Non-Markovian analysis for model-driven engineering of real-time software

Laura Carnevali, Marco Paolieri, Alessandro Santoni, Enrico Vicario

Dipartimento di Ingegneria dell'Informazione – Università di Firenze
Via di Santa Marta, 3 – 50139 Firenze – Italy
{laura.carnevali, marco.paolieri, alessandro.santoni, enrico.vicario}@unifi.it

ABSTRACT

Quantitative evaluation of models with stochastic timings can decisively support schedulability analysis and performance engineering of real-time concurrent systems. These tasks require modeling formalisms and solution techniques that can encompass stochastic temporal parameters firmly constrained within a bounded support, thus breaking the limits of Markovian approaches. The problem is further exacerbated by the need to represent suspension of timers, which results from common patterns of real-time programming. This poses relevant challenges both in the theoretical development of non-Markovian solution techniques and in their practical integration within a viable tailoring of industrial processes.

We address both issues by extending a method for transient analysis of non-Markovian models to encompass suspension of timers. The solution technique addresses models that include timers with bounded and deterministic support, which are essential to represent synchronous task releases, timeouts, offsets, jitters, and computations constrained by a Best Case Execution Time (BCET) and a Worst Case Execution Time (WCET). As a notable trait, the theory of analysis is amenable to the integration within a Model Driven Development (MDD) approach, providing specific evaluation capabilities in support of performance engineering without disrupting the flow of design and documentation of the consolidated practice.

Keywords

Software performance engineering, non-Markovian stochastic analysis, model driven development, real-time systems.

1. INTRODUCTION

Model Driven Development (MDD) provides a way to incorporate formal methods in the development process of safety-critical real-time systems, so as to support formal verification of design correctness, automatic compilation of specifications into code skeletons and other artifacts, unit and integration testing [40, 32]. This practice is explicitly encouraged within various regulatory standards whenever the system exhibits complex behaviors mixing concurrency and timing [36, 26].

Various approaches and tools have been developed to support the aim for concurrent systems with non-deterministic temporal parameters. Notable examples include Uppaal [4] with the extensions TIGA [20] and TRON [29], based on the theory of timed automata [1]; Giotto [28], a tool-supported programming methodology for the validation and synthesis of control software; Charon [2], a language for modular specification of interacting hybrid systems, relying on the notions of agent and mode; Simulink [35], a block diagram environment for simulation and model-based design of complex control systems.

The Oris tool [10] supports MDD by leveraging the theory of preemptive Time Petri Nets (pTPNs) [11], an extension of Time Petri Nets (TPNs) [43] that encompasses priority-driven preemptive scheduling with possible suspension of computations, attaining an expressive power that compares with stopwatch automata [21] and Petri nets with hyper-arcs [37]. The MDD approach of [18] integrates the theory of pTPNs and the Oris tool within a tailoring of the V-Model life cycle [14], supporting formal verification of SW design, generation of real-time code, measurement-based profiling of temporal parameters, and testing. As a relevant trait, the designer is allowed to create a model through a semi-formal specification, which captures design choices of the periods and the concurrency structure of tasks, as well as design assumptions on the duration of computations, initially guessed by analogy with previous realizations. During development iterations, the results of formal analysis and execution time profiling are used to refine the specification and implementation until SW requirements are satisfied. The formal methodology of [18] is extended in [25] to enable a smooth integration within an industrial process of SW development, supporting the documentation process prescribed by Military Standard 498 (MIL-STD-498) [42] while avoiding to disrupt consolidated industrial practice.

With the evolution of certification standards for safety-critical software towards RAMS (Reliability, Availability, Maintainability, Safety) practices, safety requirements excluding the *possibility* of anomalous behaviors are often recast into quantitative values of guaranteed *probability* [26, 27]. This motivates the extension of MDD approaches so as to incorporate quantitative evaluation of RAM indices into the formal verification process. For this purpose, modeling and solution techniques should encompass inherent characteristics of real-time software, such as multiple concurrent temporal parameters characterized by general (GEN) distributions over bounded or deterministic supports. Computations usually have execution times firmly constrained within finite intervals; task periods, offsets, and timeouts are necessarily deterministic temporal parameters; jitters and inter-release times of asynchronous tasks are often lower-bounded. At the same time, the analysis techniques must be able to deal with suspension in the advancement of timers, as

ICPE'13, April 21–24, 2013, Prague, Czech Republic.
doi:10.1145/2479871.2479889

an effect of preemptive scheduling, which corresponds to the *Preemptive Resume* (PRs) policy, as opposed to cooperative scheduling represented by the *Preemptive Repeat Different* (PRD) policy [7].

In this paper, we present an approximated solution technique for the transient analysis of non-Markovian preemptive models that include deterministic and bounded timers. Transient analysis based on stochastic state classes [30] is extended from a PRD policy (where all enabled timers advance with the same speed and disabled timers are re-sampled at newly-enabling) to a PRs policy (where some enabled timers may be suspended and later resumed). The addition of suspension has a major impact on the analysis technique with respect to the PRD policy: in this case, the geometric shape of the joint support of enabled timers is no more in the form of a Difference Bounds Matrix zone (DBM zone), but becomes a linear convex polyhedron, leading to exponential complexity in the derivation and encoding of the state space. The issue is addresses in [11] for non-deterministic models by resorting to an overapproximation of polyhedral supports with DBM zones. For stochastic models, the problem is exacerbated by the fact that the joint probability density function (PDF) of enabled timers has a piecewise form over a partition of the polyhedral support into polyhedral subdomains. The computational complexity of such partitioning excludes the generalization of approaches based on DBM zone partitioning [44], and requires a solution technique based on the approximation of both supports and distributions. The proposed approach allows to represent deterministic timers that are never suspended, which allow to represent synchronous task releases and timeouts, as resulting from common patterns of real-time programming. In contrast, suspension may occur on stochastic timers, which represent computation times associated with non-pointlike support, both to allow a margin of laxity in the implementation and to be robust with respect to possible changes in the embedding environment.

The rest of the paper is organized as follows: in Sect. 2, we discuss related work; in Sect. 3, we present a novel solution technique for the transient analysis of non-Markovian preemptive models including temporal parameters with bounded and deterministic supports; in Sect. 4, we present experimental results; in Sect. 5, we draw conclusions.

2. RELATED WORK

Quantitative evaluation of real-time systems faces the analysis of models that encompass both stochastic temporal parameters with bounded support and priority-driven preemptive scheduling with suspension and resumption of timers. This mix of requirements is partially fulfilled by a few discrete time approaches and basically unsolved in the continuous time perspective.

The numerical approach presented in [31] for the analysis of Non-Markovian Stochastic Petri Nets (NMSPNs) relies on a time-discretization algorithm that approximates the firing times of transitions through Discrete Phase-Type (DPH) random variables, mapping the evolution of the marking process into an expanded Discrete Time Markov Chain (DTMC). The technique allows any kind of general distribution, combining together the PRs policy and the PRD policy [7]. The inherent state-space explosion problem is alleviated in [39] by using a Kronecker algebra, which avoids an explicit enumeration of timed states while requiring a finite number of reachable markings. A similar approach based on DPHs is proposed in [45] for the model of Discrete Deterministic and Stochastic Petri Nets (DDSPNs), which encompass race policies equivalent to PRs and PRD policies. In [12], a discrete-time variant of time Petri nets [43] supports the representation of preemptive behavior and associates a stochastic characterization with all non-deterministic logical and temporal choices. The analysis technique relies on a maximal step semantics which conveniently separates the effects of logical concurrency from those of non-Markovian timed behavior, circumventing the problems of confusion and non-well-definedness arising in discrete-time models with an interleaved semantics of concurrency.

As a common trait, discretization approaches suffer a trade-off between the accuracy attained by the analysis and the computational complexity, which largely depends on the time scale used to cover the ranges of variation of temporal parameters. The problem is further exacerbated in those methods that approximate the distribution of timers with finite support through DPHs, which may produce unexpected results when the finiteness of behavior actually depends on the sequencing constraints induced by timing.

Managing non-Markovian and preemptive behavior is a much harder challenge in continuous time, where the limits of the analysis reduce the expressive capabilities of the model. In [23, 8], GEN timers are approximated by means of Continuous Phase-Type (CPH) distributions, following an approach that can be regarded as a continuous-time version of the phase-type expansion technique of [31]. The symbolic technique of [33] supports efficient computation and encoding of the reachability graph of the untimed model, although the dimension of the state space still comprises a limitation in the step of solution and the obtained approximants are not able to firmly preserve bounded supports. An analytical approach based on a continuous abstraction of time was proposed in [9] for Deterministic and Stochastic Petri Nets (DSPNs), which include exponential and deterministic timers, and extended in [7] to manage a combined use of different preemption polices. The approach leverages Markov Renewal Theory under the so-called enabling restriction, which rules out concurrent enabling of multiple GEN transitions. As a major limit, this structurally clashes with the need to represent sequencing restrictions due to mutual timing.

The case of multiple concurrent GEN timers with possibly overlapping activity cycles is managed by the method of stochastic state classes. Steady state analysis can be performed exactly in the case that every cyclic behavior that changes the enabling status of GEN transitions visits at least one regeneration point [44]. Transient analysis [30] can be performed exactly provided that the model does not include cycles that must be completed in zero time (no Zeno behaviors), and it can be always guaranteed to be finite in probability under the assumption of an approximation threshold $\epsilon > 0$ on the total error. However, the solution techniques of [44, 30] address models that do not encompass suspension and resumption of timers. In [15], the method of [44] is extended to support steady-state analysis of systems that run under static-priority preemptive scheduling. Nevertheless, this still maintains two major limits in the perspective of application to the development of real-time systems. In fact, the technique is developed under the restriction that all timers have non-pointlike support, which precludes deterministic timers and thus prevents the representation of structural patterns of real-time programming such as timeouts and periodic task releases. In addition, the approach does not support transient analysis, which instead turns out to take major relevance in the verification of most real-time requirements.

3. MODELING AND EVALUATION OF STOCHASTIC REAL-TIME SYSTEMS

We first introduce the model of stochastic preemptive Time Petri Nets, which aims at providing sufficient expressive capabilities for the representation of realistic patterns of real-time programming (Sect. 3.1). We then present a novel solution technique that extends the method of [30] to encompass transient analysis of non-Markovian preemptive models (Sect. 3.2).

3.1 Modeling task-sets through spTPN

A stochastic preemptive Time Petri Net represents a discrete-event system whose state is made of a logic condition called *marking* and a set of continuous timers called *times-to-fire*, each associated with a transition enabled by the marking. Each enabled transition may request one or more resources according to a static priority preemptive scheduling: it is called *progressing* or *suspended* depending on whether it is assigned each of its requested resources or not, respectively. The progressing transition with minimum time-to-fire is selected as the next event and its execution leads to a new state with different marking and times-to-fire. After the marking update, transitions can persist or be newly-enabled, disabled, reset. Times-to-fire of newly-enabled and reset transitions are sampled according to static distributions; times-to-fire of persistent transitions are decreased by that of the fired transition if they are progressing, and maintained if they are suspended; times-to-fire of disabled transitions are discarded.

3.1.1 Syntax

A stochastic preemptive Time Petri Net (spTPN) is a tuple $\langle P; T; A^-; A^+; A^{\cdot}; m_0; EFT^s; LFT^s; Res; Req; Prio; \mathcal{F}; \mathcal{C}\rangle$.

The first eight elements comprise the model of *time Petri nets* [43]. P is a set of places. T is a set of transitions disjoint from P. $A^- \subseteq P \times T$, $A^+ \subseteq T \times P$, and $A^{\cdot} \subseteq P \times T$ are sets of precondition, postcondition, and inhibitor arcs. $m_0 : P \to \mathbb{N}$ is the initial marking associating each place with an initial non-negative number of tokens. $EFT^s : T \to \mathbb{R}_0^+$ and $LFT^s : T \to \mathbb{R}_0^+ \times (\mathbb{R}_0^+ \cup \{\infty\})$ associate each transition with a *static Earliest Firing Time* and a (possibly infinite) static *Latest Firing Time* ($EFT^s(t) \leq LFT^s(t)\ \forall\, t \in T$). As usual in Petri Nets, a place p is said to be an *input*, an *output*, or an *inhibitor* place for a transition t if $\langle p, t\rangle \in A^-$, $\langle t, p\rangle \in A^+$, or $\langle p, t\rangle \in A^{\cdot}$, respectively.

Res, Req, and $Prio$ comprise a mechanism of resource assignment in the style of *preemptive Time Petri Nets* (pTPNs) [11], which makes the progress of timed transitions dependent on the availability of a set of preemptable resources. Res is a set of preemptable resources disjoint from P and T. $Req : T \to 2^{Res}$ and $Prio : T \to \mathbb{N}$ associate each transition with a subset of Res representing its resource request and with a static priority level, respectively. We assume that low priority numbers correspond to high priority levels.

As in *stochastic Time Petri Nets* (sTPNs) [44, 16], $\mathcal{F}$ and $\mathcal{C}$ define a measure of probability for non-deterministic choices. Specifically, $\mathcal{C} : T \to \mathbb{R}^+$ associates each transition with a positive weight and $\mathcal{F} : T \to F_t^s$ associates each transition with a static Cumulative Distribution Function (CDF) supported over its static firing interval $[EFT^s(t), LFT^s(t)]$. As usual in Stochastic Petri Nets, a transition t is called *immediate* (IMM) if $EFT^s(t) = LFT^s(t) = 0$ and *timed* otherwise. A timed transition t is called *exponential* (EXP) if $F_t^s(x) = 1 - e^{\lambda x}$ over $[0, \infty]$ for some rate $\lambda \in \mathbb{R}_0^+$ and *general* (GEN) otherwise. In particular, a GEN transition t with $EFT^s(t) = LFT^s(t) > 0$ is called *deterministic* (DET). Moreover, for each GEN transition t with $EFT^s(t) \neq LFT^s(t)$, which we call *distributed* transition, we assume that F_t^s is absolutely continuous and, thus, that there exists a Probability Density Function (PDF) f_t^s such that:

$$F_t^s(x) = \int_0^x f_t^s(y)dy.$$

3.1.2 Semantics

The *state* of an spTPN is a pair $\langle m, \tau\rangle$, where $m : P \to \mathbb{N}$ is a marking that associates each place with a non-negative number of tokens and $\tau : T^e(m) \to \mathbb{R}_0^+$ associates each transition enabled by m with a (dynamic) real-valued time-to-fire, where a transition t is said to be *enabled* by m if each of its input places contains at least one token and none of its inhibitor places contains any token (i.e., $m(p) \geq 1\ \forall\ \langle p, t\rangle \in A^-$ and $m(p) = 0\ \forall\ \langle p, t\rangle \in A^{\cdot}$).

An enabled transition t is *progressing* if every resource it requires is not required by any other enabled transition with a higher level of priority (i.e., $Prio(t) \leq Prio(t')\ \forall\, t' \in T^e(M)$ such that $Req(t) \cap Req(t') \neq \emptyset$); otherwise, it is *suspended.*

A progressing transition t is *firable* in state s if its time-to-fire is not higher than that of any other transition that is progressing in s (i.e., $\tau(t) \leq \tau(t')\ \forall\ t' \in T_p(s)$, where $T_p(s)$ is the set of transitions that are progressing in s). When multiple transitions are firable, the choice is resolved by the random switch determined by $\mathcal{C}$: $Prob\{t \text{ is selected}\} = \mathcal{C}(t)/\sum_{t_i \in T_f} \mathcal{C}(t_i)$.

When a transition t fires, the state $\langle m, \tau\rangle$ is replaced by a new state $s' = \langle m', \tau'\rangle$, which we write as $s \xrightarrow{t} s'$. Marking m' is derived from m by removing a token from each input place of t (i.e., if $\langle p, t\rangle \in A^-$ then $m_{tmp}(p) = m(p) - 1$, else $m_{tmp}(p) = m(p)$) and by adding a token to each output place of t (i.e., if $\langle t, p\rangle \in A^+$ then $m'(p) = m_{tmp}(p) + 1$, else $m'(p) = m_{tmp}(p)$). Transitions that are enabled both by the intermediate marking m_{tmp} and by m' are said *persistent*, while those that are enabled by m' but not by m_{tmp} or m are said *newly-enabled.* If t is still enabled after its own firing, it is always regarded as newly enabled [5, 43].

For any transition t_n newly-enabled after the firing of t, the time-to-fire takes a random value sampled in the static firing interval according to the static probability distribution $F_{t_n}^s$, i.e., $EFT^s(t_n) \leq \tau'(t_n) \leq LFT^s(t_n)$ and $Prob\{\tau'(t_n) \leq x\} = F_{t_n}^s(x)$. For any transition t_p that was progressing in the previous state and is persistent after the firing of t, the time-to-fire is reduced by the time elapsed in the previous state (which is equal to the time-to-fire of t measured at the entrance in the previous state), i.e., $\tau'(t_p) = \tau(t_p) - \tau(t)$. For any transition t_s that was suspended in the previous state and is persistent after the firing of t, the time-to-fire remains unchanged, i.e., $\tau'(t_s) = \tau(t_s)$.

3.1.3 An illustrative example

Fig. 1 represents a periodic real-time task with a semaphore synchronization within a wider (not shown) task-set. A transition with no input places nor resource requests models a task release, and fires repeatedly with inter-firing times falling within its firing interval. According to this, transition t_{10} models periodic task releases with period of 50 time units. A time-consuming operation performed on one or more processors is represented by a transition associated with a resource request, a static priority, and a firing interval corresponding to its range of execution time. For instance, transitions t_{12} and t_{13} model two computations performed on resource *cpu* with priority level 1 in a time ranging within $[5, 10]$ and $[2, 4]$ time units, respectively. A binary semaphore is represented as a place initially marked with one token, e.g., place $mutex$. While a *wait* operation is modeled as a transition having the semaphore place as an input place, a *release* operation is usually represented by a transition that has the semaphore place as an output place and that also accounts for the completion of a synchronized computation. According to this, transition t_{11} represents a wait operation on semaphore $mutex$, while transition t_{13} models a release operation on semaphore $mutex$ as well as the completion of the second computation of the task.

Some temporal parameters of a real-time task-set attain a non-deterministic range of variation (e.g., the execution time of computations and the inter-release time of asynchronous tasks), and they are modeled as transitions that, in principle, can be associated with any kind of distribution (e.g., t_{11}, t_{12}, and t_{13}). Other temporal parameters are intrinsically deterministic (e.g., timeouts and the inter-release time of synchronous tasks), and thus their are modeled as IMM or DET transitions, which are associated with a Dirac delta function (e.g., t_{10} is associated with a Dirac delta function $f^s_{t_{10}}(y) = \delta(y - 50)$, centered at $y = 50$ time units).

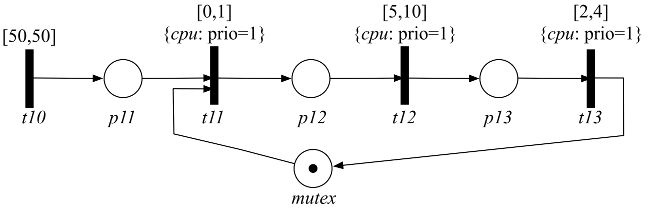

Figure 1: A fragment of the spTPN model of a task-set, representing a periodic real-time task.

In general, the construction of the model of a real-time task-set can be conveniently cast into a minor extension of the development process devised in [18, 25], which was proven viable in the industrial practice [25] but did not include any probabilistic quantification. To this end, during the early stages of SW Design, the distributions associated with the execution times of computations and with the inter-release times of asynchronous tasks are guessed by analogy with previous or similar implementations. Next, during the iterations of the development process, they are progressively refined according to the histograms of temporal parameters observed during the execution of the real-time code, which are obtained as a by-product of the approach implemented in [18, 25]. In doing so, the additional effort demanded to the designer to manage probabilities consists in finding GEN distributions in the class of expolynomial functions that fit the shape and support of the measured histograms.

Note that the spTPN semantics combines the PRD and PRs policies of [7, 9], also known as *enabling memory* policy and *age* policy [22], respectively. In the formulation of [7, 9], the clock of a transition disabled by the lack of a token is reset or maintained depending on whether the transition is associated with PRD or PRs policy, respectively. Conversely, in our treatment, the progress of times-to-fire depends both on the presence of tokens into input places and on the availability of preemptable resources. Specifically, when a transition is disabled by the lack of a token in some of its input places, its time-to-fire is reset, following the PRD policy; otherwise, when a transition is suspended by the lack of some of its required resources, its time-to-fire is maintained and resumed when the transition is assigned the resource again, following the PRs policy. This enables separate modeling of inter-task communication mechanisms from real-time concurrency on resources, facilitating the representation of usual patterns of real-time programming [13]. As shown by the example of Fig. 1, while the presence of tokens into places models logic conditions such as the availability of a semaphore, preemptable resources of the net are used to account for processors that are necessary to perform a computation.

3.2 Transient analysis of spTPN

We recall here the salient aspects of the method of transient stochastic state classes (Sect. 3.2.1), for which we also refer the reader to [30]. We then develop a novel and substantial extension of the approach of [30] that encompasses suspension and resumption of timers by relying on a continuous-time approximation in the state-space (Sects. 3.2.3, 3.2.2, and 3.2.4). We finally discuss how the complexities arising from the interaction of suspension with DET transitions can be managed (Sect. 3.2.5).

3.2.1 Transient classes

The state of an spTPN evolves depending on the times-to-fire sampled by transitions and the resolution of random switches according to transitions weights. Within a given time bound, the method of transient stochastic state classes [30] characterizes the state of an spTPN after each transition firing and the absolute time of the firing. Specifically, a transient stochastic state class provides the joint PDF for the vector of times-to-fire of enabled transitions augmented with a timer, called τ_{age}, which accounts for the time elapsed since when the class was entered. For convenience, τ_{age} encodes the opposite of the elapsed time, so that it evolves with the same slope as that of times-to-fire of progressing transitions and it can be treated as an always-persistent-progressing time-to-fire.

DEFINITION 3.1. *A transient stochastic state class (transient class for short) is a tuple $\langle m, New, D, f_{\langle\tau_{age},\underline{\tau}\rangle}\rangle$, where: $m : P \to \mathbb{N}$ is a marking; $\langle\tau_{age}, \underline{\tau}\rangle$ is a random variable made of the scalar variable τ_{age} and the vector $\underline{\tau} = \langle\tau_0, \tau_1, ..., \tau_{N-1}\rangle$ of times-to-fire of transitions enabled by m; New is the set of transitions that are newly enabled when the class is entered; and, $f_{\langle\tau_{age},\underline{\tau}\rangle}$ is the joint PDF of the variable in $\langle\tau_{age}, \underline{\tau}\rangle$ supported over D.*

A reachability relation is defined among transient classes:

DEFINITION 3.2. *We say that $\Sigma' = \langle m', D', New', f_{\langle\tau'_{age},\underline{\tau}'\rangle}\rangle$ is a successor of $\Sigma = \langle m, New, D, f_{\langle\tau_{age},\underline{\tau}\rangle}\rangle$ through the firing of t_0 with probability μ_0, which we write $\Sigma \overset{t_0,\mu_0}{\Longrightarrow} \Sigma'$, iff, given that the marking of the net is m and the time vector $\langle\tau_{age}, \underline{\tau}\rangle$ is a random variable with support D and PDF $f_{\langle\tau_{age},\underline{\tau}\rangle}$, then the firing of t_0 occurs with probability μ_0 and leads to a new marking m' and a new time vector $\langle\tau'_{age}, \underline{\tau}'\rangle$ with support D', PDF $f_{\langle\tau'_{age},\underline{\tau}'\rangle}$, and the set New' of newly-enabled transitions.*

In the initial transient class $\Sigma_0 = \langle m_0, D, f_{\langle\tau_{age},\underline{\tau}\rangle}\rangle$, τ_{age} is equal to 0 and all enabled transitions are newly-enabled with times-to-fire independently distributed according to their PDFs supported over their firing intervals. According to this:

$$D = [0, 0] \times \prod_{t_i \in T_e(m_0)} [EFT^s(t_i), LFT^s(t_i)],$$

$$f_{\langle\tau_{age},\underline{\tau}\rangle}(x_{age}, \underline{x}) = \delta(x_{age}) \cdot \prod_{t_i \in T_e(m)} f_{t_i}(x_i),$$

where $\delta(x_{age})$ is the Dirac delta function centered at $x_{age} = 0$. Given an initial transient class Σ_0, the transitive closure of the reachability relation among transient classes enumerates a *transient stochastic tree*, where vertexes are transient classes and edges are labeled with a transition t and a probability μ. This enables the derivation of continuous-time transient probabilities of reachable markings within any given time bound [30].

3.2.2 Transient classes in partitioned form

When the spTPN model includes IMM and DET transitions, a class $\Sigma = \langle m, New, D, f_{\langle\tau_{age},\underline{\tau}\rangle}\rangle$ may have a domain D with null measure in $\mathbb{R}^{N+1}$, where N is the cardinality of the set $T^e(m)$ of transitions enabled by marking m [44]. To avoid a stochastic characterization of deterministic and deterministically dependent timers, we resort to the $\langle Q, U, V\rangle$ partition of [44], where: Q is the set (of maximal cardinality) of *distributed* variables, i.e., variables that have a non-deterministic value and mutual delay (i.e., $\tau_i \in Q$ iff $b_{i*} + b_{*i} \neq 0 \;\wedge\; b_{ij} + b_{ji} \neq 0 \;\; \forall \tau_j \in Q$ with $i \neq j$); U is the set of *deterministic* variables, i.e., variables that have a deterministic value (i.e., $\tau_h \in U$ iff $b_{h*} + b_{*h} = 0$); and, V is the set of *synchronized* variables, i.e., variables that have a deterministic delay with respect to some distributed variable (i.e., $\tau_k \in V$ iff $b_{k*} + b_{*k} \neq 0 \wedge \exists\, \tau_{\iota_k} \in Q$ such that $b_{k\iota_k} + b_{\iota_k k} = 0$).

According to this, the normal form of D can be rewritten in a *partitioned form* that combines the set D_Q of constraints limiting the variables of Q with a set of equalities expressing all the remaining variables of U and V:

$$D_Q = \begin{cases} \tau_i - \tau_j \leq b_{ij} \\ \tau_i - \tau_* \leq b_{i*} \\ \tau_* - \tau_i \leq b_{*i} \\ \forall\, \tau_i, \tau_j \in Q \end{cases} \quad D = \begin{cases} D_Q \\ \tau_h = b_{h*} + \tau_* \\ \tau_k = b_{k\iota_k} + \tau_{\iota_k} \\ \forall\, \tau_h \in U,\ \forall\, \tau_k \in V \end{cases} \tag{1}$$

When domain D is represented in partitioned form, the joint PDF of variables in $Q \cup U \cup V$ is completely defined by the joint PDF of variables in Q. According to this, $f_{\langle \tau_{age}, \underline{\tau} \rangle}$ is completely defined by the joint PDF $f_{\langle \tau_{age}, \underline{\tau}_Q \rangle}$ of τ_{age} and $\underline{\tau}_Q$. According to this, a transient class $\langle m, New, D, f_{\langle \tau_{age}, \underline{\tau} \rangle} \rangle$ will be represented as $\langle m, New, D, f_{\langle \tau_{age}, \underline{\tau}_Q \rangle} \rangle$.

3.2.3 Enumeration of transient classes

In the derivation of successor classes, we use the following notational conventions: *i)* given a vector $\underline{x} = \langle x_0, \ldots, x_{N-1} \rangle \in \mathbb{R}^N$ and a scalar $\delta \in \mathbb{R}$, $\underline{x} + \delta$ denotes the vector $\langle x_0 + \delta, \ldots, x_{N-1} + \delta \rangle$; *ii)* if $D \subseteq \mathbb{R}^N$ and $n \in \{0, \ldots, N-1\}$, then $D \downarrow_{\tau_n}$ denotes the projection of D that eliminates τ_n: $D \downarrow_{\tau_n} = \{ \langle x_0, \ldots, x_{n-1}, x_{n+1}, \ldots, x_{N-1} \rangle \in \mathbb{R}^{N-1} \mid \exists\, x_n \in \mathbb{R} \,.\, \langle x_0, \ldots, x_{n-1}, x_n, x_{n+1}, \ldots, x_{N-1} \rangle \in D \}$; *iii)* $D \downarrow_{A_1, \ldots, A_N}$ denotes the projection of D that eliminates all variables in $A_1 \cup \ldots \cup A_N$; *iv)* given a transient class $\Sigma = \langle m, New, D, f_{\langle \tau_{age}, \underline{\tau} \rangle} \rangle$, the number of enabled transitions (i.e., the number of times-to-fire) is denoted by M; the number of distributed times-to-fire is denoted by N and their vector is denoted by $\underline{\tau}_Q = \langle \tau_0, \tau_1, \ldots, \tau_{N-1} \rangle$; the time vector is represented as $\langle \tau_{age}, \underline{\tau} \rangle = \langle \tau_{age}, \tau_0, \tau_1, \ldots, \tau_{N-1}, \tau_N, \ldots, \tau_{M-1} \rangle$ so as to encode τ_{age} and distributed times-to-fire in the first $N+1$ positions.

Successors detection and calculus of their probability. According to the spTPN semantics, the firing of a transition t_0 is a possible outgoing event from a transient class $\Sigma = \langle m, New, D, f_{\langle \tau_{age}, \underline{\tau} \rangle} \rangle$ iff t_0 is progressing in Σ and has a time-to-fire not higher than that of any other progressing transition. This occurs iff the restricted domain $D^0 = D \cap \{ \tau_0 \leq \tau_p \; \forall\, t_p \in T^{progr}(\Sigma) \}$ is not empty, where $T^{progr}(\Sigma)$ denotes the set of transitions that are progressing in Σ. The probability that t_0 fires is derived in a different manner by distinguishing the following cases:

- If all progressing transitions have deterministic time-to-fire, then t_0 is selected in the random switch with progressing transitions t_p having the same time-to-fire as that of t_0 (i.e., $t_p \in T^{progr}(\Sigma)$ such that $b_{p0} = b_{0p} = 0$):

$$\mu_0 = \frac{\mathcal{C}(t_0)}{\mathcal{C}(t_0) + \displaystyle\sum_{\substack{t_n \neq t_0 \in T^{progr}(\Sigma) \\ \text{with } b_{p0} = b_{0p} = 0}} \mathcal{C}(t_p)}. \tag{2}$$

- If at least a progressing transition has distributed time-to-fire and the projection of D^0 that eliminates all variables in U and V has null measure (i.e., $||D^0 \downarrow_{U,V}\ || = 0$), then $\mu_0 = 0$. This occurs iff a time-to-fire exists that was distributed before the conditioning that yields D^0 and becomes synchronized or deterministic after the conditioning.

- If at least a progressing transition has distributed time-to-fire and the projection of D^0 that eliminates all variables in U and V has non-null measure (i.e., $||D^0 \downarrow_{U,V}\ || \neq 0$), then μ_0 is the joint probability that the vector of times-to-fire $\underline{\tau}$ takes a value such that τ_0 is not lower than that of any other progressing transition and t_0 is selected in the random switch among all of the progressing transition that have the same time-to-fire as that of t_0:

$$\mu_0 = \frac{\mathcal{C}(t_0)}{\mathcal{C}(t_0) + \displaystyle\sum_{\substack{t_n \neq t_0 \in T^{progr}(\Sigma) \\ \text{with } b_{p0} = b_{0p} = 0}} \mathcal{C}(t_p)} \cdot \int_{D^0 \downarrow_{U,V}} f_{\langle \tau_{age}, \underline{\tau}_Q \rangle}(x_{age}, \underline{x}_Q)\, dx_{age}\, d\underline{x}_Q. \tag{3}$$

If $\mu_0 > 0$, the marking m' and the set of newly-enabled transitions are derived according to the spTPN semantics. The support D' of the time vector $\langle \tau'_{age}, \underline{\tau}' \rangle$ after the firing of t_0 and the PDF $f_{\langle \tau'_{age}, \underline{\tau}'_Q \rangle}$ are derived according to the following steps, regarding τ_{age} as a persistent time-to-fire.

Conditioning. The assumption that t_0 is the next transition to fire conditions $\langle \tau_{age}, \underline{\tau} \rangle$ and yields a new random variable $\langle \tau^a_{age}, \underline{\tau}^a \rangle = \langle \tau_{age}, \underline{\tau} \mid \tau_0 \leq \tau_p \; \forall\, t_p \in T^{progr}(\Sigma) \rangle$ ranging within D^0. Thus, $\langle \tau^a_{age}, \underline{\tau}^a_Q \rangle$ is distributed over $D^0 \downarrow_{U,V}$ according to:

$$f_{\langle \tau^a_{age}, \underline{\tau}^a_Q \rangle}(x_{age}, \underline{x}) = \frac{f_{\langle \tau_{age}, \underline{\tau}_Q \rangle}(x_{age}, \underline{x})}{\mu_0}, \tag{4}$$

where $\underline{x} = \langle x_0, x_1, \ldots, x_{N-1} \rangle$. The assumption that $\mu_0 > 0$ guarantees that the conditioning leaves the $\langle Q, U, V \rangle$ partition unchanged so that $\langle Q^a, U^a, V^a \rangle = \langle Q, U, V \rangle$.

Time advancement. At the firing of t_0, the age and times-to-fire of progressing transitions are reduced by the time-to-fire value of t_0 given by τ^a_0, while those of suspended transitions remain unchanged. This yields a new random variable $\langle \tau^b_{age}, \underline{\tau}^b \rangle = \langle \tau^a_{age} - \tau^a_0, \tau^a_1 - \tau^a_0, \ldots, \tau^a_{P-1} - \tau^a_0, \tau^a_P, \ldots, \tau^a_{N-1}, \ldots \tau^a_{M-1} \rangle$, where: $\underline{\tau}_P = \langle \tau^a_1, \ldots, \tau^a_{P-1} \rangle$ is the vector of distributed times-to-fire that are progressing; $\underline{\tau}_S = \langle \tau^a_P, \ldots, \tau^a_{N-1} \rangle$ is the vector of distributed times-to-fire that are suspended; and, τ^a_N, ..., τ^a_{M-1} are times-to-fire that are either deterministic or synchronized. As synchronized times-to-fire result from multiple progressing deterministic timers that persist at the firing of a distributed timer, each of them is always progressing and synchronized with a distributed time-to-fire.

This step is then developed in a different manner depending on whether τ^a_0 is deterministic, distributed, or synchronized with a distributed variable.

- If τ^a_0 is *deterministic*, the $\langle Q, U, V \rangle$ partition does not change. The support D^b is derived from D^a through a constant shift of progressing timers and $f_{\langle \tau^b_{age}, \underline{\tau}^b_Q \rangle}$ is expressed as:

$$\begin{aligned} & f_{\langle \tau^b_{age}, \underline{\tau}^b_Q \rangle}(x_{age}, \underline{x}) \\ & = f_{\langle \tau^a_{age}, \underline{\tau}^a_Q \rangle}(x_{age} + x_0, x_0, \underline{x}_P + x_0, \underline{x}_S)\, dx_0, \end{aligned} \tag{5}$$

  where $\underline{x} = \langle x_1, x_2, \ldots, x_{N-1} \rangle$, $\underline{x}_P = \langle x_1, x_2, \ldots, x_{P-1} \rangle$, and $\underline{x}_S = \langle x_P, x_{P+1}, \ldots, x_{N-1} \rangle$.

- If τ^a_0 is *distributed*, the $\langle Q, U, V \rangle$ partition may change. Specifically, distributed times-to-fire remain distributed; progressing deterministic times-to-fire become distributed but remain synchronized among themselves, thus one is added to Q^b while the other ones are added to V^b; suspended deterministic times-to-fire remain deterministic; progressing times-to-fire that are synchronized with τ_0^a become deterministic; progressing times-to-fire that are synchronized with a distributed time-to-fire $\tau_h^a \neq \tau_0^a$ remain synchronized with it.

The treatment proceeds in a different manner depending on whether *i)* all distributed timers are progressing, *ii)* all distributed timers are suspended, or *iii)* some distributed timer is progressing and some is suspended. In all these cases, the derivation also distinguishes whether some time-to-fire is deterministic or not.

i) If all distributed timers are progressing and no time-to-fire is deterministic, then $f_{\langle \tau_{age}^b, \underline{\tau}_Q^b \rangle}$ is obtained by shifting all the components by the value of τ_0^a and by eliminating it through a projection:

$$
\begin{aligned}
& f_{\langle \tau_{age}^b, \underline{\tau}^b \rangle}(x_{age}, \underline{x}) \\
& = \int_{Min^0(x_{age}, \underline{x})}^{Max^0(x_{age}, \underline{x})} f_{\langle \tau_{age}^a, \underline{\tau}^a \rangle}(x_{age} + x_0, x_0, \underline{x}_P + x_0) dx_0
\end{aligned}
\tag{6}
$$

with $\underline{x} = \langle x_1, ..., x_{P-1} \rangle$ and $[Min^0(x_{age}, \underline{x}), Max^0(x_{age}, \underline{x})]$ being the support of all possible values of τ_0^a. As in [16, 15], this may partition $D^b \downarrow_{U,V}$ into a finite set of DBM subdomains over which $f_{\langle \tau_{age}^b, \underline{\tau}^b \rangle}$ accepts a piece-wise representation.

Otherwise, if all distributed timers are progressing and some time-to-fire is deterministic, let τ_P^a be the deterministic time-to-fire that becomes distributed. Then $f_{\langle \tau_{age}^b \underline{\tau}_Q^b \rangle}$ has global analytic representation over DBM domain D^b:

$$
\begin{aligned}
f_{\langle \tau_{age}^b, \underline{\tau}^b \rangle}(x_{age}, \underline{x}) = f_{\langle \tau_{age}^a, \underline{\tau}^a \rangle}(& x_{age} - x_P - B_{p*}, \\
& -x_P - B_{p*}, \underline{x} - x_P - B_{p*}),
\end{aligned}
\tag{7}
$$

where $\underline{x} = \langle x_1, \dots, x_{P-1} \rangle$.

ii) If some distributed timer is progressing and some is suspended, D^b takes the form of a linear convex polyhedron.

If no time-to-fire is deterministic, then $f_{\langle \tau_{age}^b, \underline{\tau}^b \rangle}$ accepts a piece-wise representation over a partition of D^b into a finite set of polyhedral subdomains. The problem is overcome by approximating D^b through its tightest embedding DBM and by approximating $f_{\langle \tau_{age}^b, \underline{\tau}^b \rangle}$ through Bernstein polynomials.

Otherwise, if some time-to-fire is deterministic, then a deterministic timer becomes distributed and the other ones become synchronized with it. According to this, $f_{\langle \tau_{age}^b, \underline{\tau}^b \rangle}$ accepts a global analytic representation over D^b:

$$
\begin{aligned}
f_{\langle \tau_{age}^b, \underline{\tau}^b \rangle}(x_{age}, \underline{x}) = f_{\langle \tau_{age}^a, \underline{\tau}^a \rangle}(& x_{age} - x_P, \\
& -x_P + B_{p*}, \underline{x}_P - x_P + B_{p*}, \underline{x}_S),
\end{aligned}
\tag{8}
$$

where $\underline{x} = \langle x_1, \dots, x_{P-1} \rangle$. To resort to a global analytic representation over a DBM domain, also in this case D^b and $f_{\langle \tau_{age}^b, \underline{\tau}^b \rangle}$ are approximated as in the previous case.

iii) If all distributed timers are suspended and no time-to-fire is deterministic, then $f_{\langle \tau_{age}^b, \underline{\tau}^b \rangle}$ is derived through a projection that eliminates τ_0^a:

$$
\begin{aligned}
& f_{\langle \tau_{age}^b, \underline{\tau}^b \rangle}(x_{age}, \underline{x}) \\
& = \int_{Min^0(x_{age}, \underline{x})}^{Max^0(x_{age}, \underline{x})} f_{\langle \tau_{age}^a, \underline{\tau}^a \rangle}(x_{age}, x_0, \underline{x}) dx_0,
\end{aligned}
\tag{9}
$$

where $\underline{x} = \langle x_P, \dots, x_{N-1} \rangle$. Also in this case, $f_{\langle \tau_{age}^b, \underline{\tau}^b \rangle}$ accepts a piece-wise representation over a partition of $D^b \downarrow_{U,V}$ into a finite set of DBM subdomains.

Otherwise, if all distributed timers are suspended and some time-to-fire is deterministic, then a deterministic timer becomes distributed and the other ones become synchronized with it. Hence, this case can be reduced to the case where some distributed timer is progressing and some is suspended.

- If τ_0^a is *synchronized* with some $\tau_{\iota_0}^b \in Q^b$, this case can be reduced to the previous one by swapping τ_0^a and $\tau_{\iota_0}^b$ in the partition of the sets V^b and Q^b.

Disabling. At the firing of t_0, multiple transitions may be disabled and a new time vector is derived by repeatedly eliminating the time-to-fire of each of them according to the following step. Let t_d be a disabled transition and $\langle \tau_{age}^c, \underline{\tau}^c \rangle$ be the random variable yielded by the elimination of its time-to-fire τ_d^b. Three cases are distinguished depending on whether τ_d^b is deterministic, or distributed and synchronized with some element of V^b, or distributed and not synchronized with any element of V^b.

- If τ_d^b is deterministic or synchronized, it can be removed without changing $\underline{\tau}_Q^b$ and $f_{\langle \tau_{age}^b, \underline{\tau}_Q^b \rangle}$, i.e., $\underline{\tau}_Q^c = \underline{\tau}_Q^b$ and $f_{\langle \tau_{age}^c, \underline{\tau}_Q^c \rangle} = f_{\langle \tau_{age}^b, \underline{\tau}_Q^b \rangle}$.
- If τ_d^b is distributed and synchronized with some $\tau_{\iota_1}^b \in V^b$, this case can be reduced to the previous one by swapping τ_d^b and $\tau_{\iota_1}^b$ in the partition of the sets Q^b and V^b.
- If τ_d^b is distributed but not synchronized with any element of V^b, then its elimination reduces by one element $\underline{\tau}_Q^b$. Let τ_d^b occupy the first position in $\underline{\tau}_Q^b$, i.e., $\tau_d^b = \tau_1^b$. Its elimination yields a new random variable $\langle \tau_{age}^c, \underline{\tau}^c \rangle = \langle \tau_{age}^b, \tau_2^b, \dots, \tau_{M-1}^b \rangle$ ranging within $D^c = D^b \downarrow_{\tau_1}$ and $f_{\langle \tau_{age}^c, \underline{\tau}_Q^c \rangle}$ is derived by marginalizing $f_{\langle \tau_{age}^b, \underline{\tau}_Q^b \rangle}$ with respect to τ_1^b:

  $$
  \begin{aligned}
  & f_{\langle \tau_{age}^c, \underline{\tau}^c \rangle}(x_{age}, \underline{x}) \\
  & = \int_{Min^1(x_{age}, \underline{x})}^{Max^1(x_{age}, \underline{x})} f_{\langle \tau_{age}^b, \underline{\tau}^b \rangle}(x_{age}, x_1, \underline{x}) dx_1
  \end{aligned}
  \tag{10}
  $$

  where $\underline{x} = \langle x_2, \dots, x_{N-1} \rangle$.

  As in [16, 15], this step may partition $D^c \downarrow_{U,V}$ into a finite set of DBM subdomains over which $f_{\langle \tau_{age}^c, \underline{\tau}^c \rangle}$ accepts a piece-wise representation.

Newly enabling. At the firing of t_0, multiple transitions may be newly-enabled and a new time vector is derived by adding the time-to-fire of each of them according to the following step. Let t_N be a newly-enabled transition with static PDF $f_N(x_N)$ supported over $[EFT_N, LFT_N]$. The time vector is extended with its time-to-fire τ_N^d, yielding a new random variable $\langle \tau_{age}^d, \underline{\tau}^d \rangle = \langle \tau_{age}^d, \tau_2^d, ..., \tau_{M-1}^d, \tau_N^d \rangle$ ranging over $D' = D^c \times [EFT_N, LFT_N]$. In particular, if t_N has point-like firing interval (i.e., $EFT_N = LFT_N$), then τ_N^d is added to U' without changing $\underline{\tau}_Q^c$ and $f_{\langle \tau_{age}^c, \underline{\tau}_Q^c \rangle}$, i.e., $\underline{\tau}_Q^d = \underline{\tau}_Q^c$ and $f_{\langle \tau_{age}^d, \underline{\tau}_Q^d \rangle} = f_{\langle \tau_{age}^c, \underline{\tau}_Q^c \rangle}$. Otherwise, if t_N has distributed firing interval (i.e., $EFT_N \neq LFT_N$), then τ_N^d is added to Q' and the joint PDF of τ_{age} and $\underline{\tau}_Q^d$ is obtained as:

$$
f_{\langle \tau_{age}^d, \underline{\tau}_Q^d \rangle}(x_{age}, \underline{x}, x_N) = f_{\langle \tau_{age}^c, \underline{\tau}_Q^c \rangle}(x_{age}, \underline{x}) \cdot f_N(x_N), \tag{11}
$$

where $\underline{x} = \langle x_2, \dots, x_{N-1} \rangle$.

3.2.4 *Approximation of transient classes*

If the model does not include transitions that can be suspended, the support of the time vector $\langle \tau_{age}, \underline{\tau} \rangle$ can be encoded as a Different Bounds Matrix (DBM) zone [43, 5, 24], i.e., the set of solutions of a system of linear inequalities constraining the difference between the times-to-fire of any two enabled transitions (including τ_{age}, which is regarded as an always-persistent time-to-fire, and a fictitious time-to-fire τ_*, which denotes the ground time of the class): $\tau_i - \tau_j \leq b_{ij} \; \forall \, i \neq j \in \{*, age, 0, 1, ..., N-1\}$ and $b_{ij} \in \mathbb{R} \cup \{\infty\}$. For instance, we consider the example of Fig. 2, where all transitions are assumed to be uniformly distributed over their supports. In the initial class, transitions t_{10}, t_{20}, t_{30}, and t_{40} are newly enabled and thus independently distributed. According to this, in the initial stochastic class Σ_0, the marking is $m_0 = p_{10}\, p_{30}\, p_{40}$ and the time vector $\langle \tau_{10}, \tau_{20}, \tau_{30}, \tau_{40} \rangle$ is distributed according to $f^1(\tau_{10}, \tau_{20}, \tau_{30}, \tau_{40}) = 1$ over D^0:

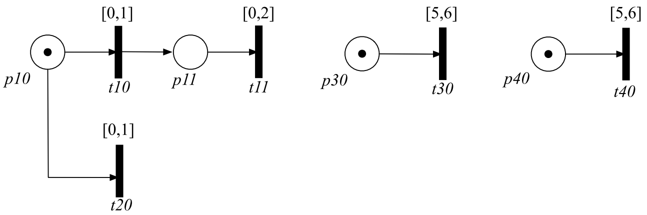

Figure 2: An example with uniformly distributed transitions.

$$D^0 = \begin{cases} 0 \leq \tau_{10} - \tau_\star \leq 1 \\ 0 \leq \tau_{20} - \tau_\star \leq 1 \\ 5 \leq \tau_{30} - \tau_\star \leq 6 \\ 5 \leq \tau_{40} - \tau_\star \leq 6 \end{cases}$$

In the initial class, τ_{age} is a deterministic timer (equal to 0) and thus it is not included in the support D^0 which refers to distributed timers, as it will better emerge in Sect. 3.2.5.

The DBM form is closed with respect to the steps of derivation of successor classes, enabling their efficient computation and encoding in polynomial time $O((N+1)^2)$ with respect to the cardinality $(N+1)$ of the time vector $\langle \tau_{age}, \underline{\tau} \rangle$. Moreover, the joint PDF of τ_{age} and $\underline{\tau}$ accepts a piece-wise representation over a partition of the support into DBM sub-domains and its symbolic form can be derived through an efficient closed-form calculus [16] if all transitions of the model have expolynomial distribution. For instance, in the example of Fig. 2, after the subsequent firings of t_{10} and t_{11}, a stochastic class Σ_2 is reached with marking $m_2 = p_{30} p_{40}$. In Σ_2, t_{30} and t_{40} are both persistent while t_{10}, t_{11}, and t_{20} are all disabled. The time vector $\underline{\tau}' = \langle \tau'_{age}, \tau'_{30}, \tau'_{40} \rangle$ turns out to be distributed according to a piece-wise density function f_2 over D^2:

$$D^2 = \begin{cases} -3 \leq \tau'_{age} - \tau'_* \leq 0 \\ 2 \leq \tau'_{30} - \tau'_* \leq 6 \\ 2 \leq \tau'_{40} - \tau'_* \leq 6 \\ -1 \leq \tau'_{30} - \tau'_{40} \leq 1 \\ 5 \leq \tau'_{30} - \tau'_{age} \leq 6 \\ 5 \leq \tau'_{40} - \tau'_{age} \leq 6 \end{cases}$$

$$f^2(\tau'_{30}, \tau'_{40}, \tau'_{age}) = \begin{cases} 0.5 & \text{if } \underline{\tau}' \in D^2_1 \\ -\tau'_{age} - 0.5\, (\tau'_{age})^2 & \text{if } \underline{\tau}' \in D^2_2 \\ 4.5 + 3\, \tau'_{age} + 0.5\, (\tau'_{age})^2 & \text{if } \underline{\tau}' \in D^2_3 \end{cases}$$

with $D^2_1 = D^2 \cap \{-2 \leq \tau'_{age} \leq -1, 3 \leq \tau'_{30} \leq 5, 3 \leq \tau'_{40} \leq 5\}$, $D^2_2 = D^2 \cap \{-1 \leq \tau'_{age} \leq -0, 4 \leq \tau'_{30} \leq 6, 4 \leq \tau'_{40} \leq 6\}$, and, $D^2_3 = D^2 \cap \{-3 \leq \tau'_{age} \leq -2, 2 \leq \tau'_{30} \leq 4, 2 \leq \tau'_{40} \leq 4\}$. As an example, Fig. 3 shows the (DBM) projection of D^2 on the space of timers τ'_{30} and τ'_{40}:

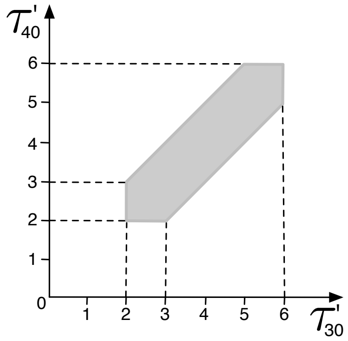

Figure 3: The joint support of $\langle \tau'_{30}, \tau'_{40} \rangle$ of the model of Fig. 2 after the subsequent firings of t_{10} and t_{11}.

Conversely, if the model includes transitions that can be suspended, the support of the time vector $\langle \tau_{age}, \underline{\tau} \rangle$ breaks the DBM structure and takes the form of a linear convex polyhedron [11, 6]. With reference to the example of Fig. 2, we consider the case when t_{30} and t_{40} require a resource res with priority level 1 and 2, respectively. At the firing of t_{10}, transitions t_{30} is progressing while transition t_{40} is suspended, reaching a stochastic class Σ_1 with marking $m_1 = p_{11}\, p_{30}\, p_{40}$ and time vector $\langle \tau'_{age}, \tau'_{11}, \tau'_{30}, \tau'_{40} \rangle$ distributed over the polyhedral support D^1:

$$D^1 = \begin{cases} -1 \leq \tau'_{age} \leq 0 & 0 \leq \tau'_{20} \leq 1 \\ 4 \leq \tau'_{30} \leq 6 & 5 \leq \tau'_{40} \leq 6 \\ -6 \leq \tau'_{20} - \tau'_{30} \leq -4 & -7 \leq \tau'_{20} - \tau'_{40} \leq -4 \\ 0 \leq \tau'_{20} - \tau'_{age} \leq 1 & -2 \leq \tau'_{30} - \tau'_{40} \leq 1 \\ 5 \leq \tau'_{30} - \tau'_{age} \leq 6 & -7 \leq \tau'_{age} - \tau'_{40} \leq -5 \\ & \\ -7 \leq -\tau'_{40} - \tau'_{20} \leq -4 & (*) \\ -12 \leq -\tau'_{40} - \tau'_{30} \leq -9 & (*) \\ -6 \leq -\tau'_{40} - \tau'_{age} \leq -4 & (*) \\ 9 \leq -\tau'_{20} + \tau'_{30} + \tau'_{40} \leq 12 & (*) \\ -1 \leq \tau'_{20} - \tau'_{30} + \tau'_{40} \leq 2 & (*) \\ 5 \leq -\tau'_{age} + \tau'_{20} + \tau'_{40} \leq 7 & (*) \\ 4 \leq \tau'_{age} - \tau'_{20} + \tau'_{40} \leq 6 & (*) \\ 10 \leq -\tau'_{age} + \tau'_{30} + \tau'_{40} \leq 12 & (*) \\ -1 \leq \tau'_{age} - \tau'_{30} + \tau'_{40} \leq 1 & (*) \end{cases}$$

Specifically, the constraints denoted by $(*)$ are not linear inequalities constraining the difference between two timers, but they rather involve more than two timers or the sum of two timers. This actually makes D^1 break the form of a DBM zone.

In principle, polyhedral domains could be managed through the Parma Polyhedra Library [3], which is currently used by several applications in the field of analysis and verification of hardware and software systems. In the practice, manipulation of polyhedral domains would lead to exponential complexity for derivation and encoding of successor classes, impairing efficient enumeration of the state-space. In [11], in a non-deterministic perspective, the problem of polyhedral domains is circumvented by resorting to an over-approximation that replaces supports with their tightest embedding DBM zones. The analysis technique of [11] also supports the exact identification of feasible timings of selected paths through an algorithm that cleans up false behaviors produced by the approximation, enabling efficient verification of reachability properties under timing constraints and real-time deadlines. Porting the approximation approach of [11] from a non-deterministic to a stochastic perspective comprises a more difficult challenge. In fact, the support takes the form of a linear convex polyhedron and the joint PDF of the time vector $\langle \tau_{age}, \underline{\tau} \rangle$ assumes a global analytic form over a partition of the support into sub-domains that are no longer DBM zones but rather polyhedra themselves. The presence of polyhedral constraints also increases the complexity of the analytic form of PDFs. In fact, in each subdomain, the lower and the upper bound of a timer can be not only a constant or a monovariate linear function of a single component of the time vector, but also a bivariate linear function of two components of the time vector. This takes relevance in the subsequent integrations performed to enumerate successor classes.

We address both aspects of the problem by relying on a twofold approximation of timing domains and density functions. On the one hand, we replace polyhedral supports with their tightest embedding DBM as in the non-deterministic approach of [11]. This adds false behaviors but still preserves all timings that would be feasible for the exact representation of domains. On the other hand, we approximate piece-wise density functions through multivariate Bernstein polynomials [34, 38], extending the approximation technique that was devised and experimented in [16] for steady-state analysis of non-preemptive models. Bernstein approximants can be derived in a straightforward manner by weighting a kernel of multivariate monomials according to the samples of the approximated density functions taken over a regular grid. As a matter of fact, replacing piece-wise density functions with global approximants actually perturbs the distribution of feasible behaviors and also associates false behaviors with a non-null probability. In positive, the probability of false behaviors can be reduced by assigning null value to the samples that belong to the tightest embedding DBM but not to the polyhedron.

In principle, both the limits deriving from the approximation of domains and densities could be overcome by working on the complexity of the Bernstein approximant. In fact, as the number of samples goes to infinity and those samples that belong to the extended DBM support but not to the exact polyhedron are assigned zero value, the Bernstein approximant uniformly converges to the approximated function with an error bounded by a Lipschitz inequality [34], provided that the approximated function is continuous (which is the case of our density functions). Of course, the number of samples used in the approximation is finite. Actually, Bernstein polynomials turn out to be very good at providing a rough approximation of density functions with the benefit of simplicity of derivation, much less in converging to the approximated function with tight accuracy. Yet, other approximants could be considered.

3.2.5 On the effect of suspension on DET transitions

In stochastic analysis of non-preemptive models [44, 30], multiple timers $\{\tau_i\}_{i \in I}$ may have a deterministic delay Δ with respect to a distributed timer τ_k and we call them *synchronized* with τ_k: $\tau_i = \tau_k + \Delta \quad \forall\, i \in I$, i.e., $\tau_i - \tau_k = \Delta$, $\forall\, i \in I$. For instance, consider the case when a timer τ_1 takes values over the interval $[0, 3]$ and a timer τ_2 is bound to have a deterministic delay $\Delta = 5$ with respect to τ_1, thus varying within $[5, 8]$ (for illustration purposes, the variable τ_{age} is not considered in the example). The domain D_{12} of the time vector $\langle \tau_1, \tau_2 \rangle$ has the following form:

$$D_{12} = \begin{cases} 0 \le \tau_1 - \tau_* \le 3 \\ 5 \le \tau_2 - \tau_* \le 8 \\ 5 \le \tau_2 - \tau_1 \le 5 \end{cases}$$

where all constraints are in DBM form. In particular, the constraint $\tau_2 - \tau_1 = 5$ is expressed by the couple of linear inequalities $\tau_1 - \tau_2 \le -5$ and $\tau_2 - \tau_1 \le 5$. Fig. 4 illustrates domain D_{12}, making evident that it has null measure due to the presence of the synchronized timer τ_2.

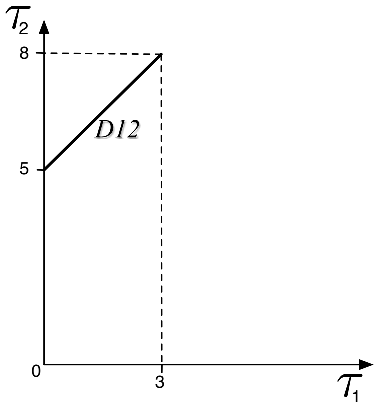

Figure 4: A bivariate DBM domain D_{12} with dimension $||D_{21}|| = 0$ as $\tau_2 - \tau_1$ is bound to assume a deterministic value.

Synchronized timers arise when a set of DET transitions, enabled in the same class or in classes with a deterministic distance between their entering times, persist at the firing of a distributed transition. In the specific application context addressed in this paper, this occurs for transition accounting for periodic task releases, which are simultaneously enabled at each hyper-period and enabled within a deterministic time gap at any other task activation. This condition is illustrated by the model of Fig. 5, which includes two deterministic transitions t_1 and t_2 accounting for the periodic releases of two tasks with period of 5 time units and 10 time units, respectively. Both t_1 and t_2 are enabled at each multiple of 10 time units, and t_1 is also enabled 5 time units after each time it is enabled with t_2. The model also includes a distributed t_3 with support $[2, 5]$, accounting for the release time of a sporadic task with minimum and maximum inter-arrival time equal to 2 and 5, respectively.

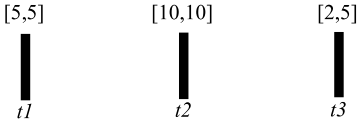

Figure 5: A trivial model with two deterministic transitions t_1 and t_2 and a uniformly distributed transition t_3.

When multiple DET transitions persist at the firing of a distributed transition, one of these DET transitions becomes distributed while the other ones are *synchronized* with it. This is the case that occurs in the example of Fig. 5 when the deterministic transitions t_1 and t_2 persist at the firing of the distributed transition t_3. In this example, in the initial class, each transition takes values within its static firing interval and the domain D_{123} of the time vector $\langle \tau_1, \tau_2, \tau_3 \rangle$ has the following form:

$$D_{123} = \begin{cases} 5 \le \tau_1 - \tau_* \le 5 \\ 10 \le \tau_2 - \tau_* \le 10 \\ 2 \le \tau_3 - \tau_* \le 5 \end{cases}$$

At the firing of t_3, the times-to-fire of t_1 and t_2 are reduced by the time-to-fire of t_3 while t_3 is regarded as newly enabled, leading to a new class where τ_1 is distributed over the support $[5, 8]$, τ_2 is bound to have a deterministic delay equal to 5 with respect to τ_1, and τ_3 is distributed over the interval $[2, 4]$. According to this, the time domain of the class reached through the firing of t_3 is D'_{123}:

$$D'_{123} = \begin{cases} 0 \leq \tau_1 - \tau_* \leq 3 \\ 5 \leq \tau_2 - \tau_* \leq 8 \\ 5 \leq \tau_2 - \tau_1 \leq 5 \\ 2 \leq \tau_3 - \tau_* \leq 5 \end{cases}$$

which is actually equal to domain D_{12} shown in Fig. 4 plus the inequalities that constrain τ_3 within its static firing interval $[0, 2]$, i.e., $D'_{123} = D_{12} \cup \{2 \leq \tau_3 - \tau_* \leq 5\}$. It is worth noting that, when a synchronized timer persists at the firing of a distributed timer, then it either becomes deterministic (if it is synchronized with the firing transition), or it remains synchronized with the same distributed transition with the same deterministic delay (otherwise). This maintains synchronization constraints in DBM form across subsequent derivations of successor classes.

As shown by the example of Fig. 4, when a class includes DET timers and/or synchronized timers, its time domain has null measure. The problem is circumvented by restraining the stochastic characterization to a minimal set of timers that are sufficient to determine all feasible values for variables of the time domain. This basically consists in maintaining the joint PDF of distributed timers and τ_{age}, while avoiding a stochastic characterization of deterministic and synchronized timers. The solution also reduces the number of managed variables, limiting the complexity of encoding and computation of classes.

Unfortunately, when the solution technique encompasses suspension and resumption of timers, the way how deterministically dependent timers synchronize with each other becomes much more complicated to manage. In fact, it may be the case that multiple timers become synchronized according to a constraint that breaks the limit of a DBM equality and takes the form of a linear convex polyhedral equality. Specifically, the unfavorable case occurs when multiple DET or synchronized timers persist at the firing of a distributed timer and at least on of them is *i) progressing* and synchronized with a *suspended* (distributed) timer or *ii) suspended* and synchronized with a *progressing* distributed timer that is not coincident with the time-to-fire of the firing transition. For instance, consider the model of Fig. 6, which extends the example of Fig. 5 by chaining transition t_3 with a deterministic transition t_4 requiring a resource res with priority level 1 and with a transition t_5 distributed over $[0, 3]$. Resource res is also required by transition t_2 with (lower) priority level 2. In the initial class, the enabled transitions are all progressing (as res is requested only by t_2) and the vector $\langle \tau_1, \tau_2, \tau_3 \rangle$ is supported over D_{123}. When t_3 fires: t_1 and t_2 are persistent and progressing and their times-to-fire are reduced by that of t_3, which is regarded as newly enabled; t_4 and t_5 are also enabled. According to this, in the arrival class, the time vector $\langle \tau_1, \tau_2, \tau_3, \tau_4, \tau_5 \rangle$ is distributed over $D_{1234} = D'_{123} \cup \{1 \leq \tau_4 \leq 1, 0 \leq \tau_5 \leq 3\}$. Specifically, in this class: τ_1, τ_3, and τ_5 are progressing distributed timers, τ_2 is a suspended timer synchronized with τ_1, and τ_4 is a deterministic progressing timer. At the subsequent firing of t_5 (which is disabled in the class reached through its firing), the values of τ_1, τ_3, and τ_4 are reduced by the value of τ_1, while τ_2 is not changed, yielding a new time vector $\langle \tau'_1, \tau'_2, \tau'_3, \tau_4 \rangle = \langle \tau_1 - \tau_5, \tau_2, \tau_3 - \tau_5, \tau'_4 - \tau_5 \rangle$. According to this, τ'_2 remains equal to $\tau_1 + 5$. By subsequently applying the substitutions $\tau_1 = \tau'_1 + 5$ and $\tau_5 = \tau_4 - \tau'_4 = 1 - \tau'_4$, we obtain that $\tau'_2 = \tau'_1 - \tau'_4 + 6$, which is not an equality in DBM form but rather a linear convex polyhedral equality. The derivation of the exact form of these synchronizations requires exponential complexity in the number of synchronized timers and would impair efficient enumeration of the state space. Note that these polyhedral constraints are additional with respect to those polyhedral constraints that may involve distributed timers, which are well managed through the approximation of both domains and density functions as sketched in Sect. 3.2.4.

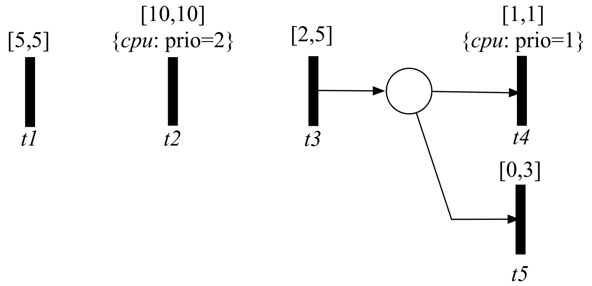

Figure 6: A variant of the model of Fig. 5.

In the context of a model-driven approach to the development of real-time SW components, a timer may arise from a timeout, from the inter-release time of a periodic or a sporadic task, from a deterministic offset or a non-deterministic jitter between subsequent task releases, or from the execution time of a computation. In the practice, DET timers mainly represent temporal parameters that are never suspended (i.e., timeouts, task periods, and release offsets), while distributed timers account both for temporal parameters that are never suspended (i.e., inter-release times of asynchronous tasks and release jitters) and for temporal parameters that can be suspended (i.e., the execution time of a computation). Whereas DET temporal parameters representing the execution time of computations that can be suspended are often considered in the literature, they actually correspond to an approximation abstraction that replaces the actual execution time with its Worst Case Execution Time (WCET). In fact, in the practice, the code of a chunk computation is usually characterized by multiple execution paths depending on the input values, which results in a non-pointlike spectrum of possible execution times. According to this, without loss of generality, we can safely assume that DET transitions are never suspended, i.e., DET transitions do not require resources (or require resources with higher priority level than that of any other transition requiring at least one of those resources). Thus, DET transitions are always progressing in every class where they are enabled. As a direct consequence, any synchronized timer is always progressing and the distributed timer that it is synchronized with is also progressing. In the theoretical perspective, this rules out the case of polyhedral constraints among synchronized timers, maintaining synchronization constraints in the form of DBM equalities which can be efficiently managed as in [44, 30]. In the applicative perspective, this permits to use DET transitions to model timeouts, delays, and synchronous release times, but not computation chunks, which nevertheless does not reduce the expressive capabilities of the approach with respect to the models at hand.

Note that the applicability of the proposed solution technique could be lifted to encompass various other special cases, e.g., a single possibly suspended DET transition (which mainly accounts for a preemptable computation with DET execution time), multiple DET transitions that are bound to be all progressing or all suspended in each class (which primarily model concurrent computations with DET execution time that can undergo preemption only simultaneously), and multiple DET transitions requiring some resources with higher priority level than that of any other transition requiring at least one of those resources (such DET transitions cannot be suspended by construction and represent concurrent computations with a DET execution time running at maximum priority).

4. COMPUTATIONAL EXPERIENCE

We illustrate the potentials of the proposed solution technique in the evaluation of real-time task-sets. The approach supports the derivation of transient probabilities of reachable states of concurrency, which comprise all the possible combinations of task computations that are concurrently ready, running, or blocked. As an example of probabilistic measures that can be evaluated, we derive the transient probability that a task holds a semaphore and the transient probability that a task misses a deadline. To this end, the task-set purposely includes a trivial case of priority inversion, which causes a job of the high-priority task to be released before the previous job is completed. Priority inversion could be avoided through many techniques of program structuring, for instance by applying a priority ceiling emulation protocol [41]. Accuracy of the approximated analysis is assessed by comparing analytic transient probabilities against simulation results. Both analysis and simulation were performed through the Sirio Framework [17, 19] on an Intel Core i7 laptop processor.

4.1 Three concurrent synchronized tasks

Fig. 7 shows the representation of a non-deterministic task-set introduced in [11], which is here extended with a stochastic characterization that associates each transition with a distribution (times are expressed in ms). The task-set includes a high-priority periodic task Tsk_1 with period of 50 ms, a mid-priority sporadic task Tsk_2 with an inter-arrival time falling within $[100, 300]$ ms, and a low-priority periodic task Tsk_3 with period of 150 ms. Moreover, Tsk_1 and Tsk_3 are synchronized on a binary semaphore named $mutex$. To fulfill the assumption that deterministic timers are always progressing, transitions t_{11} and t_{31}, which were IMM in the original model of [11], are here associated with a non-pointlike firing interval. Note that this change increases not only the complexity of the model, but also its validity, as semaphore operations are actually time-consuming. With respect to [11], the task-set complexity is also increased by reducing the maximum inter-release time of Tsk_2 from $+\infty$ to 300 ms. Moreover, the support of the computations (i.e., t_{12}, t_{21}, and t_{32}) is enlarged to increase the probability of a deadline miss of Tsk_1 and observe this behavior also in simulation, obtaining a ground truth for analytical results.

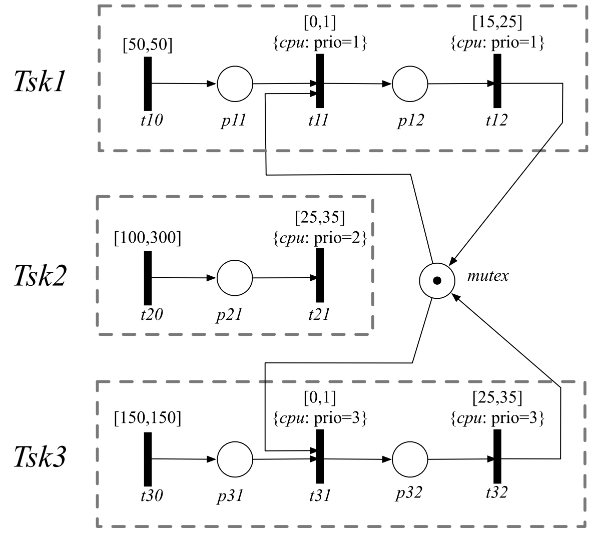

Figure 7: The model of a task-set with three concurrent tasks and a binary semaphore.

In the model of Fig 7, t_{10} and t_{30}, representing periodic task releases of Tsk_1 and Tsk_2, are associated with Dirac delta distributions centered at 50 and 150 ms, respectively. We assume that the remaining transitions have uniform distribution over their firing intervals, i.e., t_{20}, representing sporadic releases of Tsk_2; t_{12}, t_{21} and t_{32}, accounting for the execution times of computations; t_{11} and t_{31}, modeling semaphore acquisition operations.

4.2 Experimental results

Transient analysis of the task-set of Fig. 7 is repeated for values 3, 5, and 7 of the maximum polynomial degree of each variable in the multivariate Bernstein approximation (degree for short), adopting an error threshold of $\epsilon = 0.01$ on the total unallocated probability within the time limit of 300 ms and evaluating transient probabilities at multiples of 0.5 ms. The enumeration yields 938, 935 and 851 transient classes for degree 3, 5, and 7 in 3, 8, and 97 minutes, respectively, while the simulation of $2,000,000$ runs took about an hour. For increasing values of the degree, the decreasing number of transient classes can be ascribed to the increasing number of samples of the approximated PDFs that belong to the tightest embedding DBM but not to the polyhedron. As these samples are assigned null probability, false behaviors introduced by the approximation of supports have lower probability and, thus, the number of transient classes decreases.

The complexity of the analysis is impacted by the number of transient classes. This depends on the number feasible events within the time bound, which is determined by: the number of concurrent tasks (especially sporadic tasks), the ratio between the minimum temporal parameter and the hyper-period of the task-set, and the mean time-to-fire duration [11]. The complexity of the analysis is also impacted by the complexity of density functions, which is affected by different factors for approximated and exact transient classes. In the former case, it depends on the polynomial complexity of Bernstein approximants, determined by the number of samples. In the latter case, it depends on the complexity of domain partitioning and expolynomial functions [16]. For the task-set of Fig. 7, a significant number of transient classes include preemptive behaviors, and thus the computation time increases with the degree.

As an example, Fig. 8 plots the transient probability that the semaphore $mutex$ is held by Tsk_1, which corresponds to the probability of any marking where place p_{12} contains a non-null number of tokens (and, by construction, place $mutex$ contains no tokens). The analysis follows the trend of the simulation with sufficient accuracy, but the error becomes more relevant for behaviors undergoing repeated approximations of domains and densities.

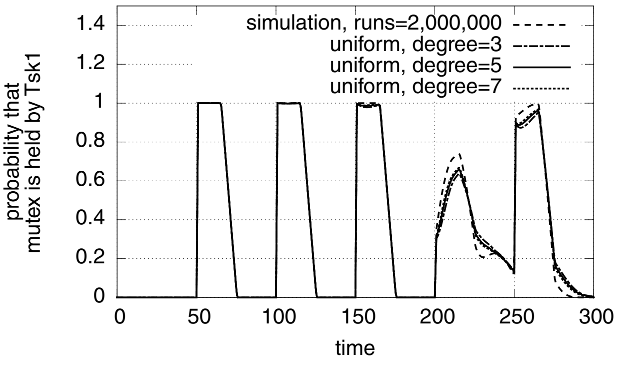

Figure 8: Probability that $mutex$ is held by Tsk_1 for the task-set of Fig. 7 with uniformly distributed execution times.

Transient analysis also evidences traces where the high priority task Tsk_1 undergoes a priority inversion with respect to Tsk_3. This occurs when Tsk_1 is blocked on semaphore $mutex$ (acquired and not yet released by Tsk_3) and a job of Tsk_2 preempts Tsk_3, leading Tsk_1 to miss its deadline. Fig. 9 shows the transient probability that two jobs of Tsk_1 are active, i.e., Tsk_1 is in a marking m such that $m(p_{11}) + m(p_{12}) > 1$. This condition is caused by a deadline miss of Tsk_1, which is deterministically bound to occur at time 250 ms. The probability associated with the peak at 250 ms (0.11 in the simulation and nearly 0.14 in the analysis) corresponds to the probability of having a deadline miss for Tsk_1 before 300 ms. After the deadline miss, the task-set recovers from the condition of having two active jobs of Tsk_1, reaching a marking m' such that $m'(p_{11}) + m'(p_{12}) \leq 1$ before the subsequent release of Tsk_1 at 300 ms.

In this case, the approximation affects the transient reward in a more substantial manner, by prolonging the maximum time required for the recovery, which is 270 ms in the simulation and 290 ms according to the analysis. This is a combined effect of the approximation of supports and density functions, which results in false behaviors with non-null probability that lower the probability of feasible behaviors. As a future direction, the error introduced by the extension of domains could be evaluated on a per-class basis and employed in the computation of an error bound on the analysis results, thus extending the cleanup procedure of [11] in a probabilistic perspective.

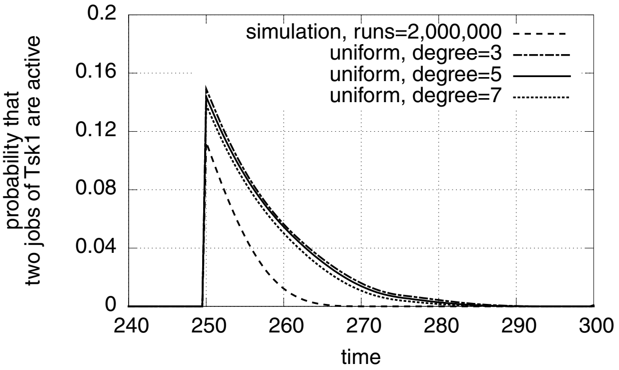

Figure 9: Probability that a deadline of Tsk_1 is missed for the task-set of Fig. 7 with uniformly distributed execution times.

5. CONCLUSIONS

We proposed a novel solution technique for transient analysis of real-time systems having non-Markovian temporal parameters and running under fixed-priority preemptive scheduling. The approach extends [30] so as to encompass suspension and resumption of timers taking values over a non-deterministic support, which enables the representation of concurrent computations with an execution time constrained between a minimum and a maximum value. As a notable trait, the approach addresses models that may also include deterministic timers that are bound to be never suspended, which is crucial to represent key features of real-time programming such as timeouts, offsets, and periodic task releases.

In the theoretical perspective, the approach faces major challenges due to the presence of progressing and suspended non-Markovian timers. This exacerbates the complexity of the analysis, yielding time vectors distributed over polyhedral supports according to probability density functions that assume a piece-wise analytical form over a partition of the support in polyhedral subdomains. The issue is circumvented through the approximation of both domains and density functions, by replacing the exact distributions with global Bernstein approximants supported over the tightest embedding DBM zones of polyhedral domains. It is worth stressing that the proposed solution technique relies on the approximation of joint multivariate PDFs in the state-space rather than on the approximation of monovariate PDFs in the model structure. This opens the way to the analysis of other classes of models that break the structure of DBM time domains, e.g., hybrid systems and, notably, multi-rate systems.

In the applicative perspective, the approach is amenable to smooth integration within a MDD methodology that was proven to be viable in a non-deterministic perspective [18, 25], providing decisive support for schedulability analysis and performance engineering of real-time systems while avoiding to disrupt consolidated practices of design and documentation. Specifically, since the very initial steps of software design, the approach permits to derive a quantitative formal specification of the task-set where unknown distributions of temporal parameters are tentatively guessed by analogy with previous realizations. This supports rapid exploration of the design space and early performance validation of design assumptions, long before the model is actually implemented on a real-time system. During the iterations of the development process, the temporal distributions are refined according to the results of the execution time profiling technique implemented in [18, 25], by deriving expolynomial distributions that fit the measured histograms. As a significant aspect, this comprises the main effort on the part of the developer to manage a stochastic characterization of timers.

6. REFERENCES

[1] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, Apr. 1994.

[2] R. Alur, I. Lee, and O. Sokolsky. Compositional refinement for hierarchical hybrid systems. In *Hybrid Systems: Computation and Control, LNCS 2034*, pages 33–48. Springer–Verlag, 2001.

[3] R. Bagnara, P. M. Hill, and E. Zaffanella. The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Science of Computer Programming*, 72(1–2):3–21, 2008.

[4] G. Behrmann, A. David, and K. Larsen. A tutorial on UPPAAL. *Formal methods for the design of real-time systems*, pages 33–35, 2004.

[5] B. Berthomieu and M. Diaz. Modeling and Verification of Time Dependent Systems Using Time Petri Nets. *IEEE Trans. on SW Eng.*, 17(3):259–273, March 1991.

[6] B. Berthomieu, D. Lime, O. H. Roux, and F. Vernadat. Reachability Problems and Abstract State Spaces for Time Petri Nets with Stopwatches. *LAAS Report 04483*, 2004.

[7] A. Bobbio, A. Puliafito, and M. Telek. A modelling framework to implement preemption policies in non-Markovian SPNs. *IEEE Trans. on SW Eng.*, 26(1), January 2000.

[8] A. Bobbio, A. Puliafito, M. Telek, and K. S. Trivedi. Recent Developments in Non-Markovian Stochastic Petri Nets. *Journal of Circuits, Systems, and Computers*, pages 119–158, 1998.

[9] A. Bobbio and M. Telek. Transient Analysis of a Preemptive Resume M/D/1/2/2 through Petri Nets. *Periodica Politechnica*, 41:123–146, September 1997.

[10] G. Bucci, L. Carnevali, L. Ridi, and E. Vicario. Oris: a tool for modeling, verification and evaluation of real-time systems. *Int. Journal of Software Tools for Technology Transfer*, 12(5):391 – 403, 2010.

[11] G. Bucci, A. Fedeli, L. Sassoli, and E. Vicario. Timed state space analysis of real time preemptive systems. *IEEE Trans. SW Eng.*, 30(2):97–111, Feb. 2004.

[12] G. Bucci, L. Sassoli, and E. Vicario. Correctness verification and performance analysis of real time systems using stochastic preemptive time petri nets. *IEEE Trans. on SW Eng.*, 31(11):913–927, November 2005.

[13] G. Buttazzo. *Hard Real-Time Computing Systems*. Springer, 2005.

[14] BWB - Federal Office for Military Technology and Procurement of Germany. *V-Model 97, Lifecycle Process Model-Developing Standard for IT Systems of the Federal Republic of Germany. General Directive No. 250*, June 1997.

[15] L. Carnevali, J. Giuntini, and E. Vicario. A symbolic approach to quantitative analysis of preemptive real-time systems with non-markovian temporal parameters. In *VALUETOOLS*, May 2011.

[16] L. Carnevali, L. Grassi, and E. Vicario. State-Density Functions over DBM Domains in the Analysis of Non-Markovian Models. *IEEE Trans. on SW Eng.*, 35(2):178–194, 2009.

[17] L. Carnevali, L. Ridi, and E. Vicario. A framework for simulation and symbolic state space analysis of non-markovian models. In *SAFECOMP*, pages 409–422, 2011.

[18] L. Carnevali, L. Ridi, and E. Vicario. Putting preemptive Time Petri Nets to work in a V-Model SW life cycle. *IEEE Trans. on SW Engineering*, 37(6), Nov./Dec. 2011.

[19] L. Carnevali, L. Ridi, and E. Vicario. Sirio: A framework for simulation and symbolic state space analysis of non-Markovian models. In *QEST'11*, pages 153–154, 2011.

[20] F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *Proc. Int. Conf. on Concurrency Theory*, pages 66–80. Springer-Verlag, 2005.

[21] F. Cassez and K. G. Larsen. The Impressive Power of Stopwatches. In *Proceedings of the 11th International Conference on Concurrency Theory*, CONCUR '00, pages 138–152, London, UK, UK, 2000. Springer-Verlag.

[22] G. Ciardo, A. Blakemore, P. Chimento, J. Muppala, and K. Trivedi. Automated generation and analysis of Markov reward models using Stochastic Reward Nets. *IMA Volumes in Mathematics and its Applications*, 48:145–191, 1993.

[23] A. Cumani. ESP: A Package for the Evaluation of Stochastic Petri Nets with Phase-Type Distributed Transition Times. In *PNPM*, pages 144–151, 1985.

[24] D. Dill. Timing assumptions and verification of finite-state concurrent systems. *Proc. Workshop on Computer Aided Verification Methods for Finite State Systems*, 1989.

[25] J. B. Dugan. Galileo: A tool for dynamic fault tree analysis. In *Computer Performance Evaluation / TOOLS*, pages 328–331, 2000.

[26] European Committee for Electrotechnical Standardization. *CENELEC EN 50128 Railway applications - Communications, signalling and processing systems - Software for railway control and protection systems*, July 2009.

[27] European Cooperation for Space Standardization. *ECSS-Q-ST-80C Space product assurance - Software product assurance*, March 2009.

[28] T. A. Henzinger, B. Horowitz, and C. M. Kirsch. Giotto: A time-triggered language for embedded programming. In *Proc. of the IEEE*, pages 84–99. IEEE, 2003.

[29] A. Hessel, K. Larsen, M. Mikucionis, B. Nielsen, P. Pettersson, and A. Skou. Testing Real-Time systems using UPPAAL. In *Formal Methods and Testing*, pages 77–117. 2008.

[30] A. Horváth, M. Paolieri, L. Ridi, and E. Vicario. Transient analysis of non-Markovian models using stochastic state classes. *Performance Evaluation*, 2012.

[31] A. Horvath, A. Puliafito, M. Scarpa, and M. Telek. Analysis and Evaluation of non-Markovian Stochastic Petri Nets. In *Proc. of the* 11^{th} *Int. Conf. Computer Performance Evaluation (TOOLS)*, pages 171–187, 2000.

[32] G. Karsai, J. Sztipanovits, A. Ledeczi, and T. Bapty. Model-integrated development of embedded software. *Proc. of the IEEE*, 91:145–164, Jan. 2003.

[33] F. Longo and M. Scarpa. Applying Symbolic Techniques to the Representation of Non-Markovian Models with Continuous PH Distributions. In *Proc. European Perf. Eng. Workshop on Computer Perf. Eng.*, EPEW '09, pages 44–58, Berlin, Heidelberg, 2009. Springer-Verlag.

[34] G. Lorentz. *Bernstein Polynomials*. University of Toronto Press, 1953.

[35] The Mathworks. *Simulink*. www.mathworks.com/products/simulink.

[36] Radio Technical Commission for Aeronautics. *DO-178B, Software Considerations in Airborne Systems and Equipment Certification*, 1992.

[37] O. H. Roux and D. Lime. Time Petri Nets with inhibitor hyperarcs: formal semantics and state-space computation. 25^{th} *Int. Conf. on Theory and Application of Petri Nets*, 3099:371–390, 2004.

[38] T. Sauer. Multivariate Bernstein polynomials and convexity. *Computer Aided Geometric Design*, 8(6):465 – 478, 1991.

[39] M. Scarpa and A.Bobbio. Kronecker Representation of Stochastic Petri Nets With Discrete PH Distribution. In *IPDS Proceedings*, pages 52–62, 1998.

[40] D. C. Schmidt. Model–driven engineering. *IEEE Computer*, pages 1–2, February 2006.

[41] L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority inheritance protocols: An approach to real-time synchronization. *IEEE Trans. Comput.*, 39(9):1175–1185, 1990.

[42] US Department of Defense. MIL-STD-498: Military standard for software development and documentation. Technical report, 1994.

[43] E. Vicario. Static analysis and dynamic steering of time dependent systems using Time Petri Nets. *IEEE Trans. on SW Eng.*, 27(1):728–748, August 2001.

[44] E. Vicario, L. Sassoli, and L. Carnevali. Using stochastic state classes in quantitative evaluation of dense-time reactive systems. *IEEE Trans. SW Eng.*, 35(5):703–719, 2009.

[45] R. Zijal, G. Ciardo, and G. Hommel. Discrete Deterministic and Stochastic Petri Nets. In *In Proc. of Measurement, Modeling, and Valuation of Computer and Communication Systems*, pages 103–117, 1997.