Realizability of Schedules by Stochastic Time Petri Nets with Blocking Semantics (regular paper)

Loïc Hélouët, Karim Kecir

Realizability of Schedules by Stochastic Time Petri Nets with Blocking Semantics (regular paper)

Loïc Hélouët, Karim Kecir

INRIA Rennes, France
loic.helouet@inria.fr, karim.kecir@inria.fr

Abstract. This paper considers realizability of schedules by stochastic concurrent timed systems. Schedules are high level views of desired executions represented as partial orders decorated with timing constraints, while systems are represented as elementary stochastic time Petri nets. We first consider logical realizability: a schedule is realizable by a net $\mathcal{N}$ if it embeds in a time process of $\mathcal{N}$ that satisfies all its constraints. However, with continuous time domains, the probability of a time process that realizes a schedule is null. We hence consider probabilistic realizability, of schedules up to some imprecision α, that holds if the probability that $\mathcal{N}$ logically realizes S with constraints enlarged by α time units is strictly positive. We show that upon a sensible restriction guaranteeing time progress, logical and probabilistic realizability of a schedule can be checked on the finite set of symbolic prefixes extracted from a bounded unfolding of the net. We give a construction technique for these prefixes and show that they represent all time processes of a net occurring up to a given maximal date. We then show how to verify existence of an embedding and compute the probability of its realization.

1 Introduction

Correct scheduling of basic operations in automated systems such as manufacturing or transport systems is a way to manage at best available resources, avoid undesired configurations, or achieve an objective within a bounded delay. Following a predetermined schedule is also a way to meet qualitative objectives. This is for instance the case for metro networks, in which changes to predetermined schedules may cause network congestion and impact users satisfaction. Schedules provide high-level views for correct ordering of operations in a system, consider time issues and provide optimal dates for a production plan. They can be seen as partial orders among basic tasks, decorated with dates and timing constraints.

Designing a correct and optimal schedule for a system is a complex problem. Occurrence dates of events can be seen as variables, and correct and optimal schedules as optimal solutions (w.r.t. some criteria) for a set of constraints over these variables. Linear programming solutions have been proposed to optimize scheduling in train networks [9, 10]. The size of models for real systems that run for a full day call for approximated solutions and is usually addressed by experts. When a high-level schedule and the low-level system that implements it are designed in a separate way, nothing guarantees that the system is able to realize the expected

schedule. This calls for tools to check realizability of a schedule by a system. One can notice that optimal and realizable schedules are not necessarily robust if they impose thight realization dates to systems that are subject to random variations (delays in productions, faults...). In metro networks, trains delays are expected and are part of the normal behavior of the system. To overcome this problem, metro schedules integrate small recovery margins that avoid the network performance to collapse as soon as a train is late. Note also that for systems where time issues are defined with continuous variables, the probability to execute a given event at a precise date is zero. Furthermore, being able to realize a schedule does not mean that the probability to meet optimal objectives is high enough. Beyond logical realizability, a schedule shall hence be considered as realizable if it can be approached with a significant probability.

This paper addresses realizability of schedules by stochastic timed systems. We define schedules as labeled partial orders decorated with dates and timing constraints, and represent systems with elementary stochastic time Petri nets (STPN for short), a model inspired from [14]. We particularly emphasize on resources: non-availability of a resource (represented by a place) may block transitions. This leads to the definition of a blocking semantics for STPNs that forbids firing a transition if one of its output places is filled. We then propose a notion of realizability: a schedule S is *realizable* by a net $\mathcal{N}$ if S embeds in a symbolic process of $\mathcal{N}$ that meets constraints of S. We prove that upon some reasonable time progress assumption, realizability can be checked on a finite set of symbolic processes, obtained from a bounded untimed unfolding [17, 13] of $\mathcal{N}$. Symbolic processes are processes of the unfolding with satisfiable constraints on occurrence dates of events. A symbolic framework to unfold time Petri nets was already proposed in [5, 7] but blocking semantics brings additional constraints on firing dates of transitions.

Embedding of a schedule in some process of $\mathcal{N}$ only guarantees *logical realizability*: the probabilty of a time process in which, at least, one event is forced to occur at a precise date is 0. We use transient analysis of STPNs [14] to compute the probability that a schedule is realized by a symbolic time process of $\mathcal{N}$ up to imprecision of δ. This allows to compute the probability that $\mathcal{N}$ realizes $S \pm \delta$ and define *probabilistic realizability* that holds if $\mathcal{N}$ realizes $S \pm \delta$ with strictly positive probability.

The paper is organized as follows: Section 2 introduces schedules and our variant of stochastic time Petri nets with blocking semantics. Section 3 defines a notion of symbolic processes. Section 4 shows how to verify that a schedule is compatible with at least one process of the system and measure the probability of such realization. Due to lack of space, proofs and several technical details are omitted, but can be found in Appendix.

2 Schedules and Stochastic Time Petri Nets

A schedule describes causal dependencies among tasks, and timing constraints on their respective starting dates. Schedules are defined as decorated partial orders. We allow timing constraints among tasks that are not causally related.

Definition 1 (schedule). *A* schedule *over a finite alphabet $\mathcal{A}$ is a quadruple $S \triangleq \langle N, \to, \lambda, C\rangle$ where N is a set of nodes, $\to \subseteq N \times N$ is an acyclic precedence relation, $\lambda : N \to \mathcal{A}$ is a labeling of nodes, and $C : N \times N \nrightarrow \mathbb{Q}_{>0}$ is a partial function that associates a time constraint to pairs of nodes. A* dating function *for a schedule S is a function $d : N \to \mathbb{Q}_{\geq 0}$ that satisfies all constraints of C and $\to$: $\langle n, n'\rangle \in \to$ implies $d(n') \geq d(n)$, and $C(n, n') = x$ implies $d(n') - d(n) \geq x$.*

This model for schedules is inspired from [9, 10]. Intuitively, if $C(n, n') = x$, then n' cannot occur earlier than x time units after n, and if $\langle n, n'\rangle \in \to$, then n (causally) precedes n'. Constraints model the minimal times needed to perform tasks and initiate the next ones in production cells, the times needed for trains to move from a station to another, etc.

A schedule S is *consistent* if the graph $\langle N, \to \cup \{\langle n, n'\rangle \mid C(n, n') \text{ is defined}\}\rangle$ does not contain cycles. Obviously, consistent schedules admit at least one dating function. A frequent approach is to associate costs to dating functions and to find optimal functions that meet a schedule. A cost example is the earliest completion date. Optimizing this cost amounts to assigning to each node the earliest possible execution date. However, these optimal schedules are not the most probable ones. For the earliest completion date objective, if an event n occurs later than prescribed by d, then all its successors will also be delayed. In real systems running in an uncertain environment (e.g., with human interactions or influenced by weather conditions), tight timings are impossible to achieve. Finding a good schedule is hence a tradeof between maximization of an objective and of the likelihood to stay close to optimal realizations at runtime.

We want to check whether a consistent schedule S with its dating function d can be realized by a system. Systems are described with a variant of Petri nets with time and probabilities, namely stochastic time Petri nets [14]. We will show how to check that (S, d) is realizable by a net $\mathcal{N}$, and then how to measure the probability that (S, d) is realized by $\mathcal{N}$. Roughly speaking, an STPN is a time Petri net with distributions on firing times attached to transitions. As for Petri nets, the semantics of our model moves tokens from the preset of a transition to its postset. The time that must elapse between enabling of a transition and its firing is sampled according to the distribution attached to the transition. The major difference with [14] is that we equip our STPNs with a blocking semantics. Due to blockings, stochastic time Petri nets are *safe* (1-bounded). This semantics restriction is justified by the nature of the systems we address: in production chains, places symbolize tools that can process only one item at a time. Similarly, when modeling train networks, an important security requirement is that two trains cannot occupy the same track portion, which can only be implemented with such a blocking semantics. Standard time or stochastic Petri nets do not assume a priori bounds on their markings. A way to force boundedness is to add complementary places to the original Petri net and then study it under the usual semantics [8]. However, this trick does not allow to preserve time and probability issues in STPNs with blockings.

For simplicity, we only consider closed intervals of the form $[a, b]$ with $a < b$ and open intervals of the form $[a, +\infty)$. A *probability density function* (PDF) for a

continuous random variable X is a function $f_X : \mathbb{R} \to [0,1]$ that describes the relative likelihood for X to take a given value. Its integral over the domain of X is equal to 1. A *cumulative distribution function* (CDF) $F_X : \mathbb{R} \to [0,1]$ for X describes the probability for X to take a value less than or equal to a chosen value. We denote by Σ_{pdf} the set of PDFs, Σ_{cdf} the set of CDFs, and we will only consider PDFs for variables representing durations, i.e., whose domains are included in $\mathbb{R}_{\geq 0}$. The CDF of X can be computed from its PDF as $F_X(x) = \int_0^x f_X(y)\,dy$. Given a finite set of places P, a *marking* is a function that assigns 0 or 1 token to each place $p \in P$.

Definition 2 (stochastic time Petri net). *A stochastic time Petri net (STPN for short) is a tuple $\mathcal{N} = \langle P, T, {}^\bullet(), ()^\bullet, m_0, \mathsf{eft}, \mathsf{lft}, \mathcal{F}, \mathcal{W} \rangle$ where P is a finite set of places; T is a finite set of transitions; ${}^\bullet() : T \to 2^P$ and $()^\bullet : T \to 2^P$ are pre and post conditions depicting from which places transitions consume tokens, and to which places they output produced tokens; $m_0 : P \to \{0,1\}$ is the initial marking of the net; $\mathsf{eft} : T \to \mathbb{Q}_{\geq 0}$ and $\mathsf{lft} : T \to \mathbb{Q}_{\geq 0} \cup \{+\infty\}$ respectively specify the minimum and maximum time-to-fire that can be sampled for each transition; and $\mathcal{F} : T \to \Sigma_{\mathsf{pdf}}$ and $\mathcal{W} : T \to \mathbb{R}_{>0}$ respectively associate a PDF and a strictly positive weight to each transition.*

For a given place or transition $x \in P \cup T$, ${}^\bullet x$ will be called the preset of x, and $x^\bullet$ the postset of x. We denote by f_t the PDF $\mathcal{F}(t)$, and by F_t the associated CDF. To be consistent, we assume that for every $t \in T$, the support of f_t is $[\mathsf{eft}(t), \mathsf{lft}(t)]$. This syntax of STPNs is similar to [14], but we equip them with a blocking semantics, defining sequences of discrete transition firings, and timed moves. We will say that a transition t is *enabled* by a marking m iff $\forall p \in {}^\bullet t, m(p) = 1$. We denote by $\mathsf{enab}(m)$ the set of transitions enabled by a marking m.

For a given marking m and a set of places P', we will denote by $m - P'$ the marking that assigns $m(p)$ tokens to each place $p \in P \setminus P'$, and $m(p) - 1$ tokens to each place $p \in P'$. Similarly, we will denote by $m + P'$ the marking that assigns $m(p)$ tokens to each place $p \in P \setminus P'$, and $m(p) + 1$ tokens to each place $p \in P'$. Firing a transition t is done in two steps and consists in: (1) consuming tokens from ${}^\bullet t$, leading to a *temporary* marking $m_{\text{tmp}} = m - {}^\bullet t$, then (2) producing tokens in $t^\bullet$, leading to a marking $m' = m_{\text{tmp}} + t^\bullet$.

The blocking semantics can be informally described as follows. A variable τ_t is attached to each transition t of the net. As soon as the preset of a transition t is marked, τ_t is set to a random value ζ_t (called the *time-to-fire* of t, or TTF for short) sampled from $[\mathsf{eft}(t), \mathsf{lft}(t)]$ according to f_t. We will assume that every CDF F_t is strictly increasing on $[\mathsf{eft}(t), \mathsf{lft}(t)]$, which allows to use *inverse transform sampling* to choose a value (see for instance [18] for details). Intuitively, this TTF represents a duration that *must* elapse before firing t once t is enabled. The value of τ_t then decreases as time elapses but cannot reach negative values. When the TTF of a transition t reaches 0, then if $t^\bullet$ is empty in m_{tmp}, t becomes *urgent* and has to fire unless another transition with TTF 0 and empty postset fires; otherwise (if $t^\bullet$ is not empty in m_{tmp}), t becomes *blocked*: its TTF stops decreasing and keeps a null value, and its firing is delayed until the postset of t becomes empty; in the meantime, t can

be disabled by the firing of another transition. The semantics of STPNs is *urgent*: time can elapse by durations that do not exceed the minimal remaining TTF of enabled transitions that are not blocked. If more than one transition is urgent, then the transition that fires is randomly chosen according to the respective weights of urgent transitions. We formalize the semantics of STPNs in terms of discrete and timed moves between *configurations* that memorize markings and TTFs for enabled transitions.

Definition 3 (configuration of an STPN). *A* configuration of an STPN *is a pair* $\mathcal{C}_N \triangleq \langle m, \tau \rangle$ *where* m *is a marking, and* $\tau : \mathsf{enab}(m) \to \mathbb{R}_{\geq 0}$ *is a function that assigns a positive real TTF* $\tau_i \triangleq \tau(t_i)$ *to each transition* t_i *enabled by* m*. A transition* t *is* enabled *in a configuration* $\langle m, \tau \rangle$ *iff it is enabled by* m*.*

Definition 4 (firable and blocked transitions). *A transition* t *is* firable *in* $\langle m, \tau \rangle$ *iff it is enabled by* m*, all places of its postset are empty in* $m - {}^\bullet t$*, and its TTF is equal to* 0*. We denote by* $\mathsf{fira}(\langle m, \tau \rangle)$ *the set of firable transitions of* $\langle m, \tau \rangle$*. A transition* t *is* blocked *in* $\langle m, \tau \rangle$ *iff it is enabled by* m*, its TTF* $\tau(t)$ *is equal to* 0*, and one of its postset places is marked in* $m - {}^\bullet t$*. We denote by* $\mathsf{blck}(\langle m, \tau \rangle)$ *the set of blocked transitions in* $\langle m, \tau \rangle$*.*

Timed moves: A *timed move* $\langle m, \tau \rangle \xrightarrow{\delta} \langle m, \tau' \rangle$ lets a strictly positive duration δ elapse. To be allowed, δ must be smaller or equal to all TTFs of transitions enabled by m and not yet blocked. The new configuration $\langle m, \tau' \rangle$ decreases TTFs of every enabled and non-blocked transition t by δ time units ($\tau'(t) = \tau(t) - \delta$). Blocked transitions keep a TTF of 0, and m remains unchanged.

Discrete moves: A discrete move $\langle m, \tau \rangle \xrightarrow{t} \langle m', \tau' \rangle$ consists in firing a transition t from a configuration $\langle m, \tau \rangle$ to reach a configuration $\langle m', \tau' \rangle$. Discrete moves change the marking of a configuration, and sample new times to fire for transitions that become enabled after the move. To define the semantics of discrete moves, we first introduce newly enabled transitions.

Definition 5 (newly enabled transitions). *Let* m *be a marking and* t *a transition enabled by* m*. A transition* t' *is* newly enabled *after firing of* t *from* m *iff it is enabled by marking* $m' = (m - {}^\bullet t) + t^\bullet$ *and either it is not enabled by* $m - {}^\bullet t$ *or* $t' = t$*. We denote by* $\mathsf{newl}(m, t) \triangleq \mathsf{enab}(m') \cap (\{t\} \cup (T \setminus \mathsf{enab}(m - {}^\bullet t)))$ *the set of transitions newly enabled by firing of* t *from* m*.*

The transition t fired during a discrete move is chosen among all firable transitions of $\langle m, \tau \rangle$. The new marking reached is $m' = (m - {}^\bullet t) + t^\bullet$, and τ' is obtained by sampling a new TTF for every newly enabled transition and keeping unchanged TTFs of transitions that were enabled by m and are still enabled by m'.

For completeness, we give operational rules for moves of STPNs in Appendix A. We will write $\langle m, \tau \rangle \to \langle m', \tau' \rangle$ iff there exists a timed or discrete move from $\langle m, \tau \rangle$ to $\langle m', \tau' \rangle$, and $\langle m, \tau \rangle \xrightarrow{*} \langle m', \tau' \rangle$ iff there exists a sequence of moves leading from $\langle m, \tau \rangle$ to $\langle m', \tau' \rangle$. An initial configuration for $\mathcal{N}$ is a configuration $\langle m_0, \tau_0 \rangle$ where τ_0 attaches a sampled TTF to each transition enabled by m_0.

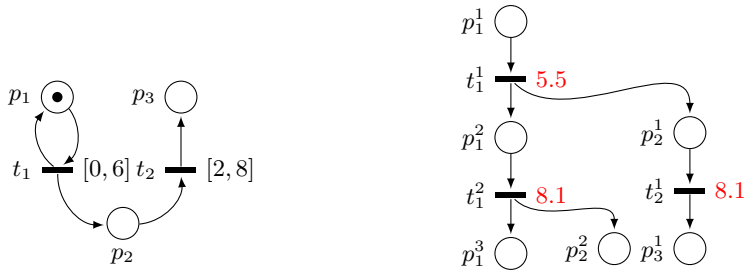

Fig. 1: a) An example STPN $\mathcal{N}_1$ and b) a time process of $\mathcal{N}_1$

Consider the STPN $\mathcal{N}_1$ of Figure 1, and suppose that $\mathcal{N}_1$ is in configuration $\langle m, \tau \rangle$, with $m(p_1) = 1$, $m(p_2) = m(p_3) = 0$, $\tau(t_1) = 5.5$. From this configuration, one can let 5.5 time units elapse, and then fire t_1. After this firing, the STPN reaches marking m' with $m'(p_1) = m'(p_2) = 1$, $m'(p_3) = 0$. New TTFs d_1, d_2 are sampled for t_1, t_2, leading to a configuration $\langle m', \tau' \rangle$, where $\tau'(t_1) = d_1$ and $\tau'(t_2) = d_2$. Let us suppose that $d_1 = 1.5$ and $d_2 = 2.6$. Then one can let 1.5 time units elapse, but after this timed move, transition t_1 cannot fire, as place p_2 contains a token. $\mathcal{N}_1$ is hence in a configuration $\langle m', \tau'' \rangle$, where $\tau''(t_1) = 0$ and $\tau''(t_2) = 1.1$, where t_1 is blocked. After letting 1.1 time units elapse, transition t_2 can fire, leading to marking $m''(p_1) = m''(p_3) = 1, m''(p_2) = 0$, and transition t_1 immediately fires at the same date.

Let us now assign probabilities to STPN moves. Randomness in STPNs semantics mainly comes from sampling of TTFs. However, when several transitions are firable from a configuration, weights are used to determine the probability for a transition to fire first. Timed moves are achieved with probability 1: once TTFs are set, there is a unique configuration allowing discrete moves. In a move $\langle m, \tau \rangle \xrightarrow{t} \langle m', \tau' \rangle$, m' is built deterministically, but τ' is obtained by sampling a random value ζ_t for each newly enabled transition t. Each ζ_t is chosen according to CDF F_t, i.e., we have $\mathbb{P}(\zeta_t \leq x) = F_t(x)$ (for any $x \in [\mathsf{eft}(t), \mathsf{lft}(t)]$). When more than one transition is firable from $\langle m, \tau \rangle$, the transition that fires is randomly chosen, and each transition t_k in $\mathsf{fira}(\langle m, \tau \rangle)$ has a probability to fire $\mathbb{P}_{\mathsf{fire}}(t_k) = \mathcal{W}(t_k) / \sum_{t_i \in \mathsf{fira}(\langle m, \tau \rangle)} \mathcal{W}(t_i)$. Note that, as STPNs have continuous probability laws, the probability to choose a particular value ζ_t is the probability of a point in a continuous domain and is hence null. However, in the next sections, we will consider probabilities for events of the form $\tau(t_i) \leq \tau(t_j)$, which may have strictly positive probability.

STPNs define sequences of moves $\rho = (\langle m, \tau \rangle \xrightarrow{e_i} \langle m', \tau' \rangle)_{i \in 1 \dots k}$, where e_i is a transition name in discrete moves and a real value in timed moves. Leaving probabilities for the moment, STPNs can also be seen as generators for timed words over T. A *timed word* over an alphabet $\mathcal{A}$ is a sequence $\langle a_1, d_1 \rangle \dots \langle a_q, d_q \rangle \dots$ in $(\mathcal{A} \times \mathbb{R}_{\geq 0})^*$, where each a_i is a letter from $\mathcal{A}$, each d_i defines the occurrence date of a_i, and $d_1, \dots, d_q$ is an increasing sequence of positive real numbers. Letting $i_1, \dots, i_q$ denote the indices of discrete moves in ρ, we can build a timed word $u_\rho = \langle a_{i_1}, d_1 \rangle \dots \langle a_{i_q}, d_q \rangle \in (T \times \mathbb{R}_{\geq 0})^q$ that associates dates to transitions firings, where $d_1 = \sum_{j < i_1} e_j$, and $d_j = d_{j-1} + \sum_{i_{j-1} < k < i_j} e_k$ for $j \in \{2, \dots, q\}$. The *timed*

language of an STPN $\mathcal{N}$ is the set $\mathcal{L}(\mathcal{N})$ of timed words associated with its sequences of moves. We denote by $\mathcal{L}_{\leq D}(\mathcal{N})$ the set of words in $\mathcal{L}(\mathcal{N})$ whose maximal date is lower than D.

As already highlighted in [2] for TPNs, timed languages give a sequential and interleaved view for executions of inherently concurrent models. A non-interleaved semantics can be defined using *time processes*, i.e., causal nets equipped with dating functions. We recall that *causal nets* are finite acyclic nets of the form $CN \triangleq \langle B, E, {}^\bullet(), ()^\bullet \rangle$, where for every $b \in B$, $|b^\bullet| \leq 1$ and $|{}^\bullet b| \leq 1$. Intuitively, a causal net contains no conflict (pairs of transition with common places in their presets) nor place receiving tokens from more than one transition.

Definition 6 (time process). *A* time process *is a tuple $TP \triangleq \langle CN, \theta \rangle$, where $CN \triangleq \langle B, E, {}^\bullet(), ()^\bullet \rangle$ is a causal net, and $\theta : E \to \mathbb{R}_{\geq 0}$ associates a positive real date to transitions of the net, and is such that $\forall e, e' \in E$ with ${e}^\bullet \cap {}^\bullet e' \neq \emptyset$ we have $\theta(e) \leq \theta(e')$. In time processes, places in B are called* conditions*, and transitions in E are called* events*. The* depth *of a time process is the maximal number of events along a path of the graph $\langle B \cup E, {}^\bullet() \cup ()^\bullet \rangle$. We will write $e \prec e'$ iff ${e}^\bullet \cap {}^\bullet e' \neq \varnothing$, and denote by $\preceq$ the transitive and reflexive closure of $\prec$.*

Intuitively, conditions in B represent occurrences of places fillings, and events in E are occurrences of transitions firings. Given an STPN $\mathcal{N}$, for every timed word $u = \langle a_1, d_1 \rangle \dots \langle a_n, d_n \rangle$ in $\mathcal{L}(\mathcal{N})$, we can compute a time process $TP_u = \langle B, E, {}^\bullet(), ()^\bullet, \theta \rangle$. The construction described below is the same as in [2]. It does not consider probabilities and, as the construction starts from an executable word, it does not have to handle blockings either. To differentiate occurrences of transitions firings, an event will be defined as a pair $e \triangleq \langle X, t \rangle$, where t is the transition whose firing is represented e and X is the set of conditions it consumes. Similarly, a condition is defined as a pair $b \triangleq \langle p, e \rangle$, where p is the place whose filling is represented by b, and e is the event whose occurrence created b.

We will denote by $\mathsf{tr}(e)$ the transition t attached to an event e, and by $\mathsf{pl}(b)$ the place p associated with a condition b. The flow relations are hence implicit: ${}^\bullet e = \{b \mid e = \langle X, t \rangle \wedge b \in X\}$, and similarly ${e}^\bullet = \{b \mid b = \langle p, e \rangle\}$, and for $b = \langle p, e \rangle$, ${}^\bullet b = e$ and ${b}^\bullet = \{e \in E \mid b \in {}^\bullet e\}$. We will then drop flow relations and simply refer to time processes as triples $TP \triangleq \langle B, E, \theta \rangle$.

The time process TP_u obtained from a timed word $u = \langle t_1, d_1 \rangle \langle t_2, d_2 \rangle \dots \langle t_k, d_k \rangle \in \mathcal{L}(\mathcal{N})$ is built inductively as follows. We assume a dummy initial event $\perp$ that initializes the initial contents of places according to m_0. We start from the initial process $TP_0 = \langle B_0, E_0, \theta_0 \rangle$ with a set of conditions $B_0 = \{(p, \perp) \mid p \in m_0\}$, a set of events $E_0 = \{\perp\}$, and a function $\theta_0 : \{\perp\} \to \{0\}$.

Let $TP_{u,i} = \langle B_i, E_i, \theta_i \rangle$ be the time process built after i steps for the prefix $\langle t_1, d_1 \rangle \dots \langle t_i, d_i \rangle$ of u, and let $\langle t, d_{i+1} \rangle$ be the $(i+1)^{th}$ entry of u. We denote by $\mathsf{last}(p, E_i, B_i)$ the last occurrence of place p in $TP_{u,i}$, i.e., the only condition $b = \langle p, e \rangle$ with an empty postset. Then, we have $E_{i+1} = E_i \cup \{e\}$, where $e = \langle t, X \rangle$ with $X = \{b \mid b = \mathsf{last}(p, E_i, B_i) \wedge p \in {}^\bullet t\}$ and $B_{i+1} = B_i \cup \{\langle p, e \rangle \mid p \in t^\bullet\}$. We also set $\theta(e) = d_{i+1}$. The construction ends with $TP_u = TP_{u,|u|}$.

Figure 1-b is an example of a time process for STPN $\mathcal{N}_1$. In this example, event t_i^j (resp. condition p_i^j) denotes the j^{th} occurrence of transition t_i (resp. place p_i). This time process corresponds to the time word $u = \langle t_1, 5.5\rangle\langle t_2, 8.1\rangle\langle t_1, 8.1\rangle \in \mathcal{L}(\mathcal{N}_1)$. It contains causal dependencies among transitions (e.g., from t_1^1 to t_2^1). Event t_1^2 cannot occur before t_2^1 as t_1 cannot fire as long as place p_2 is filled. However, this information is not explicit in the process. The timed language $\mathcal{L}(\mathcal{N})$ of a TPN can be reconstructed as the set of linearizations of its time processes. In these linearizations, ordering of events considers both causality and dates of events: e must precede $e' \neq e$ in a linearization of a process if $\theta(e) < \theta(e')$ or if $e \preceq e'$. With blocking semantics, some causality and time-preserving interleaving may not be a valid timed word of $\mathcal{L}(\mathcal{N})$: in the process of Figure 1-b, t_1^2 cannot occur before t_2^1, even if both transitions have the same date. To avoid wrong ordering among events with identical dates, one has to check whether the chosen ordering does not imply firing a transition that should be blocked in one of the reached configurations.

3 Unfolding of STPNs

A time process emphasizes concurrency but only gives a partial order view of a *single* timed word. Many time processes of $\mathcal{N}_1$ have the same structure as the process of Figure 1-b, but different dating functions. Indeed, there can be uncountably many time processes with identical structure, but different real dates. It is hence interesting to consider symbolic (time) processes, that define constraints on events dates instead of exact dates. Similarly, to avoid recomputing the structural part of each symbolic process, we will work with *unfoldings*, i.e., structures that contain all symbolic processes of an STPN, but factorize common prefixes. Symbolic unfoldings were already introduced for TPNs in [19] and later used in [6]. In this section, we show how to unfold STPNs with blockings and extract symbolic processes out of this unfolding. Our aim is to find the minimal structure that represents prefixes of all symbolic processes that embed a schedule of known duration. We show that if a system cannot execute arbitrary large sets of events without progressing time, unfolding an STPN up to some bounded depth is sufficient.

Definition 7 (time progress). *An STPN $\mathcal{N}$ guarantees* time progress *iff there exists $\delta \in \mathbb{Q}_{>0}$ such that $\forall t \in T, i \in \mathbb{N}$, and for every time word $u = \langle t_1, d_1\rangle \ldots \langle t^i, \theta_1\rangle \ldots \langle t^{i+1}, \theta_2\rangle \ldots \langle t_k, d_k\rangle \in \mathcal{L}(\mathcal{N})$ where t^i denotes the i^{th} occurrence of t, we have $\theta_2 - \theta_1 \geq \delta$.*

Time progress is close to non-Zenoness property, and is easily met (e.g., if no transition has an earliest firing time of 0). Any execution of duration Δ of an STPN that guarantees time progress is a sequence of at most $|T| \cdot \lceil \frac{\Delta}{\delta} \rceil$ transitions.

As in processes, unfoldings will contain occurrences of transitions firings (a set of events E), and occurrences of places fillings (a set of conditions B). We associate to each event $e \in E$ positive real valued variables $\mathsf{doe}(e)$, $\mathsf{dof}(e)$ and $\theta(e)$ that respectively define the enabling, firability and effective firing date of the occurrence of transition $\mathsf{tr}(e)$ represented by event e. Similarly, we associate to each condition b positive real valued variables $\mathsf{dob}(b)$ and $\mathsf{dod}(b)$ that respectively represent the date

of birth of the token in place $\mathsf{pl}(b)$, and the date at which the token in place $\mathsf{pl}(b)$ is consumed. We denote by $\mathsf{var}(E,B)$ the set of variables $\bigcup_{e\in E}\mathsf{doe}(e)\cup\mathsf{dof}(e)\cup\theta(e)\cup\bigcup_{b\in B}\mathsf{dob}(b)\cup\mathsf{dod}(b)$. A *constraint* over $\mathsf{var}(E,B)$ is a boolean combination of atoms of the form $x\bowtie y$, where $x\in\mathsf{var}(E,B)$, $\bowtie\in\{<,>,\leq,\geq\}$ and y is either a variable from $\mathsf{var}(E,B)$ or a constant value. A set of constraints C over a set of variables V is *satisfiable* iff there exists at least one valuation $v:V\to\mathbb{R}$ such that replacing each occurrence of each variable x by its valuation $v(x)$ yields a tautology. We denote by $SOL(C)$ the set of valuations that satisfy C.

Definition 8 (unfolding). *A* (structural) unfolding *of an STPN $\mathcal{N}$ is a pair $\mathcal{U}\triangleq\langle E,B\rangle$ where E is a set of events and B a set of conditions.*

Unfoldings can be seen as processes with branching. As for processes, each event $e\in E$ is a pair $e=\langle{}^\bullet e,\mathsf{tr}(e)\rangle$ where ${}^\bullet e\subseteq B$ is the set of predecessor conditions of e (the conditions needed for e to occur). A condition $b\in B$ is a pair $b\triangleq\langle{}^\bullet b,\mathsf{pl}(b)\rangle$ where ${}^\bullet b\subseteq E$ is the predecessor of b, i.e., the event that created condition b. We assume a dummy event $\perp$ that represents the origin of the initial conditions in an unfolding. Function ${}^\bullet()$, $()^\bullet$, $\mathsf{pl}()$ and $\mathsf{tr}()$ keep the same meaning as for time processes. The main change between processes and unfoldings is that conditions may have several successor events. Using relations $\prec$ and $\preceq$ as defined for processes, we define the *causal past* of an event $e\in E$ as $\uparrow e\triangleq\{e'\in E\mid e'\preceq e\}$. A set of events $E'\subseteq E$ is *causally closed* iff $\forall e\in E',\uparrow e\subseteq E'$. We also extend this notion to conditions. We say that two events e,e' are in *conflict*, and write $e\sharp' e$, iff ${}^\bullet e\cap{}^\bullet e'\neq\varnothing$. A set of events $E'\subseteq E$ is *conflict free* if it does not contain conflicting pairs of events. Similarly, we will say that two events e,e' are *competing* iff $\mathsf{tr}(e)^\bullet\cap\mathsf{tr}(e')^\bullet\neq\varnothing$. Intuitively, competing events try to fill the same place.

Definition 9 (pre-processes of an unfolding). *A* pre-process *of a finite unfolding $\mathcal{U}=\langle E,B\rangle$ is a pair $\langle E',B'\rangle$ such that $E'\subseteq E$ is maximal (i.e., there is no larger pre-process containing E',B'), causally closed and conflict free set of events, and $B'={}^\bullet E'\cup E'^\bullet$. We denote by $\mathcal{PE}(\mathcal{U})$ the set of pre-processes of $\mathcal{U}$.*

We say that a condition $b\in B$ is *maximal* in $\mathcal{U}=\langle E,B\rangle$ or in a pre-process of $\mathcal{U}$ when it has no successor event ($b^\bullet=\varnothing$), and denote the set of maximal conditions of B by $\mathsf{max}(B)$. As for time processes construction, given a finite pre-process $\langle E',B'\rangle\in\mathcal{PE}(\mathcal{U})$, and a place p of the considered STPN, we denote by $\mathsf{last}(p,E',B')$ the maximal occurrences of place p w.r.t. $\prec$ in $\langle E',B'\rangle$. A *cut* of a pre-process is an unordered set of conditions. We denote by $Cuts(E,B)$ the set of cuts of pre-process $\langle E,B\rangle$.

The unfolding of a net up to a fixed depth K is performed inductively, without considering time. We will then use this structure to find processes. Timing issues will be considered through addition of constraints on occurrence dates of events.

Structural unfolding: Following [13], we build inductively unfoldings $\mathcal{U}_0,\ldots,\mathcal{U}_K$. Each step k adds new events at depth k and their postset to the preceding unfolding $\mathcal{U}_{k-1}$. We start with the initial unfolding $\mathcal{U}_0\triangleq\langle\varnothing,B_0\rangle$, where $B_0=\{\langle\perp,p\rangle\mid p\in m_0\}$. Each induction step that builds $\mathcal{U}_{k+1}$ from $\mathcal{U}_k$ adds new events and conditions to $\mathcal{U}_k$ as follows. Letting $\mathcal{U}_k=\langle E_k,B_k\rangle$ be the unfolding obtained at step k, we have

$\mathcal{U}_{k+1} = \langle E_k \cup \hat{E}, B_k \cup \hat{B} \rangle$ where $\hat{E} \triangleq \{\langle B, t\rangle \in (2^{B_k} \times T) \setminus E_k \mid \exists \langle X, Y\rangle \in \mathcal{PE}(\mathcal{U}_k), B \subseteq Cuts(X, Y), {}^\bullet t = \mathsf{pl}(B)\}$, and $\hat{B} \triangleq \{\langle e, p\rangle \in \hat{E} \times T \mid e = \langle B, t\rangle \in \hat{E} \wedge p \in t^\bullet\}$. Intuitively, $\hat{E}$ adds an occurrence of a transition if its preset is contained in the set of conditions representing the last occurrences of places contained in some pre-process of $\mathcal{U}_k$, and $\hat{B}$ adds the conditions produced by $\hat{E}$.

The structural unfolding of an STPN does not consider timing issues nor blockings. Hence, an (untimed) pre-process of $\mathcal{PE}(\mathcal{U}_K)$ need not be the untimed version of a time process obtained from a word in $\mathcal{L}(\mathcal{N})$. Indeed, urgent transitions can forbid firing of other conflicting transitions. Similarly, blockings prevent an event from occurring as long as a condition in its postset is filled. They may even prevent events in a pre-process from being executed if a needed place is never freed. We will show later that, once constrained, time processes of $\mathcal{N}$ are only prefixes of pre-processes in $\mathcal{PE}(\mathcal{U}_K)$ with associated timing function. To introduce timing aspects, we now attach constraints on events and conditions of pre-processes as follows:

Constraints: Let $\mathcal{U}_K = \langle E_K, B_K\rangle$ be the unfolding of an STPN $\mathcal{N}$ up to depth K, and let $E \subseteq E_K$ be a conflict free and causally closed set of events, and $B = {}^\bullet E \cup E^\bullet$ (B is contained in B_K). We define $\Phi_{E,B}$ as the set of constraints attached to events and conditions in E, B, assuming that executions of $\mathcal{N}$ start at a fixed date d_0. Constraints must be set to guarantee that occurrence dates of events are compatible with the earliest and latest firing times of transitions in $\mathcal{N}$, and that urgency or blocking is never violated. Let us first define the constraints associated with each condition $b = \langle e, p\rangle$. Recalling that variable $\mathsf{dob}(b)$ represents the date at which condition b is created, $\Phi_{E,B}$ must impose that for every $b \in B_0$, $\mathsf{dob}(b) = d_0$.

For all other conditions $b = \langle e, p\rangle$, as the date of birth is exactly the occurrence date of e, we set $\mathsf{dob}(b) = \theta(e)$ for every $b = \langle e, p\rangle$. Thank to this equality, we need not use both variables $\theta(e)$ and $\mathsf{dob}(b)$ in our constraints. However, we will use both variables for readability purposes. Recall that $\mathsf{dod}(b)$ is a variable that designates the date at which a place is emptied by some transition firing, $\mathsf{dod}(b)$ is hence the occurrence date of an event that has b as predecessor. Within a conflict free set of events, this predecessor is unique. In the considered subset of conditions B, several conditions may represent fillings of the same place, and B can hence be partitioned into $B_1 \uplus B_2 \uplus \cdots \uplus B_{|P|}$, where conditions in B_i represent fillings of place p_i. Due to blocking semantics, all conditions in a particular subset $B_i = \{b_{i,1}, b_{i,2}, \ldots, b_{i,k}\}$ must have disjoint existence dates, that is for every $j, j' \in \{1, 2, \ldots, k\}$ with $j \neq j'$, the intersection between $[\mathsf{dob}(b_{i,j}), \mathsf{dod}(b_{i,j})]$ and $[\mathsf{dob}(b_{i,j'}), \mathsf{dod}(b_{i,j'})]$ is either empty, or limited to a single value. This constraint can be encoded by the disjunction

$$\mathsf{no\text{-}overlap}(b_{i,j}, b_{i,j'}) \triangleq \begin{cases} \mathsf{dod}(b_{i,j}) \leq \mathsf{dob}(b_{i,j'}) \vee \mathsf{dod}(b_{i,j'}) \leq \mathsf{dob}(b_{i,j}) & \text{if } {b_{i,j}}^\bullet \neq \varnothing \wedge {b_{i,j'}}^\bullet \neq \varnothing, \\ \mathsf{dod}(b_{i,j}) \leq \mathsf{dob}(b_{i,j'}) & \text{if } {b_{i,j}}^\bullet \neq \varnothing \wedge {b_{i,j'}}^\bullet = \varnothing, \\ \mathsf{dod}(b_{i,j'}) \leq \mathsf{dob}(b_{i,j}) & \text{otherwise.} \end{cases}$$

Note that if $b_j \preceq b_{j'}$, then the constraint among events and transitions immediately ensures that $\mathsf{dob}(b_{j,i}) \leq \mathsf{dod}(b_{j,i}) \leq \mathsf{dob}(b_{j',i}) \leq \mathsf{dod}(b_{j',i})$. However, we need to add

a consistency constraint for every pair of concurrent conditions $b_{i,j}, b_{i,j'}$ that belong to the same B_i. Hence, calling $I(b_{i,j}, E, B)$ the set of conditions that represent the same place as $b_{i,j}$ and are concurrent with $b_{i,j}$ in $\langle E, B\rangle$, we have to ensure the constraint $\mathsf{non\text{-}blocking}(b_{i,j}) \triangleq \bigwedge_{b_{i,j'} \in I(b_{i,j},E,B)} \mathsf{no\text{-}overlap}(b_{i,j}, b_{i,j'})$. In words, condition $b_{i,j}$ does not hold during the validity dates of any concurrent condition representing the same place. This means in particular that a time process of $\mathcal{N}$ cannot contain two maximal conditions with the same place.

Let us now consider the constraints attached to events. An event $e = \langle B, t\rangle$ is an occurrence of a firing of transition t that needs conditions in B to be fulfilled to become enabled. Calling $\mathsf{doe}(e)$ the date of enabling of e, we necessarily have $\mathsf{doe}(e) = \max\{\mathsf{dob}(b) \mid b \in B\}$. Event e is firable at least $\mathsf{eft}(t)$ time units, and at most $\mathsf{lft}(t)$ time units after being enabled. We hence have $\mathsf{doe}(e) + \mathsf{eft}(t) \leq \mathsf{dof}(e) \leq \mathsf{doe}(e) + \mathsf{lft}(t)$. However, execution of e does not always occur immediately when e is firable. Execution of e occurs after e is firable, as soon as the places filled by e are empty, i.e., e occurs at a date $\theta(e)$ that guarantees that no place in $t^\bullet$ is occupied. This is guaranteed by attaching to every event e the constraints $\theta(e) = \mathsf{dob}(b_1), \theta(e) = \mathsf{dob}(b_2), \ldots, \theta(e) = \mathsf{dob}(b_k)$, where $\{b_1, b_2, \ldots b_k\} = e^\bullet$, and constraints $\mathsf{non\text{-}blocking}(b_1), \mathsf{non\text{-}blocking}(b_2), \ldots, \mathsf{non\text{-}blocking}(b_k)$. Last, as semantics of STPN is urgent, once firable, e has to fire at the earliest possible date. This is encoded by the constraint $\theta(e) = \min\{x \in \mathbb{R}_{\geq 0} \mid x \notin]\mathsf{dob}(b), \mathsf{dod}(b)[$ for some $b \in \bigcup I(b_i) \wedge x \geq \mathsf{dof}(e)\}$. Figure 2 shows the effect of blocking and possible free firing dates for some event with a condition b in its postset. Horizontal lines represent real lines, and intervals values in interval $[\mathsf{dob}(b_i), \mathsf{dod}(b_i)]$ for $i \in 0, 1, 2$. Suppose that $I(b) = \{b_0, b_1, b_2\}$. Then $[\mathsf{dob}(b), \mathsf{dod}(b)]$ have to be fully inscribed in one of these thick segments. An event with b in its postset can occur only at dates contained in these thick segments.

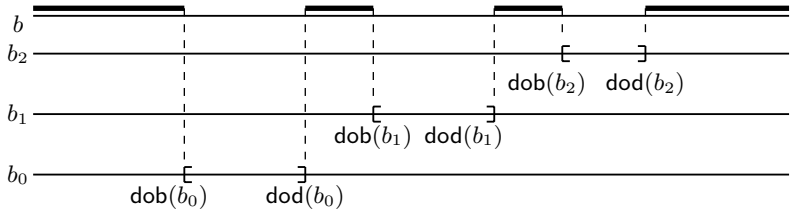

Fig. 2: Constraints on dates of birth of tokens in a shared place.

Written differently,

$$\theta(e) = \begin{cases} \mathsf{dof}(e) \text{ if } \bigwedge_{b \in I(b_1) \cup \ldots I(b_k)} \mathsf{dof}(e) \leq \mathsf{dob}(b), \text{ and} \\ \min\{\mathsf{dod}(b) \mid \forall b' \in \bigcup_{b_i \in e^\bullet} I(b_i), \mathsf{dod}(b) \notin]\mathsf{dob}(b'), \mathsf{dod}(b')[\} \text{ otherwise.} \end{cases}$$

This formula can be translated in boolean combinations of inequalities over variables of $\mathsf{var}(E, B)$. Similarly, event $e = \langle B, t\rangle$ must occur before all its conflicting events. If an event e' in conflict with e is executed, at least one condition in B is consumed, and e cannot occur in a time process containing e'. We hence need the additional constraint $\bigwedge_{e' \sharp e} \mathsf{notMoreUrg}(e, e')$ to guarantee that there exists no other event

that is forced to occur before e due to urgency. We define $\mathsf{notMoreUrg}(e, e')$ as the following constraint:

$$\mathsf{notMoreUrg}(e, e') \triangleq \theta(e) \geq \mathsf{doe}(e') + \mathsf{lft}(\mathsf{tr}(e')) \Rightarrow \mathsf{tiled}(e, e') \vee \bigvee_{e''||e} \mathsf{preempts}(e', e'')$$

where $\mathsf{tiled}(e, e') \triangleq \mathsf{free}(e') \cap [\mathsf{doe}(e') + \mathsf{lft}(\mathsf{tr}(e')), \theta(e)] = \varnothing$, $e''||e$ refers to event that are concurrent with e in the considered set of events, $\mathsf{free}(e') = \mathbb{R}_{\geq 0} \setminus \{[\mathsf{dob}(b), \mathsf{dod}(b)] \mid \exists b' \in e'^{\bullet}, b \in I(b')\}$ is the set of intervals in which places attached to conditions in $e'^{\bullet}$ are empty, and $\mathsf{preempts}(e', e'') \triangleq \theta(e'') \leq \min(]\mathsf{doe}(e') + \mathsf{lft}(\mathsf{tr}(e')), \theta(e)[\cap \mathsf{free}(e'))$ means e'' disabled e' by consuming a condition in ${}^{\bullet}e''$.

Constraint $\mathsf{notMoreUrg}(e, e')$ means that if e' is in conflict with e, then at least one condition in ${}^{\bullet}e'$ is consumed before e' can fire, or if e' becomes firable before e fires, the urgent firing of e' is delayed by blockings so that e can occur. As for constraint attached to blockings, $\mathsf{notMoreUrg}(e, e')$ can be expressed as a boolean combination of inequalities. One can also notice that $\mathsf{notMoreUrg}(e, e')$ can be expressed without referring to variables attached to event e' nor $e'^{\bullet}$, as $\mathsf{doe}(e') = \max_{b_i \in {}^{\bullet}e'} \mathsf{dob}(b_i)$ and the intersection of $I(b)$ and $e'^{\bullet}$ is void.

For causally closed sets of events and conditions $E \cup B$ contained in some pre-process of $\mathcal{U}_K$, the constraint $\Phi_{E,B}$ applying on events and conditions of $E \cup B$ is now defined as $\Phi_{E,B} = \bigwedge_{x \in E \cup B} \Phi_{E,B}(x)$ where:

$$\forall b \in B, \Phi_{E,B}(b) = \mathsf{non\text{-}blocking}(b) \wedge \begin{cases} \mathsf{dob}(b) = d_0 \text{ if } b \in B_0, \text{ and } b \text{ is maximal,} \\ \mathsf{dob}(b) = d_0 \wedge \mathsf{dob}(b) \leq \mathsf{dod}(b) \text{ if } b \in B_0, \\ \mathsf{dob}(b) = \theta({}^{\bullet}b) \text{ if } b \notin B_0 \text{ and } b \text{ is maximal,} \\ \mathsf{dob}(b) = \theta({}^{\bullet}b) \wedge \mathsf{dob}(b) \leq \mathsf{dod}(b) \text{ otherwise.} \end{cases}$$

$$\forall e \in E, \Phi_{E,B}(e) = \begin{cases} & \mathsf{doe}(e) = \max_{b \in {}^{\bullet}e} \mathsf{dob}(b) \\ \wedge & \mathsf{doe}(e) + \mathsf{eft}(\mathsf{tr}(e)) \leq \mathsf{dof}(e) \leq \mathsf{doe}(e) + \mathsf{lft}(\mathsf{tr}(e)) \\ \wedge & \mathsf{dof}(e) \leq \theta(e) \wedge \bigwedge_{b \in {}^{\bullet}e} \mathsf{dod}(b) = \theta(e) \\ \wedge & \bigwedge_{b \in e^{\bullet}} \theta(e) = \mathsf{dob}(b) \\ \wedge & \bigwedge_{e' \sharp e} \mathsf{notMoreUrg}(e, e') \end{cases}$$

We can now define symbolic processes, and show how instantiation of their variables define time processes of $\mathcal{N}$. Roughly speaking, a symbolic process is a prefix of a pre-process of $\mathcal{U}_K$ (it is hence a a causal net) decorated with a *satisfiable* set of constraints on occurrence dates of events. Before formalizing symbolic processes, let us highlight three important remarks. **Remark 1:** an unfolding up to depth K misses some constraints on occurrence dates of events due to blockings by conditions that do not belong to $\mathcal{U}_K$ but would appear in some larger unfolding $\mathcal{U}_{K'}$, with $K' > K$. We will however show (Prop. 1 and 2 that with time progress assumption, unfolding $\mathcal{N}$ up to a sufficient depth guarantees that all constraints regarding events with $\theta(e) \leq D$ are considered. This allows to define symbolic processes representing the time processes of $\mathcal{N}$ that are executable in less than D time units. **Remark 2:** unfoldings consider depth of events, and not their dates. Hence if a process contains an event e occurring at some date greater than d, and another event e' that belongs to the same pre-process and becomes urgent before date d, then e' must belong to

the process, even if it lays at a greater depth than e. **Remark 3:** Every pre-process $\langle E, B\rangle$ of $\mathcal{U}_K$ equipped with constraint $\Phi_{E,B}$ is not necessarily a symbolic process. Indeed, some events in a pre-process might be competing for the same resource. Consider for instance the net of Figure 3-a). Its unfolding is represented in b), and two of its (symbolic) processes in c) and d). For readability, we have omitted constraints. One can however notice that there exists no symbolic process containing two occurrences of transition t_3, because conditions b_5 and b_6 are maximal and represent the same place p_2.

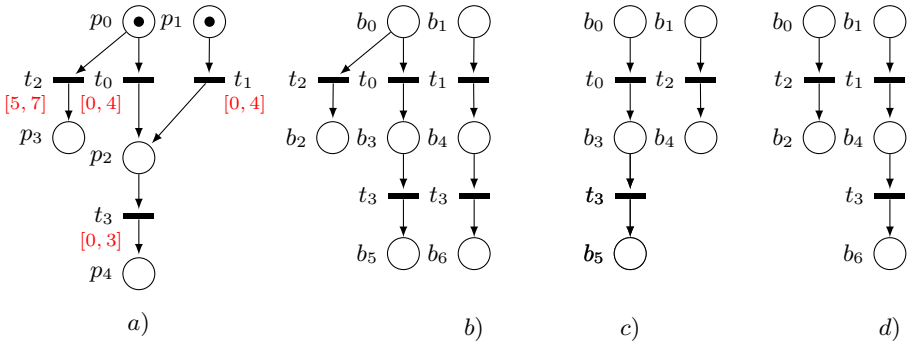

Fig. 3: An STPN with conflicts and blockings a), its symbolic unfolding b), and two of its symbolic processes c) and d).

Definition 10 (prefixes of an unfolding). *Let $SPP = \langle E, B\rangle$ be a pre-process of $\mathcal{U}_K$. A* symbolic prefix *of SPP is a triple $\langle E', B', \Phi_{E',B'}\rangle$ where $E' \subseteq E$ is a causally closed set of elements contained in E, and $B' = {}^\bullet E' \cup E'^\bullet$.*

Note that symbolic prefixes define causally closed parts of pre-processes, decorated with constraints inherited from the unfolding $\mathcal{U}_K$, that may not be satisfiable.

Definition 11 (symbolic processes). *A* symbolic process *of $\mathcal{U}_K$ is a triple $\mathcal{E}^s = \langle E', B', \Phi_{E,B}\rangle$ where $\langle E', B'\rangle$ is a symbolic prefix of some pre-process $PP = \langle E, B\rangle$ of $\mathcal{U}_K$, $\Phi_{E,B}$ is satisfiable, and E' is* maximal *w.r.t. urgent events firing in PP, that is for every $f \in B'^\bullet \cap E$, and letting $C_f = \mathsf{pl}^{-1}(f^\bullet) \cap B'$ denote the set of conditions whose place appears in the postset of e, the following constraint is not satisfiable.*

$$\Phi_{\max}(f) \triangleq \begin{cases} \Phi_{E',B'} \\ \wedge\ \theta(f) \leq \max_{e' \in E'} \theta(e') \ \text{(f fires before the last event in E')} \\ \wedge\ \mathsf{eft}(f) + \max_{b \in {}^\bullet f} \mathsf{dob}(b) \leq \theta(f) \ \text{(f is urgent)} \\ \wedge\ \bigvee_{X \in 2^{C_f}} \max_{x \in X} \mathsf{dod}(x) \leq \theta(f) \leq \min_{x \in C_f \setminus X} \mathsf{dob}(x) \\ \quad \text{(f is not blocked for the whole duration of the process)} \end{cases}$$

Intuitively, $\Phi_{\max}(f)$ indicates that the event f that is not in the symbolic process becomes urgent, is not blocked by conditions in B', and has to fire before the execution of the last event in E'. If this constraint is satisfiable, then f should appear in the process.

A crux in the construction of symbolic processes of $\mathcal{U}_K$ is to find appropriate maximal and causally closed sets of events with satisfiable constraints. This can be costly: as exemplified by the example of Figure 3, satisfiability of constraints is not monotonous: the constraints for processes in Fig 3$-c$) and d) are satisfiable. However, adding one occurrence of transition t_3 yields unsatisfiable constraints. Satisfiability of a prefix of size n hence does not imply satisfiability of a larger prefix of size $n+1$. The converse implication is also false: when constraints of a prefix of size n are not satisfiable, appending new events may introduce new blockings and delays urgent transitions, yielding satisfiability of a constraint on a prefix of size $n+1$. This means that satisfiability of constraints cannot be a criterion to stop unfolding.

Definition 12 (executions of symbolic processes). *Let $\mathcal{E}^s = \langle E, B, \Phi \rangle$ be a symbolic process of an unfolding $\mathcal{U}_K$. An* execution *of $\mathcal{E}^s$ is a time process $TP = \langle E, B, \theta \rangle$ where θ is a solution for Φ. For a chosen θ, we denote by $\mathcal{E}^s_\theta = \langle E, B, \theta \rangle$ the time process obtained from $\mathcal{E}^s$. $TP = \langle E, B, \theta \rangle$ is a* time process *of $\mathcal{U}_K$ if there exists a symbolic process $\mathcal{E}^s = \langle E, B, \Phi \rangle$ of $\mathcal{U}_K$ s.t. TP is an execution of $\mathcal{E}^s$.*

Informally, symbolic pre-processes select maximal conflict-free sets of events in an unfolding. Symbolic processes extract executable prefixes from symbolic pre-processes, and executions attach dates to events of symbolic processes to obtain time processes. In the rest of the paper, we respectively denote by $\mathcal{E}^s(\mathcal{U}_K)$ and by $\mathcal{E}(\mathcal{U}_K)$ the set of symbolic processes and time processes of $\mathcal{U}_K$.

We can now show that upon time progress hypothesis, unfoldings and their symbolic processes capture the semantics of STPNs with blockings. Given an STPN that guarantees time progress with a minimal elapsing of δ time units between successive occurrences of every transition, and given a maximal date D, we want to build an unfolding $\mathcal{U}^D$ of $\mathcal{N}$ that contains all events that might be executed before D, but also all places and events which may impact firing dates of these events. We can show that $\mathcal{U}^D$ is finite and its processes are of depth $H = \lceil \frac{D-d_0}{\delta} \rceil \cdot |T|$ at most.

Let $b = \langle e, p \rangle$ be a condition of an unfolding $\mathcal{U}_n$ obtained at step n. We denote by $\mathsf{block}(b)$ the set of conditions that may occur in the same process as b, represent the same place, and are not predecessors or successors of b in any unfolding $\mathcal{U}_{n+k}$ obtained from $\mathcal{U}_n$.

Clearly, dates of birth and death of conditions in $\mathsf{block}(b)$ may influence the date of birth and death of b, or even prevent b from appearing in the same process as some conditions in $\mathsf{block}(b)$. However, in general, $\mathsf{block}(b)$ need not be finite, and at step n, $\mathsf{block}(b)$ is not fully contained in a pre-process of $\mathcal{U}_n$. Fortunately, upon time progress assumption, we can show that elements of $\mathsf{block}(b)$ that can influence $\mathsf{dob}(b)$ appear.

Proposition 1. *Let $\mathcal{N}$ be a STPN guaranteeing time progress of δ time units (between consecutive occurrences of each transition). For every date $D \in \mathbb{R}_{\geq 0}$ and condition b in an unfolding $\mathcal{U}_n$, there exists $K \geq n$ s.t. $\{b' \in \mathsf{block}(b) \mid \mathsf{dob}(b') \leq D\}$ is contained in $\mathcal{U}_K$.*

This proposition means that if some event cannot occur at $\mathsf{dof}(e)$ due to a blocking, then one can discover all conditions that prevent this firing from occuring in a bounded extension of the current unfolding.

Proposition 2. *Let $\mathcal{N}$ be a STPN guaranteeing time progress of δ time units. The set of time processes executable by $\mathcal{N}$ in D time units are prefixes of time processes of $\mathcal{U}_K$, with $K = \lceil \frac{D}{\delta} \rceil \cdot |T|^2$ containing only events with date $\leq D$.*

4 Realizability of schedules

We can now describe a way to check whether a given schedule S, i.e., a high-level description of operations of a system and of their timing constraints can be realized by a system represented as a STPN $\mathcal{N}$ depicting low-level operations and distributions over possible delays between enabledness and firing of transitions. First of all, the connection between the high-level view of operations in S and the lowest level implementation provided by $\mathcal{N}$ is defined through a realization function.

Definition 13 (realization function). *A* realization function *for a schedule S and an STPN $\mathcal{N}$ is a map $r : \mathcal{A} \to 2^T$ that associates a subset of transitions from T to each letter of $\mathcal{A}$, and such that $\forall a \neq a' \in \mathcal{A}, r(a) \cap r(a') = \varnothing$.*

A realization function describes which low-level actions implement a high-level operation of a schedule. Each letter a from $\mathcal{A}$ can be interpreted as an operation performed through the firing of any transition from the subset of transitions $r(a)$. Allowing $r(a)$ to be a subset of T provides some flexibility in the definition of schedules: in a production cell, for example, a manufacturing step a for an item can be implemented by different processes on different machines. Similarly, in a train network, a departure of a train from a particular station in the schedule can correspond to several departures using different track portions, which is encoded with several transitions in a net. Realization functions hence relate actions in schedules to several transitions in a net. The condition on realization functions prevents ambiguity by enforcing each transition to appear at most once in the image of r. Note that $r(\mathcal{A}) \subseteq T$, that is the realization of a schedule may need many intermediate steps that are depicted in the low-level description of a system, but are not considered in the high-level view provided by a schedule. We will call transitions that belong to $r(\mathcal{A})$ *realizations* of $\mathcal{A}$.

Definition 14 (embedding, realizability). *Let $S = \langle N, \to, \lambda, C \rangle$ be a schedule, $\mathcal{E}^s = \langle E, B, \Phi \rangle$ be a symbolic process of $\mathcal{N}$ and $r : N \to T$ be a realization function. We say that S* embeds *into $\mathcal{E}^s$ (w.r.t. r and d) and write $S \hookrightarrow \mathcal{E}^s$ iff there exists an injective function $\psi : N \to E$ such that:*

$$\begin{cases} \forall n \in N, \mathsf{tr}\big(\psi(n)\big) \in r\big(\lambda(n)\big) \\ \forall \langle n, n' \rangle \in \to, \psi(n) \preceq \psi(n') \\ \nexists f \leq \psi(\min(n)), \mathsf{tr}(f) \in r(\mathcal{A}) \\ \forall e \leq f \leq g, e = \psi(n) \wedge g = \psi(n'') \wedge \mathsf{tr}(f) \in r(\mathcal{A}) \\ \qquad\qquad \Rightarrow \exists n', f = \psi(n') \wedge n \to^* n' \to^* n'' \end{cases}$$

Intuitively, S embeds in $\mathcal{E}^s$ iff there is a way to label every node n of S by a letter from $r(\lambda(n))$ and obtain a structure that is contained in some restriction of a prefix of $\mathcal{E}^s$ to events that are realizations of actions from $\mathcal{A}$ and to a subset of its causal ordering. This way, a process respects the ordering described in S, does not "forget" actions, and does not "insert" realizations that are not the image by ψ of any high-level operation between two mapped realizations, or before the image by ψ of minimal nodes in the schedule. Note that there can be several ways to embed S into a process of $\mathcal{N}$.

Definition 15 ((boolean) realizability). *Let d be a dating function for a schedule S, r be a realization function. S is* realizable *by $\mathcal{E}^s$ (w.r.t. r and d) iff there exists an embedding ψ from S to $\mathcal{E}^s$, and furthermore, $\Phi_{\psi,S,d} \triangleq \Phi \wedge \bigwedge_{n \in N} \theta\big(\psi(n)\big) = d(n)$ is satisfiable. S is* realizable *by $\mathcal{N}$ (w.r.t. r and d) iff there exists a symbolic process $\mathcal{E}^s$ such that S is* realizable *by $\mathcal{E}^s$.*

We write $\mathcal{E}^s \models S$ when S is realizable by $\mathcal{E}^s$, and $\mathcal{N} \models S$ when S is realizable by $\mathcal{N}$. Appendix D gives an algorithm to compute a set $\Psi_{S,\mathcal{E}^s}$ of embeddings of a schedule S in a process $\mathcal{E}^s$. Once $\Psi_{S,\mathcal{E}^s}$ is obtained, it remains to show that for at least one embedding $\psi \in \Psi_{S,\mathcal{E}^s}$, $\Phi_{\psi,S,d}$ is satisfiable to prove that S is realizable by $\mathcal{E}^s$. We can then compute the set of symbolic processes $\mathcal{E}^S \triangleq \{\mathcal{E}^s_0, \mathcal{E}^s_1, \ldots, \mathcal{E}^s_{N-1}\}$ of $\mathcal{U}_K$ that embed S and similarly for each $\mathcal{E}^s_i \in \mathcal{E}^S$ the set of possible embedding functions $\Psi_i \triangleq \{\psi_{i,0}, \psi_{i,1}, \ldots, \psi_{i,N_i-1}\}$ for which the constraints $\Phi_{\psi_{i,j},S,d}$ are satisfiable.

To illustrate the construction of unfoldings and of processes, let us consider the example of figure 4. This simple toy example depicts two train carousels that serve stations. Line 1 serves stations A, B and C, and line 2 serves stations A', B' and C'. Both lines share a common track portion between stations B, C and B', C', and line 1 uses two trains. A possible required schedule is that one train leaves every 10 time units from station A on line 1, starting from date 10, and one train leaves station C' every 10 time units, but starting from date 15. The leftmost picture shows the aspect of both lines and stations. The center picture is an STPN model for this example. We do not precise distributions, and focus on the structural unfolding, on the right of the example. One can notice on this picture that the topmost occurrence of place OK, that plays the role of a boolean flag in a critical section can be both consumed by occurrences t_1^1 and t_1^2 of transition t_1. However, the second occurrence t_1^2 has an occurrence B^2 of place B in its preset, produced by t_6^1. Due to blocking semantics, t_6^1 can fire only at dates greater that the date of death of the initial occurrence B^1 of B. As a consequence, no time process of the net can contain at the same time t_1^1 and t_1^2, even if these transitions are not in conflict.

Boolean satisfiability is not sufficient. Consider the net of Figure 5, and the two symbolic processes: one in which transition t_1 fires, and another one in which t_2 fires. The probability of the first process is the probability that a value v_1 sampled to assign a TTF for t_1 is smaller or equal to another value v_2 sampled independently to assign a TTF for t_2. Clearly, the probability that $v_1 \leq v_2$ is equal to the probabilty that $v_1 \in [0, 1]$. The probability of the second process is equal to the probability that $v_1 \geq v_2$, but the set of values allowing this inequality is restricted to a single

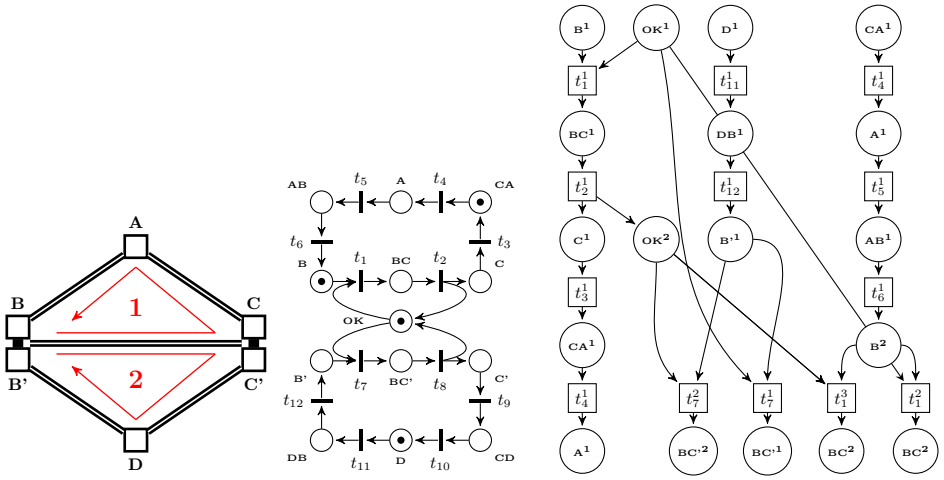

Fig. 4: A toy example : realizability of a partial schedule for two train carroussels with shared track portions.

point $v_1 = 1, v_2 = 1$. Conforming to continous probability distribution semantics, the probability of this point is equal to zero. A schedule composed of a single node n with date 1 such that $r(\lambda(n)) = \{t_2\}$ is realizable according to Definition 14, but with *null probability*. A more accurate notion of realizability is to require that schedules embed into symbolic processes of $\mathcal{U}_K$ with non-null probability.

This raises a second issue: requiring a schedule to be realized with an exact timing also leads to realizations with null probabilities. Consider the former example: a schedule composed of a single node n, a realization function r s.t. $r(\lambda(n)) = \{t_2\}$, and a dating function d s.t. $d(n) = 2$. Assign interval $[0, 3]$ to transition t_1 in the net of Figure 5-a) and interval $[1, 4]$ to transition t_2. The probability that t_2 fires from the initial marking is equal to the probability that $v_1 \geq v_2$, which is not null (we will explain later how to compute the probability of such domain and the joint probability of v_1, v_2), and is equal to the probability of the domain for values of v_1, v_2 depicted by the colored zone in Figure 5-b). However, within this continuous domain of possible values, the probability to fire t_2 exactly at precise date 2 as required by dating function d is still null. We hence consider realizability of a schedule up to some admissible delay α. The constraint to meet once an injection ψ is found for a schedule S into a symbolic process $\mathcal{E}^s$ hence becomes: $\Phi_{\psi,S,d\pm\alpha} = \Phi \wedge \bigwedge_{n\in N} \max(d(n) - \alpha, 0) \leq \theta(\psi(n)) \leq d(n) + \alpha$.

Definition 16 (probabilistic realizability). *A schedule with maximal date D is* realizable with non-null probability *iff there exists an embedding ψ of S into a symbolic process $\mathcal{E}^s$ of $\mathcal{U}_K$ s.t. $\mathbb{P}(\mathcal{E}^s \wedge \textsc{Sol}(\Phi_{\psi,S,d\pm\alpha})) > 0$.*

Intuitively, this definition requires that a symbolic process embeds S, and that the probability that this process is executed and satisfies all timing constraints imposed by the STPN and by the dating function is non-null. This probability can be evalu-

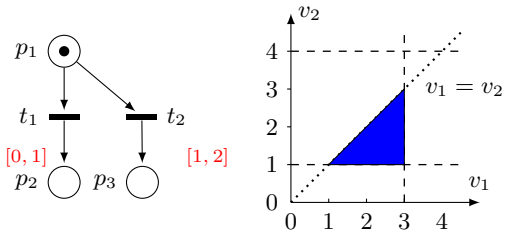

Fig. 5: *a*) An example STPN *b*) A domain for $\tau(t_1), \tau(t_2)$ allowing firing of t_2.

ated using a transient execution tree, as proposed in [14]. Roughly speaking, nodes of this tree are abstract representations of time domains for sampled values attached to enabled transitions (this is the usual notion of state class, already used in [3, 16] to analyze time Petri nets). In addition to state classes, transient tree nodes contain abstract representations of probability distributions. If the definition of distribution is appropriately chosen, for instance, using truncated sums of exponentials of the form

$$f(x) \quad = \quad \begin{cases} \sum c_k x^{a_k} e^{-\lambda_k x} \text{ if } x \in [a,b] \\ 0 \text{ otherwise} \end{cases}$$

then the distributions obtained by projection, multiplication, or variable elimination can still be encoded as sums of exponentials, and memorized using a finite set of parameters. The probability to fire a particular transition from a state and move to a successor node is computed as an integration over the time domains allowing this transition to fire first. The children of a node give a probabilistic distribution on possible classes of successor states. As time progress is guaranteeed in our model, a finite tree representing executions of an STPN or of one of its processes up to some bounded duration can be built. As explained in [14], the sum of probabilities attached to all paths of the tree can be used to compute the probability of some properties. In our case, the sum of probabilities of all paths that end with the execution of a chosen symbolic process gives the probability to realize this process. Details on the construction of this transient tree are provided in Appendix.

5 Conclusion

Related work : This work addresses realizability of partially ordered timed schedules by timed and stochastic concurrent systems with blocking semantics. Realizability in a timed setting has been addressed as a timed game problem [12], with a boolean answer. The objective in this work is to check whether a player in a timed game has a strategy to ensure satisfaction of a formula written in a timed logic called Metric Interval Temporal Logic. Brought back to the setting of realizability of schedules, the work of [12] can be used to answer a boolean realization question, by translating a schedule to a formula. However, the work of [12] lies in an interleaved setting: a sequential formula can not differentiate interleaved and concurrent actions. It does not address randomness in systems and hence can not quantify realizability. Scheduling of train networks was already addressed as a contraint satisfaction problems [9]. The input of the problem is given as an alternative graph (that can be seen as some kind of unfolding of a systems's behavior, decorated with time constraints. The algorithms provided by [9] returns an optimal schedule for the

next 2 hours of operation of a train network, using a branch and bound algorithm. However randomness was not addressed in this work. In [4], stability of timetables with a stochatics model is considered.

Realizability is also close to diagnosis question: given a log (a partial observation of a run of a system), and a model for this system, diagnosis aims at finding all possible runs of the model of the system which partial observation complies with the log. Diagnosis was addressed for stochastic Petri nets in [1]. In this work, the likelihood of a process that complies with an observation is evaluated, and time is a seen as a sequence of discrete instants. Diagnosis was addressed for timed Petri nets in [5], where unfolding of a timed Petri net is built to explain an observed log. [11] proposes temporal patterns called chronicles that represent possible evolutions of an observed system. A chronicle is a set of events, linked together by time constraints. The diagnosis proposed is to explain stream of time-stamped events as combinations of chronicles. assembling chronicles can be seen as some kind of timed unfolding. However, the run to explain is not a concurrent model as for schedules, and chronicles do not address randomness in timed systems.

Schedulability can also be seen as conformance of an expected behavior (the schedule) to an implementation (the Petri net model). Conformance was defined as a timed IOCO relation between timed input/output automata in [15]. More precisely, $\mathcal{A}_1$ IOCO $\mathcal{A}_2$ iff after some timed word, the set of outputs produced by $\mathcal{A}_1$ is included in the outputs produced by $\mathcal{A}_1$. This relation can not be verified in general (as inclusion of timed automata languages is not decidable), but can be tested. Boolean realizability can be seen as some kind of conformance test. Note however that TIOCO is defined for an interleaved timed model without probabilization of transitions.

Conclusion: The techniques described in this work first build an unfolding $\mathcal{U}_K$ up to a depth K that depends on the maximal date appearing in the schedule, finds symbolic processes of $\mathcal{U}_K$ that embed the schedule, and then checks that at least one of them has non-null probability. So far, we did not consider complexity issues. The size of an unfolding can grow exponentially w.r.t. its depth. Checking satisfiability of a set of constraints with disjunctions can also be costly. Satisfiability of constraints is not monotonous and hence cannot be used to stop unfolding. However, embedding verification and unfolding can be done jointly: one can stop a branch of unfolding as soon as a schedule does not embed in the pre-process on this branch. Most of the constraints presented in this paper can be simplified, and refer mainly to event variables. One can also notice that atoms in constraints are rather simple inequalities, which could simplify their verification. Computation of realization probability for processes can also be improved. We use the transient tree construction of [14], that builds a symbolic but *interleaved* representation of some processes. This is obviously very costly. We are currently investigating the possibility to attach probabilities to symbolic processes without computing an interleaved representation.

As future work, we would like to implement and improve this realizability verification framework, and use it as a basis to prove more properties. For instance, it might be interesting to prove that a schedule can be realized while ensuring that the

overall sum of delays w.r.t the expected schedule does not exceed some threshold. Another improvement would be to provide means to compute an exact value for the realization probability. We are also interested in the design of controllers that maximize the probability of realization.

References

1. A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, and C. Jard. Fault detection and diagnosis in distributed systems: An approach by partially stochastic Petri nets. *Discrete Event Dynamic Systems*, 8(2):203–231, 1998.
2. T. Aura and J. Lilius. Time processes for time Petri-nets. In *18th conference on Application and Theory of Petri Nets*, volume 1248 of *LNCS*, pages 136–155, 1997.
3. B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. Software Eng.*, 17(3):259–273, 1991.
4. M. Carey. Ex ante heuristic measures of schedule reliability. *Transportation research*, 33(7):473494, 1999.
5. T. Chatain and C. Jard. Symbolic diagnosis of partially observable concurrent systems. In *FORTE'04*, volume 3235 of *LNCS*, pages 326–342, 2004.
6. T. Chatain and C. Jard. Time supervision of concurrent systems using symbolic unfoldings of time Petri nets. In *FORATS'05*, volume 3829 of *LNCS*, pages 196–210. Springer, 2005.
7. T. Chatain and C. Jard. Complete finite prefixes of symbolic unfoldings of safe time Petri nets. In *Petri Nets and Other Models of Concurrency-ICATPN 2006*, pages 125–145. Springer, 2006.
8. J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev. Synthesizing Petri nets from state-based models. In *Proceedings of ICCAD'95*, pages 164–171, 1995.
9. A. D'Ariano, D. Pacciarelli, and M. Pranzo. A branch and bound algorithm for scheduling trains in a railway network. *European Journal of Operational Research*, 183(2):643–657, 2007.
10. A. D'Ariano, M. Pranzo, and I.A. Hansen. Conflict resolution and train speed coordination for solving real-time timetable perturbations. *IEEE Trans. on Intelligent Transportation Systems*, 8(2):208–222, 2007.
11. C. Dousson. Extending and unifying chronicle representation with event counters. In *ECAI'2002*, pages 257–261, 2002.
12. L. Doyen, G. Geeraerts, J.F. Raskin, and J. Reichert. Realizability of real-time logics. In *Formal Modeling and Analysis of Timed Systems, 7th International Conference, FORMATS 2009*, volume 5813 of *LNCS*, pages 133–148, 2009.
13. J. Esparza, S. Römer, and W. Vogler. An improvement of mcmillan's unfolding algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.
14. A. Horváth, M. Paolieri, L. Ridi, and E. Vicario. Transient analysis of non-Markovian models using stochastic state classes. *Performance Evaluation*, 69(7):315–335, 2012.
15. M. Krichen and S. Tripakis. Conformance testing for real-time systems. *Formal Methods in System Design*, 34(3):238–304, 2009.
16. D. Lime and O.H. Roux. Model checking of time Petri nets using the state class timed automaton. *Journal of Discrete Event Dynamic Systems*, 16(2):179–205, 2006.
17. Kenneth L. McMillan. A technique of state space search based on unfolding. *Formal Methods in System Design*, 6(1):45–65, 1995.
18. R. Y. Rubinstein and D.P. Kroese. *Simulation and the Monte Carlo Method*. Wiley, 2 edition, 2008.
19. A.L. Semenov and A. Yakovlev. Verification of asynchronous circuits using time Petri net unfolding. In *DAC*, pages 59–62, 1996.

A Formal semantics of STPNs

Timed moves are formally defined by the following rule:

$$\frac{\begin{array}{l}\delta > 0 \\ \wedge\ \forall t \in \mathsf{enab}(m) \setminus \mathsf{blck}(\langle m, \tau\rangle), \tau(t) \geq \delta \wedge \tau'(t) = \tau(t) - \delta \\ \wedge\ \forall t \in \mathsf{blck}(\langle m, \tau\rangle), \tau'(t) = \tau(t)\end{array}}{\langle m, \tau\rangle \xrightarrow{\delta} \langle m, \tau'\rangle}$$

A transition t' is *persistent* after firing of t from m iff it is enabled in m, $t \neq t'$, and t' is enabled in $m - {}^\bullet t$. We denote by $\mathsf{pers}(m,t) = (\mathsf{enab}(m) \cap \mathsf{enab}(m_{tmp})) \setminus \{t\}$ the set of persistent transitions after firing of t from m.

Discrete moves are formally defined by the following operational rule:

$$\frac{\begin{array}{l}t \in \mathsf{fira}(\langle m, \tau\rangle) \\ \wedge\ m' = (m - {}^\bullet t) + t^\bullet \\ \wedge\ \forall t_i \in \mathsf{pers}(m,t), \tau'(t_i) = \tau(t_i) \\ \wedge\ \forall t_i \in \mathsf{newl}(m,t), \tau'(t_i) \in [\mathsf{eft}(t), \mathsf{lft}(t)]\end{array}}{\langle m, \tau\rangle \xrightarrow{t} \langle m', \tau'\rangle}$$

B Proof of proposition 1

Proposition 1. Let $\mathcal{N}$ be a STPN guaranteeing time progress of δ time units (between consecutive occurrences of each transition). For every date $D \in \mathbb{R}_{\geq 0}$ and condition b in an unfolding $\mathcal{U}_n$, there exists $K \geq n$ s.t. $\{b' \in \mathsf{block}(b) \mid \mathsf{dob}(b') \leq D\}$ is contained in $\mathcal{U}_K$.

Proof. Consider a pre-process PP of $\mathcal{U}_n$, which depth is more than $\lceil \frac{D}{\delta} \rceil.|T|$ events. Every event of the unfolding appended at depth i consumes conditions that were created at depth $j < i$, and at least one condition that was produced at step i of the unfolding. Hence, for every event e_n and b_n condition created at depth n, there exists a sequence $b_0 < e_1 < b_1 < \cdots < e_n < b_n$ of events and conditions of increasing depth (and also increasing dates). With the time progress assumption, we know that every consecutive pair of events representing the same transition occurs at lest at dates that differ by δ. Hence, an event created at depth n has an occurrence date of at least $\delta.\lfloor n/|T| \rfloor$. The occurrence date of an event created at depth greater than $\frac{D}{\delta}.|T|$ is hence greater than D. The number of events and conditions created at step n and appearing in the same pre-process of $\mathcal{U}_n$ is finite (as creating an event uses exclusively at least one condition of the preceding step). It is hence sufficient to unfold a net up to depth $\frac{D}{\delta}.|T|$ to obtain the (finite) set of conditions that refer to the same place as some condition b before a given date D.

C Proof of Proposition 2

Proposition 2. Let $\mathcal{N}$ be a STPN guaranteeing time progress of δ time units. The set of time processes executable by $\mathcal{N}$ in D time units are prefixes of time processes of $\mathcal{U}_K$, with $K = \lceil \frac{D}{\delta} \rceil \cdot |T|$ containing only events with date $\leq D$.

Proof. We will show inclusion of the set of processes in the two directions. First of all, we define an ordering on symbolic processes. Let $\mathcal{E}^s = \langle E, B, \Phi \rangle$ and $\mathcal{E}^{s\prime} = \langle E', B', \Phi' \rangle$ be two symbolic processes. We will say that $\mathcal{E}^s \sqsubseteq \mathcal{E}^{s\prime}$ iff there exists an event e' such that $E' = E \cup \{e'\}$, $B' = B \cup e'^{\bullet}$, and $\Phi = \Phi'_{|\mathsf{var}(E,B)}$.

Lemma 1. *Let $\mathcal{E}^s$ be a symbolic process of unfolding $\mathcal{U}_K$, starting from m_0, d_0, that is satisfiable and complete. Let θ be one of its solutions guaranteeing $\forall e \in E, \theta(e) \leq D$. Then, there exists a sequence $\mathcal{E}^{s,0} = \langle E_0, B_0, \Phi_0 \rangle \sqsubseteq \mathcal{E}^{s,1} = \langle E_1, B_1, \Phi_1 \rangle \cdots \sqsubseteq \mathcal{E}^s$ of symbolic processes of $\mathcal{U}_K$ such that $E_0 = \emptyset$, $B_0 = \{\langle \perp, p \rangle \mid p \in m_0\}$ $\Phi_0 = \{\theta(\perp) = d_0 \wedge \bigwedge_{b \in B_0} \mathsf{dob}(b) = d_0\}$ and θ is a solution for every $\mathcal{E}^{s,i}$ and $\theta(e_i) \leq \theta(e_i + 1)$.*

Proof. We can show this property by induction on the size of prefixes of $\mathcal{E}^s$. The base hypothesis is straightforward, taking the sequence with only one symbolic process $\mathcal{E}^{s,0}$ without events. Suppose that this property is satisfied for symbolic processes up to size n, and consider a satisfiable and complete symbolic process $\mathcal{E}^{s,n+1}$ of size $n+1$. Let θ_{n+1} denote a solution for this process. A growing sequence from $\mathcal{E}^{s,0}$ to $\mathcal{E}^{s,n+1}$ exists.In this sequence, the difference between $\mathcal{E}^{s,n+1}$ and $\mathcal{E}^{s,n}$ is a single event e that is maximal in $\mathcal{E}^{s,n+1}$ w.r.t. ordering on events $\preceq$, and such that $\theta(e) \geq \theta(x)$ for every event x in $\mathcal{E}^{s,n+1} \setminus \{e\}$, and $\theta(e) \geq \mathsf{dob}(b)$ for every b in $\mathcal{E}^{s,n+1} \setminus \{e\}$. Let E^n, B^n denote the set of events in $\mathcal{E}^{s,n+1} \setminus \{e\}$. Let us denote by $\Phi_{n+1|E^n,B^n}$ the restiction of Φ_{n+1} to variables attached to events and conditions E^n, B^n. One has to remove variables $\theta(e)$, $\mathsf{dod}(b)$ for every $b \in {}^{\bullet}e$, and $\mathsf{dob}(b)$ for every $b \in e^{\bullet}$ using an elimination technique such as Fourier-Motzkin. Using the properties of elimination, θ satisfies Φ_{n+1} if and only if the restriction of θ to $\mathsf{var}(E^n, B^n)$ satisfies $\Phi_{n+1|E^n,B^n}$. However, the restriction of θ is exactly θ_n, and as $\theta(e)$, $\mathsf{dod}(b)$ for $b \in e^{\bullet}$, and $\mathsf{dod}(b)$ for $b \in {}^{\bullet}e$ are all greater than variables in $\mathsf{var}(E^n, B^n)$, the elimination of variables is simply a projection on atoms that do not contain variables related to e, and $\Phi_{n+1|E^n,B^n} = \Phi_n$. □

Lemma 2. *Given a symbolic process $\mathcal{E}^s$ of $\mathcal{U}_K$, one of its solutions θ, and an ordering $\mathcal{E}^{s,0} = \langle E_0, B_0, \Phi_0 \rangle \sqsubseteq \mathcal{E}^{s,1} = \langle E_1, B_1, \Phi_1 \rangle \cdots \sqsubseteq \mathcal{E}^s$ as above, then the word $u_{\mathcal{E}^s,\theta} = \langle t_1, \theta(e_1) \rangle \ldots \langle t_{|E|}, \theta(e_{|E|}) \rangle$ is a timed word of $\mathcal{L}(\mathcal{N})$.*

Proof. Again we can prove this lemma by induction. The base case is obvious, as the empty word ϵ is a timed word of $\mathcal{L}(\mathcal{N})$. Let us suppose that the property is satisfied up to n, that is for every process $\mathcal{E}_n$ of size n and solution θ_n meeting all constraints of $\mathcal{E}_n$, there exists an increasing sequence of prefixes of $\mathcal{E}_n$ such that the word associated with this sequence is a timed word of $\mathcal{L}(\mathcal{N})$.

Let us now consider a time process $\mathcal{E}_{n+1}$ with $n+1$ events and one of its solutions θ_{n+1}. As in Lemma 1, one can find an event e_{n+1} and a process $\mathcal{E}_n$ such that $\mathcal{E}_n$ and $\mathcal{E}_{n+1}$ only differ by addition of this single event. There exists a timed word $u_n = \langle e_1, \theta_{n+1}(e_1)\rangle \dots \langle e_n, \theta_{n+1}(e_n)\rangle \in \mathcal{L}(\mathcal{N})$ corresponding to $\mathcal{E}_n$. This word may lead the net to any configurations in a set $Conf_n$ with identical markings, but distinct times to fire attached to transitions. However, as we know that θ_{n+1} meets all constraints of $\mathcal{E}_{n+1}$, there exists a configuraton in $Conf_n$ whose times to fire allow firing of e_{n+1} at date $\theta(e_{n+1})$, and $u_{n+1} = u_n.\langle e_{n+1}, \theta(e_{n+1})\rangle \in \mathcal{L}(\mathcal{N})$.□

Note that assuming time progress, the dates attached to an event of a process of $\mathcal{U}_K$ that occur at a date smaller than D cannot be further constrained by addition of constraints coming from events that are not in $\mathcal{U}_K$. The two lemmas above hence allow to conclude that for a given symbolic process $\mathcal{E}^s$ of unfolding $\mathcal{U}_K$, in which one considers events that occur before date D, and for each solution of $\mathcal{E}^s$, we have $\mathcal{E}^s_\theta = TP_{u_{\mathcal{E}^s,\theta}}$ for some word $u_{\mathcal{E}^s,\theta} \in \mathcal{L}(\mathcal{N})$. Hence, the set of time processes of $\mathcal{U}_K$ whose events occur before D is contained in the set of time processes $TP(\mathcal{L}_{\leq D}(\mathcal{N}))$. All time processes of some pre-process of $\mathcal{U}_K$ (and hence all time processes of unfolding $\mathcal{U}_K$) can be built from a timed word that is executable by $\mathcal{N}$ in less than D time units, and are hence time processes of $\mathcal{N}$.

We now have to prove the converse direction, i.e., every time process associated with a word $u \in \mathcal{L}_{\leq D}(\mathcal{N})$ is a time process of $\mathcal{U}_K$.

Lemma 3. *Let $u \in \mathcal{L}_{\leq D}(\mathcal{N})$. Then, $TP(u)$ is a time process of $\mathcal{U}_K$.*

Proof. We proceed by induction on the size of words. First, for the empty words, the time process with only initial conditions is clearly a time process of $\mathcal{U}_K$. Let us now assume that for every $u_n = \langle e_1, \theta(e_1)\rangle \dots \langle e_n, \theta(e_n)\rangle \in \mathcal{L}_{\leq D}(\mathcal{N})$ of length n, $TP(u_n)$ is a time process of $\mathcal{U}_K$. Let us consider a word $u_{n+1} = \langle e_1, \theta(e_1)\rangle \dots \langle e_n, \theta(e_n)\rangle.\langle e_{n+1}, \theta(e_{n+1})\rangle \in \mathcal{L}_{\leq D}(\mathcal{N})$. One can build a time process $\mathcal{E}_u$ for $u = \langle e_1, \theta(e_1)\rangle \dots \langle e_n, \theta(e_n)\rangle$. Clearly, as $u_{n+1} \in \mathcal{L}_{\leq D}(\mathcal{N})$, word u leads from marking m_0 to a marking that enables e_{n+1}. Let $e_{p_1}, \dots, e_{p_k}$ denote the k events that produce the tokens that are consumed by e_{n+1}. If event e_{n+1} is a firing of some transition t that occurs exactly when its time to fire has expired, $\theta(e_{n+1})$ meets the constraint $\mathsf{eft}(t) + \max\{\theta(e_{p_i})\} \leq \theta(e_{n+1}) \leq \mathsf{lft}(t) + \max\{\theta(e_{p_i})\}$. In any case, we have $eft(t) + \max\{\theta(e_{p_i})\} \leq \theta(e_{n+1})$ (which is the only constraint w.r.t predecessors imposed by constraint in the unfolding. Similarly, let $e_{b_1}, \dots e_{b_q}$ denote the last events of u that free places in which t outputs some tokens (and hence may have blocked the execution of t before $\theta(e_{n+1})$). We have $\theta(e_{n+1})$ meets the constraint $\max(\{\theta(e_{b_i})\}) \leq \theta(e_{n+1})$. Hence, any event that had to occur before $\theta(e_{n+1})$ (due to urgency, causality, or blockings) also appears in $\mathcal{E}_u$. Hence, θ witnesses satisfiability of a set of constraints over occurrence dates of events $e_1, \dots, e_n$, and one can safely append $e_{n+1} = (B, t)$ to maximal places of $\mathcal{E}_u$, and obtain a symbolic prefix $\mathcal{E}_{u_{n+1}}$ (satisfiable, conflict free and complete). It now remains to show that $\mathcal{E}_{u_{n+1}}$ is a symbolic process of $\mathcal{U}_K$. As $\theta(e_{n+1}) \leq D$, e_{n+1} appears in the unfolding of $\mathcal{N}$ at depth at most $\frac{D}{\delta}$, which is lower than $K = \lceil \frac{D}{\delta} \rceil \cdot |T|$. Hence, $\mathcal{E}_{u_{n+1}}$ is an causally

closed set of events that also contains all mandatory urgent transition firings and place unblockings whose set of constraints is satisfiable, and contained in $\mathcal{U}_K$, i.e., it is a symbolic process of $\mathcal{U}_K$.

D Algorithm to compute embeddings of a schedule in a process

Finding the set of embedding functions $\Psi \triangleq \{\psi_0, \psi_1, \ldots, \psi_k\}$ that satisfy the conditions of definition 14 is achieved building iteratively injective functions meeting the embedding requirements, by matching at every step minimal yet unexplored nodes of S with minimal unmatched nodes of $\mathcal{E}^s$. This construction is depicted in Algorithm 1. We will define an embedding function f as a set pairs of the form $\langle n, e\rangle$, interpreted as $f(n) = e$. We define in particular the empty embedding $f_\perp$, that is undefined for every node of N, and will be used as starting point of the algorithm. For a given function f, the function $f \cup \langle n, e\rangle$ is the function that associates e to node n, and $f(n')$ to every other node $n' \in \mathsf{domain}(f)$.

Algorithm 1: Computation of embeddings of a schedule in a symbolic process

input: a schedule $S = \langle N, \to, \lambda, C\rangle$, a symb. process $\mathcal{E} = \langle E, B, \Phi\rangle$;
$\Psi \coloneqq \varnothing$; `// the set of solutions is initially empty`
$F \coloneqq \{f_\perp\}$; `// the exploration starts from the undefined map`
while $F \neq \varnothing$ **do**
  choose $f \in F$;
  $\textsc{Min}_{S,f} \coloneqq \min_{\to}\big(N \setminus \mathsf{domain}(f)\big)$;
  if $\textsc{Min}_{S,f} = \varnothing$ **then** ; `// all nodes of` S `have an image in` $\mathcal{E}$
    $\Psi \coloneqq \Psi \cup \{f\}$; `// ` f `is an embedding`
  else
    $F \coloneqq F \setminus \{f\}$; `// we will explore extensions of partial embedding` f
    $\textsc{Min}_{\mathcal{E},f} \coloneqq \min_{\preceq}\big(E \setminus \mathsf{image}(f)\big)$;
    $\textsc{Found} \coloneqq \mathsf{false}$;
    while $\textsc{Min}_{S,f} \neq \varnothing \wedge \textsc{Found} = \mathsf{false}$ **do**
      choose $n \in \textsc{Min}_{S,f}$;
      $\textsc{Min}_{S,f} \coloneqq \textsc{Min}_{S,f} \setminus \{n\}$;
      $\textsc{cand} \coloneqq \{e \in \textsc{Min}_{\mathcal{E},f} \mid \mathsf{tr}(e) \in r\big(\lambda(n)\big) \wedge \forall n' \in \underset{\to}{\mathsf{dpred}}(n), f(n') \preceq e\}$;
      if $\textsc{Cand} \neq \varnothing$ **then**
        $F \coloneqq F \cup \bigcup_{e \in \textsc{Cand}} \{f \cup \langle n, e\rangle\}$; `// update` f `with pairs` $\langle n, e\rangle$ `that meet the matching criteria and add the new functions to candidate embeddings`
        $\textsc{Found} \coloneqq \mathsf{true}$;
      end
    end
  end
end

E Stochastic state class tree

In this part of the appendix, we detail how to build a stochastic state class tree for a particular process of a stochastic Petri net with blocking seantics.

Definition 17 (transient stochastic state class). *A* transient stochastic state class *(or class for short) of an STPN $\mathcal{N}$ is a tuple $\Sigma \triangleq \langle m, D, f_{\underset{\sim}{\tau}}, \text{BLK}, \text{URG}\rangle$ where m is a marking, $\underset{\sim}{\tau} \triangleq \langle \tau_{\text{age}}, \underline{\tau}\rangle$ is a vector where τ_{age} is the opposite of the elapsed time and D is a domain for $\underset{\sim}{\tau}$. $\underline{\tau} \triangleq \langle \tau_0, \tau_1, \ldots, \tau_{N-1}\rangle$ is a vector of random variables with support $D_{\downarrow\tau_{\text{age}}}$ representing the possible TTFs of transitions enabled by marking m s.t. for every τ_i in $\underline{\tau}$, the elimination of all other variables from D yields a non-empty set of possible values that is different from $\{0\}$; $f_{\underset{\sim}{\tau}}$ is a PDF over D,* BLK *is a set of blocked transitions, and* URG *is a set of urgent transitions.*

The domain D of $\underline{\tau}$ is simply the domain of a class as defined in the usual state class graph construction of TPNs (see for instance [16, 3]). It represents possible values for TTFs attached to transitions. As our STPN is bounded, the number of domains that can be generated inductively at construction time is finite (as proved by [3]). Urgent and blocked transitions are also finite subsets of T. However, as shown in [14], the number of distributions that can be iteratively computed needs not be finite.

Definition 18 (transient stochastic state class tree). *A* transient stochastic state class tree *for an STPN $\mathcal{N}$ (that we shall call tree, for short) is a directed acyclic graph $\mathfrak{S} \triangleq \langle V, \circ\!\!\rightarrow \cup \bullet\!\!\rightarrow\rangle$ where vertices in V are classes, edges in $\circ\!\!\rightarrow$ represent firing transitions after (symbolically) elapsing time, and edges in $\bullet\!\!\rightarrow$ represent firings of urgent transitions. Every vertex in the tree has only one predecessor except for the root of the tree, denoted v_0, that has no predecessor. Edges carry probabilistic informations on transitions firings and the sum of probabilities of all edges leaving the same vertex is equal to* 1.

The construction of a tree starts from the initial class Σ_0 (with marking m_0, a domain D_0 for the TTFs of transitions enabled in m_0 and all other components defined accordingly, see appendix E) and inductively computes edges and reachable classes. Edges $\Sigma \xrightarrow{t,\mu} \Sigma'$ from a class Σ to a successor class Σ' are labeled by a transition name t and by the probability μ to fire t from Σ, and are of two forms:

Firing after elapsing time: A move $\Sigma \overset{t_i,\mu_i}{\circ\!\!\rightarrow} \Sigma'$ from $\Sigma = \langle m, D, f_{\underset{\sim}{\tau}}, \text{BLK}, \text{URG}\rangle$ to $\Sigma' = \langle m', D', f_{\underset{\sim}{\tau}'}, \text{BLK}', \text{URG}'\rangle$, achievable with probability μ_i, consists in firing transition t_i after symbolically elapsing its TTF. Such a move is only allowed if $\text{URG} = \varnothing$ and the TTF τ_i of t_i is less than or equal to TTFs of all other transitions that could fire from Σ. The time domain D^i from which t_i can fire is hence $D^i \triangleq D \cap \bigcap_{\tau_j \in \underline{\tau}} \{\tau_i \leq \tau_j\}$, and the probability of firing t_i from Σ is $\mu_i = \int_{D^i} f_{\underset{\sim}{\tau}}(\underset{\sim}{x})\, d\underset{\sim}{x}$. We have $m' = m - {}^\bullet t_i + t_i^\bullet$. The new domain D' and distribution $f'_{\underset{\sim}{\tau}}$ are computed as for STPNs with non-blocking semantics: Vector $\underset{\sim}{\tau}$ is obtained by advancing time, removing variables of disabled transitions and adding those of

newly enabled transitions [14], and then removing variables of transitions whose domain is the singleton $\{0\}$. The domain of a single variable τ_i in a domain D over several variables can be obtained by eliminating all variables but τ_i from D. As soon as a variable has domain $\{0\}$, the time to fire of the associated transition is necessarily zero, and the transition has to fire. It is then stored in the set of urgent transitions if it is not blocked, and in the set of blocked transitions otherwise. The set BLK$'$ is obtained by removing from BLK transitions that were disabled by firing of t_i and transitions that are not blocked anymore in m' thanks to the places freed by firing of t_i (they become urgent), and adding transitions which are enabled in m' with a firing domain in D' that is $\{0\}$. Finally, the set URG$'$ contains all enabled transitions that became urgent when firing t_i, i.e., transitions with firing domain $\{0\}$ among enabled transitions, and formerly blocked transitions unblocked by t_i.

Firing urgent transitions: In STPNs semantics, when more than one transition is firable from a configuration, their weights are used to compute the probability of firing each transition. This case can occur because of blocking semantics: an STPN can keep several transitions blocked, and firing a transition can also unlock several of them at the same time (all unblocked transitions become urgent). When a class Σ has urgent transitions (URG $\neq \varnothing$), only moves of the form $\Sigma \overset{t_i,\mu_i}{\bullet\!\!\rightarrow} \Sigma'$ are allowed. They consist in firing a transition t_i among urgent transitions in URG with probability $\mu_i = \mathcal{W}(t_k) \big/ \sum_{t_j \in \text{URG}} \mathcal{W}(t_j)$. Components m', D' and $f_{\underset{\sim}{\tau}'}$ of the successor class are computed as for timed moves, with the only difference that no time elapses before the firing of t_i. Set BLK$'$ is obtained by removing from BLK transitions that are unlocked or disabled by the firing of t_i and adding those that become blocked in m', and URG' contains transitions from URG that were not disabled by firing of t_i and transitions from BLK that were unblocked when firing t_i.

Transient trees are a priori infinite, but very often, one can work with a bounded horizon. This is our case when evaluating the probabilty of realization in $\mathcal{U}_K$. Let $\Psi_i \triangleq \{\psi_{i,0}, \psi_{i,1}, \ldots, \psi_{i,n-1}\}$ denote all possible embeddings of a schedule S into a symbolic process $\mathcal{E}_i^s$ of $\mathcal{N}$. We denote by $\mathbb{P}(\Phi_{\psi_{i,j},d\pm\alpha})$ the probability that $\mathcal{N}$ executes a time process $\mathcal{E}_i^s$ and realizes S within a precision of $\pm\alpha$ when $\psi_{i,j}$ is the embedding of S in $\mathcal{E}_i^s$. We adapt the tree construction to consider only $\mathcal{E}_i^s$ and embedding $\psi_{i,j}$, and compute $\mathbb{P}(\mathcal{E}_i^s \wedge \Phi_{\psi_{i,j},d\pm\alpha})$. We build a tree whose vertices of the form $\langle \Sigma, S \rangle$ memorize a class and a suffix of $\mathcal{E}_i^s$ not yet executed. We start from vertex $\langle \Sigma_0, \mathcal{E}_i^s \rangle$. We create an edge $\langle \Sigma, S \rangle \xrightarrow{t_k,\mu_k} (\Sigma', S')$ representing firing of a transition t_k if there is a minimal event e in the remaining suffix of $\mathcal{E}_i^s$, and $\mathsf{tr}(e) = t_k$. S' is the new suffix obtained by removing e from S. Σ' is the successor class obtained after firing this transition from Σ. Edges are built as before in the tree, but with an additional constraint: edges with label t_k, μ_k and components m, D, $f_{\underset{\sim}{\tau}}$, BLK, URG of classes are built with the additional requirement that when creating an edge from an event e that is in the image of $\psi_{i,j}$, the firing time domain is restricted to impose that e occurs in the time interval $I_k = [\max(0, d(\psi_{i,j}^{-1}(e)) - \alpha), d(\psi_{i,j}^{-1}(e)) + \alpha]$. In this case, the probability of firing $t_k = \mathsf{tr}(e)$ becomes $\mu_k = \int_{D^k \cap D'^k} f_{\underset{\sim}{\tau}}(\underset{\sim}{x})\, d\underset{\sim}{x}$, where D'^k is the part of D in which $\tau_k - \tau_{\text{age}}$ (the firing date for e) belongs to I_k. We stop

developing a branch of the tree at vertices whose suffix does not contain events that are images of nodes in S via $\psi_{i,j}$. The construction ends with a tree $\mathfrak{S}_{\mathrm{i,j},\alpha}$.

Transient tree $\mathfrak{S}_{\mathrm{i,j},\alpha}$ measures the probability of solutions and of occurrence of a particular process. The probability $\mathbb{P}(\mathcal{E}_i^s \wedge \Phi_{\psi_{i,j},\alpha})$ is computed as $\mathbb{P}(\mathcal{E}_i^s \wedge \Phi_{\psi_{i,j},\alpha}) = \sum_{\rho \in \textsc{Path}(\mathfrak{S}_{\mathrm{i,j},\alpha})} \mathbb{P}(\rho)$ where $\textsc{Path}(\mathfrak{S}_{\mathrm{i,j},\alpha})$ is the set of paths from $\langle \Sigma_0, \mathcal{E}_i^s \rangle$ to a leaf of $\mathfrak{S}_{\mathrm{i,j},\alpha}$, and the probability $\mathbb{P}(\rho)$ of a path $\rho = \Sigma_0 \xrightarrow{t_i,\mu_i} \dots \xrightarrow{t_{l-1},\mu_{l-1}} \Sigma_l$ that start at Σ_0 and end on a leaf Σ_l is the product $\Pi_{i \in \{0,\dots,l-1\}} \mu_i$. As soon as $\mathbb{P}(\mathcal{E}_i^s \wedge \Phi_{\psi_{i,j},\alpha}) > 0$, S has a non-null probability to be realized (with a tolerance of $\pm\alpha$). Noticing that different embeddings yield disjoint paths in their respective transient stochastic state class trees, the probability for a schedule to be realized by a process is hence $\mathbb{P}(\mathcal{E}_i^s \models S) = \sum_{\psi_{i,j} \in \Psi_i} \mathbb{P}(\mathcal{E}_i^s \wedge \Phi_{\psi_{i,j},\alpha})$.

Finally, denoting by $\mathcal{E}(PP, S)$ the symbolic processes of a pre-process PP that embed S, the probability $\mathbb{P}(\mathcal{N} \models S)$ is greater than $\max\{\mathbb{P}(\mathcal{E}_i^s \wedge \Phi_{\psi_{i,j},\alpha}) \mid PP \in \mathcal{PE}(U_K) \wedge \mathcal{E}_i^s \in \mathcal{E}(PP, S)\}$. It is difficult to obtain more than a lower bound for realization, as symbolic processes of $\mathcal{E}(PP, S)$ might have overlapping executions.

F Derivation of components of successor class

We hereafter provide details on how to compute components D' and $f_{\underset{\sim}{\tau}'}$ of a class Σ' obtained from a class Σ through a transition $\overset{t_i,\mu_i}{\circ\!\!\rightarrow}$. Derivation of components m', $\textsc{blk}'$ and $\textsc{urg}'$ has already been covered and need not further explanations.

We shall use the following notations:

- given a time domain D delimiting the possible values of a set of variables $\underline{x} = \langle x_0, x_1, \dots, x_{N-1} \rangle$, we will denote by $D\downarrow_{x_i}$ the projection of D that eliminates variable x_i from $\underline{x}$. The elimination is done via the Fourier-Motzkin method detailed in Appendix G;
- given a vector $\underline{x} = \langle x_0, x_1, \dots, x_{N-1} \rangle$, we denote by $\underline{x} \setminus x_i$ the vector $\underline{x}$ from which variable x_i is removed, with $i \in \{0, 1, \dots, N-1\}$;
- the addition of a scalar x_0 to each element of a vector $\underline{x} = \langle x_1, x_2, \dots, x_n \rangle$ is simply written $\underline{x} + x_0$.

Fist of all, the initial class is $\Sigma_0 \triangleq \langle m_0, D_0, f_{\tau_0}, blk_0, urg_0 \rangle$. Marking m_0 is the initial marking of the net, domain D_0 is a product of domains $[eft(t), lft(t)$ for all transitions enabled in m_0 (their domain is not $\{0\}$ as we have $eft(t) < lft(t)$. Vector $\tau_0 =< 0, \tau_1, \tau_{enab(m_0)}$ is a vector representing elapsed time (in the inital class it must be 0, and times to fires of enabeled variables. As domains for τ_i's are not singletons, we can safely set $blk_0 = \emptyset$, and $urg_0 = \emptyset$.

Probability of firing: A transition t_i can fire from class Σ iff t_i is enabled by m and no postset place of t_i is occupied; that is $\forall p \in t^\bullet, m(p) = 0$. We also need its TTF τ^i to be less than or equal to TTFs of all variables in $\underset{\sim}{\tau}$; transition t_i will then

fire from Σ with probability μ^i, with:

$$\mu^i = \int_{D^i} f_{\underset{\sim}{\tau}}(\underset{\sim}{x})\, d\underset{\sim}{x}$$

D^i is the time domain from which t_i shall fire with all its TTF less than or equal to every variable in $\underset{\sim}{\tau}$.

$$D^i \triangleq D \cap \bigcap_{t^j \in \tau} \{\tau^i \leq \tau^j\}$$

Precedence condition: The assumption that t_i fires before any other transition adds conditions on the time vector and thus leads to a new random variable $\underset{\sim}{\tau}_a$ distributed over D^i according to the following conditional PDF:

$$f_{\underset{\sim}{\tau}_a}(\underset{\sim}{x}) = f_{\underset{\sim}{\tau}}(\underset{\sim}{x}) \Big/ \mu^i$$

Time elapsing and elimination: According to the semantics of STPNs, when t_i fires, TTFs of activated transitions are decreased by the value of the TTF of t_i, namely τ^i. This yields a new random variable $\underset{\sim}{\tau}_b = \underset{\sim}{\tau}_a - \tau^i$ distributed over the domain $D_b = D^i{\downarrow}_{\tau^i}$ in which the variable attached to the fired transition t_i is eliminated. The PDF of the new multivariate random variable $\underset{\sim}{\tau}_b$ is then:

$$f_{\underset{\sim}{\tau}_b}(\underset{\sim}{x}) = \int_{\text{MIN}^i}^{\text{MAX}^i} f_{\underset{\sim}{\tau}_a}(\underset{\sim}{x} + x_i, x_i)\, dx_i$$

where MIN^i and MAX^i denote the bounds of the support of variable τ^i.

Disabling: If the firing of t_i disables a transition t_j, variable τ_b^j has to be eliminated from the time vector, yielding a new vector $\underset{\sim}{\tau}_c = \underset{\sim}{\tau}_b \setminus \tau_b^j$ distributed over $D_c = D_b{\downarrow}_{\tau_b^j}$ with PDF:

$$f_{\underset{\sim}{\tau}_c}(\underset{\sim}{x}) = \int_{\text{MIN}^j}^{\text{MAX}^j} f_{\underset{\sim}{\tau}_b}(\underset{\sim}{x}, x_j)\, dx_j$$

The same procedure is repeated for every disabled transition by the firing of t_i. Let $\underset{\sim}{\tau}_{c^*}$, D_{c^*} and $f_{\underset{\sim}{\tau}_{c^*}}$ then respectively denote the resulting time vector, domain and PDF.

Newly enabling: If the firing of t_i enables a transition t_k, with PDF f_{t_k} over $[\mathsf{eft}(t_k), \mathsf{lft}(t_k)]$, then the new time vector, that we denote by $\underset{\sim}{\tau}_d$, shall include an additional component τ_d^k and shall be distributed over $D_d = D_{c^*} \times [\mathsf{eft}(t_k), \mathsf{lft}(t_k)]$ according to the PDF:

$$f_{\underset{\sim}{\tau}_d}(\underset{\sim}{x}, x_k) = f_{\underset{\sim}{\tau}_{c^*}}(\underset{\sim}{x}) \times f_{t_k}(x_k)$$

The same procedure is similarly repeated for every newly enabled transition to finally obtain the PDF of the successor class Σ'.

G Fourier–Motzkin elimination method

The *Fourier–Motzkin elimination method* is an algorithm for eliminating variables from a system of linear inequalities. Let ϕ be a system of linear inequalities with variables $x_1, x_2, \ldots, x_r$ where x_r is the variable to be removed. ϕ can be written as $\phi^+ \wedge \phi^- \wedge \phi^{\varnothing}$ where $\phi^- = \bigwedge_{i=1}^{m} -x_r \leq b_i - \sum_{k=1}^{r-1} a_{ik}x_k$ and $\phi^+ = \bigwedge_{i=1}^{n} x_r \leq b_i - \sum_{k=1}^{r-1} a_{ik}x_k$ are the sets of inequations where the coefficients of x_r are respectively negative and positive, and $\phi^{\varnothing}$ is the inequations subsystem in which x_r doesn't appear. Eliminating the variable x_r from ϕ refers to the creation of another system of inequations in which x_r doesn't appear and which has the same solutions over the remaining variables. The original system can be written as

$$\max_{i=1,\ldots,m}(-b_i + \sum_{k=1}^{r-1} a_{ik}x_k) \leq x_r \leq \min_{i=1,\ldots,n}(b_i - \sum_{k=1}^{r-1} a_{ik}x_k) \wedge \phi^{\varnothing}$$

Which, by eliminating x_r, gives

$$\max_{i=1,\ldots,m}(-b_i + \sum_{k=1}^{r-1} a_{ik}x_k) \leq \min_{i=1,\ldots,n}(b_i - \sum_{k=1}^{r-1} a_{ik}x_k) \wedge \phi^{\varnothing}$$

Example 1. Let ϕ be the following system of linear inequalities:

$$\phi = \begin{cases} \alpha_1 \leq x_1 \leq \beta_1 \\ \alpha_2 \leq x_2 \leq \beta_2 \\ \alpha_3 \leq x_3 \leq \beta_3 \\ -\gamma_{21} \leq x_1 - x_2 \leq \gamma_{12} \\ -\gamma_{31} \leq x_1 - x_3 \leq \gamma_{13} \\ -\gamma_{32} \leq x_2 - x_3 \leq \gamma_{23} \end{cases}$$

Suppose that we want to eliminate the variable x_1. We start by identifying $\phi^{\varnothing}$, the subsystem of inequations in which x_1 doesn't appear, as follows:

$$\phi = \left.\begin{cases} \alpha_1 \leq x_1 \leq \beta_1 \\ -\gamma_{21} \leq x_1 - x_2 \leq \gamma_{12} \\ -\gamma_{31} \leq x_1 - x_3 \leq \gamma_{13} \\ \alpha_2 \leq x_2 \leq \beta_2 \\ \alpha_3 \leq x_3 \leq \beta_3 \\ -\gamma_{32} \leq x_2 - x_3 \leq \gamma_{23} \end{cases}\right\} = \phi^{\varnothing}$$

We then isolate x_1 by rewriting the inequations left, as follows:

$$\phi = \begin{cases} \alpha_1 \leq x_1 \leq \beta_1 \\ x_2 - \gamma_{21} \leq x_1 \leq x_2 + \gamma_{12} \\ x_3 - \gamma_{31} \leq x_1 \leq x_3 + \gamma_{13} \\ \phi^{\varnothing} \end{cases}$$

One can easily see that an equivalent solution of this system is the one where x_1 is bounded with the maximum value of its left bounds and the minimum of its right bounds in the system of inequations; adding to that $\phi^{\varnothing}$.

$$\phi \iff \max(\alpha_1, x_2 - \gamma_{21}, x_3 - \gamma_{31}) \leq x_1 \leq \min(\beta_1, x_2 + \gamma_{12}, \leq x_3 + \gamma_{13}) \wedge \phi^{\varnothing}$$

Finally, eliminating x_1 consists in saying that the left bound is inferior or equal to the right bound and we get the following:

$$\phi' \iff \max(\alpha_1, x_2 - \gamma_{21}, x_3 - \gamma_{31}) \leq \min(\beta_1, x_2 + \gamma_{12}, \leq x_3 + \gamma_{13}) \wedge \phi^{\varnothing}$$