



**HAL**  
open science

# Realizability of Schedules by Stochastic Time Petri Nets with Blocking Semantics

Loïc Hélouët, Karim Kecir

► **To cite this version:**

Loïc Hélouët, Karim Kecir. Realizability of Schedules by Stochastic Time Petri Nets with Blocking Semantics: (Extended Version). [Research Report] INRIA Rennes - Bretagne Atlantique. 2017. hal-01646920v2

**HAL Id: hal-01646920**

**<https://inria.hal.science/hal-01646920v2>**

Submitted on 27 Nov 2017

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# Realizability of Schedules by Stochastic Time Petri Nets with Blocking Semantics

Loïc Hélouët

*INRIA Rennes, loic.helouet@inria.fr*

Karim Kecir

*INRIA Rennes & Alstom, karim.kecir@alstom.com*

---

## Abstract

This paper considers realizability of expected schedules by production systems with concurrent tasks, bounded resources that have to be shared among tasks, and random behaviors and durations. Schedules are high level views of desired executions of systems represented as partial orders decorated with timing constraints. Production systems (production cells, train networks. . .) are modeled as stochastic time Petri nets STPNs with an elementary (1-bounded) semantics. We detail their interleaved operational semantics, and then propose a non-interleaved semantics through the notion of time processes. We then consider *boolean realizability*: a schedule  $S$  is realizable by a net  $\mathcal{N}$  if  $S$  embeds in a time process of  $\mathcal{N}$  that satisfies all its constraints. However, with continuous time domains, the probability of a time process with exact dates is null. We hence consider *probabilistic realizability* up to  $\alpha$  time units, that holds if the probability that  $\mathcal{N}$  realizes  $S$  with constraints enlarged by  $\alpha$  is strictly positive. Upon a sensible restriction guaranteeing time progress, boolean 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.

*Keywords:* Petri nets, unfolding, scheduling, realizability.

---

## 1. Introduction

Scheduling basic operations a priori in automated systems (e.g., manufacturing or transport systems, etc.) is a way to manage at best available resources, avoid undesired configurations, or achieve an objective within a bounded delay.

- 5 Following a predetermined schedule is also a way to meet QoS objectives. For instance, operating a metro network or a fleet of buses usually amounts to implementing at best a predetermined timetable to meet users needs. In many

cases, schedules are designed to avoid deadlocks, races or congestion problems, and ease up recovery from minor delays that are part of the normal behavior of the system. Failing to implement a chosen schedule may then result in a loss of QoS, lead to congestion when delays accumulate, cause conflicting accesses to shared resources and, in the worst cases, cause deadlocks. Schedules are high-level views for correct ordering of important or critical operations (i.e., that use shared resources) in a system. They consider time issues such as delays between tasks, optimal dates, durations... in a production plan. They can be seen as partial orders among basic tasks, that abstract low-level implementation details, and are decorated with dates and timing constraints.

Designing a correct and optimal schedule for a system is a complex problem. On one hand, 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 [1, 2]. On the other hand, optimal solutions (for instance, w.r.t. completion date) are not necessarily the most probable nor the most robust ones: indeed, systems such as metro networks are subject to small delays (variations in trips durations from a station to another, passengers misbehavior...). Delays are hence expected and considered as 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. Consequently, optimal and realizable schedules are not necessarily robust enough if they impose tight realization dates to systems that are subject to random variations (delays in productions, faults...). Note also that the size of timetabling problems for real systems running for hours cannot be handled in a completely automated way by tools, that usually have to be guided by experts to return quasi-optimal solutions. Hence, timetables design involves expert competences that differ from those needed to design systems.

When a schedule and a low-level system description are designed separately, nothing guarantees that the system is able to realize the expected schedule. This calls for tools to check realizability of a schedule. One can expect systems designers and timetable experts to share a common understanding of the behavior of their system, so in general, the answer to a boolean realizability question should be positive. However, being able to realize a schedule does not mean that the probability to meet optimal objectives is high enough. Obviously, in systems with random delays, the probability to realize a schedule with precise dates is null. Beyond boolean realizability, a schedule shall hence be considered as *realizable* if it can be realized up to some bounded imprecision, and with a significant probability.

This paper addresses realizability of schedules by timed systems with concurrently running tasks, working in a stochastic environment. Such systems range from automated production cells, to transport networks and human organizations. The nature of considered systems calls for the use of concurrency models, for which one can expect efficient decision procedures as long as global states need not be computed. Addressing realizability also forces to consider time: one cannot consider that a schedule is realized if it cannot be implemented in

a finite and reasonable amount of time. Last, realization of a schedule can not  
55 be addressed in a purely boolean setting: one can not consider that a system  
realizes a schedule if the probability of executions that implement a schedule is  
null. This calls for models with probabilities, and for a quantitative notion of  
realizability.

The first contribution of the paper is the formalization of the implementation  
60 model with stochastic time Petri nets (STPN for short) and of its partial order  
semantics. STPNs are inspired from [3], but with a *blocking semantics*. In  
this semantics, a transition cannot fire if its postset is not empty. Considering  
blocking is essential to model systems with shared resources. This semantics  
guarantees 1-boundedness of places along all runs. We show that this blocking  
65 semantics can be captured in a concurrent setting by *time processes*. However,  
in a timed setting, blocking has non-trivial consequences on both interleaved  
and partial-order semantics. First, it forces to adapt the notion of urgency, as  
an urgent transition may have to wait for the postset of a transition to be free  
to fire. A second difficulty is related to the partial order semantics. Processes  
70 are obtained by unfolding STPNs with standard algorithms [4, 5], and then  
adding dates to events of these untimed processes in a way that is consistent  
with the time constraints attached to transitions of the net. A major difficulty is  
that under blocking semantics, process construction is not monotonic anymore.  
Indeed, due to blocking, one cannot decide to stop unfolding as soon as a set of  
75 constraints is unsatisfiable.

The second contribution of this paper is a boolean framework to address  
realizability of schedules by a STPN. Schedules are specified as partial orders  
decorated with time constraints. Boolean realizability of a schedule  $S$  by a STPN  
 $\mathcal{N}$  is defined as the existence of a process of  $\mathcal{N}$  that *embeds*  $S$ , and whose dates  
80 are compatible with the time constraint specified in  $S$ . We prove that upon some  
reasonable time progress assumption that guarantees time progress, realizability  
can be checked on a finite set of *symbolic processes*, obtained from a bounded  
untimed unfolding of  $\mathcal{N}$ . Symbolic processes are processes of the unfolding with  
satisfiable constraints on occurrence dates of events. A symbolic framework to  
85 unfold time Petri nets was already proposed in [6, 7] but blocking semantics  
brings additional constraints on firing dates of transitions.

Embedding a schedule  $S$  in a process of  $\mathcal{N}$  only guarantees *boolean realizability*:  
the probability of a time process in which at least one event is forced to occur at  
a precise date can be null. The third contribution of the paper is the study of a  
90 quantitative notion of realizability ensuring that a schedule can be realized up  
to some bounded imprecision, with a non null probability. Uncertainty in the  
systems we consider originates from random delays with low individual penalty.  
We hence do not consider major failures that result in hours of delays: such  
failures call for use of recovery scenarios that lay outside the normal behavior  
of a system, and realizability addresses normal situations perturbed by small  
95 deviations with respect to the expected most probable values. These deviations  
are defined via continuous probability distributions for the actual value that  
a duration may take. We use transient analysis techniques [3] for STPNs to  
compute the probability of occurrence of the set of processes that realize a

100 schedule, and consequently prove its realizability up to an imprecision of  $\alpha$  with a strictly positive probability. The last contribution of the paper is the application of boolean and quantitative realizability to a practical case study, namely verification of correct scheduling of train trips in a metro network.

105 The paper is organized as follows: Section 2 defines the models used in the paper, namely stochastic time Petri nets with a blocking semantics and schedules, and gives a concurrent semantics to STPNs. Section 3 defines a notion of unfolding and symbolic process for STPNs. 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. Section 5 shows how to check realizability on a case study, namely correct realization of trips in a train network. Section 6 compares the contents of this paper with former models and former works related to scheduling, before conclusion. Due to lack of space, some proofs and several technical details are omitted, but can be found in Appendices.

## 2. Formal models for Timetables and metro networks

### 115 2.1. Preliminaries

For simplicity, we will consider that random durations appearing in our systems are sampled from 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  $f_X : \mathbb{R} \rightarrow [0, 1]$  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)  $F_X : \mathbb{R} \rightarrow [0, 1]$  for  $X$  describes the probability for  $X$  to take a value less than or equal to a chosen value. We denote by  $\Sigma_{\text{pdf}}$  the set of PDFs,  $\Sigma_{\text{cdf}}$  the set of CDFs, and we 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  $F_X(x) = \int_0^x f_X(y) dy$ .

125 Several standard solutions can be used to model continuous distributions: Gaussian distributions, exponential laws. . . A drawback of Gaussian distributions is that, considering a duration as a randomly chosen value  $x$  around a pivot value  $a$ , a Gaussian distribution attributes the same probability to event  $x < a$  and  $x > a$ . Intuitively, if such a distribution is used to represent a trip duration for a train, this train has the same probability to be late and to be in advance. Everyday's experience shows that the probability for a train to be delayed is higher, so Gaussian distributions do not match exactly the need for asymmetric distributions in transport systems. In the rest of the paper, we will adopt distributions defined with truncated expolynomial functions, that have all the features needed to model random delays. A *truncated expolynomial* function is a function  $f(x)$  defined over an interval  $[a, b]$  as a sum of  $K \in \mathbb{N}$  weighted exponentials of the form:

$$f(x) = \begin{cases} \sum_{k \in 1 \dots K} c_k \cdot x^{a_k} \cdot e^{-\lambda_k \cdot x} & \text{if } x \in [a, b] \\ 0 & \text{otherwise} \end{cases}$$

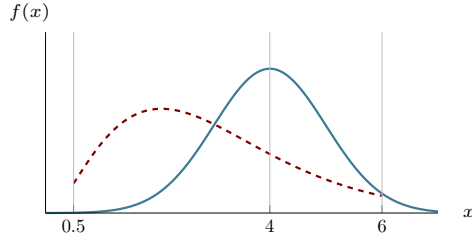


Figure 1: Expressing probability of a delay with a Gaussian distribution (plain line) and an expolynomial function (dashed line).

When  $f(x)$  defines a distribution, the integral  $\int_a^b f(x) dx$  must be equal to 1. Figure 1 illustrates the advantages of using expolynomial functions to model *continuous* distributions. The curves represented in this figure are a standard Gaussian distribution centered around value 4, and an expolynomial function  $f(x) = 0.58 \cdot x^2 \cdot e^{-1.7x} + 0.29 \cdot x^3 \cdot e^{-1.2x}$  defined over domain  $[0.5, 6]$ . As already explained in [3], expolynomial functions can be used to model phase functions, that is functions where distributions have the shape of several bell-like curves centered around several pivot values. This is particularly interesting, for instance, to represent speed profiles of trains: pivot values correspond to choices of a targeted travel time, and pseudo-bell shapes around these values to perturbations of these target durations. Last, expolynomial functions compose well and are easy to manipulate: joint distributions are products of exponentials, projections on a variable or on a domain can be done as simple integrations, and remain expolynomial functions.

## 2.2. Petri nets

Petri nets are good models to represent workflows (e.g., to show how trains can move in a network), but are of little use without timing information and randomness. A frequent question asked for train networks is whether the service is properly ensured. This means in particular whether trains serve stations at a regular enough pace, or in case departure dates of trains are very sparse, whether trains arrivals and departures match an expected timetable. This question can be addressed only in a timed setting, where actions take time, but also where timed guards can force execution of actions after a measured delay (this is captured by the essential notion of *urgency*). We propose a variant of stochastic time Petri nets inspired by [3] that handles time, urgency, and randomness, with a blocking semantics allowing to represent competition for shared resources.

**Definition 1 (stochastic time Petri net).** *A stochastic time Petri net (STPN for short) is a tuple  $\mathcal{N} = \langle P, T, \bullet(), ()^\bullet, m_0, \text{eft}, \text{lft}, \mathcal{F}, \mathcal{W} \rangle$  where  $P$  is a finite set of places;  $T$  is a finite set of transitions;  $\bullet(): T \rightarrow 2^P$  and  $()^\bullet: T \rightarrow 2^P$  are pre and post conditions depicting from which places transitions consume tokens, and to which places they output produced tokens;  $m_0: P \rightarrow \{0, 1\}$  is the initial*

marking of the net;  $\text{eft} : T \rightarrow \mathbb{Q}_{\geq 0}$  and  $\text{lft} : T \rightarrow \mathbb{Q}_{\geq 0} \cup \{+\infty\}$  respectively specify the minimum and maximum time-to-fire that can be sampled for each transition; and  $\mathcal{F} : T \rightarrow \Sigma_{\text{pdf}}$  and  $\mathcal{W} : T \rightarrow \mathbb{R}_{> 0}$  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  $[\text{eft}(t), \text{lft}(t)]$ . The semantics of STPNs can be summarized in a few lines: markings associate 0 or 1 token to places. A transition is enabled if places in its preset are filled. When a transition  $t$  becomes enabled, a value  $x_t$  is sampled from the interval and the distribution attached to  $t$ . Then, this value decreases over time. A transition can fire as soon as its clock has reached 0, its preset is filled, and its postset is empty. We give a precise semantics for STPNs in section 2.3.

To illustrate how such nets can be used, consider the piece of network in Figure 2. A similar mechanism is used later in section 5 to model a simple shunting mechanism in a train network. Places are used to represent tracks (tokens symbolize trains), and transitions model departures and arrivals of trains. Once at station  $D$  (represented by place  $P_D$ ), a train can be controlled to move towards station  $E$ , it will then enter the track portion from  $D$  to  $E$ , represented by place  $P_{D,E}$ , or it will follow another itinerary, and enter another track (represented by place  $P_{D,\bar{D}}$ ) to move towards station  $\bar{D}$ . This decision is represented by transition  $t_2$ . When a train is on its way to station  $E$ , entering station  $E$  (represented by place  $P_E$ ) is symbolized by firing of transition  $t_3$ . Decision to fire  $t_1$  or  $t_2$  is controlled by additional places (the dotted places  $P_{c1}$  and  $P_{c2}$ ). One can notice that firing  $t_1$  empties  $P_{c1}$  and fills  $P_{c2}$ , and firing  $t_2$  empties  $P_{c2}$  and fills  $P_{c1}$ . With such a simple controller, it is assumed that one train out of two goes from  $D$  to  $E$ , and the other goes from  $D$  to  $\bar{D}$ . The distributions  $f_1, f_2, f_3$  attached to transitions  $t_1, t_2, t_3$  are represented in the right part of Figure 2. Intuitively, a train staying at station  $D$  will leave after a sojourn time comprised between 60 and 80 seconds, as long as its destination place is empty. Sojourn times in place  $P_D$  have different distributions in interval  $[60, 80]$ , depicted by function  $f_1, f_2$ , and that depend on whether the train leaves for station  $E$  or station  $\bar{D}$ . The trip from  $D$  to  $E$  lasts between 130 and 140 seconds, and the distribution of trip duration is depicted by function  $f_3$ . Now, if a train is ready to leave station  $D$  to go to  $E$  after a sufficient dwell time, but a train already occupies the track portion from  $D$  to  $E$ , then departure is forbidden. In our model, this is implemented by the blocking semantics that says that a transition can fire only when its postset is empty. In particular, in the model of Figure 2, this means that a token will enter place  $P_{D,E}$  (resp. place  $P_{D,\bar{D}}$ ) at earliest 60 time units after  $P_D$  was filled and at latest 80 time units after if  $P_{c1}$  (resp.  $P_{c2}$ ) contains a token. However, even if  $P_{c1}$  (resp.  $P_{c2}$ ) contain a token, there is no guarantee that the sojourn time of a token in  $P_D$  is smaller than 80 seconds, despite urgency. Indeed, if places  $P_{c2}$  and  $P_{D,E}$  are filled, transition  $t_1$  has to wait for  $P_{D,E}$  to be empty to fire, which may occur at

earliest 130 seconds and at latest 140 seconds later.

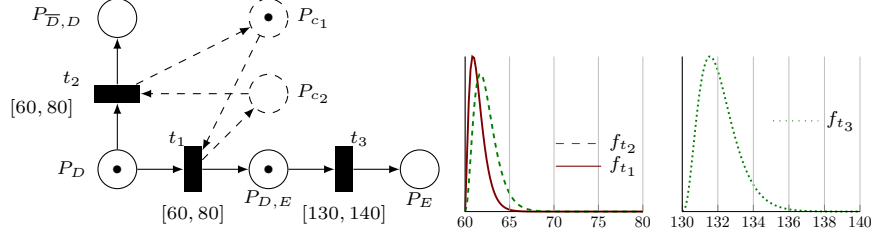


Figure 2: Using STPNs to model trains moves

A possible execution for the STPN of Figure 2 is as follows. Assuming that the value sampled for  $t_1$  is  $x_{t_1} = 62$ , and the value sampled for  $t_3$  is  $x_{t_3} = 132$ , then clock  $x_{t_1}$  expires after 62 time units, but  $t_1$  cannot fire as place  $P_{D,E}$  is filled. Hence, one has to wait 132 time units, fire  $t_3$ , and then fire  $t_1$ . In terms of timed word, this execution can be represented as  $w = \langle t_3, 132 \rangle \cdot \langle t_1, 132 \rangle$ . Equivalently, one can describe this execution as a *time process*  $TP$  of  $\mathcal{N}$ . Roughly speaking, a time process unfolds the net and associates an execution date to occurrences of transitions (the formal definition of time processes and their construction is given in Section 2.3). The time process corresponding to timed word  $w$  is given in Figure 3. Note on this example that even if the first occurrences of  $t_1$  and  $t_3$  seem to be concurrent (they occur at the same dates, and are not causally related), the blocking semantics imposes that  $t_3^1$  fires before  $t_1^1$ .

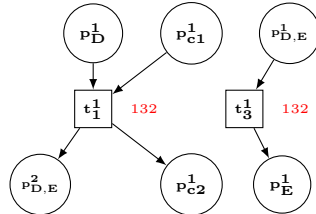


Figure 3: Time process for the net of Figure 2

### 2.3. Semantics of Stochastic Time Petri Nets

Roughly speaking, an STPN is a time Petri net with distributions on firing times attached to transitions. The semantics of this model describes how tokens move 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. This model borrows its main features from [3], with the major difference that the semantics is *blocking*, i.e. it forces nets to remain *safe* (1-bounded), as in *elementary nets*. This restriction is justified by the nature of the systems we address: in production chains, places



symbolize tools that process only one item at a time. Similarly, when modeling  
 225 train networks, security requirements impose that two trains cannot occupy the  
 same track portion. 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 all time and probability issues of  
 230 the original net. Enforcing a bound via a blocking semantics is hence a practical  
 way to guarantee a priori that models do not allow specification of undesired  
 situations.

Let  $\mathcal{N} = \langle P, T, \bullet(\cdot), (\cdot)^\bullet, m_0, \text{eft}, \text{lft}, \mathcal{F}, \mathcal{W} \rangle$  be a STPN. A *marking* of  $\mathcal{N}$  is a  
 function that assigns 0 or 1 token to each place  $p \in P$ . We will say that a  
 235 transition  $t$  is *enabled* by a marking  $m$  iff  $\forall p \in \bullet t, m(p) = 1$ . We denote by  
 $\text{enab}(m)$  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  
 240 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 described by timed and discrete moves among  
 245 configurations, and can be described informally as follows. A variable  $\tau_t$  is  
 attached to each transition  $t$  of the STPN. As soon as the preset of a transition  
 $t$  is marked,  $\tau_t$  is set to a random value  $\zeta_t$  (called the *time-to-fire* of  $t$ , or TTF  
 for short) sampled from  $[\text{eft}(t), \text{lft}(t)]$  according to  $f_t$ . We will assume that  
 every CDF  $F_t$  is strictly increasing on  $[\text{eft}(t), \text{lft}(t)]$ , which allows to use *inverse*  
 250 *transform sampling* to choose a value (see for instance [9] for details). Intuitively,  
 this TTF represents a duration that *must* elapse before firing  $t$  once  $t$  is enabled.  
 The value of  $\tau_t$  then decreases as time elapses but cannot reach negative values.  
 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.  
 255 When the TTF of a transition  $t$  reaches 0, then if  $t^\bullet$  is empty in  $m_{\text{tmp}}$ ,  $t$  becomes  
*urgent* and has to fire unless another transition with TTF 0 and empty postset  
 fires; otherwise (if  $t^\bullet$  is not empty in  $m_{\text{tmp}}$ ),  $t$  becomes *blocked*: its TTF stops  
 decreasing and keeps value 0, and its firing is delayed until the postset of  $t$   
 becomes empty; in the meantime,  $t$  can be disabled by the firing of another  
 260 transition. If more than one transition is urgent, then the transition that fires  
 is randomly chosen according to the respective weights of urgent transitions.  
 Weight are used only when several transitions are fireable at the same instant.  
 In general, the probability that two transitions become urgent at the same instant  
 is null, but in some degenerate models where time intervals are singletons, such  
 265 situation can occur. In this situation, weights are arbitrary values set to favor  
 transitions that are the most probable. In a train setting, for instance, if a  
 freight train and a passenger train can leave at the same instant, one can favor  
 departure of the passenger train by assigning a high weight to the transition  
 symbolizing its departure, and a low weight to the other departure. In general,

270 one may also assume that all transitions have the same weight, and that the  
urgent transition that fires first is drawn from an uniform discrete distribution  
on these transitions. We formalize the semantics of STPNs in terms of discrete  
and timed moves between *configurations* that memorize markings and TTFs for  
enabled transitions.

275 **Definition 2 (configuration of an STPN).** A configuration of an STPN is  
a pair  $\mathcal{C}_N = \langle m, \tau \rangle$  where  $m$  is a marking, and  $\tau : \text{enab}(m) \rightarrow \mathbb{R}_{\geq 0}$  is a function  
that assigns a positive real TTF  $\tau_i = \tau(t_i)$  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 3 (firable and blocked transitions).** A transition  $t$  is firable in  
280  $\langle m, \tau \rangle$  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  $\text{fira}(\langle m, \tau \rangle)$  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  
 $\text{blck}(\langle m, \tau \rangle)$  the set of blocked transitions in  $\langle m, \tau \rangle$ .

285 **Timed moves:** A timed move  $\langle m, \tau \rangle \xrightarrow{\delta} \langle m, \tau' \rangle$  lets a strictly positive duration  
 $\delta \in \mathbb{R}$  elapse. To be allowed,  $\delta$  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  $\delta$  time units ( $\tau'(t) = \tau(t) - \delta$ ).  
Blocked transitions keep a TTF of 0, and  $m$  remains unchanged. Timed moves  
290 are formally defined by the following operational rule:

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

**Discrete moves:** A discrete move  $\langle m, \tau \rangle \xrightarrow{t} \langle m', \tau' \rangle$  consists in firing a transi-  
tion  $t$  from a configuration  $\langle m, \tau \rangle$  to reach a configuration  $\langle m' = m - \bullet t + t^\bullet, \tau' \rangle$ .  
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  
295 of discrete moves, we first introduce newly enabled transitions.

**Definition 4 (newly enabled transitions).** Let  $m$  be a marking and  $t$  a tran-  
sition 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  $\text{newl}(m, t) = \text{enab}(m') \cap (\{t\} \cup (T \setminus \text{enab}(m - \bullet t)))$  the  
300 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  $\tau'$  is  
obtained by sampling a new TTF for every newly enabled transition and keeping  
unchanged TTFs of transitions already enabled by  $m$  and still enabled by  $m'$ .

305 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  $\text{pers}(m, t) = (\text{enab}(m) \cap \text{enab}(m_{tmp})) \setminus \{t\}$  the set of persistent transitions after firing of  $t$  from  $m$ . Discrete moves are formally defined by the following operational rule:

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

310 We will consider that runs of a STPN  $\mathcal{N}$  start from an initial configuration  $\langle m_0, \tau_0 \rangle$  where  $m_0$  is the initial marking of  $\mathcal{N}$ , and  $\tau_0$  attaches a sampled TTF to each transition enabled by  $m_0$ . We will write  $\langle m, \tau \rangle \rightarrow \langle m', \tau' \rangle$  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  $\langle m', \tau' \rangle$ .

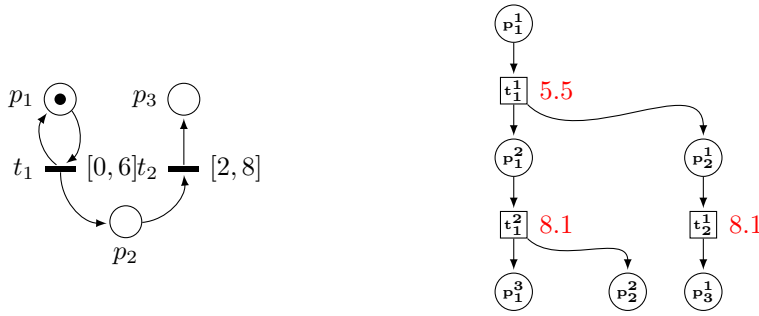


Figure 4: a) An example STPN  $\mathcal{N}_1$  and b) a time process of  $\mathcal{N}_1$

315 Consider the STPN  $\mathcal{N}_1$  of Figure 4, 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  $\tau''(t_1) = 0$ ,  $\tau''(t_2) = 1.1$ , and  $t_1$  is blocked. After letting 1.1 time units elapse, transition  $t_2$  can fire, leading to marking  $m''(p_1) = m''(p_3) = 1$ ,  $m''(p_2) = 0$ . Once  $t_2$  has fired,  $t_1$  is urgent and firable, and immediately fires, at the same date.

325 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. 330 In a discrete move  $\langle m, \tau \rangle \xrightarrow{t} \langle m', \tau' \rangle$ ,  $m'$  is built deterministically, but  $\tau'$  is

obtained by sampling a random value  $\zeta_t$  for each newly enabled transition  $t$ . Each  $\zeta_t$  is chosen according to CDF  $F_t$ , i.e., we have  $\mathbb{P}(\zeta_t \leq x) = F_t(x)$  (for any  $x \in [\text{eft}(t), \text{lft}(t)]$ ). When more than one transition is fireable from  $\langle m, \tau \rangle$ , the transition that fires is randomly chosen according to the respective weight of each fireable transition: each transition  $t_k$  in  $\text{fira}(\langle m, \tau \rangle)$  has a probability to fire  $\mathbb{P}_{\text{fire}}(t_k) = \mathcal{W}(t_k) / \sum_{t_i \in \text{fira}(\langle m, \tau \rangle)} \mathcal{W}(t_i)$ . Note that, as STPNs have continuous probability laws, the probability to choose a particular value  $\zeta_t$  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 probabilities.

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  $\rho$ , 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  $\mathcal{L}(\mathcal{N})$  with a maximal date lower than  $D$ .

#### 2.4. Partial order semantics of Stochastic Time Petri Nets

As already highlighted in [10] 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  $CN = \langle B, E, \bullet(\cdot), (\cdot)^\bullet \rangle$ , where for every  $b \in B$ ,  $|b^\bullet| \leq 1$  and  $|\bullet b| \leq 1$ . Intuitively, a causal net contains no conflict (pairs of transitions with common places in their presets) nor places receiving tokens from more than one transition.

**Definition 5 (time process).** A time process is a tuple  $TP = \langle CN, \theta \rangle$ , where  $CN = \langle B, E, \bullet(\cdot), (\cdot)^\bullet \rangle$  is a causal net, and  $\theta : E \rightarrow \mathbb{R}_{\geq 0}$  associates a positive real date to transitions of net  $CN$ , 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  $\langle B \cup E, \bullet(\cdot) \cup (\cdot)^\bullet \rangle$ . We will write  $e \prec e'$  iff  $e^\bullet \cap \bullet e' \neq \emptyset$ , 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 of the underlying STPN. We denote by  $\text{tr}(e)$  the transition  $t$  attached to an event  $e$ , and by  $\text{pl}(b)$  the place  $p$  associated with a condition  $b$ . To differentiate occurrences of transitions firings, an event will be defined as a pair  $e = \langle X, t \rangle$ , where  $t$  is the transition whose firing is represented by  $e$  and  $X$  is the set of conditions it consumes. Similarly, a condition

is defined as a pair  $b = \langle p, e \rangle$ , where  $p$  is the place whose filling is represented by  $b$ , and  $e$  is the event whose occurrence created  $b$ . 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 often drop flow relations and simply refer to time processes as triples  $TP = \langle B, E, \theta \rangle$ .

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  $TP_u = \langle B, E, \bullet(\cdot), (\cdot)^\bullet, \theta \rangle$ . The construction described below is the same as in [10]. It does not consider probabilities and, as the construction starts from an executable word, it does not have to handle blockings either. We denote by  $TP_u$  the time process obtained from a timed word  $u = \langle t_1, d_1 \rangle \langle t_2, d_2 \rangle \dots \langle t_k, d_k \rangle \in \mathcal{L}(\mathcal{N})$ . It can be built incrementally by adding transitions occurrences with their associated dates one after another, starting from a set of conditions  $B_0 = \{(\perp, p) \mid p \in m_0\}$ . We give a detailed construction in Appendix A.

Figure 4-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  $j^{\text{th}}$  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: for a pair of events  $e, e'$  with  $e \neq e'$ ,  $e$  must precede  $e'$  in a linearization of a process if  $\theta(e) < \theta(e')$  or if  $e \preceq e'$ . With blocking semantics, some causality and time-preserving interleavings may not be valid timed words of  $\mathcal{L}(\mathcal{N})$ : in the process of Figure 4-b,  $t_1^2$  cannot occur before  $t_2^1$ , even if both transitions have the same date. A correct ordering among events with identical dates in a process  $TP_u$  can however be found by checking that a chosen ordering does not prevent occurrence of other transitions.

## 2.5. Schedules

Schedules are objects of everyday's life, and are both used as documentation for a system (for instance, bus and train schedules indicate to users when and where to pick up their transport), and to guide a system along behaviors that guarantee some quality of service. Though schedules are often perceived as linear orderings of actions, a partial ordering of actions is often useful, for instance to order tasks in a Gantt diagram, manage bus fleets, or train moves in metro networks...

Blocking semantics allows to handle safety requirements for systems with critical resources. However, this semantics makes reasoning on systems harder, and brings no guarantee on the liveness of a system. On our example of Figure 2, a possible wish of designers is that trains stay no longer than 80 seconds at station  $D$ , and one train out of two goes to  $E$ . The second requirement is easily implemented by the simple additional control represented by the dashed places and flows. However, nothing guarantees a priori the first requirement, i.e., that place  $P_{D,E}$  is emptied at latest 80 seconds after a token has entered  $P_D$ . So,

even if the model seems correct, trains may have to wait longer than expected a  
420 priori. The objective of this paper is to make sure that a system can implement  
a predetermined schedule. The principle of schedules appears in many places of  
our everyday lives. Trains, metros, buses, follow predetermined schedules, also  
called *timetables*. The view that everyday users of transports have of a schedule  
is only a partial view corresponding to the station where they catch a train or  
425 a bus, listing all possible departure hours. Now, this list of dates is simply a  
projection of more complex objects.

Timetables describe a trajectory for every object: for instance the list of  
stops that a bus has to visit, and the arrival/departure dates from these stops.  
Distinct objects cannot use the same resource at the same date. Hence, although  
430 individual trajectories can be seen as very linear, and could for instance be  
modeled as timed words, dependencies exist: a train can enter a station only  
after its predecessor on the same line has left it.

The natural notion to encode timetables is hence that of partially ordered  
sets of events decorated with dates. Events represent the beginning/end of a task,  
435 the arrival/departure of a train or bus, etc. Partial ordering among events allows  
to account for linearity in individual trajectories and for causal dependencies due  
to exclusive resource use (a train can enter a station only after its predecessor  
has left). The dates decorating these partial order are dates that comply with  
the dependencies, and also with some physical characteristics of the modeled  
440 systems: a train move from a station  $S_i$  to the following one  $S_{i+1}$  in its itinerary  
takes time, which shall be reflected by the dates attached to departures and  
arrivals at different stations.

A schedule can be seen as a solution for a set of constraints over variables  
representing dates of events. Even if a proposed solution satisfies all constraints  
445 attached to a system, it is usually a high-level view of the overall expected  
behavior of a system. This raises the question of whether a particular schedule  
can be effectively realized by the system. As schedules and systems models are  
descriptions of the same system, one can expect the answer to be positive in  
most of cases. However, solutions returned by a solver for a constraint problem  
450 are optimal solutions w.r.t. some criteria, but not necessarily the most plausible  
nor the most robust ones. Indeed, for obvious reasons, one cannot ask a bus to  
reach each stop at the earliest possible date: such solution makes systems poorly  
robust to random delays, that necessarily occur in transport systems. Providing  
the ability to check that a schedule is realizable with reasonable chances is hence  
455 an essential tool to design schedules.

In many transport systems, schedules are only ideal behaviors of vehicles, that  
are realized up to some imprecision, and used mainly to guide the system. Systems  
such as bus fleets or metros often deviate from the expected timing prescribed  
by a timetable at peak hours. This deviation mainly concerns occurrence dates  
460 of events. Re-ordering of events is rare, and mainly occurs in case of a major  
failure of the system that forces working in a degraded setting, where following  
a timetable makes no sense. Systems are also adaptive: production cells can  
be equipped with controllers, bus drivers receive instructions to avoid bunching  
phenomena [11], and metro systems are controlled by regulation algorithms that

465 help trains sticking to predetermined schedules. In this setting, a key question  
 is to asses how much regulation and control is needed to make a system run  
 as expected in its schedule. A first way to answer this question is to ask the  
 probability that a schedule is realized by the system. In section 4.2, we give ways  
 to compute the probability that a schedule is realized up to some minor shift  
 470 in occurrence dates of its events. This measure does not consider adaptation  
 techniques (control, regulation) that help a system stick to a predetermined  
 schedule. However, this measure still makes sense, as it quantifies the need for  
 correction in unregulated systems.

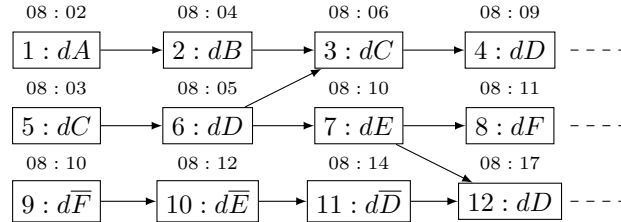


Figure 5: A possible schedule for trains departures in a metro network

475 Consider the example of Figure 5. This Figure represents the beginning  
 of a schedule for three trains, where only departure dates are planned. The  
 departures for each train are depicted as boxes, carrying a label of the form  $i : dJ$ ,  
 where  $i$  is the event number, and  $dJ$  means that this event is a departure from  
 station  $J$ . Furthermore, an execution date is attached to each node. One can see  
 on this drawing that schedules are partial orders containing train trajectories.  
 480 However, there are some dependencies among events from distinct trains: for  
 instance, our schedule imposes that the second departure of the day at station  $D$   
 (event 6 :  $dD$ ) precedes departure 3 :  $dC$  from station  $C$ . Similarly, 7 :  $dE$  must  
 precede 12 :  $dD$ . Now, a sensible question is: can a network realize this schedule?  
 More precisely, can the STPN designed for this network (with additional control  
 485 places) realize such a schedule starting at 08 : 00 with a train at station  $A$ ,  
 a train at station  $C$  and a train at station  $\bar{F}$ ? The answer to this question is not  
 straightforward. Even if the answer is positive, the next step is to ask whether  
 this schedule, with a tolerance of 1 min delay for each departure, is probable  
 enough. If the answer is that the probability to realize our schedule is very low,  
 490 then this schedule should not be considered as operational.

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 6 (schedule).** A schedule over a finite alphabet  $\mathcal{A}$  is a quadruple  
 495  $S = \langle N, \rightarrow, \lambda, C \rangle$  where  $N$  is a set of nodes,  $\rightarrow \subseteq N \times N$  is an acyclic precedence  
 relation,  $\lambda : N \rightarrow \mathcal{A}$  is a labeling of nodes, and  $C : N \times N \rightarrow \mathbb{Q}_{>0}$  is a partial  
 function that associates a time constraint to pairs of nodes. A dating function for

a schedule  $S$  is a function  $d : N \rightarrow \mathbb{Q}_{\geq 0}$  that satisfies all constraints of  $C$  and  $\rightarrow$ , i.e.,  $\langle n, n' \rangle \in \rightarrow$  implies  $d(n') \geq d(n)$ , and  $C(n, n') = x$  implies  $d(n') - d(n) \geq x$ .

500 This model for schedules is inspired from [1, 2]. Intuitively, if  $C(n, n') = x$ , then  $n'$  cannot occur earlier than  $x$  time units after  $n$ , and if  $\langle n, n' \rangle \in \rightarrow$ , 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  $\langle N, \rightarrow \cup \{\langle n, n' \rangle \mid C(n, n') \text{ is defined}\} \rangle$  does not contain cycles. 505 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. 510 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 515 trade-off between maximization of an objective and of the likelihood to stay close to optimal realizations at runtime.

**Realizability:** Accuracy of schedules can be formally defined as follows: we want to check whether a consistent schedule  $S$  can be realized by a system, described as a STPN  $\mathcal{N}$ . More formally, this amounts to verifying that there 520 exists a timed execution  $TP$  of  $\mathcal{N}$  and an embedding function  $\psi$  mapping abstract events of  $S$  onto concrete events of  $TP$ , such that causally related events of  $S$  are causally dependent in  $TP$  and dates of events in  $TP$  meet the constraints on their abstract representation. In the rest of the paper, we show how to check realizability of a schedule  $C$  by a STPN  $\mathcal{N}$ , and how to compute a lower bound 525 on the probability that  $S$  is realized by  $\mathcal{N}$ . This allows in particular to check that the probability of realization of a schedule is not null.

### 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 530 as the process of Figure 4-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 535 *unfoldings*, i.e., structures that contain all symbolic processes of an STPN, but factorize common prefixes. Symbolic unfoldings were introduced for TPNs in [12] and used in [13]. 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 540 schedule of known duration. We show that if a system cannot execute arbitrary



large sets of events without progressing time, unfolding 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  $u = \langle t_1, d_1 \rangle \dots \langle t^i, \theta_1 \rangle \dots \langle t^{i+1}, \theta_2 \rangle \dots \langle t_k, d_k \rangle \in \mathcal{L}(\mathcal{N})$  where  $t^i$  denotes the  $i^{\text{th}}$  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). The example of Figure 4 does not guarantee time progress as according to the semantics of STPNs, this net allows an arbitrary number of occurrences of transition  $t_1$  to fire at date 0. However, such specification can be considered as ill-formed. Let us consider our motivating examples: in a production cell, the same kind of processing by a tool necessarily takes time, and conveying an item from a tool to another cannot either be instantaneous. Similarly, in train networks, some time must elapse between two consecutive arrivals of a train at a station, as well as between two consecutive arrivals/departures of the same train. Hence, real-life systems ensure properties that are often stronger than this time progress property. Many STPNs modeling real-life cases contain exclusively transitions with rational intervals of the form  $[a, b]$  or  $[a, \infty)$ , where  $a > 0$ . An interesting consequence of time progress is that any execution of duration  $\Delta$  of an STPN that guarantees time progress is a sequence of at most  $|T| \cdot \lceil \frac{\Delta}{\delta} \rceil$  transitions. It means that for most of usual systems that run for a predetermined period (e.g., a metro network operates from 05:00 to 01:00), it is sufficient to consider behaviors of a model up to a bounded horizon.

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  $\text{doe}(e)$ ,  $\text{dof}(e)$  and  $\theta(e)$  that respectively define the enabling, firability and effective firing date of the occurrence of transition  $\text{tr}(e)$  represented by event  $e$ . Similarly, we associate to each condition  $b$  positive real valued variables  $\text{dob}(b)$  and  $\text{dod}(b)$  that respectively represent the date of birth of the token in place  $\text{pl}(b)$ , and the date at which the token in place  $\text{pl}(b)$  is consumed. We denote by  $\text{var}(E, B)$  the set of variables  $\bigcup_{e \in E} \text{doe}(e) \cup \text{dof}(e) \cup \theta(e) \cup \bigcup_{b \in B} \text{dob}(b) \cup \text{dod}(b)$  (with values in  $\mathbb{R}_{\geq 0}$ ). A *constraint* over  $\text{var}(E, B)$  is a boolean combination of atoms of the form  $x \bowtie y$ , where  $x \in \text{var}(E, B)$ ,  $\bowtie \in \{<, >, \leq, \geq\}$  and  $y$  is either a variable from  $\text{var}(E, B)$  or a constant value. A set of constraints  $C$  over a set of variables  $V$  is *satisfiable* iff there exists at least one valuation  $v : V \rightarrow \mathbb{R}$  such that replacing each occurrence of each variable  $x$  by its valuation  $v(x)$  yields a tautology. We denote by  $\text{SOL}(C)$  the set of valuations that satisfy  $C$ .

**Definition 8 (unfolding).** *A (structural) unfolding of an STPN  $\mathcal{N}$  is a pair  $U = \langle E, B \rangle$  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  $e = \langle \bullet e, \text{tr}(e) \rangle$  where  $\bullet e \subseteq B$  is the set of predecessor

585 conditions of  $e$  (the conditions needed for  $e$  to occur). A condition  $b \in B$  is  
 a pair  $b = \langle \bullet b, \text{pl}(b) \rangle$  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  $\bullet(\cdot)$ ,  $(\cdot)^\bullet$ ,  $\text{pl}(\cdot)$  and  $\text{tr}(\cdot)$  keep the  
 same meaning as for time processes. The main difference between processes  
 and unfoldings is that conditions may have several successor events. Using  
 590 relations  $\prec$  and  $\preceq$  as defined for processes, we define the *causal past* of  $e \in E$   
 as  $\uparrow e = \{e' \in E \mid e' \preceq e\}$ . A set of events  $E' \subseteq E$  is *causally closed* iff  
 $\forall e \in E', \uparrow e \subseteq E'$ . This notion extends to conditions. Two events  $e, e'$  are in  
*conflict*, and write  $e \# e'$ , iff  $\bullet e \cap \bullet e' \neq \emptyset$ . A set of events  $E' \subseteq E$  is *conflict free* if  
 it does not contain conflicting pairs of events. Two events  $e, e'$  are *competing* iff  
 595  $\text{tr}(e)^\bullet \cap \text{tr}(e')^\bullet \neq \emptyset$  (they fill a common 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 a maximal (i.e., there  
 is no larger pre-process containing  $E', B'$ ), causally closed and conflict free set  
 of events, and  $B' = \bullet E' \cup E'^\bullet$ .  $\mathcal{PE}(\mathcal{U})$  denotes the set of pre-processes of  $\mathcal{U}$ .

600 Unfolding an STPN up to depth  $K$  is performed inductively, without con-  
 sidering time. We will then use this structure to find processes. Timing issues  
 will be considered through addition of constraints on occurrence dates of events.  
 Unfoldings can be built inductively, following the procedure proposed for in-  
 stance in [5] and recalled in Appendix B. When building  $\mathcal{U}_0, \dots, \mathcal{U}_K$ , each step  
 605  $k$  adds new events at depth  $k$  and their postset to the preceding unfolding  
 $\mathcal{U}_{k-1}$ . The construction starts with the initial unfolding  $\mathcal{U}_0 = \langle \emptyset, B_0 \rangle$  where  
 $B_0 = \{\langle \perp, p \rangle \mid p \in m_0\}$ .

The structural unfolding of an STPN does not consider timing issues nor  
 blockings. Hence, an (untimed) pre-process of  $\mathcal{PE}(\mathcal{U}_K)$  needs not be the un-  
 610 timed 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. However, time processes of a net  $\mathcal{N}$  can be built from  
 615 processes of its untimed unfolding. 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 that satisfy requirement on dates that only depend on the con-  
 sidered pre-process. We re-introduce time in unfoldings by attaching constraints  
 to events and conditions of pre-processes. Let  $\mathcal{U}_K = \langle E_K, B_K \rangle$  be the unfolding  
 620 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  $E, B$  (i.e., defined  
 over set of variables  $\text{var}(E, B)$ ), assuming that executions of  $\mathcal{N}$  start at a fixed  
 date  $d_0$ . Constraints in  $\Phi_{E,B}$  are set to guarantee:

- 625 • (net constraints): occurrence dates of events are compatible with the  
 earliest and latest firing times of transitions in  $\mathcal{N}$ ,

- (causal precedence): if event  $e$  precedes event  $e'$ , then  $\theta(e) \leq \theta(e')$ .
- (no overlapping conditions): if  $b, b'$  represent occurrences of the same place, then intervals  $[\text{dob}(b), \text{dod}(b)]$  and  $[\text{dob}(b'), \text{dod}(b')]$  have at most one common point,
- (urgency): an urgent transition with empty postset fires if no other urgent transition fires before; and, in particular, for every event  $e \in E$ , there is no event that becomes fireable and urgent before  $e$ .

Overall,  $\Phi_{E,B}$  is a boolean combination of inequalities. We do not detail its construction here: except for the additional constraint on place occupancy due to blocking semantics, it is almost the solution proposed by [13]. Interested readers can also find a complete description of the construction of  $\Phi_{E,B}$  in Appendix C. 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 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, and make  $\Phi_{E,B}$  unsatisfiable. Consider for instance the STPN of Figure 6-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  $p_4^1$  and  $p_4^2$  are maximal and represent the same place  $p_4$ .

**Definition 10 (prefixes of an unfolding).** Let  $PP = \langle E, B \rangle$  be a pre-process of  $\mathcal{U}_K$ . A symbolic prefix of  $PP$  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  $B' = \bullet E' \cup E'^\bullet$ . The constraint  $\Phi_{E',B'}$  is the constraint that helps fulfilling net constraints, causality, overlapping and urgency constraints (see appendix Appendix C for exact definition).

Symbolic prefixes are causally closed parts of pre-processes, but their constraints inherited from the unfolding  $\mathcal{U}_K$  may not be satisfiable. Let  $SPP =$

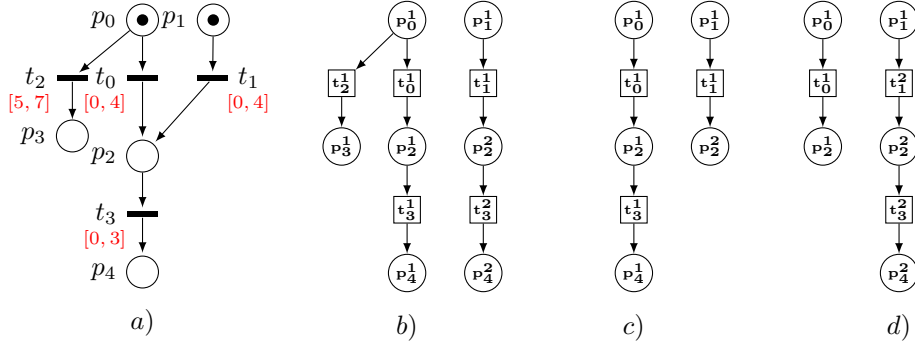


Figure 6: An STPN with conflicts and blockings a), its symbolic unfolding b), and two of its symbolic processes c) and d).

$\langle E', B', \Phi_{E', B'} \rangle$  be a symbolic prefix of pre-process  $PP = \langle E, B \rangle$ . We will say  
 670 that *SPP* is *maximal w.r.t. urgent events firing* iff no more event of  $PP$  **have**  
**to** belong to *SPP*. This property of *SPP* holds if every event  $f \in B' \bullet \cap E$  that  
 could have become urgent before the last date of all events in  $E'$  was prevented  
 from firing due to blocking. We show in Appendix C that this property of  
 symbolic prefixes can be verified as unsatisfiability of a property  $\Phi_{\max}(f)$  for  
 675 every  $f \in B' \bullet \cap E$ .

**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  $\langle E', B', \Phi_{E', B'} \rangle$  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  $PP$ .

680 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 illustrated by the example of Figure 6, satisfiability of constraints is  
 not monotonous: the constraints for processes in Fig 6–c) and d) are satisfiable.  
 However, adding one occurrence of transition  $t_3$  yields unsatisfiable constraints.  
 685 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: if a constraint  
 associated with a prefix of size  $n$  is not satisfiable, appending a new event may  
 introduce blockings that delay urgent transitions, yielding satisfiability of a  
 constraint on a prefix of size  $n + 1$ . So, unsatisfiability of constraints cannot be  
 690 used as a criterion to stop incremental unfoldings construction.

**Definition 12 (executions of symbolic processes).** Let  $\mathcal{E}^s = \langle E, B, \Phi_{E, B} \rangle$   
 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  $\theta$  is a solution for  $\Phi$ . For a chosen  $\theta$ , we denote by  
 $\mathcal{E}_\theta^s = \langle E, B, \theta \rangle$  the time process obtained from  $\mathcal{E}^s$ .  $TP = \langle E, B, \theta \rangle$  is a time  
 695 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  $\delta$  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 that its processes are of depth at most  $H = \lceil \frac{D-d_0}{\delta} \rceil \cdot |T|$ .

Let  $b = \langle e, p \rangle$  be a condition of an unfolding  $\mathcal{U}_n$  obtained at step  $n$ . Let  $\mathbf{block}(b)$  be 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  $\mathbf{block}(b)$  may influence the date of birth and death of  $b$ , or even prevent  $b$  from appearing in the same process as some conditions in  $\mathbf{block}(b)$ . However, in general,  $\mathbf{block}(b)$  need not be finite, and at step  $n$ ,  $\mathbf{block}(b)$  is not fully contained in a pre-process of  $\mathcal{U}_n$ . Fortunately, upon time progress assumption, we can show that elements of  $\mathbf{block}(b)$  that can influence  $\mathbf{dob}(b)$  appear in some bounded unfolding  $\mathcal{U}_K$ .

**Proposition 1.** *Let  $\mathcal{N}$  be a STPN guaranteeing time progress of  $\delta$  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.  $\{b' \in \mathbf{block}(b) \mid \mathbf{dob}(b') \leq D\}$  is contained in  $\mathcal{U}_K$ .*

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

**Proposition 2.** *Let  $\mathcal{N}$  be a STPN guaranteeing time progress of  $\delta$  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 address the question of realizability of a high-level description of operations (a schedule  $S$ ) by a system (described by a STPN  $\mathcal{N}$ ). Considered in a purely boolean setting, this question can be rewritten as: is there an execution of  $\mathcal{N}$  that implements  $S$ ? In many cases, a positive answer to this question is not sufficient: as STPNs are equipped with continuous probability distributions, the probability of a particular execution  $TP = \langle E, B, \theta \rangle$  is always 0. A sensible way to address realizability is a quantitative approach requiring that the set of executions of  $\mathcal{N}$  implementing  $S$  has a positive probability. In this section, we

first formalize the notion of realization of a schedule by an execution, and define boolean realizability. We then define probabilistic realizability, and show how an under-approximation of the probability to realize a schedule can be computed using a transient tree construction [3].

#### 4.1. Boolean realizability of schedules

First of all, the connection between high-level description of operations in  $S$  and their implementation in  $\mathcal{N}$  is defined via a realization function.

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

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 tracks, or to departure with different speed profiles, which is encoded with several transitions in an STPN. Realization functions hence relate actions in schedules to several transitions in an STPN. The condition  $r(a) \cap r(a') = \emptyset$  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. This allows in particular to define schedules that constrain dates for a subset of events, and leave dates of other events free from any constraint. In the context of a train network, this allows for the verification of realizability of schedules that focus on a subset of stations, e.g., requiring a departure from a chosen station every  $x$  minutes during a normal day of operation. We will call transitions that belong to  $r(\mathcal{A})$  *realizations* of  $\mathcal{A}$ .

**Definition 14 (embedding).** *Let  $S = \langle N, \rightarrow, \lambda, C \rangle$  be a schedule,  $\mathcal{E}^s = \langle E, B, \Phi \rangle$  be a symbolic process of  $\mathcal{N}$  and  $r : \mathcal{A} \rightarrow 2^T$  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  $\psi : N \rightarrow E$  such that:*

$$\left\{ \begin{array}{ll} \forall n \in N, \text{tr}(\psi(n)) \in r(\lambda(n)) & \text{(embedding is consistent with labeling)} \\ \forall \langle n, n' \rangle \in \rightarrow, \psi(n) \preceq \psi(n') & \text{(causal precedence is respected)} \\ \nexists f \leq \psi(\min(n)), \text{tr}(f) \in r(\mathcal{A}) & \text{(embedding starts on 1st compatible events)} \\ \forall e \leq f \leq g, e = \psi(n) \wedge g = \psi(n'') \wedge \text{tr}(f) \in r(\mathcal{A}) & \text{(embedding "misses"} \\ \Rightarrow \exists n', f = \psi(n') \wedge n \rightarrow^* n' \rightarrow^* n'' & \text{no compatible event)} \end{array} \right.$$

$S$  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  $\psi$  of any high-level operation between two mapped realizations, or before the image by  $\psi$  of minimal nodes in the schedule. Note that there can be several ways to embed  $S$  into a process of  $\mathcal{N}$ .

**Definition 15 (realizability).** *Let  $d$  be a dating function for a schedule  $S = \langle N, \rightarrow, \lambda, C \rangle$ ,  $r$  be a realization function. The pair  $(S, d)$  is realizable by  $\mathcal{E}^s = \langle E, B, \Phi \rangle$  (w.r.t.  $r$ ) iff there exists an embedding  $\psi$  from  $S$  to  $\mathcal{E}^s$ , and furthermore,  $\Phi_{\psi, S, d} = \Phi \wedge \bigwedge_{n \in N} \theta(\psi(n)) = d(n)$  is satisfiable.  $(S, d)$  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  $\mathcal{E}^s$ .*

Realizability of a schedule  $S = \langle N, \rightarrow, \lambda, C \rangle$  with constraints  $C$  stands for realizability of any of the dating functions  $d$  meeting constraints  $C$ . Very often, we also address realizability of a schedule with respect to a fixed dating function  $d$ . Letting  $C_\psi$  denote the conjunction of inequalities obtained by replacing every assignment  $(i, j) \rightarrow v$  in map  $C$  by inequalities  $\theta(\psi(n_j)) - \theta(\psi(n_i)) \geq v$ , we will say that  $S$  is *realizable* by  $\mathcal{E}^s$  (w.r.t.  $r$ ) iff there exists an embedding  $\psi$  from  $S$  to  $\mathcal{E}^s$  and  $\Phi_{\psi, S, C} = \Phi \wedge C_\psi$  is satisfiable. Similarly, when nodes of schedules are attached a precise date by a map  $d(\cdot)$ , we can define  $C_d = C_\psi \wedge \bigwedge_{n \in N} \theta(\psi(n)) = d(n)$ , and  $\Phi_{\psi, S, d} = \Phi \wedge C_d$ . 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 some symbolic process of  $\mathcal{N}$ . Algorithm 1 in appendix Appendix F computes the set  $\Psi_{S, \mathcal{E}^s}$  of embeddings of a schedule  $S$  in a process  $\mathcal{E}^s$ . If there exists an embedding  $\psi \in \Psi_{S, \mathcal{E}^s}$  from  $S$  to a symbolic process  $\mathcal{E}^s$ , such that  $\Phi_{\psi, S, C}$  (resp.  $\Phi_{\psi, S, d}$ ) is satisfiable, then  $S$  is realizable by  $\mathcal{E}^s$ . Realizability hence consists in finding at least one symbolic process of  $\mathcal{N}$  with an appropriate embedding in  $\psi \in \Psi_{S, \mathcal{E}^s}$ . When a maximal occurrence date  $D$  for operations in  $S$  is provided, and when  $\mathcal{N}$  guarantees time progress, such a process is a process of unfolding  $\mathcal{U}_K$  (where  $K$  is the bound given in Prop. 2). We can then compute the set of symbolic processes  $\mathcal{E}^S = \{\mathcal{E}_0^s, \mathcal{E}_1^s, \dots, \mathcal{E}_{N-1}^s\}$  of  $\mathcal{U}_K$  that embed  $S$  and similarly for each  $\mathcal{E}_i^s \in \mathcal{E}^S$ , the set of possible embedding functions  $\Psi_i = \{\psi_{i,0}, \psi_{i,1}, \dots, \psi_{i, N_i-1}\}$  for which constraint  $\Phi_{\psi, S, C}$  (resp.  $\Phi_{\psi, S, d}$ ) is satisfiable.

To illustrate the construction of unfoldings and of processes, let us consider the example of figure 7. This toy example depicts two train carousels: line 1 serves stations  $A$ ,  $B$  and  $C$ , and line 2 serves stations  $D$ ,  $B'$  and  $C'$ . Both lines share a common track portion between stations  $B, C$  and  $B', C'$ , and line 1 uses two trains. The upper left picture shows the aspect of both lines and stations, and the bottom left figure a STPN model of this network (we do not show distributions). Stations are represented by places labeled by station names, and track portions between two stations by places labeled by pairs of letters representing the connected stations (e.g., place  $CA$  represents the track from  $C$  to  $A$ ). Transitions consuming tokens from a station place represent trains

departures, and transitions consuming tokens from a track place are arrivals. A possible required schedule (middle of the figure) is that one train leaves every 10 time units from station  $A$  on line 1, starting from date 10, and one train leaves station  $B'$  every 10 time units, but starting from date 15. Arrivals of trains and departures from other stations are not represented, and are hence not constrained. Departures from  $A$  are nodes labeled by  $d_A$  and departures from  $B'$  are nodes labeled by  $d_{B'}$ . The rightmost part of the figure is a structural unfolding of the net. We set  $r(d_A) = \{t_5\}$  and  $r(d_{B'}) = \{t_7\}$ . Note 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  $t_1$ , which is a standard conflict. Note also that events  $t_4^1$  and  $t_4^2$  output a token in place  $A$ . Even if these events are not in conflict, due to non-blocking semantics, their firing dates may influence one another. The way operations of the schedule inject in a process of the net is symbolized by dotted lines. Notice that if  $t_5^1$  has to fire at date 10, then according to intervals attached to transitions,  $t_4^1$  has to fire at date 5. This means that there exists a unique way to guarantee a departure from station  $A$  at date 10, which is to sample the smallest trip durations from  $C$  to  $A$  and the smallest possible dwell time at station  $A$ . The schedule of Figure 7 is hence realizable. However, the probability of occurrence of this schedule with precise dates is obviously 0.

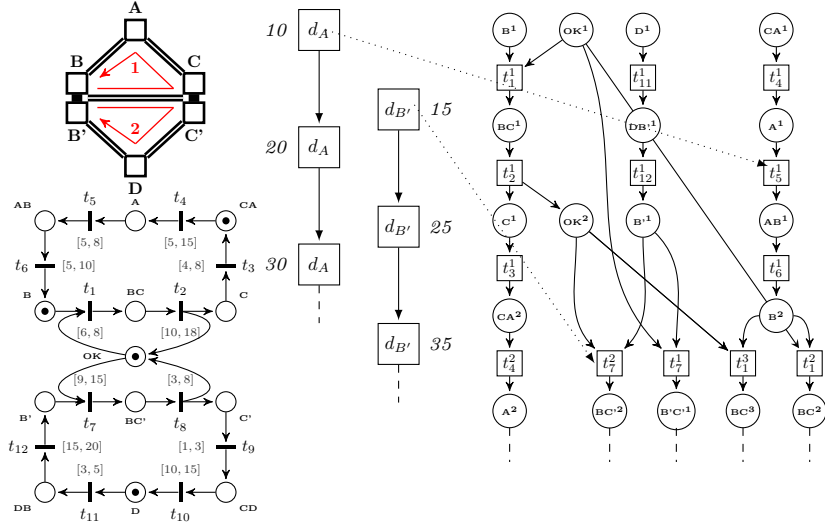


Figure 7: Realizability of a schedule for a metro network with two lines and a shared track.

#### 4.2. Probabilistic Realizability

The example of Figure 7 shows that boolean realizability characterizes an embedding that is consistent with time constraints, but not the probability to realize a schedule. Consider the STPN of Figure 8. This net has two symbolic



840 processes:  $\mathcal{E}_1^s$  in which transition  $t_1$  fires, and  $\mathcal{E}_2^s$  in which  $t_2$  fires. The probability  
 of process  $\mathcal{E}_1^s$  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 probability that  $v_1 \in [0, 1]$   
 (and is hence equal to 1). The probability of the second process  $\mathcal{E}_2^s$  is equal to  
 845 the probability that  $v_1 \geq v_2$ , but the set of values allowing this inequality is  
 restricted to a single point  $v_1 = 1, v_2 = 1$ . Conforming to continuous probability  
 distributions semantics, the probability of this point, and consequently the  
 probability of executing a time process that is consistent with constraints in  $\mathcal{E}_2^s$   
 is 0. A schedule  $S$  composed of a single node  $n$  with a realization function such  
 850 that  $r(\lambda(n)) = \{t_2\}$  and a date  $d(n) = 1$  are realizable according to Definition 14,  
 but with *null probability*. Let us slightly change the example of Figure 8-a).  
 We now assign interval  $[0, 3]$  to transition  $t_1$  in the STPN and interval  $[1, 4]$   
 to transition  $t_2$ . We keep the same schedule  $S$  and realization function  $r$ , but  
 require that  $d(n) = 2$ . The probability that  $t_2$  fires from the initial marking is  
 855 equal to the probability that  $v_1 \geq v_2$ , which is not null (we explain below and  
 in Appendix H how to compute the probability of such domain and the joint  
 probability of  $v_1, v_2$ ), and is equal to the joint probability of values of  $v_1, v_2$  laying  
 in domain  $v_1 \geq v_2$  depicted by the Grey zone in Figure 8-b). However, within  
 this continuous domain of possible values, the probability to fire  $t_2$  exactly at  
 860 date 2 as required by dating function  $d$  is still null. Nevertheless, if  $t_2$  is allowed  
 to fire at date 2 with some imprecision  $\alpha$ , then the probability to realize the  
 expected schedule is equal to the integration of the joint probability distribution  
 over the domain where  $(v_1 \geq v_2) \wedge (2 - \alpha \leq v_2 \leq 2 + \alpha)$  (represented as a dashed  
 part in Figure 8-b), which can be strictly positive if distributions attached to  $t_1$   
 and  $t_2$  are properly set. Note that requiring dates to be implemented up to some  
 865 imprecision does not necessarily increase the probability to realize a schedule: in  
 the example of Figure 8-a with intervals  $[0, 1]$  and  $[1, 2]$  the only way to realize  
 the schedule  $S$  mentioned before with date  $d(n) = 1$  up to some imprecision is to  
 execute the unique time process in which  $t_2$  fires at date 1, and the probability  
 870 of this process is null. More generally, the probability to realize a schedule when  
 the embedding relation leaves a single possible occurrence date for at least one  
 event in the chosen symbolic process is always null.

Boolean realizability is a first step to check that a schedule and an imple-  
 mentation are not totally orthogonal visions of a system. However, examples 7  
 875 and 8 demonstrate that it is not precise enough. They also show that boolean  
 realizability up to imprecision still allows to consider sets of processes with null  
 probabilities as realizations of a schedule. An accurate notion of realizability  
 should require that schedules embed into symbolic processes of  $\mathcal{U}_K$  with strictly  
 positive probability **and** up to some admissible imprecision on dates of events,  
 880 bounded by some value  $\alpha \in \mathbb{Q}_{\geq 0}$ . When a schedule is constrained by a precise  
 map  $d(\cdot)$ , every operation  $x$  in a schedule should now be implemented by an  
 occurrence  $t_i^j$  of a transition  $t$  at date  $\theta(t_i^j) \in [\max(d(x) - \alpha, 0), d(x) + \alpha]$ . Once  
 an injection  $\psi$  from a schedule  $S = \langle N, \rightarrow, \lambda, C \rangle$  to a symbolic process  $\mathcal{E}^s$  is found,  
 the constraint to obtain realizability of a dating function  $d$  up to imprecision  
 885 of  $\alpha$  becomes:  $\Phi_{\psi, S, d \pm \alpha} = \Phi \wedge \bigwedge_{n \in N} \max(d(n) - \alpha, 0) \leq \theta(\psi(n)) \leq d(n) + \alpha$ .

One can similarly require constraints  $C$  in  $S$  to be realized up to imprecision of  $\alpha$ , that is, require that constraints of the form  $d(n_j) - d(n_i) \geq v$  imposed by map  $C$  are implemented in the low level net by a process satisfying a relaxed constraint of the form  $\theta(\psi(n_j)) - \theta(\psi(n_i)) \geq v - \alpha$ . From a practical point of view, this notion of realization up to bounded imprecision is more natural than boolean realizability. Indeed, for systems such as train networks, one cannot expect a schedule to be precisely realized (trains are subject to random delays), but rather that differences between realized and scheduled dates are usually not too important. To measure the probability of realizing a schedule, we define as  $\mathbb{P}(\mathcal{E}^s)$  the probability of the set of time processes of  $\mathcal{E}^s$ . When an embedding  $\psi$  from  $S$  to  $\mathcal{E}^s$  exists, we define as  $\mathbb{P}(\mathcal{E}^s \wedge \text{SOL}(\Phi_{\psi,S,d\pm\alpha}))$  the probability of time processes that are realizations of  $\mathcal{E}^s$  in which dates of events are solutions of  $\Phi_{\psi,S,d\pm\alpha}$ .

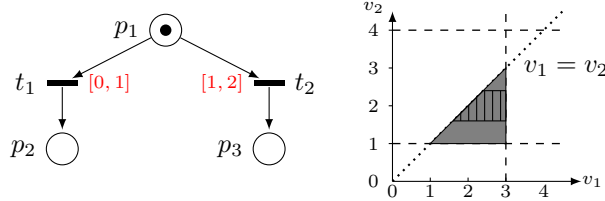


Figure 8: a) An example STPN b) A domain for  $\tau(t_1), \tau(t_2)$  allowing firing of  $t_2$ , assuming  $I(t_1) = [0, 3]$  and  $I(t_2) = [1, 4]$ .

**Definition 16 (probabilistic realizability).** Let  $d$  be a dating function with maximal date  $D$  for a schedule  $S = \langle N, \rightarrow, \lambda, C \rangle$ ,  $r$  be a realization function. The pair  $(S, d)$  is realizable with non-null probability (w.r.t.  $r$ ) up to imprecision  $\alpha$  iff there exists an embedding  $\psi$  of  $S$  into a symbolic process  $\mathcal{E}^s$  of  $\mathcal{U}_K$  such that:  $\mathbb{P}(\mathcal{E}^s \wedge \text{SOL}(\Phi_{\psi,S,d\pm\alpha})) > 0$ .

This definition requires that a symbolic process of  $\mathcal{N}$  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 strictly positive. We will now show how to compute this probability using a transient execution tree, as proposed in [3]. A transient execution tree is a tree which nodes are stochastic state classes.

**Definition 17 (transient stochastic state class).** A transient stochastic state class (or class for short) of an STPN  $\mathcal{N}$  is a tuple  $\Sigma = \langle M, C, D, \text{BLK}, \text{URG} \rangle$  where  $M$  is a marking. For a given stochastic state class, we define a set of variables  $X_M$  with support  $C$  representing the possible TTFs of transitions enabled by marking  $M$  s.t. for every  $x_i$  in  $X_M$ , the elimination of all other variables from  $C$  yields a non-empty set of possible values that is different from  $\{0\}$ ;  $D$  is a PDF over  $C$ ,  $\text{BLK}$  is a set of blocked transitions, and  $\text{URG}$  is a set of urgent transitions.

Roughly speaking, stochastic state classes are abstract representations of markings, time domains for sampled values attached to enabled transitions, and

of their distribution over the abstract domain. The notion of *state class* was  
 920 already used in [14, 15] to analyze time Petri nets, stochastic state class hence  
 only add a probabilistic dimension. In a stochastic state class transitions that  
 are enabled but not blocked nor urgent are associated a time to fire. We will  
 denote by  $x_i$  the variable describing the time to fire associated to a transition  $t_i$ ,  
 and by  $X_M$  the set of all variables attached to enabled transitions in  $M$ . The  
 925 domain  $C$  can be described by a set of inequalities of the form  $x_i - x_j \bowtie v$  or  
 $x_i \bowtie v$  with  $\bowtie \in \{<, >, \leq, \geq, =\}$  and  $v \in \mathbb{Q}$ . It represents possible values for  
 TTFs attached to transitions. Similarly, the distribution  $D$  is a PDF defined on  
 $\mathbb{R}^{|X_M|}$  giving probability of values of  $x_{i_1}, \dots, x_{i_{|X_M|}}$ . When needed, we can also  
 include in  $X_M$  a particular variable  $x_{age}$  representing the time elapsed since the  
 930 beginning of an execution. The probability to fire a particular transition from a  
 class and move to a successor class is computed as an integration over the time  
 domain allowing this transition to fire first. On the example of Figure 8 (with  
 intervals  $[0, 3]$  and  $[1, 4]$ ), this corresponds to integration of a joint distribution  
 for values  $v_1, v_2$  over the domain in Grey.

One can assume that the system under study starts from a known initial marking  
 935  $M_0$  with initial known TTFs, represented as a point of  $\mathbb{R}^{|X_{M_0}|}$ . The construction  
 of the transient tree starts from stochastic state class  $(M_0, C_0, D_0, \text{BLK}_0, \text{URG}_0)$ ,  
 where  $C_0$  is a single point defining known TTFs,  $D_0$  associates probability 1 to  
 the single point in  $C_0$  and 0 to  $\mathbb{R}^{|X_{M_0}|} \setminus C_0$ .

940 Then, the transient tree is built by iterating a construction of successors for  
 already found state classes. Let  $(M, C, D, \text{BLK}, \text{URG})$  be a state class such that  
 $\text{URG} = \emptyset$ , and let  $t_i$  be an enabled transition in this class. Then, one can compute  
 the domain defined by constraint  $C' = C \wedge \{x_i \leq x_j \mid x_i \neq x_j \in X_M\}$ , that  
 imposes that  $t_i$  is the first transition to fire, i.e., it has the smallest TTF in the  
 945 state class. If  $C'$  is satisfiable, then  $t_i$  is effectively fireable, and one can compute  
 the probability  $p_i$  that  $t_i$  fires first, and the successor class  $(M_i, C_i, D_i, \text{BLK}_i, \text{URG}_i)$   
 reached after firing  $t_i$ . First of all, we have  $p_i = \int_{C'} D(x_{i_1} \dots x_{i_{|X_M|}})$ , that is,  
 the probability of all values for  $X_M$  in which  $x_i$  is the smallest variable. The  
 successor class  $(M_i, C_i, D_i, \text{BLK}_i, \text{URG}_i)$  is then obtained as follows:

- 950 •  $M_i$  is the marking  $M \setminus \bullet t_i \cup t_i \bullet$ . Sets  $\text{BLK}_i, \text{URG}_i$  can be updated as follows:  
 a blocked transition remains blocked if it is enabled in  $M_i$  and its postset  
 is not freed by firing of  $t_i$ . It can also become urgent if its postset is freed,  
 or be disabled if firing  $t_i$  removes a token from its preset. Similarly, urgent  
 transitions can become blocked, remain urgent, or be disabled.
- 955 • We reuse the technique of [15] to compute  $C_i$ . We start from  $C'$ . We first  
 make a variable change of the form  $x_j = x_i + x'_j$  (to consider the fact that  
 all TTFs are decreased by the value of  $x_i$ ). We then eliminate all variables  
 associated to transitions disabled by firing of  $t_i$  (for instance using the  
 Fourier-Motzkin elimination method). Last, we add new constraints of the  
 960 form  $x'_k \in I(t_k)$ , for every newly enabled transition  $t_k$ . This represents the  
 fact that a new TTF is sampled. Last, we rename all  $x'_k$ s to  $x_k$  to obtain  
 $C_i$ .

- $D_i$  is obtained in an almost similar way, using the procedure given by [3]. We have already computed the probability  $p_i$  to fire  $t_i$  from  $(M, C, D, \text{BLK}, \text{URG})$ . As  $t_i$  fires before any other transition from class  $(M, C, D, \text{BLK}, \text{URG})$  we first compute a new distribution  $D^a$  defined over  $C'$  such that  $D^a(X_M) = \frac{D(X_M)}{p}$  that conditions values of variables in  $X_M^a = X_M \setminus \{x_i\}$  knowing that  $x_i$  has the smallest value in  $X_M$ . The next step is to build the distribution of probabilities once  $x_i$  is eliminated, i.e., a distribution  $D^b(X_M^a)$  computed as  $D^b(X_M^a) = \int_{\min_i}^{\max_i} D^a(\{x_j + x_i \mid i \neq j\})$ , where  $\min_i$  is the minimal value of  $x_i$  in  $C'$  and  $\max_i$  its maximal value. This integration is repeated for every variable attached to a transition disabled by firing of  $t_i$ . The last step consists in integrating the newly created variable and their distribution to  $D^b$ . Let  $X_{M_i} \setminus X_M = \{x_{k_1} \dots x_{k_q}\}$ , where  $t_{k_1} \dots t_{k_q}$  are newly enabled transitions. As the value of each  $x_{k_j}$  represent a new sampled TTF, it does not depend on former values of variables, we have  $D_i = D^b \cdot \mathcal{F}(x_{k_1}) \dots \mathcal{F}(x_{k_q})$ .  $D_i$  is defined over domain  $C_i$  defining possible values of variables in  $X_{M_i}$ .

The construction of a successor class when  $\text{URG} \neq \emptyset$  follow similar lines, i.e., consists in computing successor marking, domain and distribution. The main difference is that an urgent transition necessarily has a TTF equal to 0. Futhermore, it may compete with other urgent transitions. The probability to fire  $t_i \in \text{URG}$  is hence  $p_i = \mathcal{W}(t_i) / \sum_{t_j \in \text{URG}} \mathcal{W}(t_j)$ .

One can iteratively compute successors of stochastic state classes to obtain a transient execution tree. As our STPN is bounded, the number of markings and domains that can be generated inductively at construction time is finite (as proved by [14]). Urgent and blocked transitions are also finite subsets of  $T$ . The distributions attached to transitions of STPNs are expolynomial functions. Beyond their expressive power, expolynomial functions are closed under projections, integrations, or multiplication. Further, (joint) distributions of clock values in a node can always be encoded as expolynomial functions. This way, one can iteratively build a tree whose nodes contain markings, state classes and expolynomial distributions over these classes. However, as shown in [3], the number of distributions that can be iteratively computed need not be finite. However, as time progress is guaranteed, one can limit the construction to a bounded horizon. In our case, we can be even more directive, and guide the tree construction to match executions of a particular symbolic process  $\mathcal{E}^s$ . Indeed, we can consider only executions of a tree that are executions of  $\mathcal{E}^s$  by remembering at every step which transitions have been executed, and forbidding any transition that is not among the possible next ones. Details on the construction of this transient tree are provided in Appendix G.

As shown in [3], after this transient tree construction the (sum of) probabilities attached to paths of the tree can be used to compute the probability of properties such as safety of a system within a bounded horizon. 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. If this probability is not null, then there is a positive probability to realize the considered schedule.

Note that the computed value is only a lower bound for the exact probability to realize a schedule: indeed there can be more than one process realizing a schedule. However, computing the exact probability is rather involved, as distinct realizations of a schedule are not necessarily independent.

Consider a transient tree that collects all symbolic executions of a particular process  $\mathcal{E}^s$ . Each of its paths

$$\rho = (M_0, C_0, D_0, \text{BLK}_0, \text{URG}_0) \xrightarrow{t_1, p_1} (M_1, C_1, D_1, \text{BLK}_1, \text{URG}_1) \dots \xrightarrow{t_k, p_k} (M_k, C_k, D_k, \text{BLK}_k, \text{URG}_k)$$

defines a feasible set of executions of  $\mathcal{E}^s$  by  $\mathcal{N}$ . The probability of execution  $\rho$  is the product  $p_1 \cdot p_2 \dots p_k$ . By summing up probabilities of all path of the transient tree that are executions of  $\mathcal{E}^s$ , one obtains the probability of  $\mathcal{E}^s$ .

This immediately gives us an informal algorithm to check probabilistic realizability of a schedule  $S$  with a maximal date by a STPN  $\mathcal{N}$  (that guarantees time progress). First, unfold  $\mathcal{N}$  up to a depth  $K$  computed according to the maximal date for the schedule. Find the set  $E$  of all processes of  $\mathcal{U}_K$  that embed  $S$ . For each symbolic process  $\mathcal{E}^s \in E$ , compute the part of the transient tree that corresponds to executions of  $\mathcal{E}^s$  in  $\mathcal{N}$ . If the schedule imposes precise dates, include in state classes a variable  $x_{age}$  that remembers the time elapsed since the beginning of an execution, and add the constraint  $x_{age} \in [d(n) - \alpha, d(n) + \alpha]$  when firing a transition implementing some node  $n$  of the schedule. Then check the probability of each path ending on a node where all transitions of  $\mathcal{E}^s$  have been executed. If one finds a path  $\rho$  in the transient tree with a complete execution of  $\mathcal{E}^s$  and non-null probability  $p_\rho$ , then  $\mathbb{P}(\mathcal{E}^s \wedge \text{Sol}(\Phi_{\psi, S, d \pm \alpha})) \geq p_\rho > 0$  and the algorithm can stop and return a positive answer.

## 5. Use Case: realizability in a metro network

In this section, we develop a complete case study: we consider realizability of a particular schedule by a fleet of trains in a metro network. We show that one can depict a metro network and a simple shunting mechanism with stochastic time Petri nets, decide whether a particular schedule can be realized, and compute the probability of its realization.

Urban train systems are usually composed of closed imbricated loops, and the example developed below is a network of this kind, that is representative of the architecture of many metro lines. In metro networks, trains travel at predetermined speed profiles following a predetermined itinerary. They move from a station to another following a track. A network is hence not just a simple cycle: it contains forks, junctions, etc. Such complex topologies can easily be captured by the flow relations of a Petri net. Consider for instance the example network of Figure 9. This network is composed of 7 stations  $sA, sB, sC, sD, sE, sF, sG$  and bidirectional tracks. Each station hence has two platforms, one for each direction, denoted respectively  $X$  and  $\bar{X}$  for station  $sX$ . A train in the network can be scheduled to serve repeatedly platforms along trip  $A.B.C.D.E.F.G.\bar{G}.\bar{F}.\bar{E}.\bar{D}.\bar{C}.\bar{B}.\bar{A}$ , or follow smaller loop trips  $A.B.C.D.\bar{D}.\bar{C}.\bar{B}.\bar{A}$  and  $D.E.F.G.\bar{G}.\bar{F}.\bar{E}.\bar{D}$ . This

situation is a frequent one in long circular metro lines connecting suburbs and city centers. Indeed, in the mornings and evenings, it is more important to bring commuters from their home to the city center than providing long trips along the whole network.

1050 We consider a setting where trains are isolated from one another via a so-called fixed-block policy: track portions are reserved for one particular train. This prevents trains from being too close from one another. Though this is not the only nor the most efficient way to avoid collision, this mechanism is used in real systems <sup>1</sup>. With such a fixed block policy, one can address a train network  
 1055 at block level (i.e., attach a place to each portion of the network that can be entered by at most one train), and model the effect of signaling systems that avoid collision with a blocking semantics of nets.

The Petri net at the bottom part of Figure 9 can represent this network: places are used to represent stations or tracks between stations, and transitions model a possible move from a part of the network to a consecutive one. The  
 1060 flow relation of this net is almost isomorphic to the original network.

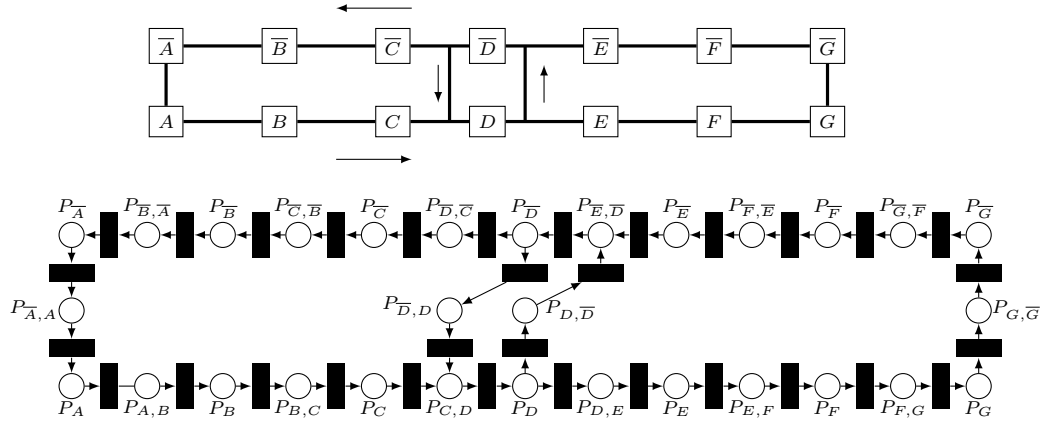


Figure 9: Modeling trains flows in a network with Petri nets

We will use the formalisms and the definition of realizability to give an answer to the following questions:

- 1065 • Given a particular schedule, is it realizable from a given configuration of the net? Being able to answer this question allows to check whether the accumulated delays of trains force rescheduling, especially to organize trains passage at track junctions.
- From a given configuration of the network, what is the probability that all trains in a fleet complete their trips within less than 39 minutes ?

<sup>1</sup> [https://en.wikipedia.org/wiki/Railway\\_signalling#Fixed\\_block](https://en.wikipedia.org/wiki/Railway_signalling#Fixed_block)

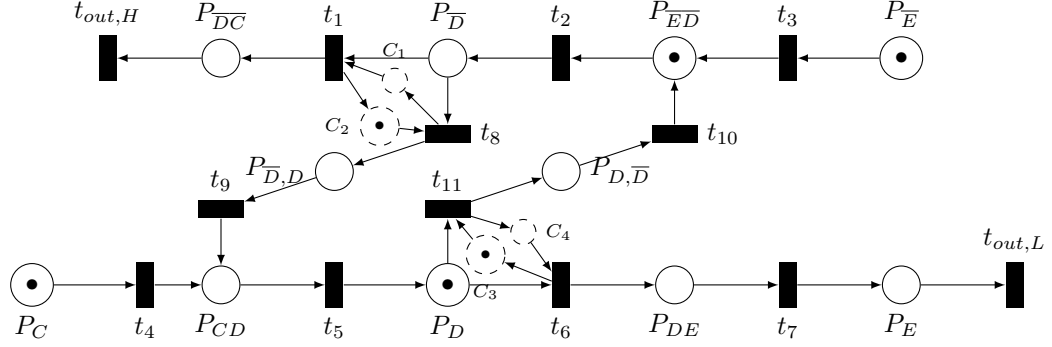


Figure 10: A zoom on the central part of the network

1070 Consider the example of Figure 10. This net provides more details on the  
 central part of the network of Figure 9. The network is decomposed into an  
 upper and lower part. In the upper part, trains circulate from right to left, with  
 places  $P_{DC}$ ,  $P_{ED}$  symbolizing track portions respectively between stations  $D, C$   
 and  $E, D$  in this direction, and two places  $P_{\bar{E}}$  and  $P_{\bar{D}}$  symbolizing platforms at  
 1075 stations  $\bar{E}$  and  $\bar{D}$ . Transitions  $t_1, t_2, t_3$  symbolize respectively a departure from  
 station  $\bar{D}$ , an arrival at station  $\bar{D}$ , and a departure from station  $\bar{E}$ . Trains leave  
 this central part via transition  $t_{out,H}$ .

Similarly, the lower part of the net of Figure 10 represents a part of the  
 network where trains circulate from left to right. It contains places  $P_{CD}$ ,  $P_{DE}$   
 1080 symbolizing track portion between stations  $C, D$  and  $D, E$ . Places  $P_C, P_D, P_E$   
 symbolize respectively platforms at stations  $C, D, E$ . Transitions  $t_4, t_5, t_6, t_7$   
 symbolize respectively departure from  $C$ , arrival in  $D$ , departure from  $D$ , and  
 arrival in  $E$ . Trains leave this central part via transition  $t_{out,L}$ .

Transitions  $t_8, t_9, t_{10}, t_{11}$  and places  $P_{\bar{D},D}, P_{D,\bar{D}}$  are used to allow trains to  
 1085 move from the upper part to the lower part, which is needed to perform small  
 loop trips. Last, places  $C_1, C_2$  implement a flip-flop shunting mechanism to  
 direct one train over two leaving platform  $\bar{D}$  towards platform  $D$ , and the other  
 towards platform  $\bar{C}$ . Places  $C_3, C_4$  play the same role to direct trains leaving  
 platform  $D$  towards  $\bar{D}$  or  $E$ .

1090 With a perfect timing of trains, this simple shunting mechanism suffices to  
 implement two crossing small loops in the network. Consider again the network  
 of Figure 10. There are four trains, represented by tokens. We will call  $train_1$   
 the train symbolized by a token in place  $P_{ED}$ ,  $train_2$  the train symbolized by a  
 token in place  $P_{\bar{E}}$ ,  $train_3$  the train symbolized by token in place  $P_D$  and  $train_4$   
 1095 the train represented by token in place  $P_C$ .

Let us now associate durations and distributions to transitions. For the sake of  
 simplicity, we consider identical distributions for dwell, running times and  
 transfer from the upper to the lower part of the net. We consider interstation  
 distances of 2 kilometers, and trains running at an average commercial speed  
 1100 of 35 km/h. That is, the sojourn time of a token in places  $P_{DC}$   $P_{ED}$ ,  $P_{CD}$  and

$P_{DE}$  should be defined by a distribution with maximal probability around 205 seconds, with more probable delays than advance. We furthermore consider that the distance needed to go from  $D$  to the track portion  $\overline{ED}$  is 0.5 km, but is performed at a speed of 20 km/h (similarly for moves from  $\overline{D}$  to track  $CD$ ). Within this setting, sojourn time in places  $P_{\overline{DD}}$  and  $P_{D\overline{D}}$  should be a distribution centered around 90 seconds. Last, we associate a dwell time to every station. We choose arbitrarily a value of 50 seconds for the most probable dwell time in the whole network. It is very frequent that trains get late at stations due to passengers misbehavior. However, we consider that dwell time cannot exceed 400 seconds. We hence associate to transitions  $t_9, t_{10}$  (transfer) the distribution  $f_1 = 1.477 \cdot 10^{-2} \cdot (x - 45)^5 \cdot e^{-1.1 \cdot (x - 45)}$ , defined only on interval  $[85, 100]$ . We associate to transitions  $t_2, t_5, t_7, t_{out,h}$  (trips) distribution  $f_3 = 1.3413 \cdot 10^{-3} \cdot (x - 200)^4 \cdot e^{-0.5 \cdot (x - 200)}$  defined only on interval  $[200, 220]$ . We last associate distribution  $f_2 = 7.8125 \cdot 10^{-3} \cdot (x - 45)^2 \cdot e^{-0.25(x - 45)}$  defined on interval  $[45, 400]$  to all other dwell transitions. The intervals associated to transitions, and the shape of distributions are depicted on Figure 11.

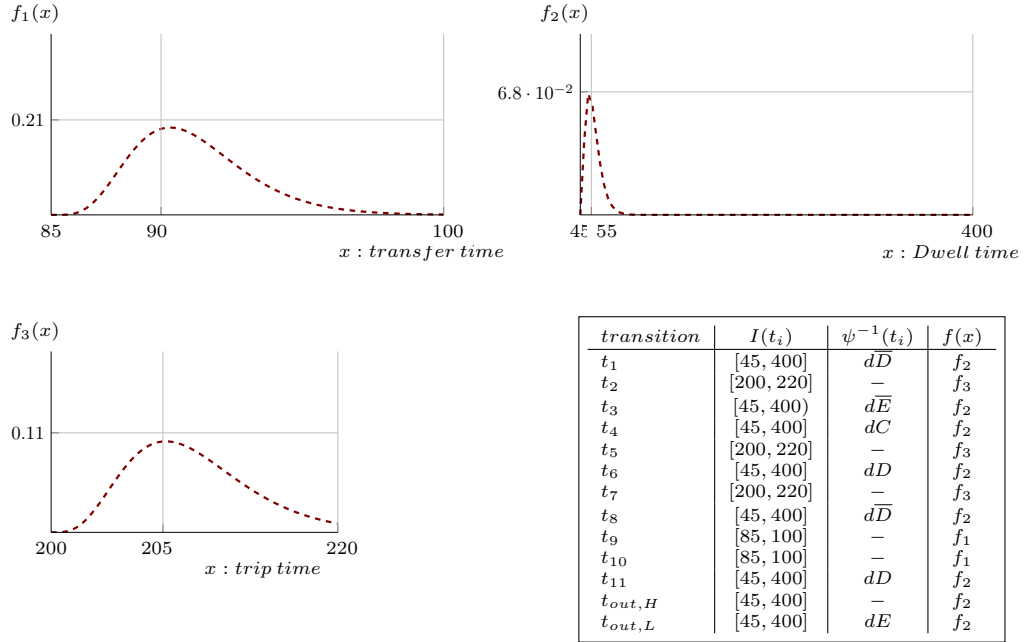


Figure 11: Distributions

Ideally, the expected trajectory of trains 1 and 2 go through stations  $\overline{D}$ ,  $D$  and  $E$ , in this order. The trajectories of trains 3 and 4 go through stations  $D$ ,  $\overline{D}$  and  $\overline{C}$ . This can be represented by the schedule of Figure 12. As we only consider departures from platforms, we associate to nodes of the schedule labels of the form  $dX$ , where  $X$  is the name of a platform. We can now relate departures



from stations in the schedule and concrete events in the net on Figure 10 using a realization function  $r$ , such that  $r(d\overline{D}) = \{t_1, t_8\}$ ,  $r(d\overline{E}) = \{t_3\}$ ,  $r(dC) = \{t_4\}$ ,  $r(dD) = \{t_6, t_{11}\}$ ,  $r(dE) = \{t_{out,L}\}$ . One can notice that there are two possible ways to implement a departure from  $D$  or  $\overline{D}$ .

Obviously, using the simple flip-flop shunting mechanism explained above, such a schedule can be realized only if the order of trains in place  $P_{\overline{D}}$  is train 1, train 3, train 2, train 4, and if the passage order in place  $P_D$  is train 3, train 1, train 4, train 2. Let us assume that we start from an initial configuration where trains 1 and 3 have a remaining trip duration of 10 seconds, that trains 2 is delayed and cannot leave  $P_{\overline{E}}$  before 320 seconds, that train 2 is delayed and cannot leave  $P_C$  before 300 seconds. This initial configuration of the network can be easily encoded by attaching adequate times to fire to enabled transitions  $t_2, t_3, t_{11}, t_4$ . Considering the schedules, one can safely add the constraint that all trips are performed after 2300 time units. As the interval attached to transitions all have lower bounds greater than 45, and as the net has 13 transitions, all timed processes of the net in Figure 10 embedding the schedule of Figure 12 in less than 2300 time units appear in an unfolding  $\mathcal{U}_K$  of depth at most  $K = \lceil \frac{2300}{45} \rceil \cdot 13^2 = 8788$ . Note however that if trains behave as expected, all trains modeled in the example of Figure 10 will eventually leave this part of the network after a finite time, and hence unfolding should stop much earlier due to lack of appendable transitions. Yet, even without this a priori information, as dwells and running consume time, this allows to represent the behavior of the network with an unfolding of bounded depth. Then, from the initial configuration, there exists time processes satisfying all time constraints due to dwell and running times. Figure 13 is an example of such process. As usual, conditions are represented by circles and events by squares. Due to lack of space, we do not give all constraints attached to this process. They will contain constraints on event variables of the form:

$$\begin{aligned}
doe(t_2^1) &= 0 \\
dof(t_2^1) &= doe(t_2^1) + 10 \\
dof(t_2^1) &\leq \theta(t_2^1) \\
\dots \\
[\theta(t_3^1), \theta(t_3^3)] \cap [\theta(t_{10}^1), \theta(t_2^2)] &= \emptyset \\
\dots
\end{aligned}$$

We indicate with red numbers associated with events a possible dating function  $d$ . One can verify that for any ordered pair of events  $e \leq e'$ , we have  $d(e) < d(e')$ , that sojourn times in places are compatible with dwell and running times, and that for any pair of conditions  $P_X^i, P_X^j$  with  $j \neq i$  we have  $[d(\bullet(P_X^i)), d((P_X^i)\bullet)] \cap [d(\bullet(P_X^j)), d((P_X^j)\bullet)] = \emptyset$ .

The time process of Figure 13 shows a possible realization of the schedule in Figure 12. As we are dealing with continuous probabilities the probability of realizing exactly this process with the specified dates is 0. However, one can easily notice that all dates indicated on this process can be shifted by several time units. Indeed, as dates are not discrete values, there is an infinite number

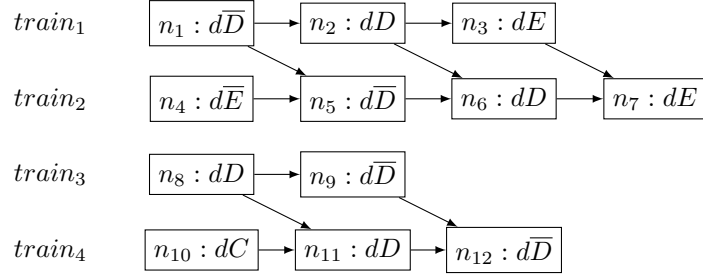


Figure 12: A possible schedule for trains departures from initial configuration in Figure 10

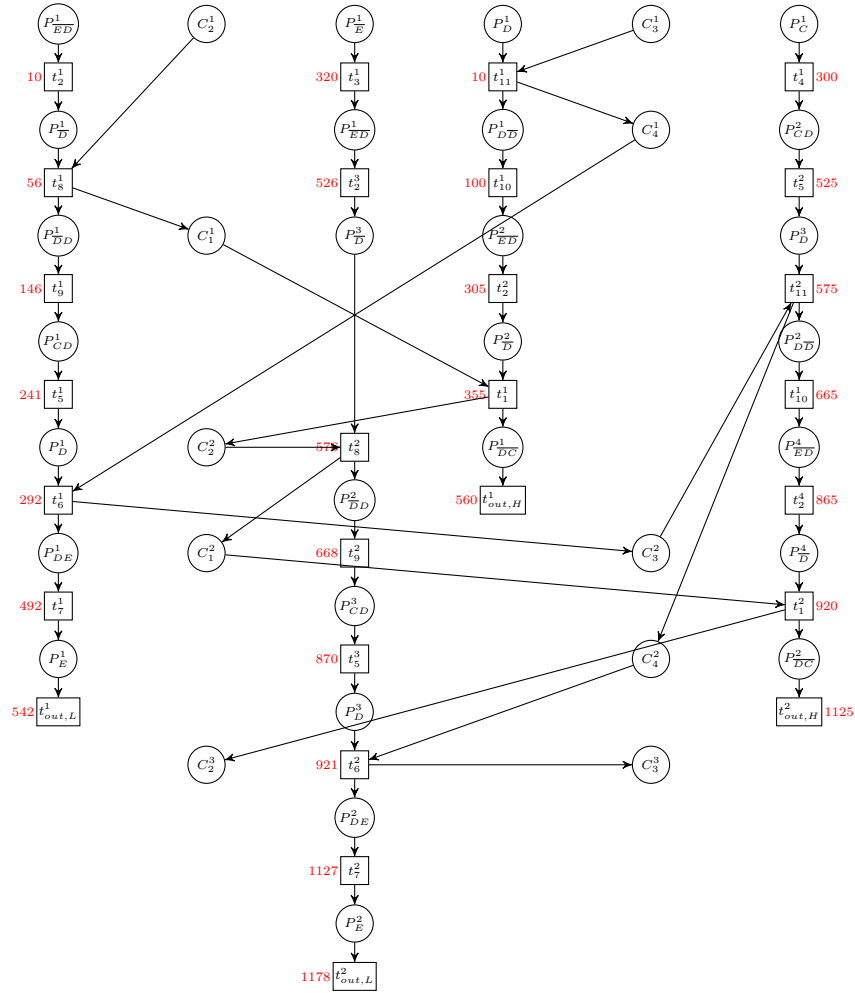


Figure 13: A witness for realizability of schedule in Figure 12

1160 of time processes with the same conditions and events but with different timing  
functions. This defines a class  $\mathcal{C}$  of time processes realizing our schedule with a  
non-null probability  $\mathbb{P}(\mathcal{C})$ . Note that the probability of realizing the schedule  
is greater than the probability  $\mathbb{P}(\mathcal{C})$  of executing a time process from this class,  
as there could be more than one way of realizing the schedule. The value  $\mathbb{P}(\mathcal{C})$   
1165 can be computed by construction of a transient execution tree, starting from the  
initial configuration, and in which nodes are of the form  $(M_i, C_i, D_i)$ , where  $M_i$  is  
a marking,  $C_i$  represents possible values of TTFs attached to enabled transitions,  
and  $D_i$  the probability distribution with positive values on the domain  $C_i$  defined  
this way.

1170 An execution of a time process from  $\mathcal{C}$  necessarily starts from the initial  
configuration given above, i.e, from node  $(M_0, C_0, D_0, \text{BLK}_0, \text{URG}_0)$ , with  $M_0 =$   
 $\{P_{\overline{E}}, P_{\overline{ED}}, P_C, P_D\}$ ,  $\text{BLK}_0 = \emptyset, \text{URG}_0 = \emptyset$ . For each enabled transition  $t_i$ , we define  
by  $x_i$  the variable symbolizing possible values of TTF (remaining time to fire)  
of  $t_i$ . We hence have  $C_0 = \{x_2 = 10; x_3 = 320; x_4 = 300; x_{11} = 10\}$ , i.e.  $C_0$  is a  
1175 point in  $\mathbb{R}_{\geq 0}^4$ . The distribution  $D_0$  associates probability 1 to the single point  
in  $C_0$ . From this situation, two events can occur: either  $t_2$  fires after 10 time  
units, or  $t_{11}$  fires after 10 time units. As both events are enabled at the same  
date, the probability to fire each one is defined according to weights attached to  
transitions. If we assume that  $t_2, t_{11}$  have the same weight, so the probability to  
1180 fire  $t_2$  from  $(M_0, C_0, D_0)$  is 0.5.

Let us consider the effect of firing  $t_2$  at date 10: it affects marking, and  
remaining TTFs. After firing  $t_2$  the possible configurations of the net are  
encoded by a node  $(M_1, C_1, D_1, \text{BLK}_1, \text{URG}_1)$  where  $M_1 = \{P_{\overline{E}}, P_{\overline{D}}, P_C, P_D\}$ ,  
 $\text{BLK}_1 = \emptyset, \text{URG}_1 = \{t_1\}$ . Upon firing of  $t_2$ , transition  $t_8$  becomes enabled,  
1185 and a new time to fire represented by variable  $x_8$  is sampled. Hence the  
possible values for TTFs are depicted by the constraints in class  $C_1 = \{x_3 =$   
 $310; x_4 = 290; 45 \leq x_8 \leq 400\}$ . The distribution  $D_1$  attached to this class is  
 $D_1(x_3, x_4, x_8) = f_2(x_8)$  if  $(x_3, x_4, x_8, x_{11}) \in C_1$ , and 0 otherwise.

From this class, transition  $t_{11}$  is urgent and has to fire immediately, with  
1190 probability 1 yielding a new marking, and sampling of a new value for newly  
enabled transition  $t_{10}$ , and a new node  $(M_2, C_2, D_2, \text{BLK}_2, \text{URG}_2)$  with  $M_2 =$   
 $\{P_{\overline{E}}, P_{\overline{D}}, P_C, P_{\overline{DD}}\}$ ,  $\text{BLK}_2 = \emptyset, \text{URG}_2 = \emptyset$ . A new value for TTF of transi-  
tion  $t_{10}$  is sampled, yielding class  $C_2 = \{x_3 = 310; x_4 = 290; 45 \leq x_8 \leq$   
 $400; 85 \leq x_{10} \leq 100\}$ , and distribution  $D_2$  such that  $D_2(x_3, x_4, x_8, x_{10}) =$   
1195  $f_2(x_8) \cdot f_1(x_{10})$  if  $(x_3, x_4, x_8, x_{10}) \in C_2$ , and 0 otherwise.

The next classes that can appear are more complex, and computing them  
is more involved. Assume that transition  $t_8$  fires from a configuration of class  
 $C_2$ . This means in particular that  $t_8$  is the transition with the smallest TTF,  
that is  $x_8$  is smaller than  $x_3, x_4, x_{10}$ . One can easily check that some values for  
1200  $x_3, x_4, x_8, x_{10}$  satisfy  $C'_2 = C_2 \cup \{x_8 \leq x_3; x_8 \leq x_4; x_8 \leq x_{10}\}$ . This additional  
constraint has to be considered, both to compute the probability of firing  
 $t_8$  and to compute the new class reached after firing  $t_8$ . The class obtained  
after  $t_8$  is  $(M_3, C_3, D_3, \text{BLK}_3, \text{URG}_3)$  and is computed as follows. First  $M_3 =$   
 $\{P_{\overline{E}}, P_{\overline{DD}}, P_C, P_{\overline{DD}}\}$ . Then, the new set of constraints is obtained using the  
1205 standard construction of state classes:

- start from  $C'_2 = C_2 \cup \{x_8 \leq x_3; x_8 \leq x_4; x_8 \leq x_{10}\}$ .
- do a variable substitution of the form  $x_i = x_8 + x'_i$  for all variables. This substitution allows to propagate the fact that new values for  $x_3, x_4, x_{10}$  are decreased by the value of  $x_8$ .
- 1210 • eliminate all variables related to disabled transitions (using the Fourier-Motzkin method). In our case, only  $x_8$  must be eliminated.
- add inequation(s) related to newly enabled transition(s). In our case, as  $t_9$  becomes enabled, this amounts to inserting constraint  $85 \leq x_9 \leq 100$ .
- rename all  $x'_i$  into  $x_i$

1215 The class  $C_3$  obtained by substitution and elimination from  $C'_2$  is  $C_3 = \{x_3 - x_4 = 20; 0 \leq x_3 \leq 265; 0 \leq x_4 \leq 245; 0 \leq x_{10} \leq 55; 210 \leq x_3 - x_{10} \leq 225; 190 \leq x_4 - x_{10} \leq 205; 85 \leq x_9 \leq 100\}$ .

Let us now consider distribution  $D_3$ . We can compute the probability  $p$  of firing transition  $t_8$  from node  $(M_2, C_2, D_2)$  as the integration over all possible values of vector  $x_3, x_4, x_8, x_{10}$  over  $C'_2$ , i.e.

$$p = \int_{C'_2} D_2(x_3, x_4, x_8, x_{10}) = \int_{45}^{100} \int_{x_8}^{100} f_2(x_8) f_1(x_{10}) d_{x_{10}} d_{x_8}$$

This value can be approximated to 0.171499, i.e., firing of  $t_8$  is not the most probable event in  $(M_2, C_2, D_2, \text{BLK}_2, \text{URG}_2)$ . We can now build  $D_3$ , using  
 1220 the procedure given by [3]. As  $t_8$  fires before any other transition from class  $(M_2, C_2, D_2)$  we first compute a new distribution  $D_2^a$  defined over  $C'_2$  such that  $D_2^a(x_3, x_4, x_8, x_{10}) = \frac{D_2(x_3, x_4, x_8, x_{10})}{p}$ . The next step is to build the distribution of probabilities once  $x_8$  is eliminated, i.e. a distribution  $D_2^b(x_3, x_4, x_{10})$  computed  
 1225 as  $D_2^b(x_3, x_4, x_{10}) = \int_{45}^{100} D_2^a(x_3 + x_8, x_4 + x_8, x_8, x_{10} + x_8)$ . In general, this integration is repeated for every variable attached to a disabled transition, but in the current case, only  $x_8$  needs to be projected away. The last step consists in integrating the newly created variable  $x_9$  and its distribution to  $D_2^b$ . As the value of  $x_9$  does not depend on former values of  $x_3, x_4, x_{10}$ , we have  $D_3 = D_2^b \cdot f_1(x_9)$ , defined over a domain  $C_3$  for variables  $x_3, x_4, x_9, x_{10}$ .

1230 We can repeat this process for all transitions that are fireable from any node to build a transient execution tree. At the end of the construction, we can distinguish a set  $\mathcal{PT}$  of paths of the tree that end after execution of all events appearing in the process of Figure 13. We can associate to each of these paths  $\rho \in \mathcal{PT}$  a probability  $p_\rho$  that is the product of probabilities of each  
 1235 transition in  $\rho$ . Last, the probability  $\mathbb{P}(\mathcal{C})$  to execute the process of Figure 13 is  $\mathbb{P}(\mathcal{C}) = \sum_{\rho \in \mathcal{PT}} p_\rho$ .

One of the key points in this technique is to integrate over variables domains. Recall that  $\int_a^b \alpha \cdot x^n \cdot e^{-\lambda \cdot x} = -\alpha \cdot \lambda \int_a^b x^n \cdot \frac{-1}{\lambda} e^{-\lambda \cdot x} = -\alpha \cdot \lambda ([x^n \cdot e^{-\lambda \cdot x}]_a^b - \int_a^b n x^{n-1} \cdot e^{-\lambda \cdot x})$ . Hence, probabilities can be computed exactly through an  
 1240 iterative process.

Let us now show how to ensure that an event  $n$  represented in a schedule occur at a fixed date  $d(n) \pm \alpha$ . As shown in [3], it is sufficient to add to state classes a variable  $x_{age}$  initially set to 0, and that is incremented by the value of every variable  $x_i$  at every state class updated resulting from the firing of a transition  $t_i$ . In other words, in every class, variable  $x_{age}$  describes possible value for the time that has elapsed since the beginning of the execution of a process. When an event of a process representing occurrence of a node  $n$  (i.e, such that  $\psi(n) = e$  which date must be  $d(n)$  up to imprecision  $\alpha$ , it suffices to add to the domains of the newly computed state class the constraint that  $d(n) - \alpha \leq x_{age} \leq d(n) + \alpha$ . Integration follow the same principles as before, but over domains with one additional dimension. For instance, if we impose that node  $n_1$  in the schedule of Figure 12 is executed at a date  $60 \pm 10$ , it suffices to maintain variable  $x_{age}$  during construction of states classes  $(M_0, C_0, D_0, \text{BLK}_0, \text{URG}_0), (M_1, C_1, D_1, \text{BLK}_1, \text{URG}_1), (M_2, C_2, D_2, \text{BLK}_2, \text{URG}_2)$  as defined above, and to impose during the construction of state class  $C_3$  the additional constraint that  $50 \leq x_{age} \leq 70$ . Similarly, completing a schedule  $S$  within a maximal amount of time  $\Delta$  amounts to imposing constraint  $d(n) \leq \Delta$  at every maximal node  $n$  of  $S$ . This means refining state classes reached by firing transitions representing these maximal node with constraint  $x_{age} \leq \Delta$ . Answering our second question on the use case then amounts to checking for every process that embeds  $S$  and every final path in the associated transient tree that constraint  $x_{age} \leq (39 * 60)$  allows firing at least one transition reaching a final node (i.e. where the schedule is completely executed) with positive probability.

## 6. Related work

We have addressed realizability of partially ordered timed schedules by timed and stochastic concurrent systems with safety requirements. The model used to represent systems such as production systems or metro networks is a variant of STPNs. Petri nets have met a lot of interest for modeling manufacturing systems (see [16] for a survey). They are also popular to represent train and metro networks (see for instance [17, 18]), as their flow relation mimics the physical architecture of the network. Realizability in a timed setting was formerly addressed as a timed game problem [19], 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. This work could be used to answer a boolean realization question, by translating a schedule to a formula, but in a fully interleaved setting, as sequential formulae cannot differentiate interleaved and concurrent actions. Furthermore, this logic does not address randomness in systems and hence cannot quantify realizability.

A standard approach to study behaviors and performance of train or metro networks is to rely on simulation with very precise dedicated models [20, 21, 22]. Another frequent approach in train systems is to address realizability as a timetable construction problem. The literature on timetabling or rescheduling is prolific, one can read [23] for a survey. Timetabling questions are often

1285 addressed with high-level descriptions such as graphs and constraints, and solved  
 as a constraint satisfaction or as an optimization problem. In [1] 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  
 solutions proposed in [1] use a branch and bound algorithm to return an optimal  
 1290 schedule for the next 2 hours of operation of a train network, but do not consider  
 randomness. Note however that this algorithm is efficient enough to be used  
 online to guide decisions of a scheduling system. Some authors consider flexible  
 solutions to achieve more robust timetables [24, 25], but the key question of  
 whether schedules are realizable with sufficiently high probability is usually not  
 1295 considered in the literature. Stability of timetables in an environment with  
 random perturbations was considered in [26]: a schedule is considered as an  
 immutable ordering on trains, and delays are modeled as probability distributions.  
 Reliability of a given timetable (with fixed dates) is then defined via probability  
 measures (probability that a given number of trains gets late by more than  
 1300 a fixed threshold. . .). In a variant of the problem, [27] proposes a notion of  
 robustness in scheduling: a timetable is said to be robust if it is able to absorb  
 small random delays by possibly applying recovery solutions. Note however that  
 considering realizability through timetables, solutions only consider constraints  
 on the possible dates of events listed in the timetable, but may not integrate  
 1305 constraints originating from the network itself. The notion of realizability that  
 we have proposed considers low-level realization of schedules, and hence differs  
 from (robust) timetable construction problem.

Realizability is also close to *diagnosis*. Given a log (a partial observation  
 of a run of a system), and a model for this system, diagnosis aims at finding  
 1310 all possible runs of the model of the system whose partial observation complies  
 with the log. Considering a log as a schedule, the ability to compute a diagnosis  
 implies realizability of this high-level log by the model. Diagnosis was addressed  
 for stochastic Petri nets in [28]. In this work, the likelihood of a process that  
 complies with an observation is evaluated, and time is seen as a sequence of  
 1315 discrete instants. Diagnosis was addressed for parameterized Petri nets in [6].  
 The proposed solution unfolds a parameterized Petri net to find explanations  
 for an observed log. Time Petri nets can be considered as a particular case of  
 this parameterized model. [29] proposes temporal patterns called chronicles that  
 represent possible evolutions of an observed system. A chronicle is a set of events,  
 1320 linked together by time constraints. The diagnosis framework explains stream  
 of time-stamped events as combinations of chronicles. Assembling chronicles  
 is some kind of timed unfolding. However, event streams are not a concurrent  
 model, and chronicles extraction does not consider randomness.

Schedulability can also be seen as conformance of an expected behavior (the  
 schedule) to an implementation (the STPN model). Conformance was defined  
 1325 as a timed input/output conformance relation (tIOCO) relation between timed  
 input/output automata in [30]. More precisely, timed automaton  $\mathcal{A}_1$  is in tIOCO  
 relation with timed automaton  $\mathcal{A}_2$  iff after some timed word, the set of outputs  
 produced by  $\mathcal{A}_1$  is included in the outputs produced by  $\mathcal{A}_2$ . This relation cannot  
 1330 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 probabilities.

Several models of time have been proposed for timed concurrency models, and especially for Petri nets. Without claiming for exhaustiveness, one can cite at least time Petri nets [31] (TPNs for short), timed Petri nets (see for instance [32]), and stochastic time Petri nets [3]. Time Petri nets associate a rational time interval  $I(t) = [l, u]$  or  $I(t) = [l, \infty)$  to every transition  $t$  in the system. The semantics of time Petri nets considers that a clock is attached to every transition, and reset every time the transition becomes enabled. A transition can fire only if its clock's value lays within the interval  $I(t)$ . A particularity of this model is that time is not allowed to progress when a clock  $x_t$  reaches the upper bound of interval  $I(t)$ . This phenomenon is called *urgency*, and allows to model mandatory limits for execution dates. This is of particular interest to handle requirements such as: 'A train has to leave at most 10 seconds after a departure order was given'. Though time Petri nets are a powerful formalism, with time concurrency and urgency, they miss blockings and stochasticity.

The second well-known timed variant of Petri nets is called timed Petri nets. In this model, markings associate a set of real values to places, and flow relations from places to transitions are constrained by intervals. Tokens are hence not blind tokens as in TPNs but rather ages: a newly created token has age 0, and a transition  $t$  can fire iff every place  $p$  in its preset contains a token satisfying the constraint attached to  $p$  and  $t$ . A drawback of this model is that transition firing is not urgent: a transition that can fire is not forced to fire, and once tokens in a place become too old to satisfy a constraint allowing them to leave the place, they can be forgotten (or not considered). A natural assumption is that trains are modeled as tokens. However, discarding tokens amounts to losing trains, which is an undesirable feature for the systems we consider. Hence, in the context of train systems, urgency seems to be an unavoidable feature. Solution to integrate urgency in timed Petri nets have been proposed [33]. Though there must be way to avoid losses in timed Petri nets with adequate extensions, we did not follow this approach and used a model with urgency.

A symbolic framework to unfold time Petri nets was already proposed in [6, 7]. As in our setting, this framework builds an untimed unfolding, and for every process a set of constraints attached to events occurrence dates. An untimed process has a timed concretization if its set of constraints is satisfiable. A nice property of this framework is that it is monotonous: if a process has an unsatisfiable set of constraints, then any larger process that also has unsatisfiable constraints. The blocking semantics brings additional constraints on firing dates of transitions that fill a common place. Due to this constraint, process construction is not monotonous anymore. This makes processes and unfolding constructions much harder, as there is no stopping criterion based on constraint satisfaction, and one has to rely on bound for the depth of unfoldings guaranteed by time progress.

The model that we use to represent timed concurrent systems is a variant of the **stochastic time Petri nets** already proposed by [3]. The original

model does not consider *blocking semantics*, but defines a transient analysis  
 technique. These techniques have been implemented in the ORIS tool [34], and  
 allow for computation of the probability of some events. Working with a blocking  
 semantics does not change the syntax of STPNs, but of course affects the legal  
 runs. Interestingly, blocking semantics has no impact on the transient analysis  
 techniques proposed. Indeed, during the construction of a transient tree, the  
 blocking semantics will only impact the fact that a transition is firable or not, but  
 not the construction of accessible state classes: the construction of a successor  
 stochastic state class in a transient tree only depends on the initial state class  
 considered, on the fired transition, and on timed intervals attached to the newly  
 enabled transitions. Hence, techniques and proof of [3] are fully applicable to  
 our setting. Note however that ORIS gives means to analyze a stochastic net  
 with a logic [35] that can express bounds on the probability that the marking of  
 the STPN satisfies a goal predicate  $\phi_2$  at some time in an interval  $[a, b]$  without  
 violating a safety predicate  $\phi_1$  in previous states. Goal and safety predicates  
 $\phi_1, \phi_2$  are properties of markings. This logic cannot describe realizability of  
 schedules, for two reasons: first, it is some kind of liveness property, and second,  
 temporal logic of this form cannot check for existence of causal dependencies  
 such as the ones appearing in example of Figure 12. The approach proposed in  
 our paper first selects the possible realizations of a schedule, and then computes  
 their probability. Note however that once a process that realizes a schedule  
 has been identified, this process could be transformed into an adequate acyclic  
 STPN, and the ORIS tool could be used to compute its probability.

## 7. Conclusion

This work has addressed the question of realizability of a schedule by a  
 detailed and low-level specification. The technique is the following: 1) given a  
 schedule  $S$  and an STPN  $\mathcal{N}$ , first build an unfolding  $\mathcal{U}_K$  up to a depth  $K$  that  
 depends on the maximal date appearing in schedule  $S$ . Then, 2) find symbolic  
 processes of  $\mathcal{U}_K$  that embed  $S$ . The next step 3) consists in checking satisfiability  
 of combinations of linear inequalities derived from the considered processes. For  
 boolean realizability, it is sufficient to compute the symbolic processes and then  
 get a positive answer at step 3 for one of them to return a positive answer.  
 Indeed, satisfiability of a system of inequalities for at least one process of  $\mathcal{N}$   
 yields existence of a time process compatible with  $S$ , and hence realizability. In  
 a probabilistic setting, answering the realizability question needs an additional  
 step 4) to build a transient execution tree and compute the probabilities of sets  
 of time processes described by realizable symbolic processes. Again, probabilistic  
 realizability holds as soon as one symbolic process of  $\mathcal{U}_K$  embedding  $S$  and  
 realizable in the boolean sense has a strictly positive probability.

Unfolding, embedding, constraint satisfaction and probability computation  
 are currently rather distinct parts of the proposed method. A first question  
 is whether some steps can be factorized. In an untimed setting, 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 built on this branch. We



have developed an unfolding tool inspired from [5] for untimed Petri nets that computes an unfolding containing all processes (and only them) that embed the untimed structure of a schedule  $S$ . This means that one can factorize unfolding and embedding of untimed schedules. This is however not sufficient, as an  
1425 untimed process must admit a timing function  $\theta$  that is compatible with the constraints originating from the low-level net and from the schedule to be a realization of a schedule  $S$ . Usually, unfolding can be stopped as soon as some criterion is met (completeness for construction of finite prefixes [4, 5], satisfaction of a property...). However, as shown in Section 3, due to blocking semantics,  
1430 satisfiability of constraints is not monotonous w.r.t. the size of unfoldings, and hence cannot be used as stopping criterion during unfolding.

A second question is of course complexity of the approach. Even when guided by embedding of schedules, the size of unfoldings can grow exponentially with their depth. This remark also applies to symbolic processes and to their con-  
1435 straints. The constraints that appear in each symbolic process can be simplified to refer only to event variables, but this results only in a linear gain in the number of used variables. One can however notice that atoms in constraints are simple binary inequalities. If a process contains no blocking place, its constraints are simple sets of linear inequalities that can be checked with polynomial  
1440 algorithms [36]. However, due to blocking semantics, constraints also contain disjunctions. As soon as blockings appear, one cannot avoid combinatorial explosion, as solving the considered systems with disjunctions amounts to choosing an ordering for conflicting places occupancy. We hence have to face a possible exponential blowup in the depth of the unfolding during the construction of  $\mathcal{U}_K$   
1445 at step 1), followed by a possible exponential blowup in the number of possible conflicts to check satisfiability of each set of constraints in symbolic processes of  $\mathcal{U}_K$ .

The cost of the algorithm needed to compute realization probability for processes is also an issue. We use the transient tree construction of [3], that builds  
1450 a symbolic but *interleaved* representation of a process. This is obviously very costly as it may impose an additional exponential blowup in the worst cases. We are currently investigating ways to evaluate probabilities of symbolic processes in a non-interleaved setting, may be at the cost of under-approximation of realization probabilities. In the current setting, this is not a real problem: one can  
1455 remark that for efficiency reasons, the algorithm proposed to check probabilistic realizability stops as soon as a symbolic process with probability greater than 0 is found. From a practical point of view, we do not yet know whether the approach can be used to solve automatically case studies of important sizes. One can notice however that the important factor to reduce the computational cost  
1460 of realizability is to master the size of unfoldings, processes, and the number of blockings. Mastering the size of unfoldings means in particular working with a reasonable depth, and avoiding branchings. This can be achieved by providing sufficiently precise schedules, which limits the number of conflicts and hence the number of branches in the unfolding. Precise schedules also limit the number  
1465 of possible embeddings. Giving a reasonable value for the maximal date of event implementing actions in a schedule is also a way to limit the depth and

hence the size of unfoldings. One can notice that when considering realizability from a practical point of view, the probability of each process decreases with its size. probabilistic realizability holds as soon as a witness process with strictly  
1470 positive probability is found, but for large processes this realizability is witnessed by a process with very low probability. On the other hand, computing the exact realization probability of very deep unfoldings will probably be too time consuming. We think that a sensible approach is to work with schedules of reasonable sizes, that shall be realized within a bounded duration. This might  
1475 be sufficient to discover that a planning is not realizable anymore starting from the current configuration of a system.

We believe that our method applies to several types of systems with concurrent elements, random environment, and where some events are constrained by time. This covers most of manufacturing or transport systems. Note however that  
1480 as unfoldings grow fast, a key ingredient to compute answers to realizability questions is to keep a reasonable growth of non-determinism w.r.t. elapsed time. As for now, the framework proposed in this paper is not yet fully implemented. The distributions  $f_1, f_2, f_3$  provided in our case study have been designed using a simple prototype tool that allows for the design, composition, projection and  
1485 integration of expolynomial functions. Integrations during the construction of state classes in our uses case were performed using Matlab. We also have developed a schedule guided unfolding tool to build pre-processes that embed a schedule. The remaining steps that need to be considered to have a fully operational software is to provide means to solve the constraint satisfaction  
1490 questions related to a symbolic processes, and to compute the probability of the set of processes it represents. For the constraint satisfaction issue, we can rely on standard solvers such as Z3. For the computation of probabilities of a symbolic process, we plan to use the ORIS tool [34]. However, as probabilistic realizability can be answered by providing a strictly positive lower bound, we  
1495 think that this leaves room for abstraction techniques, and possibly for other ad-hoc algorithms working with non-interleaved representations of processes to approximate their probability.

As future work, we would like to extend our realizability verification framework to prove more properties. For instance, it is interesting to prove that a  
1500 schedule can be realized while ensuring that the overall sum of delays w.r.t. the expected schedule does not exceed some threshold, or to associate costs to realizations symbolizing for instance energy consumption, and to impose bounds on the cost of realization.

## References

- 1505 [1] A. D’Ariano, D. Pacciarelli, M. Pranzo, A branch and bound algorithm for scheduling trains in a railway network, *European Journal of Operational Research* 183 (2007) 643–657.
- [2] A. D’Ariano, M. Pranzo, I. Hansen, Conflict resolution and train speed co-

- ordination for solving real-time timetable perturbations, *IEEE Transactions on Intelligent Transportation Systems* 8 (2007) 208–222.
- 1510 [3] A. Horváth, M. Paolieri, L. Ridi, E. Vicario, Transient analysis of non-Markovian models using stochastic state classes, *Performance Evaluation* 69 (2012) 315–335.
- [4] K. L. McMillan, A technique of state space search based on unfolding, *Formal Methods in System Design* 6 (1995) 45–65.
- 1515 [5] J. Esparza, S. Römer, W. Vogler, An improvement of McMillan’s unfolding algorithm, *Formal Methods in System Design* 20 (2002) 285–310.
- [6] T. Chatain, C. Jard, Symbolic diagnosis of partially observable concurrent systems, in: *FORTE’04*, volume 3235 of *LNCS*, pp. 326–342.
- 1520 [7] T. Chatain, C. Jard, Complete finite prefixes of symbolic unfoldings of safe time Petri nets, in: *ICATPN’06*, 2006, pp. 125–145.
- [8] J. Cortadella, M. Kishinevsky, L. Lavagno, A. Yakovlev, Synthesizing Petri nets from state-based models, in: *ICCAD’95*, pp. 164–171.
- [9] R. Y. Rubinstein, D. Kroese, *Simulation and the Monte-Carlo method*, Wiley, 2 edition, 2008.
- 1525 [10] T. Aura, J. Lilius, Time processes for time Petri nets, in: *ICATPN’97*, volume 1248 of *LNCS*, pp. 136–155.
- [11] J. Bartholdi, D. Eisenstein, A self-coordinating bus route to resist bus bunching, *Transportation Research, Part B* 46 (2012) 481–491.
- 1530 [12] A. Semenov, A. Yakovlev, Verification of asynchronous circuits using time Petri net unfolding, in: *DAC*, pp. 59–62.
- [13] T. Chatain, C. Jard, Time supervision of concurrent systems using symbolic unfoldings of time Petri nets, in: *FORMATS’05*, volume 3829 of *LNCS*, pp. 196–210.
- 1535 [14] B. Berthomieu, M. Diaz, Modeling and verification of time dependent systems using time Petri nets, *IEEE Transactions on Software Engineering* 17 (1991) 259–273.
- [15] D. Lime, O. Roux, Model checking of time Petri nets using the state class timed automaton, *Discrete Event Dynamic Systems* 16 (2006) 179–205.
- 1540 [16] J. Cecil, C. Srihari, K. Emerson, Transient analysis of non-Markovian models using stochastic state classes, *The International Journal of Advanced Manufacturing Technology* 7 (1992) 168–177.
- [17] W. Hielscher, L. Urbszat, C. Reinke, W. Kluge, On modelling train traffic in a model train system, in: *in Workshop and Tutorial on Practical Use of Coloured Petri Nets and Design/CPN*, pp. 8–12.
- 1545

- [18] P. Wang, L. Ma, R. Goverde, Rescheduling trains using Petri nets and heuristic search, *IEEE Transactions on Intelligent Transportation Systems* 17 (2016) 726–735.
- 1550 [19] L. Doyen, G. Geeraerts, J. Raskin, J. Reichert, Realizability of real-time logics, in: *FORMATS’09*, volume 5813 of *LNCS*, pp. 133–148.
- [20] M. Kettner, B. Sewczyk, C. Eickmann, Integrating microscopic and macroscopic models for railway network evaluation, *Association for European Transport* (2003).
- 1555 [21] A. Nash, D. Huerlimann, Railroad simulation using OpenTrack, in: *Computers in Railways IX*, pp. 45–54.
- [22] H. Koustopoulos, Z. Wang, Simulation of urban rail operations: model and calibration methodology, in: *Transport Simulation, Beyond Traditional Approaches*, pp. 153–169.
- 1560 [23] I. Hansen, Railway network timetabling and dynamic traffic management, *International Journal of Civil Engineering* 8 (2010) 19–32.
- [24] V. Cacchiani, P. Toth, Nominal and robust train timetabling problems, *European Journal of Operational Research* 219 (2012) 727–737.
- [25] G. Caimi, M. Fuchsberger, M. Laumanns, K. Schüpbach, Periodic railway timetabling with event flexibility, *Networks* 57 (2011) 3–18.
- 1565 [26] M. Carey, Ex ante heuristic measures of schedule reliability, *Transportation Research* 33 (1999) 473494.
- [27] S. Cicerone, G. D’Angelo, G. D. Stefano, D. Frigioni, A. Navarra, Delay management problem: Complexity results and robust algorithms, in: *Combinatorial Optimization and Applications*, volume 5165 of *LNCS*, pp. 458–468.
- 1570 [28] A. Aghasaryan, E. Fabre, A. Benveniste, R. Boubour, C. Jard, Fault detection and diagnosis in distributed systems: An approach by partially stochastic Petri nets, *Discrete Event Dynamic Systems* 8 (1998) 203–231.
- [29] C. Dousson, Extending and unifying chronicle representation with event counters, in: *ECAI’02*, pp. 257–261.
- 1575 [30] M. Krichen, S. Tripakis, Conformance testing for real-time systems, *Formal Methods in System Design* 34 (2009) 238–304.
- [31] P. Merlin, A study of the recoverability of computing systems, Ph.D. thesis, University of California, Irvine, CA, USA, 1974.
- 1580 [32] R. R. Razouk, C. V. Phelps, Performance analysis using timed petri nets, in: *Protocol Specification, Testing and Verification IV*, pp. 561–576.

- [33] S. Akshay, B. Genest, L. Hélouët, Decidable classes of unbounded Petri nets with time and urgency, in: Proc. of PETRI NETS 2016, volume 9698 of *LNCS*, pp. 301–322.
- 1585 [34] E. Vicario, G. Bucci, M. Paolieri, J. Torrini, L. Carnevali, J. Giuntini, Oris tool: Analysis of timed and stochastic petri nets, 2010. [www.oris-tool.org](http://www.oris-tool.org).
- [35] M. Paolieri, A. Horváth, E. Vicario, Probabilistic model checking of regenerative concurrent systems, *IEEE Transactions on Software Engineering* 42 (2016) 153–169.
- 1590 [36] B. Aspvall, Y. Shiloach, A polynomial time algorithm for solving systems of linear inequalities with two variables per inequality, in: 20th Annual Symposium on Foundations of Computer Science, IEEE Computer Society, 1979, pp. 205–217.

## Appendix A. Construction of time processes

1595 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  $\theta_0 : \{\perp\} \rightarrow \{0\}$ .

1600 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  $\text{last}(p, E_i, B_i)$  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  $X = \{b \mid b = \text{last}(p, E_i, B_i) \wedge p \in \bullet t\}$  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  $TP_u = TP_{u,|u|}$ .

## Appendix B. Structural unfolding of a STPN

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 ( $b^\bullet = \emptyset$ ), and denote the set of maximal conditions of  $B$  by  $\text{max}(B)$ . 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  $\text{last}(p, E', B')$  the maximal occurrences of place  $p$  w.r.t.  $<$  in  $\langle E', B' \rangle$ . Pre-processes of an unfolding are conflict free sets of events and conditions. They represent potential processes (executions) of  $\mathcal{N}$  when timing constraints are forgotten. A *cut* of a pre-process is an unordered set of conditions. As this set of conditions originates from a pre-process, conditions in a cut have no conflicting events in their causal past. They represent place contents that can be consumed by the next firable transitions at some point in an execution. We denote by  $\text{Cuts}(E, B)$  the set of cuts of pre-process  $\langle E, B \rangle$ . Unfolding of a Petri net simply consists in successively appending transitions to already built processes, i.e. to cuts of these processes.

1620 **Structural unfolding:** Following [5], we inductively build unfoldings  $\mathcal{U}_0, \dots, \mathcal{U}_K$ . 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  $\mathcal{U}_0 = \langle \emptyset, B_0 \rangle$  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  $\hat{E} \triangleq \{\langle B, t \rangle \in (2^{B_k} \times T) \setminus E_k \mid \exists \langle X, Y \rangle \in \mathcal{PE}(\mathcal{U}_k), B \subseteq \text{Cuts}(X, Y), \bullet t = \text{pl}(B)\}$ , 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  $\hat{E}$ .

## Appendix C. Constraints to reintroduce time in processes

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  $B = \bullet E \cup E^\bullet$

1635 ( $B$  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 1640  $\text{dob}(b)$  represents the date at which condition  $b$  is created,  $\Phi_{E,B}$  must impose that for every  $b \in B_0$ ,  $\text{dob}(b) = d_0$ .

For all other conditions  $b = \langle e, p \rangle$ , as the date of birth is exactly the occurrence date of  $e$ , we set  $\text{dob}(b) = \theta(e)$  for every  $b = \langle e, p \rangle$ . Despite this equality, we will 1645 use both variables  $\theta(e)$  and  $\text{dob}(b)$  for readability reasons. Recall that  $\text{dod}(b)$  is a variable that designates the date at which a place is emptied by some transition firing,  $\text{dod}(b)$  is hence the occurrence date of an event that has  $b$  as predecessor. Within a conflict free set of events, this event 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  $B_1 \uplus B_2 \uplus \dots \uplus B_{|P|}$ , where 1650 conditions in  $B_i$  represent fillings of place  $p_i$ . Due to blocking semantics, all conditions in a particular subset  $B_i = \{b_{i,1}, b_{i,2}, \dots, b_{i,k}\}$  must have disjoint existence dates, that is for every  $j, j' \in \{1, 2, \dots, k\}$  with  $j \neq j'$ , the intersection between  $[\text{dob}(b_{i,j}), \text{dod}(b_{i,j})]$  and  $[\text{dob}(b_{i,j'}), \text{dod}(b_{i,j'})]$  is either empty, or limited to a single value. This constraint can be encoded by the disjunction:

$$1655 \text{no-overlap}(b_{i,j}, b_{i,j'}) = \begin{cases} \text{dod}(b_{i,j}) \leq \text{dob}(b_{i,j'}) \vee \text{dod}(b_{i,j'}) \leq \text{dob}(b_{i,j}) & \text{if } b_{i,j}^\bullet \neq \emptyset \wedge b_{i,j'}^\bullet \neq \emptyset, \\ \text{dod}(b_{i,j}) \leq \text{dob}(b_{i,j'}) & \text{if } b_{i,j}^\bullet \neq \emptyset \wedge b_{i,j'}^\bullet = \emptyset, \\ \text{dod}(b_{i,j'}) \leq \text{dob}(b_{i,j}) & \text{otherwise.} \end{cases}$$

Note that if  $b_j \preceq b_{j'}$ , then the constraint among events and transitions immediately ensures  $\text{dob}(b_{j,i}) \leq \text{dod}(b_{j,i}) \leq \text{dob}(b_{j',i}) \leq \text{dod}(b_{j',i})$ . However, we need to add a consistency constraint for every pair of concurrent conditions  $b_{i,j}, b_{i,j'}$  1660 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  $\text{non-blocking}(b_{i,j}) = \bigwedge_{b_{i,j'} \in I(b_{i,j}, E, B)} \text{no-overlap}(b_{i,j}, b_{i,j'})$ . In words, condition  $b_{i,j}$  does not hold during the validity dates of any concurrent condition representing the same place. In particular, a time process of  $\mathcal{N}$  cannot 1665 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  $\text{doe}(e)$  the date of enabling of  $e$ , we necessarily have  $\text{doe}(e) = \max\{\text{dob}(b) \mid b \in B\}$ . Event  $e$  is firable at least  $\text{eft}(t)$  1670 time units, and at most  $\text{lft}(t)$  time units after being enabled. We hence have  $\text{doe}(e) + \text{eft}(t) \leq \text{dof}(e) \leq \text{doe}(e) + \text{lft}(t)$ . 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  $\theta(e) = \text{dob}(b_1), \theta(e) = \text{dob}(b_2), \dots, \theta(e) = \text{dob}(b_k)$ , 1675 where  $\{b_1, b_2, \dots, b_k\} = e^\bullet$ , and constraints  $\text{non-blocking}(b_1), \text{non-blocking}(b_2), \dots$ ,

non-blocking( $b_k$ ). Last, as semantics of STPNs is urgent, once fireable,  $e$  has to fire at the earliest possible date. This is encoded by the constraint  $\theta(e) = \min\{x \in \mathbb{R}_{\geq 0} \mid x \notin ]\text{dob}(b), \text{dod}(b)[ \text{ for some } b \in \bigcup I(b_i) \wedge x \geq \text{dof}(e)\}$ . Figure C.14 shows the effect of blocking and possible free firing dates for some event with a condition  $b$  in its postset. The top of the figure is a part of a pre-process, with conditions  $b, b_0, b_1, b_2$  referring to the same place  $p_1$ . Suppose that values of variables  $[\text{dob}(b_i)$  and  $\text{dod}(b_i)]$  for  $i \in 0, 1, 2$  are already known. The situation is depicted by the drawing at the bottom of Figure C.14. Horizontal lines represent real lines, and line portion between brackets represent intervals  $[\text{dob}(b_i), \text{dod}(b_i)]$  for  $i \in 0, 1, 2$ . According to the considered pre-process, we have  $I(b) = \{b_0, b_1, b_2\}$ . Then  $[\text{dob}(b), \text{dod}(b)]$  have to be fully inscribed in one of these thick segments of the Figure. An event with  $b$  in its postset (as event  $e$  in the pre-process at the top of the Figure) can occur only at dates contained in these thick segments.

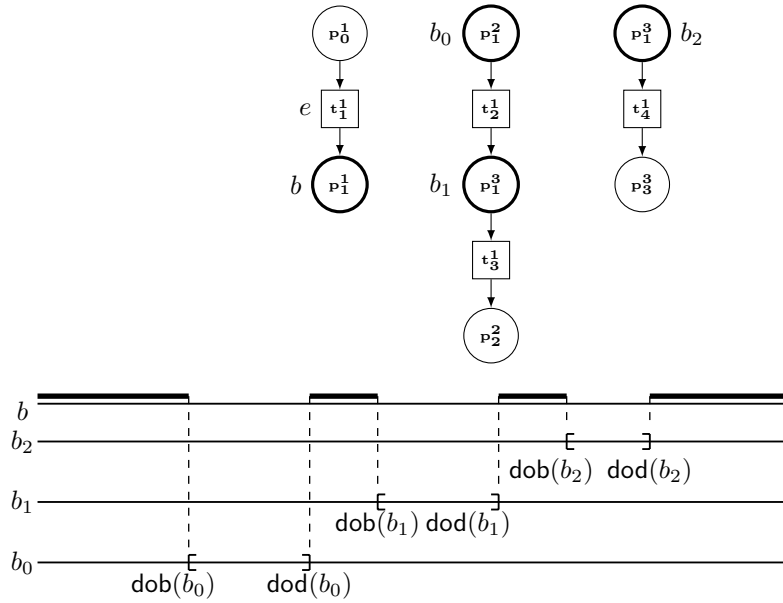


Figure C.14: Constraints on dates of birth of tokens in a shared place.

Written differently,

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

This formula can be translated in boolean combinations of inequalities over variables of  $\text{var}(E, B)$ . 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  $\bigwedge_{e' \# e} \text{notMoreUrg}(e, e')$  to guarantee



1695 that there exists no other event that is forced to occur before  $e$  due to urgency.

We define  $\text{notMoreUrg}(e, e')$  as the following constraint:

$$\text{notMoreUrg}(e, e') = \theta(e) \geq \text{doe}(e') + \text{lft}(\text{tr}(e')) \Rightarrow \text{tiled}(e, e') \vee \bigvee_{e'' \parallel e} \text{preempts}(e', e'')$$

1700 where  $\text{tiled}(e, e') = \text{free}(e') \cap [\text{doe}(e') + \text{lft}(\text{tr}(e')), \theta(e)] = \emptyset$ ,  $e'' \parallel e$  refers to events that are concurrent with  $e$  in the considered set of events  $E$ ,  $\text{free}(e') = \mathbb{R}_{\geq 0} \setminus \{[\text{dob}(b), \text{dod}(b)] \mid \exists b' \in e'^{\bullet}, b \in I(b')\}$  is the set of intervals in which places attached to conditions in  $e'^{\bullet}$  are empty, and  $\text{preempts}(e', e'') = \theta(e'') \leq \min([\text{doe}(e') + \text{lft}(\text{tr}(e')), \theta(e)] \cap \text{free}(e''))$  means  $e''$  disabled  $e'$  by consuming a condition in  $e''$ .

1705 Constraint  $\text{notMoreUrg}(e, e')$  means that if  $e'$  is in conflict with  $e$ , then at least one condition in  $e'^{\bullet}$  is consumed before  $e'$  can fire, or if  $e'$  becomes fireable before  $e$  fires, the urgent firing of  $e'$  is delayed by blockings so that  $e$  can occur. As for constraint attached to blockings,  $\text{notMoreUrg}(e, e')$  can be expressed as a boolean combination of inequalities. One can also notice that  $\text{notMoreUrg}(e, e')$  can be expressed without referring to variables attached to event  $e'$  nor  $e'^{\bullet}$ , as

$$1710 \text{doe}(e') = \max_{b_i \in e'^{\bullet}} \text{dob}(b_i) \text{ and the intersection of } I(b) \text{ and } e'^{\bullet} \text{ is void.}$$

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:

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

$$1715 \forall e \in E, \Phi_{E,B}(e) = \begin{cases} \text{doe}(e) = \max_{b \in e^{\bullet}} \text{dob}(b) \\ \wedge \text{doe}(e) + \text{eft}(\text{tr}(e)) \leq \text{dof}(e) \leq \text{doe}(e) + \text{lft}(\text{tr}(e)) \\ \wedge \text{dof}(e) \leq \theta(e) \wedge \bigwedge_{b \in e^{\bullet}} \text{dod}(b) = \theta(e) \\ \wedge \bigwedge_{b \in e^{\bullet}} \theta(e) = \text{dob}(b) \\ \wedge \bigwedge_{e' \# e} \text{notMoreUrg}(e, e') \end{cases}$$

Let us now address maximality of symbolic prefixes wrt urgent events firing. Let  $SPP = \langle E', B', \Phi_{E',B'} \rangle$  be a symbolic prefix of pre-process  $PP = \langle E, B \rangle$ . Symbolic process  $SPP$  is *maximal w.r.t urgent events firing* iff no more event of  $PP$  **have to** belong to  $SPP$ . This property of  $SPP$  holds if every event

1720  $f \in B'^{\bullet} \cap E$  that could have become urgent before the last date of all events in  $E'$  was prevented from firing due to blocking. This property prefixes can be verified as a property  $\Phi_{\max}(f)$  that have to be satisfied for every  $f \in B'^{\bullet} \cap E$ . Let  $C_f = \text{pl}^{-1}(f^{\bullet}) \cap B'$  denote the set of conditions of  $B'$  whose place appears in the postset of  $f$ . Then,  $SPP$  is maximal iff for every  $f \in B'^{\bullet} \cap E$ , the following

1725 constraint is not satisfiable.

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

Intuitively,  $\Phi_{\max}(f)$  means that  $f$ , 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  $\Phi_{\max}(f)$  is satisfiable, then  $f$  should appear in the process.

## Appendix D. Proof of proposition 1

**Proposition 1.** Let  $\mathcal{N}$  be a STPN guaranteeing time progress of  $\delta$  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.  $\{b' \in \text{block}(b) \mid \text{dob}(b') \leq D\}$  is contained in  $\mathcal{U}_K$ .

**Proof:** Consider a pre-process  $PP$  of  $\mathcal{U}_n$ , which depth is more than  $\lceil \frac{D}{\delta} \rceil \cdot |T|$  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  $b_0 < e_1 < b_1 < \dots < e_n < b_n$  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 least at dates that differ by  $\delta$ . Hence, an event created at depth  $n$  has an occurrence date of at least  $\delta \cdot \lfloor n/|T| \rfloor$ . The occurrence date of an event created at depth greater than  $\frac{D}{\delta} \cdot |T|$  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  $\frac{D}{\delta} \cdot |T|$  to obtain the (finite) set of conditions that refer to the same place as some condition  $b$  before a given date  $D$ .  $\square$

## Appendix E. Proof of Proposition 2

**Proposition 2.** Let  $\mathcal{N}$  be a STPN guaranteeing time progress of  $\delta$  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  $\mathcal{E}^{s'} = \langle E', B', \Phi' \rangle$  be two symbolic processes. We will say that  $\mathcal{E}^s \sqsubseteq \mathcal{E}^{s'}$  iff there exists an event  $e'$  such that  $E' = E \cup \{e'\}$ ,  $B' = B \cup e'^{\bullet}$ , and  $\Phi = \Phi'_{\text{var}(E, B)}$ .  $\square$

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

1765 **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  $\theta_{n+1}$  denote a solution for this process.

1770 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  $\theta(e) \geq \text{dob}(b)$  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 restriction of  $\Phi_{n+1}$  to variables attached to events and conditions  $E^n, B^n$ . One has to remove variables  $\theta(e)$ ,  $\text{dob}(b)$  for every  $b \in \bullet e$ , and  $\text{dob}(b)$  for every  $b \in e^\bullet$  using an elimination technique such as Fourier-Motzkin. Using the properties of elimination,  $\theta$  satisfies  $\Phi_{n+1}$  if and only if the restriction of  $\theta$  to  $\text{var}(E^n, B^n)$  satisfies  $\Phi_{n+1|E^n, B^n}$ . However, the restriction of  $\theta$  is exactly  $\theta_n$ , and as  $\theta(e)$ ,  $\text{dob}(b)$  for  $b \in e^\bullet$ , and  $\text{dob}(b)$  for

1775  $b \in \bullet e$  are all greater than variables in  $\text{var}(E^n, B^n)$ , the elimination of variables is simply a projection on atoms that do not contain variables related to  $e$ , and  $\Phi_{n+1|E^n, B^n} = \Phi_n$ .  $\square$

**Lemma 2.** *Given a symbolic process  $\mathcal{E}^s$  of  $\mathcal{U}_K$ , one of its solutions  $\theta$ , 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  $u_{\mathcal{E}^s, \theta} = \langle t_1, \theta(e_1) \rangle \dots \langle t_{|E|}, \theta(e_{|E|}) \rangle$  is a timed word of  $\mathcal{L}(\mathcal{N})$ .*

1785

**Proof:** Again we can prove this lemma by induction. The base case is obvious, as the empty word  $\epsilon$  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  $\theta_n$  meeting all constraints of  $\mathcal{E}_n$ , there exists an increasing sequence of prefixes of  $\mathcal{E}_n$  such

1790 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  $\theta_{n+1}$ . 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  $\mathcal{E}_n$ .

1795 This word may lead the net to any configurations in a set  $\text{Conf}_n$  with identical markings, but distinct times to fire attached to transitions. However, as we know that  $\theta_{n+1}$  meets all constraints of  $\mathcal{E}_{n+1}$ , there exists a configuration in  $\text{Conf}_n$  whose times to fire allow firing of  $e_{n+1}$  at date  $\theta(e_{n+1})$ , and  $u_{n+1} = u_n \cdot \langle e_{n+1}, \theta(e_{n+1}) \rangle \in \mathcal{L}(\mathcal{N})$ .  $\square$

1800 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

1805 of  $\mathcal{E}^s$ , we have  $\mathcal{E}_\theta^s = TP_{u_{\mathcal{E}^s, \theta}}$  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  $\mathcal{N}$ .

1810 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  
1815 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  $u_{n+1} = \langle e_1, \theta(e_1) \rangle \dots \langle e_n, \theta(e_n) \rangle \cdot \langle e_{n+1}, \theta(e_{n+1}) \rangle \in \mathcal{L}_{\leq D}(\mathcal{N})$ . 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  $e_{p_1}, \dots, e_{p_k}$   
1820 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  $\text{eft}(t) + \max\{\theta(e_{p_i})\} \leq \theta(e_{n+1}) \leq \text{lft}(t) + \max\{\theta(e_{p_i})\}$ . In any case, we have  $\text{eft}(t) + \max\{\theta(e_{p_i})\} \leq \theta(e_{n+1})$  (which is the only constraint w.r.t predecessors imposed by constraint in the  
1825 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,  $\theta$  witnesses satisfiability of a set of  
1830 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  
1835 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  $\mathcal{U}_K$ .  $\square$

## Appendix F. Algorithm to compute embeddings of a schedule in a process

1840 Finding the set of embedding functions  $\Psi = \{\psi_0, \psi_1, \dots, \psi_k\}$  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  
1845 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  $n' \in \text{domain}(f)$ .

---

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

---

**input:** a schedule  $S = \langle N, \rightarrow, \lambda, C \rangle$ , a symb. process  $\mathcal{E} = \langle E, B, \Phi \rangle$ ;  
 $\Psi := \emptyset$ ; // the set of solutions is initially empty  
 $F := \{f_{\perp}\}$ ; // the exploration starts from the undefined map

**while**  $F \neq \emptyset$  **do**  
  choose  $f \in F$ ;  
   $\text{MIN}_{S,f} := \min_{\rightarrow}(N \setminus \text{domain}(f))$ ;  
  **if**  $\text{MIN}_{S,f} = \emptyset$  **then** ; // all nodes of  $S$  have an image in  $\mathcal{E}$   
  |  $\Psi := \Psi \cup \{f\}$ ; //  $f$  is an embedding  
  **else**  
  |  $F := F \setminus \{f\}$ ; // we will explore extensions of partial  
  | embedding  $f$   
  |  $\text{MIN}_{\mathcal{E},f} := \min_{\succeq}(E \setminus \text{image}(f))$ ;  
  | **FOUND** := false;  
  | **while**  $\text{MIN}_{S,f} \neq \emptyset \wedge \text{FOUND} = \text{false}$  **do**  
  | | choose  $n \in \text{MIN}_{S,f}$ ;  
  | |  $\text{MIN}_{S,f} := \text{MIN}_{S,f} \setminus \{n\}$ ;  
  | |  $\text{CAND} := \{e \in \text{MIN}_{\mathcal{E},f} \mid \text{tr}(e) \in r(\lambda(n)) \wedge \forall n' \in \text{dpred}(n), f(n') \preceq_{\rightarrow} e\}$ ;  
  | | **if**  $\text{CAND} \neq \emptyset$  **then**  
  | | |  $F := F \cup \bigcup_{e \in \text{CAND}} \{f \cup \langle n, e \rangle\}$ ; // update  $f$  with pairs  
  | | |  $\langle n, e \rangle$  that meet the matching criteria and add the  
  | | | new functions to candidate embeddings  
  | | | **FOUND** := true;  
  | | **end**  
  | **end**  
  | **end**  
  | **end**  
**end**

---

1850 **Appendix G. 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 semantics.

**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  $\mathfrak{S} = \langle V, \circ \rightarrow \cup \bullet \rightarrow \rangle$  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  $\Sigma_0$  (with marking  $m_0$ , a domain  $C_0$  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  $\Sigma$  to a successor class  $\Sigma'$  are labeled by a transition name  $t$  and by the probability  $\mu$  to fire  $t$  from  $\Sigma$ , and are of two forms:

**Firing after elapsing time:** A move  $\Sigma \xrightarrow{t_i, \mu_i} \Sigma'$  from  $\Sigma = \langle m, C, D, \text{BLK}, \text{URG} \rangle$  to  $\Sigma' = \langle m', C', D', \text{BLK}', \text{URG}' \rangle$ , achievable with probability  $\mu_i$ , consists in firing transition  $t_i$  after symbolically elapsing its TTF. Such a move is only allowed if  $\text{URG} = \emptyset$  and the TTF  $\tau_i$  of  $t_i$  is less than or equal to TTFs of all other transitions that could fire from  $\Sigma$ . The time domain  $C^i$  from which  $t_i$  can fire is hence  $C^i = C \cap \bigcap_{x_j \in X_M} \{x_i \leq x_j\}$ , and the probability of firing  $t_i$  from  $\Sigma$  is  $\mu_i = \int_{C^i} D$ . We have  $m' = m - \bullet t_i + t_i \bullet$ . The new domain  $C'$  and distribution  $D'$  are computed as for STPNs with non-blocking semantics: it is obtained by advancing time, removing variables of disabled transitions and adding those of newly enabled transitions [3], and then removing variables of transitions whose domain is the singleton  $\{0\}$ . The domain of a single variable  $x_i$  in a domain  $C$  over several variables can be obtained by eliminating all variables but  $c_i$  from  $C$ . 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  $\text{BLK}'$  is obtained by removing from  $\text{BLK}$  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  $C'$  that is  $\{0\}$ . Finally, the set  $\text{URG}'$  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 frable 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  $\Sigma$  has urgent transitions ( $\text{URG} \neq \emptyset$ ), only moves of the form  $\Sigma \xrightarrow{t_i, \mu_i} \Sigma'$  are allowed. They consist in firing a transition  $t_i$  among urgent transitions in  $\text{URG}$  with probability  $\mu_i = \mathcal{W}(t_k) / \sum_{t_j \in \text{URG}} \mathcal{W}(t_j)$ . Components  $m'$ ,  $C'$  and  $D'$  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  $\text{BLK}'$  is obtained by removing from  $\text{BLK}$  transitions that are unlocked or disabled by the firing of  $t_i$  and adding those that become blocked in  $m'$ , and  $\text{URG}'$  contains transitions from  $\text{URG}$  that were not disabled by firing of  $t_i$  and transitions from  $\text{BLK}$  that were unblocked when firing  $t_i$ .

**Computing the probability of a process  $\mathcal{E}^s$ :**

Transient trees are a priori infinite, but very often, one can work with a bounded horizon. This is our case when evaluating the probability of realization in  $\mathcal{U}_K$ . Let  $\Psi_i = \{\psi_{i,0}, \psi_{i,1}, \dots, \psi_{i,n-1}\}$  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  $\langle \Sigma, S \rangle \xrightarrow{t_k, \mu_k} \langle \Sigma', S' \rangle$  representing firing of a transition  $t_k$  if there is a minimal event  $e$  in the remaining suffix of  $\mathcal{E}_i^s$ , and  $\text{tr}(e) = t_k$ .  $S'$  is the new suffix obtained by removing  $e$  from  $S$ .  $\Sigma'$  is the successor class obtained after firing this transition from  $\Sigma$ . Edges are built as before in the tree, but with an additional constraint: edges with label  $t_k, \mu_k$  and components  $m, C, D, \text{BLK}, \text{URG}$  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  $t_k = \text{tr}(e)$  becomes  $\mu_k = \int_{C^k \cap C'^k} D$ , where  $C'^k$  is the part of  $C$  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  $\mathfrak{S}_{i,j,\alpha}$ .

Transient tree  $\mathfrak{S}_{i,j,\alpha}$  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  $\mathbb{P}(\mathcal{E}_i^s \wedge \Phi_{\psi_{i,j}, \alpha}) = \sum_{\rho \in \text{PATH}(\mathfrak{S}_{i,j,\alpha})} \mathbb{P}(\rho)$  where  $\text{PATH}(\mathfrak{S}_{i,j,\alpha})$  is the set of paths from  $\langle \Sigma_0, \mathcal{E}_i^s \rangle$  to a leaf of  $\mathfrak{S}_{i,j,\alpha}$ , and the probability  $\mathbb{P}(\rho)$  of a path  $\rho = \Sigma_0 \xrightarrow{t_1, \mu_1} \dots \xrightarrow{t_{l-1}, \mu_{l-1}} \Sigma_l$  that start at  $\Sigma_0$  and end on a leaf  $\Sigma_l$  is the product  $\prod_{i \in \{0, \dots, l-1\}} \mu_i$ . 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.

## 1940 Appendix H. Derivation of components of successor class

We hereafter provide details on how to compute components  $C'$  and  $D'$  of a class  $\Sigma'$  obtained from a class  $\Sigma$  through a transition  $\overset{t_i, \mu_i}{\circlearrowright}$ . Derivation of components  $m'$ ,  $\text{BLK}'$  and  $\text{URG}'$  has already been covered and need not further explanations.

1945 We shall use the following notations:

- given a time domain  $C$  delimiting the possible values of a set of variables  $X_M = \{x_0, x_1, \dots, x_{N-1}\}$ , we will denote by  $C \downarrow_{x_i}$  the projection of  $C$  that eliminates variable  $x_i$  from  $X_M$ . The elimination is done via the Fourier-Motzkin method detailed in Appendix Appendix I;
- 1950 • 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  $\underline{x} + x_0$ .

**Probability of firing:** A transition  $t_i$  can fire from class  $\Sigma$  iff  $t_i$  is enabled by  $m$  and no postset place of  $t_i$  is occupied; that is  $\forall p \in t_i^\bullet, m(p) = 0$ . We also need its TTF  $x_i$  to be less than or equal to TTFs of all variables in  $X_M$ ; transition  $t_i$  will then fire from  $\Sigma$  with probability  $\mu^i$ , with:

$$\mu^i = \int_{C^i} D$$

$C^i$  is the time domain from which  $t_i$  shall fire with all its TTF less than or equal to every variable in  $X_M$ .

$$C^i = C \cap \bigcap_{x_j \in X_M} \{x_i \leq x_j\}$$

**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  $X_M^a$  distributed over  $C^i$  according to the following conditional PDF:

$$D^a = D / \mu^i$$

**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  $\tau^i$ . This yields a new random variable  $X_M^b = X_M^a - x_i$  distributed over



the domain  $C^b = C^i \downarrow_{x_i}$  in which the variable attached to the fired transition  $t_i$  is eliminated. The PDF of the new multivariate random variable  $X_M^b$  is then:

$$D^b = \int_{\text{MIN}^i}^{\text{MAX}^i} D^a$$

where  $\text{MIN}^i$  and  $\text{MAX}^i$  denote the bounds of the support of variable  $\tau^i$ .

**Disabling:** If the firing of  $t_i$  disables a transition  $t_j$ , variable  $x_j$  has to be eliminated from the time vector, yielding a new vector  $X_M^c = X_M^b \setminus x_j$  distributed over  $C^c = C^b \downarrow_{x_j}$  with PDF:

$$D^c = \int_{\text{MIN}^j}^{\text{MAX}^j} D^b$$

1955 The same procedure is repeated for every disabled transition by the firing of  $t_i$ . Let  $X_M^{c*}$ ,  $C^{c*}$  and  $D^{c*}$  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  $[\text{eft}(t_k), \text{lft}(t_k)]$ , then the new time vector, that we denote by  $X_M^d$ , shall include an additional component  $x_k$  and shall be distributed over  $C^d = C^{c*} \times [\text{eft}(t_k), \text{lft}(t_k)]$  according to the PDF:

$$D^d = D^{c*} \times f_{t_k}(x_k)$$

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

## 1960 Appendix I. Fourier–Motzkin elimination method

The *Fourier–Motzkin elimination method* is an algorithm for eliminating variables from a system of linear inequalities. Let  $\phi$  be a system of linear inequalities with variables  $x_1, x_2, \dots, x_r$  where  $x_r$  is the variable to be removed.  $\phi$  can be written as  $\phi^+ \wedge \phi^- \wedge \phi^\emptyset$  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 inequality where the coefficients of  $x_r$  are respectively negative and positive, and  $\phi^\emptyset$  is the inequality subsystem in which  $x_r$  does not appear. Eliminating the variable  $x_r$  from  $\phi$  refers to the creation of another system of inequality in which  $x_r$  does not appear and which has the same solutions over the remaining variables. The original system can be written as

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

Which, by eliminating  $x_r$ , gives

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

**example:** Let  $\phi$  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  $\phi^\emptyset$ , the subsystem of inequality in which  $x_1$  doesn't appear, as follows:

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

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

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

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 inequality; adding to that  $\phi^\emptyset$ .

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

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

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