13]	6	-	[0,13]	3	-

(a)



(b)

Fig. 8. (a) Parameters for the tasking set of Fig. 6a, with the priorities assigned according to the *subtask scheduling* strategy. (b) The time between the application of two subsequent control actions becomes independent from the high variability of *update state* computations.

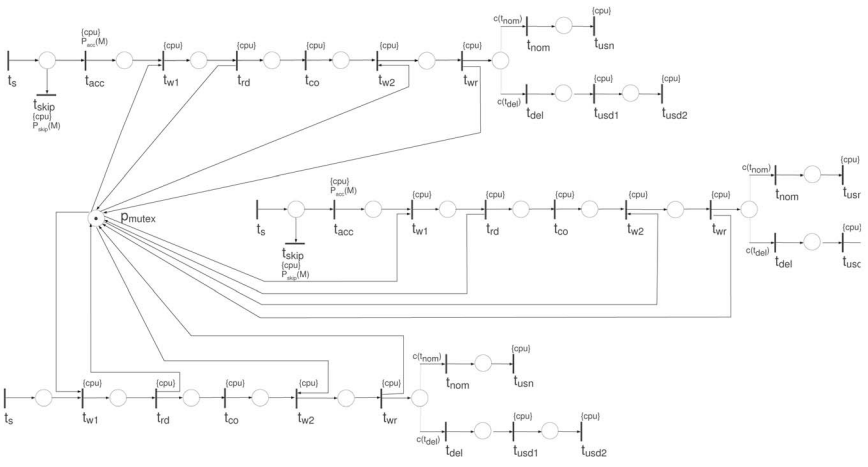
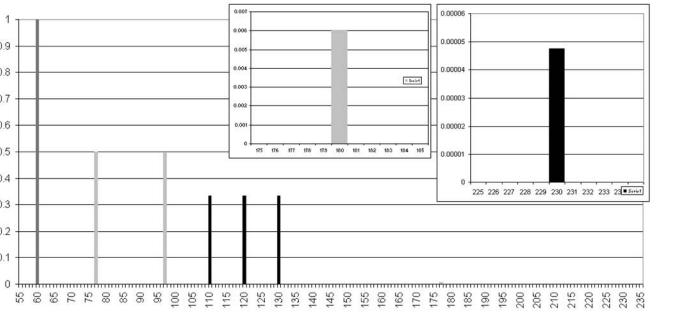


Fig. 9. The complete model of the three control threads with the implementation of *subtask scheduling* of [27] for every thread, combined with the *skip strategy* of [28] to manage occasional overruns in the low and the mid-priority threads. Temporal parameters and priorities are reported in Fig. 10a.

focus here on the skip strategy, which is represented by refining the model of each thread as in Fig. 6c so that the overall tasking set is modeled by the spTPN in Fig. 9 with the parameters of Fig. 10a. The priority of t_{skip} depends on the marking: When t_{skip} is newly enabled by the arrival of a new sample (firing of t_S) and if the processing of the previous sample is still pending (i.e., any further token is contained in any place along the thread), the priority of t_{skip} becomes higher than that of t_{acc}, so that the token is consumed to represent the skip of the sample. The analysis produces 15,567 stochastic states with 127 different markings, with the results shown in Fig. 10b. While the high priority thread is not affected by the policy, for both the mid and low-priority threads, the distribution of times between the application of two subsequent control actions exhibits a significant reduction. Nevertheless, for both of them, the maximum value increases. The value of the maximum has a limited impact on quantitative performance evaluation but definitely affects the stability of control and, thus, the qualitative correctness of design.

	High priority			Mid Priority			Low Priority		
Trans.	Time	Prio	c.f.	Time	Prio	c.f.	Time	Prio	c.f.
t_s	60	-	-	90	-	-	120	-	-
t_{w1}	0	16	-	0	13	-	0	10	-
t_{acc}	-	-	-	0	29	-	0	26	-
t_{skip}	-	-	-	0	$P_{skip}(M)$	-	0	$P_{skip}(M)$	-
t_{w2}	0	17	-	0	14	-	0	11	-
t_{rd}	1	23	-	1	21	-	1	19	-
t_{CO}	8	18	-	8	15	-	8	12	-
t_{wr}	1	24	-	1	22	-	1	20	-
t_{USn}	5	7	-	5	4	-	5	1	-
t_{nom}	0	-	0.8	0	-	0.8	0	-	0.8
t_{del}	0	-	0.2	0	-	0.2	0	-	0.2
t_{USd1}	5	8	-	5	5	-	5	2	-
t_{USd2}	[0,13]	9	-	[0,13]	6	-	[0,13]	3	-

(a)



(b)

Fig. 10. (a) Parameters for the tasking set of Fig. 9, where sampling periods are reduced by 25 percent and skip strategies are applied to low and mid priority tasks to manage occasional overruns. (b) The average time between two subsequent control actions is reduced, but occasional overruns produce an increment of the worst case.

All the analysis techniques described in the paper are implemented in the discrete time suite of the ORIS tool [21]. Running the analysis on a Pentium 4 with 512 mbytes and 1.5 gigahertz clock, enumeration is carried out in few seconds for all the cases, while stochastic analysis requires a maximum of a 10 minutes in the most complex case of skip implementation.

6 Conclusions

Stochastic preemptive Time Petri Nets (spTPN) support the representation of real-time systems running under preemptive scheduling on multiple processors, including periodic, sporadic, and asynchronous task release, with nondeterministic possibly bounded computation times, with synchronization and precedence constraints, with flexibility policies adapting behavior to dynamic conditions. They also encompass further aspects not reported in this paper including, in particular, marking-dependent temporal parameters which can represent performance polymorphism [42]. While most of this expressivity was already attained in the dense time formulation of preemptive Time Petri Nets (pTPN) [20], spTPNs assume a discrete time and they add a stochastic characterization of all nondeterministic logical and temporal choices.

The salient trait of novelty in the semantics of spTPNs consists in the use of a maximal step semantics of concurrency which enables a convenient separation of the effects of logical concurrency from those of nonMarkovian transition delays. This provides a structural solution for the problems of confusion and non-well-definedness arising in interleaved models and exacerbated by the assumption of a discrete time, and relieves the designer from the burden of identifying and characterizing conflicts which do not correspond to an explicit modeling intention.

The timed state space of the model is covered using the concept of stochastic state, a state class which integrates the enumeration of a succession relation among sets of timed states with the calculus of their probability distribution. The enumeration technique directly represents the step semantics without requiring that the concurrency of the model be recovered from the analysis of interleaving sequences, as occurring in the analysis of most (dense or discrete) timed extensions of Petri Nets.

In the resulting stochastic step transition system, information about sequencing resulting from explicit synchronization constraints and implicit precedences induced by non-Markovian timed behavior is combined with a measure of probability characterizing both logical and temporal choices. This provides a novel framework unifying the qualitative verification of correctness in the logical sequencing and timing of events with the quantitative evaluation of dependability and performance indexes.

Acknowledgments

This research was partially supported by the Italian Ministry of Instruction University and Research as a part of the FIRB project *Performance Evaluation of Complex Systems* (PERF).

References

[1] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows.* Prentice Hall, 1993.

[2] R. Alur and D.L. Dill, "Automata for Modeling Real-Time Systems," *Proc. 17th Int'l Colloquium on Automata, Languages, and Programming*, 1990.

[3] R. Alur and T.A. Henzinger, "Logics and Models of Real-Time: A Survey," *Real-Time: Theory in Practice*, 1992.

[4] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi, "Times—A Tool for Modelling and Implementation of Embedded Systems," *Proc. Eighth Int'l Conf. Tools and Algorithms for the Construction and Analysis of Systems*, 2002.

[5] H. Ben-Abdallah, J.-Y. Choi, D. Clarke, Y.-S. Kim, I. Lee, and H.-L. Xie, "A Process Algebraic Approach to the schedulability Analysis of Real-Time Systems," *Real-Time Systems*, vol. 15, no. 3, pp. 189-219, 1998.

[6] J. Bengtsson, K.G. Larsen, F. Larsson, P. Pettersson, and W. Yi, "UPPAAL: A Tool-Suite for Automatic Verification of Real-Time Systems," *Hybrid Systems III*, 1996.

[7] A. Benveniste, P. Caspi, S.A. Edwards, N. Halbwachs, P. Le Guernic, and R. DeSimone, "The Synchronous Languages 12 Years Later," *Proc. IEEE*, vol. 91, no. 1, pp. 64-83, Jan. 2003.

[8] G. Berry and G. Gonthier, "The ESTEREL Synchronous Programming Language: Design, Semantics, Implementation," *Science of Computer Programming*, vol. 19, no. 2, pp. 87-152, 1992.

[9] B. Berthomieu and M. Diaz, "Modeling and Verification of Time Dependent Systems Using Time Petri Nets," *IEEE Trans. Software Eng.*, vol. 17, no. 3, Mar. 1991.

[10] B. Berthomieu and M. Menasche, "An Enumerative Approach for Analyzing Time Petri Nets," *Proc. IFIP Congress*, Sept. 1983.

[11] B. Berthomieu, P.-O. Ribet, and F. Vernadat, "The Tool TINA-Construction of Abstract State Spaces for Petri Nets and Time Petri Nets," *Int'l J. Production Research,* 2004.

[12] A. Bobbio and A. Horvath, "Petri Nets with Discrete Phase Type Timing: A Bridge between Stochastic and Functional Analysis," *Electronic Notes in Theoretical Computer Science,* vol. 52, no. 3, 2001.

[13] A. Bobbio, A. Puliafito, M. Scarpa, and M. Telek, "WebSPN: Non-Markovian Stochastic Petri Net Tool," *Proc. 18th Int'l Conf. Application and Theory of Petri Nets,* June 1997.

[14] A. Bobbio, A. Puliafito, M. Telek, and K. Trivedi, "Recent Developments in Non-Markovian Stochastic Petri Nets," *J. Systems Circuits and Computers,* vol. 8, no. 1, pp. 119-158, 1998.

[15] A. Bobbio and M. Scarpa, "Kronecker Representation of Stochastic Petri Nets with Discrete PH Distributions," *Proc. Third IEEE Ann. Int'l Computer Performance and Dependability Symp. (IPDS '98),* Sept. 1998.

[16] R. Bruni, J. Meseguer, U. Montanari, and V. Sassone, "A Comparison of Petri Net Semantics under the Collective Token Philosophy," *Proc. Asian Computing Science Conf. (ASIAN),* pp. 225-244, 1998.

[17] G. Bucci and E. Vicario, "Compositional Validation of Time-Critical Systems Using Communicating Time Petri Nets," *IEEE Trans. Software Eng.,* vol. 21, no. 12, Dec. 1995.

[18] G. Bucci, A. Fedeli, L. Sassoli, and E. Vicario, "Modeling Flexible Real-Time Systems with Preemptive Time Petri Nets," *Proc. 15th Euromicro Conf. Real-Time Systems (ECRTS03),* July 2003.

[19] G. Bucci, L. Sassoli, and E. Vicario, "A Discrete Time Model for Performance Evaluation and Correctness Verification of Real-Time Systems," *Proc. 10th Int'l Workshop Petri Nets and Performance Models (PNPM '03),* Sept. 2003.

[20] G. Bucci, A. Fedeli, L. Sassoli, and E. Vicario, "Timed State Space Analysis of Real-Time Preemptive Systems," *IEEE Trans. Software Eng.,* vol. 30, no. 2, pp. 97-111, Feb. 2004.

[21] G. Bucci, L. Sassoli, and E. Vicario, "Oris: A Tool for State Space Analysis of Real-Time Preemptive Systems," *Proc. First Int'l Conf. the Quantitative Evaluation of Systems (QEST '04),* Sept. 2004.

[22] G. Bucci, R. Piovosi, L. Sassoli, and E. Vicario, "Introducing Probability within State Class Analysis of Dense-Time-Dependent Systems," *Proc. Int'l Conf. the Quantitative Evaluation of Systems (QEST '05),* Sept. 2005.

[23] P. Buchholz and I.V. Tarasyuk, "Net and Algebraic Approaches to Probabilistic Modeling," *Joint Novosibirsk Computing Center and Institute of Informatics Systems Bulletin, Series Computer Science,* pp. 31-64, 2001.

[24] G. Buttazzo, *Hard Real-Time Computing Systems.* Norwell, Mass.: Kluwer, 1997.

[25] M. Caccamo, L. Abeni, G.C. Buttazzo, and G. Lipari, "Elastic Scheduling for Flexible Workload Management," *IEEE Trans. Computers,* 2002.

[26] F. Cassez and K.G. Larsen, "The Impressive Power of Stopwatches," *Proc. 11th Int'l Conf. Concurrency Theory,* pp. 138-152, Aug. 2000.

[27] A. Cervin, "Improved Scheduling of Control Tasks," *Proc. 11th Euromicro Conf. Real-Time Systems,* pp. 4-10, June 1999.

[28] A. Cervin, "Merging Real-Time and Control Theory for Improving the Performance of Embedded Control Systems," research report, Sept. 2004.

[29] A. Cervin, B. Lincoln, J. Eker, K.E. Årzén, and G. Buttazzo, "The Jitter Margin and Its Application in the Design of Real-Time Control Systems," *Proc. 10th Int'l Conf. Real-Time and Embedded Computing Systems and Applications (RTCSA),* Aug. 2004.

[30] G. Chiola, M. AjmoneMarsan, G. Balbo, and G. Conte, "Generalized Stochastic Petri Nets: A Definition at the Net Level and Its Implications," *IEEE Trans. Software Eng.,* Feb. 1993.

[31] G. Ciardo, "Discrete Time Markovian Stochastic Petri Nets," *Proc. Second Workshop Numerical Solution of Markovian Chains,* 1995.

[32] G. Ciardo and R. Zijal, "Well Defined Sthocastic Petri Nets" *Proc. Fourth Int'l Workshop Modeling Analysis and Simulation of Computer and Telecomunication Systems (MASCOTS '96),* 1996.

[33] G. Ciardo, R. Zijal, and G. Hommel, "Discrete Deterministic and Stochastic Petri Nets" *Measurement, Modeling, and Valuation of Computer and Communication-Systems,* pp. 103-117, Sept. 1997.

[34] E.M. Clarke, E.A. Emerson, and A. Sistla, "Automatic Verification of Finite State Concurrent Systems Using Temporal Logic Specifications," *ACM Trans. Programming Language,* vol. 8, no. 2, pp. 244-263, 1986.

[35] C. Courcoubetis and S. Tripakis, "Probabilistic Model Checking: Formalisms and Algorithms for Discrete and Real-Time Systems," *Verification of Digital and Hybrid Systems,* pp. 183-219, 2000.

[36] C. Daws, A. Olivero, S. Tripakis, and S. Yovine, "The Tool KRONOS," *Hybrid Systems III,* 1996.

[37] R. De Nicola and F.W. Van Draager, "Action versus State Based Logics for Transition Systems," *Proc. LITP Spring School on Theoretical Computer Science,* pp. 407-419, 1990.

[38] E. Fersman, P. Pettersson, and W. Yi, "Timed Automata with Asynchronous Processes: Schedulability and Decidability," *Tools and Algorithms for Construction and Analysis of Systems,* 2002.

[39] K. Goseva-Popstojanova and K.S. Trivedi, "Stochastic Modeling Formalisms for Dependability, Performance and Performability," *Performance Evaluation—Origins and Directions,* pp. 385-404, 2000.

[40] D. Harel and A. Pnueli, "On the Development of Reactive Systems," *Logics and Models of Concurrent Systems (NATO),* 1985.

[41] A. Horvath, A. Puliafito, M. Scarpa, and M. Telek, "Analysis and Evaluation of NonMarkovian Stochastic Petri Nets," *Proc. 11th Int'l Conf. Computer Performance Evaluation (TOOLS),* pp. 171-187, 2000.

[42] K. Jay, L. Kevin, and B. Kenny, "Building Flexible Real-Time System Using the Flex Language," *Computer,* 1991.

[43] I. Lee, P. Bremond-Gregoire, and R. Gerber, "A Process Algebraic Approach to the Specification and Analysis of Resource Bound Real-Time Systems," *Proc. IEEE,* vol. 82, no. 1, pp. 158-171, Jan. 1994.

[44] I. Lee, J.-Y. Choi, H.-H. Kwak, A. Philippou, and O. Sokolsky, "A Family of Resource-Bound Real-Time Process Algebras," *Proc. 21st Int'l Conf. Formal Techniques for Networked and Distributed Systems (FORTE '01),* Aug. 2001.

[45] M. Ajmone Marsan, G. Balbo, and G. Conte, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems," *ACM Trans. Computer Systems,* vol. 2, no. 2, pp. 93-122, May 1984.

[46] J. McManis and P. Varaiya, "Suspension Automata: A Decidable Class of Hybrid Automata," *Proc. Computer Aided Verification: Sixth Int'l Conf. (CAV '94),* June 1994.

[47] P. Merlin and D.J. Farber, "Recoverability of Communication Protocols," *IEEE Trans. Comm.,* vol. 24, no. 9 Sept. 1976.

[48] X. Nicollin, J. Sifakis, and S. Yovine, "Compiling Real-Time Specifications into Extended Automata," *IEEE Trans. Software Eng.,* vol. 18, no. 9, Sept. 1992.

[49] D. Lime and O.H. Roux, "Expressiveness and Analysis of Scheduling Extended Time Petri Nets," *Proc. Fifth IFAC Conf. Fieldbus and Their Applications (FET 2003),* July 2003.

[50] O. Sokolsky, I. Lee, and H. Ben-Abdallah, "Specification and Analysis of Real-Time Systems with PARAGON," *Annals of Software Eng.,* vol. 7 1999.

[51] E. Smith, "On the Border of Causality: Contact and Confusion," *Theoretical Computer Science,* vol. 153, pp. 275-270, 1996.

[52] J.A. Stankovic and K. Ramamritham, "What Is Predictability for Real-Time Systems," *J. Real-time Systems,* vol. 2 Dec. 1990.

[53] A.S. Tanenbaum, *Modern Operating Systems.* Prentice Hall, 2001.

[54] E. Teruel, G. Franceschinis, and M. DePierro, "Well Defined Generalized Stochastic Petri Nets: A Net Level Method to Specify Priorities," *IEEE Trans. Software Eng.,* vol. 29, Nov. 2003.

[55] M.Y. Vardi, "Automatic Verification of Probabilistic Concurrent Finite-State Programs," *Proc. Symp. Foundations of Computer Science (FOCS '85),* pp. 327-338, 1985.

[56] E. Vicario, "Static Analysis and Dynamic Steering of Time Dependent Systems Using Time Petri Nets," *IEEE Trans. Software Eng.,* Aug. 2001.

[57] B. Wittenmark and N. Trngren, "Timing Problems in Real-Time Control Systems," *Proc. Am. Control Conf.,* 1995.

[58] R. Zijal, "Discrete Time Deterministic and Stochastic Petri Nets," *Proc. Int'l Workshop Quality of Comm.-Based Systems,* pp. 123-136, 1994.

[59] A. Zimmermann, J. Freiheit, and G. Hommel, "Discrete Time Stochastic Petri Nets for Modeling and Evaluation of Real-Time Systems," *Proc. Int'l Parallel and Distributed Processing Symp.,* 2001.

Giacomo Bucci (M'80) is a graduate of the University of Bologna, Italy. From 1970 to 1982, he was with the University of Bologna. During 1975, he was a visiting researcher at IBM T.J. Watson Research Center, Yorktown Heights, New York. Since 1986, he has been a full professor at the University of Florence, Faculty of Engineering, where he teaches a course in computer architectures and a course in software engineering. Currently, he is the director of the Department of Systems and Informatics and the director of the Software Technologies Lab of University of Florence. His current research interests include software development methodologies, specification and validation techniques for time-dependent systems, and computer performance evaluation. He is a member of the IEEE and the IEEE Computer Society.

Luigi Sassoli received the doctoral degree in informatics engineering in 2002 from the University of Florence where he is now completing the PhD degree in informatics, multimedia, and telecommunications engineering. His research interests are mainly focused on state space analysis methods for performance evaluation and correctness verification of real-time reactive systems, with specific interest on the integration of stochastic characterization into discrete and dense time models. He is a member of the Software Technologies Lab of the University of Florence.

Enrico Vicario (M'95) received a doctoral degree in electronics engineering and the PhD degree in informatics and telecommunications engineering received from the University of Florence in 1990, and 1994, respectively. Since 2002 he has been a full professor in the faculty of engineering of the University of Florence. His present research activity is mainly focused on correctness verification and quantitative evaluation of dependability and performance in reactive and real-time systems. He also worked on content modeling and retrieval for image and video, visual formalisms, and multimedia applications. He is a member of the steering committee of the Center for Communication and Media Integration of the University of Florence, and a member of the Software Technologies Lab of the University of Florence. His technology transfer activity mainly focuses on software engineering with specific application in software architecture and design. He is a member of the IEEE and the IEEE Computer Society.