The Right Timing: Reflections on the Modeling and Analysis of Time

Kees van Hee and Natalia Sidorova

Department of Mathematics and Computer Science
Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
{k.m.v.hee,n.sidorova}@tue.nl

Abstract. In this paper we discuss several approaches to time in Petri nets. If time is considered for performance analysis, probability distributions for choices should be included into the model and thus we need Petri nets with time and stochastics. In literature, most attention is paid to models where the time is expressed by delaying transitions and for the stochastic case to continuous time models with exponential enabling distributions, known by its software tools as GSPN. Here we focus on discrete models where the time is expressed by delaying tokens and the probability distributions are discrete, because this model class has some advantages. We show how model checking methods can be applied for the non-stochastic case. For the stochastic case we show how Markov techniques can be used. We also consider structural analysis techniques, which do not need the state space.

1 Introduction

Over the last 25 years, the modeling and analysis of time has been studied extensively in the context of Petri nets as well as of other models of concurrency. Most of the Petri net models with time are extensions of classical Petri nets, so if we discard the time aspects we obtain a classical Petri net. There are many different approaches to model time such as: delaying of instantaneous transitions, duration of transitions, aging or delays in tokens, timers and clocks. The earliest papers seem to be [27] which introduces delaying of transitions and [30] introducing duration of transitions. These model classes are often referred to as Merlin time (or Time Petri Nets) and Timed Petri Nets respectively. Many authors have contributed to these models with results on *expressiveness*, e.g. [13,15,20,36] and on *model checking* [10,9,34,3,22]. Next to the verification questions, *performance* analysis (c.f. [35,35,31,16]) is an important reason for extending models with time features. While verification is concerned with the extremities of behavior (like "will every service request be processed within 5 time units?"), performance analysis is concerned with "average" behavior or "normal" behavior, i.e. behavior within bounds with a specified probability (like "will 90% of service requests be processed within 5 time units?"). To make performance analysis possible, non-deterministic choices in models should be endowed with a probability.

J.-M. Colom and J. Desel (Eds.): PETRI NETS 2013, LNCS 7927, pp. 1–20, 2013.

There is an overwhelming literature on timed automata and their relationship to Petri nets with time,e.g. [5,24,14,12]. Most of these papers refer to timed automata with clocks as incorporated in the UPPAAL tool [40,6]. There is also a very extensive literature on stochastic Petri nets where the execution time of transitions is exponentially distributed which leads to continuous time Markov processes, see e.g. [4,26,18]. For the class of Generalized Stochastic Petri Nets, having also instantaneous transitions, there is a famous software tool for modeling and analysis GSPN [39]. This class is in fact the stochastic extension of Merlin time, so the approach with delaying transitions. Different approaches proposed for modeling time dimension in Petri nets all have their own merits and weaknesses. Some of the approaches are questionable from the modeling point of view but are handy for analysis purposes, while others are good for modeling but bad for analysis.

In this paper we focus on the approach with time in tokens since we have the feeling that this class did not obtain enough attention although it is a powerful class both for modeling and for analysis. Extending tokens with time information is studied in a number of works, see [19,2,21,8,23,11]. There are multiple examples of industrial applications of this model of time in Petri nets, often called Interval Timed Petri Nets. We restrict our focus to the discrete time setting and call the corresponding class DTPN (Discrete Time Petri Nets). The software tools CPN Tools [37] and ExSpect [38] are using a simplification of it. Next to that, we will consider a stochastic variant of DTPN, called DSPN that can be seen as a discrete alternative to the GSPN model. This class has *discrete* time, i.e. finite or countable delay sets and discrete probability distributions over the choices and it encompasses several well-known subclasses.

We start with preliminaries in Section 2. In Section 3, we compare different options for introducing timed elements in Petri nets, without claiming to be complete. We do not consider continuous Petri [32] nets there because the underlying untimed net is not a classical Petri net any more. In Section 4, we define DTPN and show how the subclasses of DTPN are related. We show by modeling inhibitor arcs that several subclasses of DTPN are Turing complete. In Section 5, we consider a stochastic version of DTPN and we show how classical Markov techniques can be used to analyse them. In particular, we consider three questions: the probability of reaching a set of states, the expected time to reach a set of states and the equilibrium distribution. For these methods we need the whole state space. For workflow nets, there are structural techniques that can be combined with the others.

2 Preliminaries

We denote the set of reals, rationals, integers and naturals by $\mathbb{R}$, $\mathbb{Q}$, $\mathbb{Z}$ and $\mathbb{N}$ (with $0 \in \mathbb{N}$), respectively. We use superscripts $+$ for the corresponding subsets containing all the non-negative values, e.g. $\mathbb{Q}^+$. A set with one element is called a *singleton*. Let $\inf(A)$, $\sup(A)$, $\min(A)$ and $\max(A)$ of the set A have the usual meaning and $\mathcal{P}(A)$ is the power set of A. We define $\max(\emptyset) = \infty$ and $\min(\emptyset) =$

$-\infty$ and we call a set $\{x \in \mathbb{Q} | a \leq x \leq b\}$ with $a, b \in \mathbb{Q}^+$ a *closed rational interval*. The set B is a *refinement* of set A denoted by $A \lhd B$ if and only if $A \subseteq \mathbb{Q}^+ \wedge A \subseteq B \wedge \sup(A) = \sup(B) \wedge \inf(A) = \inf(B)$.

A *labeled transition system* is a tuple $(S, A, \rightarrow, s_0)$ where S is the set of states, A is a finite set of *action names*, $\rightarrow \subseteq S \times A \times S$ is a transition relation and $s_0 \in S$ is an initial state. We write $(s, a, s') \in \rightarrow$ when $s \xrightarrow{a} s'$. An action $a \in A$ is called *enabled* in a state $s \in S$, denoted by $s \xrightarrow{a}$, if there is a state s' such that $s \xrightarrow{a} s'$. If $s \xrightarrow{a} s'$, we say that state s' is reachable from s by an action labeled a.

We lift the notion of reachability to sequences of actions. We say that a non-empty finite sequence $\sigma \in A^*$ of length $n \in \mathbb{N}$ is a firing sequence, denoted by $s_0 \xrightarrow{\sigma} s_n$, if there exist states $s_i, s_{i+1} \in S$ such that $s_i \xrightarrow{\sigma(i)} s_{i+1}$ for all $0 \leq i \leq n-1$. We write $s \xrightarrow{*} s'$ if there exists a sequence $\sigma \in A^*$ such that $s \xrightarrow{\sigma} s'$ and say that s' is reachable from s.

Given two transition systems $N_1 = (S_1, A_1, \rightarrow, s_0)$ and $N_2 = (S_2, A_2, \rightarrow, s_0')$, a binary relation $R \subseteq S_1 \times S_2$ is a *simulation* if and only if for all $s_1 \in S_1, s_2 \in S_2, a \in A_1$, $(s_1, s_2) \in R$ and $s_1 \xrightarrow{a} s_1'$ implies that there exist $s_2' \in S_2$ such that $s_2 \xrightarrow{a} s_2'$ and $(s_1', s_2') \in R$. We write $N_2 \preceq N_1$ if a simulation relation R exists. If R and R^{-1} are both simulations, relation R is called a *bisimulation* denoted by $\simeq$. If a simulation relation R' exists such that $N_1 \preceq N_2$ then N_1 and N_2 are *simulation equivalent*. We use the notion of *branching bisimulation* as defined in [17].

A *Petri net* is a tuple $N = (P, T, F)$, where P is the set of *places*, T is the set of *transitions*, with $P \cap T = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ is a *flow relation*. An *inhibitor net* is a tuple (P, T, F, ι), where (P, T, F) is a Petri net and $\iota : T \rightarrow \mathcal{P}(P)$ specifies the inhibitor arcs. We refer to elements of $P \cup T$ as *nodes* and elements from F as *arcs*. We define the preset of a node n as ${}^\bullet n = \{m | (m, n) \in F\}$ and the postset as $n^\bullet = \{m | (n, m) \in F\}$.

The state of a Petri net $N = (P, T, F)$ is determined by its marking which represents the distribution of tokens over the places of the net. A marking m of a Petri net N is a bag over its places P. A transition $t \in T$ is enabled in m if and only if ${}^\bullet t \leq m$. If N is an inhibitor net then for the enabling of transition t in a marking m, we additionally require that $m(p) = 0$ for all places $p \in \iota(t)$. An enabled transition may fire, which results in a new marking $m' = m - {}^\bullet t + t^\bullet$, denoted by $m \xrightarrow{t} m'$.

3 Overview of Petri Nets with Time

In this section we describe several options for extending Petri nets with time. To make the syntax uniform, we use the same definition for different classes of timed Petri nets; in this definition we add delay sets to the arcs of a classical Petri net. The semantics of these delay sets differ significantly in different model classes, and additional constraints on delays will be imposed in certain cases.

Definition 1 (Timed Petri net). *A* timed Petri net (TPN) *is a tuple* (P, T, F, δ)*, where* (P, T, F) *is a Petri net,* $\delta : F \rightarrow \mathcal{P}(\mathbb{R})$ *is a function assigning delay sets of non-negative delays to arcs.*

We consider $\delta(p,t)$, $(p,t) \in F$, as an *input delay* for transition t and $\delta(t,p)$, $(t,p) \in \cap F$, as an *output delay* for transition t. We distinguish multiple subclasses of TPNs using different combinations of the following restrictions of the delay functions:

***In-zero*:** Input arcs are non-delaying, i.e. for any $(p,t) \in F$, $\delta(p,t) = 0$.
***Out-zero*:** Output arcs are non-delaying, i.e. for any $(t,p) \in F$, $\delta(t,p) = 0$.
***In-single*:** Input delays are fixed values, i.e. for any $(p,t) \in F$, $|\delta(p,t)| = 1$.
***Out-single*:** Output delays are fixed values, i.e. for any $(t,p) \in F$, $|\delta(t,p)| = 1$.
***In-fint*:** Input delays are finite rational sets.
***Out-fint*:** Output delays are finite rational sets.
***In-rint*:** Input delays are closed rational intervals.
***Out-rint*:** Output delays are closed rational intervals.
***Tr-equal*:** For every transition, the delays on its input arcs are equal, i.e. for any $t \in T, (p_1,t), (p_2,t) \in F$, $\delta(p_1,t) = \delta(p_2,t)$
***Pl-equal*:** For every place, the delays on its input arcs are equal, i.e. for any $p \in P, (p,t_1), (p,t_2) \in F$, $\delta(p,t_1) = \delta(p,t_2)$.

There are models of time for Petri nets placing timed elements on places or transitions instead of arcs. *Tr-equal* allows to define a delay interval for a transition and *Pl-equal* – a delay interval for a place. We can combine the restrictions (e.g. *In-zero, Out-single*).

There are several *dimensions* on which different models of time for Petri nets differ semantically, such as:

- *Timed tokens vs untimed tokens*: Some models extend tokens, and thus also markings, with the time dimension, e.g. with timestamps to indicate at which moment of time these tokens become consumable or/and till which time the tokens are still consumable. A transition may fire if it has consumable tokens on its input places. Other models keep tokens untimed, meaning in fact that tokens are always consumable. The time semantics is then captured by time features of places/transitions only.
- *Instantaneous firing vs prolonged firing:* In some models, the firing of a transition takes time, i.e. the tokens are removed from the input places of a transition when the firing starts and they are produced in the output places of the transition when the firing is finished. In an alternative semantics, a potential execution delay is selected from a delay interval of a transition, but the transition firing is instantaneous.
- *Eager/urgent/lazy firing semantics*: In the eager semantics, the transition that can fire at the earliest moment is the transition chosen to fire; with the urgent semantics the transition does not have to fire at the earliest moment possible, but the firing may become urgent, i.e. there is a time boundary for the firing; in the lazy semantics the transitions do not have to fire even if they loose their ability to fire as a consequence (because e.g. the tokens on the input places are getting too old and thus not consumable any more).
- *Preemption vs non-preemption*: Preemption assumes that if a transition t gets enabled and is waiting for its firing for a period of time defined by its

delay and another transition consumes one of the input tokens of t, then the clock or timer resumes when t is enabled again, and thus its firing will be delayed only by the waiting period left from the previous enabling. The alternative is called non-preemption.
- *Non-deterministic versus stochastic choices* in the delays or order of firing.

We introduce the notion of a *timed marking* for all timed Petri nets. We assume the existence of a *global clock* that runs "in the background". The "current time" is a sort of cursor on a time line that indicates where the system is on its journey in time. We give the tokens a unique *identity* in a marking and a *timestamp*. The timestamps have the meaning that a token cannot be consumed by a transition before this time.[1]

Definition 2 (Marking of a TPN). *Let I denote a countably infinite set of identifiers. A* marking *of a TPN is a partial function $m : I \nrightarrow P \times \mathbb{Q}$ with a finite domain. For $i \in dom(m)$ with $m(i) = (p, q)$ we say that (i, p, q) is a token on place p with timestamp q. We denote the set of all markings of a TPN by $\mathbb{M}$ and we define the projection functions π, τ as $\pi((p, q)) = p$ and $\tau((p, q)) = q$.*

The semantics of the different model classes are given by different transition relations. However, they have a commonality: if we abstract from the time information, the transition relation is a subset of the transition relation of the classical Petri net. This requirement to the semantics is expressed in the following definition.

Definition 3 (Common properties of TPNs). *A transition relation $\rightarrow \subseteq \mathbb{M} \times T \times \mathbb{M}$ of a TPN satisfies the following property: if $m \xrightarrow{t} m'$ then:*

- *$dom(m' \backslash m) \cap dom(m \backslash m') = \emptyset$ (identities of new tokens differ from consumed tokens),*
- *$\{\pi(m(i)) | i \in dom(m \backslash m')\} = {}^\bullet t$ and $|m \backslash m'| = |{}^\bullet t|$ (consumption from the pre-places of t), and*
- *$\{\pi(m'(i)) | i \in dom(m' \backslash m)\} = t^\bullet$ and $|m' \backslash m| = |t^\bullet|$ (production to the post-places of t).*

Given a marking $m_0 \in \mathbb{M}$, a sequence of transitions $\langle t_1, \ldots, t_n \rangle$ is called a firing sequence *if $\exists m_1, \ldots, m_n \in \mathbb{M} : m_0 \xrightarrow{t_1} m_1 \ldots \xrightarrow{t_n} m_n$. We denote the finite set of all firing sequences of a TPN N from a marking $m \in \mathbb{M}$ by $FS(N, m)$.*

There are three main classes of TPN, based on the time passing scheme chosen:

- *Duration of firing* ($M1$)
  Time is connected to transitions (Tr-equal). As soon as transitions are enabled one of them is selected and for that transition one value is chosen from

[1] By extending the time domain to $Time \times Time$ with $Time \subseteq \mathbb{R}$ we could talk about "usability" of tokens, with tokens having the earliest and the latest consumption times.

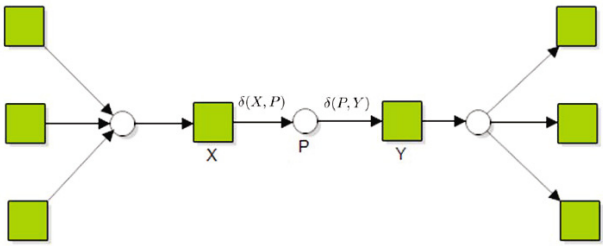

Fig. 1. Modeling task execution times

the input delay interval. Then the tokens are consumed immediately and new tokens for the output places are produced after the delay. The global clock moves till either the end of the delay or to an earlier moment where another transition is enabled. This is the Timed Petri Net model.

– *Delaying of firing* ($M2$)
Time is connected to transitions (Tr-equal). As soon as transitions are enabled, for each of them a delay is selected from the input delay interval. One of the transitions having the minimal of those delays is chosen to fire after the delay has passed. This is often called Merlin time and race semantics.
– *Delaying of tokens* ($M3$)
Time is connected to tokens (Out-rint, In-single). This is the DTPN model where a transition can fire at the time of the maximal timestamp of its input tokens. One of the earliest enabled transitions will fire. The input delay is added to the timestamp in order to determine if it is consumable or not.

For modeling systems in practice, it is often possible to model synchronization by instantaneous transitions with zero input and zero output delays. Activities that take time are modeled by a *begin* transition X and an *end* transition Y, as in Fig. 1. In this case, all model classes are equal since if X fires at time t then Y will fire at $t + a + b$, where $a \in \delta(X, P)$ and $b \in \delta(P, Y)$. Furthermore, input delays are handy for modeling time-outs.

Timed automata are another important class of timed models (see [6]). For model classes $M1$ and $M2$, it is possible to translate them to timed automata [25,14]. Some models of model class $M3$ (DTPN) can also be translated to timed automata, but because of eagerness the translations is only possible for the subclass (*Tr-equal, In-single, Out-rint*) (see [7]).

Generalized Stochastic Petri Nets (GSPN) is the famous class of stochastic Petri nets. It belongs to the class of $M2$ (race semantics with non-preemption). In general, non-preemption is a strange property from a practical point of view, because independent parts of a system are "working" while waiting for the firing of their transitions, and in case the input tokens of some transition t, being in preparation to its firing, are taken by another transition, the preparation work done by t is lost. However, preemption and non-preemption coincide in case of exponentially distributed delays: if a transition with a delayed firing is

interrupted, the rest-distribution in case of preemption is the same as the original waiting time. Therefore these models can be transformed into a continuous time Markov process and the analysis techniques for Markov processes can be applied. However, exponential distributions are very restrictive for modeling in practice. It is possible to approximate an arbitrary continuous distribution by a *phase type* distributions, which are combinations of exponential distributions, but this is not a nice solution to model arbitrary transition delay distributions, because it blows up the state space and it disturbs the preemption property. Therefore, in Section 5, we will introduce a stochastic version of the class of $M3$, which we call $DSPN$.

4 Discrete Timed Petri Nets

In this section we first define the semantics of model class DTPN and we consider two subclasses of DTPN: sDTPN and fDTPN. The subclass sDTPN satisfies (*Out-single, In-single*) and an fDTPN satisfies (*In-single, Out-fint*). We show that DTPN, sDTPN and fDTPN are equivalent and we will transform sDTPN, by reducing the time component, into a strongly bisimilar labeled transition system called rDTPN. For rDTPN we show that it has a finite reachability graph if the underlying Petri net is bounded. So rDTPN can be used for model checking and since it is equivalent to the general DTPN we are able to model check them as well. For the proofs of the formal theorems we refer to [7].

4.1 Semantics of DTPN

In order to define the firing rule of a DTPN we introduce an *activator*. For a transition t in a marking m the activator is a minimal subset of the marking that enables this transition. Like in classical Petri nets, an activator has exactly one token on every input place of t and no other tokens.

Definition 4 (Activator). *Consider a TPN with the set of all markings* $\mathbb{M}$. *A marking* $a \in \mathbb{M}$ *is called an* activator *of transition* $t \in T$ *in a marking* $m \in \mathbb{M}$ *if (1)* $a \subseteq m$ *(a coincides with* m *on* $dom(a)$*), (2)* $\{\pi(m(i)) \mid i \in dom(a)\} = {}^\bullet t$ *and* $|a| = |{}^\bullet t|$ *(*t *is classically enabled), and (3) for any* $i \in dom(a) : \tau(m(i)) = \min\{\tau(m(j)) | j \in dom(m) \wedge \pi(a(j)) = \pi(a(i))\}$ *(*i *is the oldest token on that place). We denote the set of all activators of a transition* t *in a marking* m *by* $A(m, t)$.

The *enabling time* of a transition with an activator a is the earliest possible time this transition enabled by an activator can fire. The *firing time* of a marking is the earliest possible time one of the transitions, enabled by this marking, can fire. We are assuming an *eager* system.

The time information of a DTPN is contained in its marking. This makes it possible to consider the system only at the moments when a transition fires. For this reason, we do not represent time progression as a state transition.

Definition 5 (Enabling time, firing time). *Let a be an activator of a transition $t \in T$ in a marking m of a DTPN. The* enabling time *of transition t by activator a is defined as $et(a,t) = \max\{\tau(a(i)) + \delta(\pi(a(i)), t) \mid i \in dom(a)\}$. The* firing time *of a marking m is defined as $ft(m) = \min\{et(a,t) \mid t \in T, a \in A(m,t)\}$.*

The firing time for a marking is thus completely determined by the marking itself.

The firing of a transition and its effect on a marking are described by the *transition relation*. Produced tokens are "fresh", i.e. they have new identities.

Definition 6 (Transition relation). *The* transition relation *of a DTPN satisfies Def. 3 and moreover, for any $m \xrightarrow{t} m'$:*

- *there is an $a \in A(m,t)$ such that $a = m \setminus m' \wedge ft(m) = et(a,t) < \infty$, and*
- *$\forall i \in dom(m' \setminus m) : \tau(m'(i)) = ft(m) + x$ with $x \in \delta(t, \pi(m(i)))$.*

Since the delays on arcs are nonnegative, a system cannot go back in time, i.e. $\forall m, m' \in \mathbb{M}, t \in T : m \xrightarrow{t} m' \Rightarrow \mathrm{ft}(m) \leq \mathrm{ft}(m')$.

In order to reduce infinite delay intervals to finite ones, we introduce the *refinement* of a DTPN, which is in fact a refinement of output delay intervals. Input delays are singletons and each singleton is a refinement of itself.

Definition 7 (Refinement of a DTPN). *Let $N_1 = (P, T, F, \delta)$ and $N_2 = (P, T, F, \delta')$ be DTPN. Then N_2 is a refinement of N_1 denoted by $N_1 \lhd N_2$ iff $\forall (x,y) \in F : \delta(x,y) \lhd \delta'(x,y)$.*

The refinement of a DTPN might introduce new firing sequences in the system. Due to this property, we are able to show that a refined DTPN can simulate its original DTPN with identical markings but not vice versa.

Theorem 8 (Simulation by refinement). *Consider two DTPN's N_1 and N_2 such that $N_1 \lhd N_2$. Then $\forall m \in \mathbb{M} : (N_2, m) \preceq (N_1, m)$ w.r.t. identity relation. So $\forall m \in \mathbb{M} : FS(N_1, m) \subseteq FS(N_2, m)$.*

A similar result holds for the untimed version $\overline{N}$ of a DTPN N: $\forall m \in \mathbb{M} : (\overline{N}, \overline{m}) \backsim (N, m) \wedge FS(N, m) \subseteq FS(\overline{N}, \overline{m})$ where $\overline{m}$ is the untimed version of m.

4.2 Relationship between sDTPN, fDTPN and DTPN

For a DTPN N_1 and a fDTPN N_2 such that $N_2 \lhd N_1$, we have $N_1 \preceq N_2$. The question we will address here is: *Is there an fDTPN such that $N_2 \preceq N_1$?* The answer is positive. In order to show this, we introduce a function φ assigning to a DTPN a fDTPN, called its *proxy*, which has a finite grid on each delay interval. This is only interesting for output delays, but we define it for all delay intervals. The *grid distance* is the smallest value such that all the bounds of delay sets and timestamps in the initial marking are multiples of it.

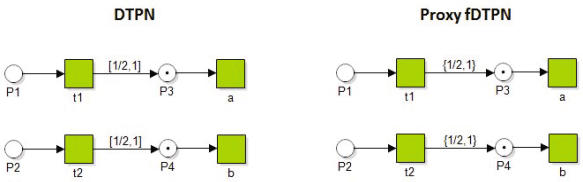

Fig. 2. Counter Example: DTPN simulates fDTPN

Definition 9 (Proxy fDTPN). *A DTPN $N = (P, T, F, \delta)$ with an initial marking m_0 has a* proxy *fDTPN $\varphi(N)$ with the same initial marking such that $\varphi(N) = (P, T, F, \delta')$ where:*

- $A = \{\tau(m(i)) \mid i \in dom(m_0)\} \cup \{\min(\delta(x,y), \max(\delta(x,y) \mid (x,y) \in F\}$,
- $\forall (x,y) \in F : \delta'(x,y) = \{i/d \mid i \in \mathbb{Z} \wedge min(\delta(x,y))/d \leq i/d \leq max(\delta(x,y)/d)\}$, *where* $d = min\{k \in \mathbb{N} | \{k.a | a \in A\} \subseteq \mathbb{N}\}$ *and* $1/d$ *is the* grid distance.

The grid distance is the least common multiple of the denominators of the non-zero elements of A expressed as non-reducible elements of $\mathbb{Q}$. We introduce the *round-off* of a marking, which is obtained if the timestamps are round-off to a grid.

Definition 10 (Round-off Relation). *For the set of all markings $\mathbb{M}$ of a DTPN: $\forall m, \bar{m} \in \mathbb{M} : m \sim \bar{m}$ iff $dom(m) = dom(\bar{m})$ and $\forall i \in dom(m) : \pi(m(i)) = \pi(\bar{m}(i))$ and $\forall i \in dom(m) : \exists k \in \mathbb{Z}, d \in \mathbb{N} :$*

$$(\tau(m(i)) \in [k/d, k/d + 1/2d] \wedge \tau(\bar{m}(i)) = k/d) \vee$$
$$(\tau(m(i)) \in (k/d + 1/2d, (k+1)/d] \wedge \tau(\bar{m}(i)) = (k+1)/d)$$

The round-off relation preserves the order of timestamps.

Corollary 11. *Let N be an DTPN with the set of all markings $\mathbb{M}$. Let $\bar{N} = \varphi(N)$ be its proxy fDTPN with the set of all markings $\bar{\mathbb{M}}$. If two markings $m \in \mathbb{M}, \bar{m} \in \bar{\mathbb{M}} : m \sim \bar{m}$ then $\forall i, j \in dom(m)$:*

$$\tau(m(i)) = \tau(m(j)) \Rightarrow \tau(\bar{m}(i)) = \tau(\bar{m}(j))$$
$$\tau(m(i)) < \tau(m(j)) \Rightarrow \tau(\bar{m}(i)) \leq \tau(\bar{m}(j))$$
$$\tau(m(i)) > \tau(m(j)) \Rightarrow \tau(\bar{m}(i)) \geq \tau(\bar{m}(j))$$

As a consequence of the preservation of the order of timestamps order we have:

Theorem 12 (Simulation by proxy). *Let N be a DTPN and $\bar{N} = \varphi(N)$ be its proxy fDTPN with the set of all common timed markings $\mathbb{M}$. Then $\forall m \in \mathbb{M} :$ $(\bar{N}, m) \preceq (N, m)$ w.r.t. the round-off relation.*

The opposite is not true, i.e. a DTPN is not simulating its proxy with respect to the round-off relation as illustrated by the example in Fig. 2. The grid distance

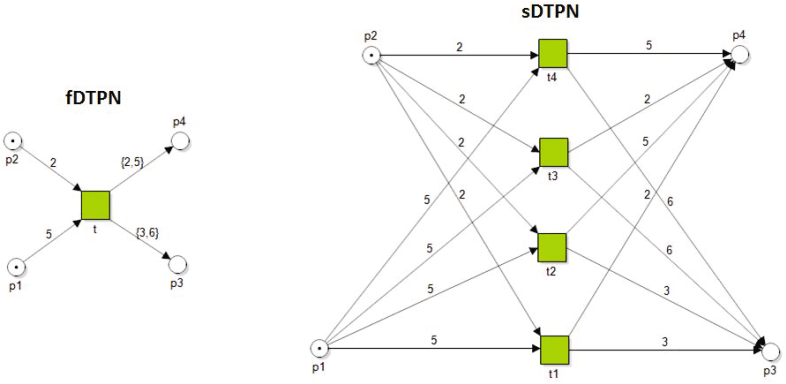

Fig. 3. Construction of a DTPN from an fDTPN

$d = 1/2$. From the initial marking with tokens in places $p1$ and $p2$ with zero timestamps, a marking m with tokens in places $p3$ and $p4$ with timestamps $11/16$ and $10/16$, respectively is reachable. In the proxy, a marking $\bar{m}$ is reachable such that $m \sim \bar{m}$, i.e. with tokens in places $p3$ and $p4$ with timestamps equal to one. From marking m only transition b is enabled but from marking $\bar{m}$ both transitions a and b are enabled. However, by Theorem 8 we know that a DTPN simulates its proxy w.r.t. the identity relation.

Corollary 13 (Simulation Equivalence). *A DTPN N and its proxy fDTPN $\bar{N} = \varphi(N)$ are* simulation equivalent *for each initial marking m_0 and hence $FS(N, m_0) = FS(\bar{N}, m_0)$.*

The final step in the reduction process is to show that any fDTPN is strongly bisimilar to a sDTPN. This can be done by making a copy of a transition, i.e. a new transition with the same preset and postset as the original one and with singleton delays for each possible combination of output delays from its output intervals. Formally this can be defined using the generalized cartesian product. We give an example in the Fig. 3.

Definition 14 (Reduction of fDTPN to sDTPN). *Let $N = (P, T, F, \delta)$ be a fDTPN. For each $t \in T$, let $A_t = \prod_{p \in t^\bullet} \delta(t, p)$ be the generalized cartesian product of all its delay sets. Then, the corresponding sDTPN is the tuple construct$(N) = (P, T', F', \delta')$, where*

$$T' = \{t_x \mid t \in T \wedge x \in A_t\},$$
$$F' = \{(p, t_x) \mid (p, t) \in F \wedge x \in A_t\} \cup \{(t_x, p) \mid (t, p) \in F \wedge x \in A_t\},$$
$$\forall p \in P : \forall t \in T : \forall x \in A_t : \delta'(p, t_x) = \delta(p, t) \wedge \delta'(t_x, p) = x(p).$$

Theorem 15 (Bisimulation of fDTPN and sDTPN). *Let N be an arbitrary fDTPN. Then $N \simeq$ construct(N).*

As a consequence, for each DTPN there exists an sDTPN that is simulation equivalent.

4.3 Analysis of sDTPN

To analyse the behavior of DTPNs it is sufficient to consider sDTPNs. However, since time is non-decreasing, the reachability graph of an sDTPN is usually infinite. Still, there is a finite *time window* that contains all relevant behavior, because new tokens obtain a timestamp bounded by a maximum in the future, i.e. the maximum of all maxima of output delays and the timestamps of tokens earlier than the current time minus an upperbound of the input delays are irrelevant, i.e. they can be updated to the current time minus this upper bound. So we can reduce the time frame of a DTPN to a finite time window. This is done by defining a reduction function that maps the timestamps into this window. We introduce a labeled transition system for an sDTPN that is strongly bisimilar to it. Therefore we call this labeled transition system rDTPN, although, strictly speaking, it is not a DTPN.

We denote by $\delta_i^{\uparrow}$ the maximal *incoming* arc delay, i.e. $\delta_i^{\uparrow} = \max\{\delta(p,t) | (p,t) \in (P \times T) \cap F\}$ and $\delta_o^{\uparrow}$ the maximal *outgoing* arc delay, i.e. $\delta_o^{\uparrow} = \max\{\delta(t,p) | (t,p) \in (T \times P) \cap F\}$. A *reduced* marking is obtained by subtracting the firing time of the marking from the timestamp of each token in the marking with a lower bound of $-\delta_i^{\uparrow}$.

Definition 16 (Reduction function). *Consider an sDTPN with the set of all markings* $\mathbb{M}$. *The reduction function* $\alpha : \mathbb{M} \to \mathbb{M}$ *satisfies* $\forall m \in \mathbb{M} : dom(\alpha(m)) = dom(m)$ *and* $\forall i \in dom(m) : \pi(\alpha(m)(i)) = \pi(m(i)) \wedge \tau(\alpha(m)(i)) = \max\{-\delta_i^{\uparrow}, \tau(m(i)) - ft(m)\}$. *The set of all* reduced markings *of a sDTPN is defined as* $\bar{\mathbb{M}} = \{m \in \mathbb{M} \mid \alpha(m) = m\}$.

The reduction function is either (1) reducing the timestamp of tokens by the firing time, or (*b*) mapping timestamps less than or equal to $-\delta_i^{\uparrow}$ to $-\delta_i^{\uparrow}$.

Corollary 17. *Let* N *be a sDTPN with the set of all markings* $\mathbb{M}$. *Then* $\forall m \in \mathbb{M}, t \in T : \forall i \in dom(m) : \tau(\alpha(m)(i)) \geq \tau(m(i)) - ft(m) \wedge A(\alpha(m), t) = \{\alpha(e) \mid e \in A(m,t)\}$.

Lemma 18. *Consider a sDTPN with the set of all markings* $\mathbb{M}$. *Then* $\forall m \in \mathbb{M} : ft(\alpha(m)) = 0$, *i.e the firing time of a reduced marking is zero.*

As a consequence of Lemma 18, the reduction function α is idempotent.

Corollary 19. *Let* N *be a sDTPN. Then* $\forall m \in \mathbb{M} : \alpha(\alpha(m)) = \alpha(m)$.

We will now show that, given a marking with an enabled transition, the same transition is also enabled in its reduced marking and the new marking created by firing this transition from both enabling markings have the same reduced marking. Furthermore, the firing time of a marking reachable from a reduced marking can be used to compute the arrival time in the concrete system.

Lemma 20. *Consider a sDTPN with the set of markings* $\mathbb{M}$. *Let* $m, m' \in \mathbb{M} : m \xrightarrow{t} m'$. *Then* $\exists \tilde{m} \in \mathbb{M} : \alpha(m) \xrightarrow{t} \tilde{m}$ *and* $ft(\tilde{m}) = ft(m') - ft(m)$ *and* $\alpha(m') = \alpha(\tilde{m})$.

For an executable firing sequence, the above theorem can be extended to the following corollary.

Corollary 21. *Let* $(N, \mathbb{M}, \rightarrow, m_0)$ *be a labeled transition system. Let* $m_0, m_1 \dots m_n \in \mathbb{M}$ *and* $m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} m_n$. *Then* $\exists \tilde{m}_0, \tilde{m}_1, \dots, \tilde{m}_n \in \mathbb{M} : \tilde{m}_0 = m_0 \wedge \forall j \in \{1, \dots, n\} : \alpha(m_{j-1}) \xrightarrow{t_j} \tilde{m}_j$ *and* $ft(m_n) = \sum_{j=0}^{n} ft(\tilde{m}_j)$

The *reduced transition relation* defines the relationship between two reduced markings, one reachable from the other.

Definition 22 (Reduced transition relation). *Let* N *be an sDTPN with the set of all markings* $\mathbb{M}$ *and the set of all reduced markings* $\bar{\mathbb{M}}$. *The* reduced transition relation $\rightsquigarrow \subseteq \bar{\mathbb{M}} \times T \times \bar{\mathbb{M}}$ *satisfies* $\forall \bar{m}, \bar{m}' \in \bar{\mathbb{M}} : \bar{m} \overset{t}{\rightsquigarrow} \bar{m}' \Leftrightarrow \exists \tilde{m} \in \mathbb{M} : \bar{m} \xrightarrow{t} \tilde{m} \wedge \alpha(\tilde{m}) = \bar{m}'$.

Definition 23 (rDTPN). *An* rDTPN *is a* labeled transition system $(\bar{\mathbb{M}}, T, \rightsquigarrow, \bar{m}_0)$, *where* $\bar{m}_0 = \alpha(m_0)$ *is an initial marking.*

For a given sDTPN, its reduced labelled transition system and timed labelled transition system are strongly bisimilar w.r.t. reduction relation. Due to Lemma 20, the time relation in the bisimulation is implicit.

Theorem 24 (Bisimulation sDTPN and rDTPN). *Consider a sDTPN with a timed labeled transition system* $N = (\mathbb{M}, T, \rightarrow, m_0)$ *and its rDTPN* $\bar{N} = (\bar{\mathbb{M}}, T, \rightsquigarrow, \bar{m}_0)$. *Then* $N \backsimeq \bar{N}$.

The number of different timestamps in the reachability graph of the rDTPN is finite. This is observed in several papers (see [8]). To see this, we consider first only timestamps in $\mathbb{Z}$. Since we have finitely many of them in the initial marking and the only operations we execute on them are: (1) selection of the maximum, (2) adding one of a finite set of delays and (3) subtracting the selected timestamp with a minimum. So the upper bound is the maximal output delay and the lower bound is zero minus the maximal input delay. Hence, we have a finite interval of $\mathbb{Z}$ which means finitely many values for all markings. In case we have delays in $\mathbb{Q}$ we multiply with the lcm of all relevant denominators, like in definition 10 and then we are in the former case.

Theorem 25. *The set of timestamps in the reachability graph of a rDTPN is finite, and so if the underlying Petri net is bounded, the reachability graph of the rDTPN is finite.*

Furthermore, using Corollary 21, given a path in the reachability graph of a rDTPN, we are able to compute the time required to execute this path in the original DTPN.

Finally, we sketch by two constructions how the two basic variants of DTPN, namely DTPN-1 with (In-single, Out-zero) and DTPN-2 with (In-zero, Out-single) can express inhibitor arcs. The models are branching bisimilar with the classical Petri net with inhibitors. In both constructions, we add a special place

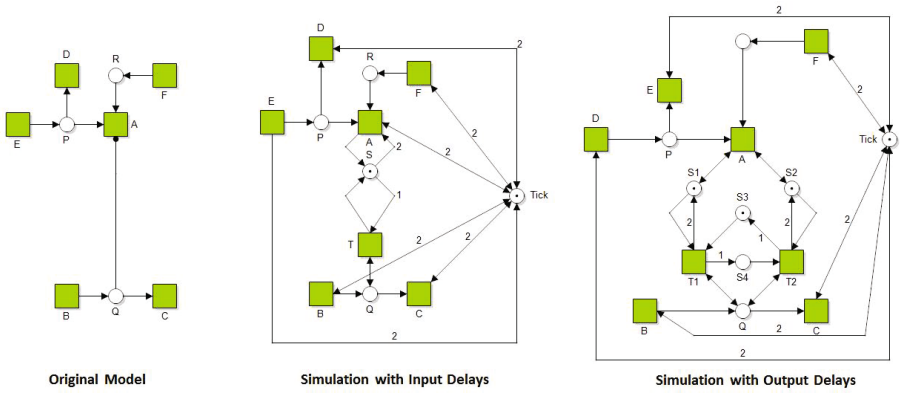

Fig. 4. Simulating Inhibitor Arcs with Input Delays and Output Delays

called *Tick* that is connected to each original transition by an input and an output arc and we replace each inhibitor by a subnet containing a silent transition T. In DTPN-1 and DTPN-2 original transitions fire at even time units while the silent transitions may also fire at odd time units.

5 Discrete Stochastic Petri Nets

In this section we endow the transition firings of a DTPN with probabilities. We do this only for DTPN with (*In-single, Out-fint*) and we assign a probability distribution to these intervals. Additionally, we should have a stochastic mechanism to choose a transition from all enabled transitions. We do this by assigning a non-negative weight to all transitions and draw an enabled transition x with probability $w(x)/\sum_{y:enabled} w(y)$. If there are no priorities for transitions, we may choose all weights to be equal. Note that we could introduce stochastics for the two other main classes, $M1$ and $M2$ in a similar way. For the class $M1$, we first select a classically enabled transition with the weights and then a duration from a finite distribution associated with the transition. For the class $M2$, we select for all classical enabled transitions a delay from a finite distribution associated with the transition and then we select with the weights one of them having the minimal delay.

Definition 26 (DSPN)
A DSPN is 6-tuple $(P, T, F, \delta, w, \phi)$ *, where*

- (P, T, F, δ) *is a DTPN with (In-single, Out-fint).*
- $w : T \to \mathbb{R}^+$ *a* weight function*, used to choose one of the simultaneously enabled transitions,*
- ϕ *is a function with domain* $F \cap T \times P$ *and for* $\forall(t,p) \in F$: $\phi(t,p) : \delta(t,p) \to [0,1]$ *such that* $\sum_{x \in \delta(t,p)} \phi(t,p) = 1$*, so* $\phi(t,p)$ *assigns probabilities to* $\delta(t,p)$*.*

We consider two transformations to derive a Markov chain for a DSPN. The first transformation is given in Def. 14, where for each value of a finite output delay interval, a transition is introduced with a one point output delay, as in Fig. 3. Here transition t has two output arcs with delay sets, one with $\{2,5\}$ and the other $\{3,6\}$. Let the probabilities of these intervals be (p_1,p_2), with $p_1+p_2=1$ and (q_1,q_2) with $q_1+q_2=1$. So $w(t_1)=w(t).p_1.q_1, w(t_2)=w(t).p_1.q_2, w(t_3)=w(t).p_2.q_1.$ and $w(t_4)=w(t).p_2.q_2$.

This transformation blows up the model and gives unreadable pictures, but it is only for automatic processing. Now we have a model of type sDTPN and we can forget the probabilities $\phi(.,.)$ because all output delays are singletons. So we only have to deal with the weight function w. By Theorem 15, we know that this model is strongly bisimilar (discarding the probabilities) with the original one so we can deal with this one. It is obvious by the construction that the probabilities over the delays of produced tokens are the same as well. So after these transformations we can consider a DSPN as a 5-tuple (P,T,F,δ,w).

The next transformation concerns this sDTPN model into the reduced labeled transition system (rDTPN) as in Theorem 24 which is strongly bisimilar with the sDTPN model. The weights can be transferred to this rDTPN model because the underlying Petri net has not changed. We call this new model class *rDSPN*. Remember that if the underlying Petri net is bounded, then rDSPN has a finite reachability graph.

We will now add two values to an arc in the reachability graph of the rDSPN, representing a transition $m,m' \in \mathbb{M} : m \xrightarrow{t} m'$: (1) *probability* of choosing this arc and (2) the *sojourn time* in a marking m' if coming from m. Remember that for each marking the firing time is uniquely determined, but the sojourn time depends on the former marking. The sojourn time can be computed during the reduction process as expressed by Lemma 20.

Definition 27 (Transition probability and sojourn time)
The transition probability $Q : \mathbb{M} \times \mathbb{M} \to [0,1]$ *satisfies:*

$$Q_{m,m'} = \sum_{x:m \xrightarrow{x} m'} w(x) \quad / \quad \sum_{y:\exists m'':m \xrightarrow{y} m''} w(y).$$

For $m,m' \in \mathbb{M} : r(m,m') = ft(m') - ft(m)$ *is the* sojourn time *in marking* m' *if coming from* m.

The transition probability contains all information of the reachability graph. Finally we are able to define the *Markov chain* that is determined by the reachability graph of the rDSPN endowed with the transition probabilities.

Definition 28 (Markov chain)
Let a rDSPN (P,T,F,δ,w) *be given and let the* Q *be the transition probability over the state space. Then the* Markov chain *of the rDSPN is a sequence of random variables* $\{X_n | n = 0,1,...\}$, *where* $X_0 = m_0$ *the initial marking and* X_n *is marking after* n *steps, such that:*

$$\mathbb{P}[X_{n+1} = m' | X_n = m, X_{n-1} = m_{n-1}, ..., X_0 = m_0] = Q(m, m')$$

for arbitrary $m_0, ..., m_{n-1} \in \mathbb{M}$.

The *Markov property* is implied by the fact that only the last marking before the transition firing is taken into account.

Since a marking and an enabled transition determine uniquely the next state, we can also consider another stochastic process $\{Y_n | n \in \mathbb{N}\}$, where $Y_n \in T$, which is a *stochastic firing sequence*. For a firing sequence $\sigma = (t_1, ..., t_n)$ with $m_0 \xrightarrow{t_1} m_1, ..., \xrightarrow{t_n} m_n$ we have

$$\mathbb{P}[Y_1 = t_1, ..., Y_n = t_n | X_0 = m_0] = \mathbb{P}[X_1 = m_1, ..., X_n = m_n | X_0 = m_0].$$

So we can compute the probability for each finite firing sequence.

Markov chains are often endowed with a *cost structure* which is a function assigning to a pair of successive markings a real value, called *cost function*. Then we can express the *total expected cost* when starting in marking m as:

$$\mathbb{E}[\sum_{n=0}^{N} c(X_n, X_{n+1}) | X_0 = m].$$

Here, $N \in \mathbb{N}$ or $N = \infty$. In particular we will use the sojourn times as "cost". In fact we may associate a *semi-Markov process* to rDSPN, because the sojourn times themselves are random variables, but in our case they are completely determined if we know the former state. The Markov chain $\{X_n | n = 0, 1, ...\}$ is then the embedded Markov chain of the semi-Markov process [33]. We will use the function $v : \mathbb{M} \to \mathbb{R}$, which is usually called the *value function* for Markov processes (see [29]): $v(m) = \mathbb{E}[\sum_{n=0}^{N} c(X_n, X_{n+1}) | X_0 = m]$ for cost functions c.

We will use the Markov chain to answer three types of important questions. We will use the cost function to express the questions and we use the Markov property to translate our questions into *Bellman equations* (see [33] and [29]).

- Probability of reaching a subset,
- Expected time to leave a subset,
- Expected sojourn times in equilibrium.

Note that all these questions concern sequences of markings or equivalently firing sequences. So they belong to LTL (see [28]).

Probability of Reaching a Subset

Let $A, B \subseteq \mathbb{M}$ be a subsets of the state space, $A \cap B = \emptyset$. We are interested in the probability of reaching A from B . Here we choose $c(m, m') = 1$ if $m \in B \wedge m' \in A$ and $c(m, m') = 0$ otherwise. Further we stop as soon as we reach A. Then

$$\forall m \in B : v(m) = \sum_{m' \in A} Q_{m,m'} + \sum_{m' \in B} Q_{m,m'}.v(m').$$

If B is finite, this can be computed, even if the underlying Petri net is unbounded. For example if B is the set of k-bounded markings (i.e. markings with at most k tokens per place), this computation is possible.

Expected Time to Leave a Subset. A of the state space.
Here we use a cost function $c(m, m') = r(m, m')$, the sojourn time in m', if $m, m' \in A$ and $c(m, m') = 0$ elsewhere. If $m \in A$ then

$$v(m) = \sum_{m' \in A} Q_{m,m'}.(r(m, m') + v(m')).$$

This can be computed even if the underlying Petri net is unbounded but A is finite.

Expected Sojourn Times in Equilibrium
We restrict us to the case where the underlying Petri net is bounded. $\mathbb{P}[X_n = m'|X_0 = m] = Q^n(m, m')$ where Q^n is the Q to power n. The *limit of averages* exists: $\pi_m(m') := \lim_{N\to\infty} \sum_{n=0}^{N} Q^n_{m,m'}$ and it satisfies

$$\pi_{m_0}(m') = \sum_{m \in \mathbb{M}} \pi_{m_0}(m).Q_{m,m'}.$$

We now assume the system is a strongly connected component (i.e. the Markov chain is irreducible), which implies that the limit distribution π is independent of the initial state m_0. Further, we know that the expected time spent in a marking m depends on the former marking, so the expected time of being in marking m' is:

$$\sum_{m \in \mathbb{M}} r(m, m').\mathbb{P}[X_{n-1} = m|X_n = m'],$$

which can be rewritten using Bayes rule to:

$$\sum_{m \in \mathbb{M}} r(m, m').\mathbb{P}[X_n = m'|X_{n-1} = m].\mathbb{P}[X_{n-1} = m]/\mathbb{P}[X_n = m'].$$

Thus, the expected sojourn time in some *arbitrary* marking is obtained by multiplying with $\mathbb{P}[X_n = m']$

$$\sum_{m \in \mathbb{M}} r(m, m').Q_{m,m'}.\mathbb{P}[X_{n-1} = m].$$

This formula could also be derived immediately as the expected sojourn time in the next marking. For $n \to \infty$, this converges either by the normal limit or limit of averages to:

$$\sum_{m \in \mathbb{M}} r(m, m').Q_{m,m'}.\pi(m).$$

If we want to solve these equations using matrix calculations, we need to compute the transition matrix of the reachability graph. However, we can also use the

method of *successive approximations* to approximate these values in an iterative way using only two functions (vectors) over the state space. As an example, the probability of reaching a set A from a set B we set: $\forall m \in B : v_0(m) = 0$ and

$$\forall m \in B : v_{n+1}(m) = \sum_{m' \in A} Q_{m,m'} + \sum_{m' \in B} Q_{m,m'}.v_n(m').$$

According to [1] we can derive for specially structured *workflow nets* the distribution of the *throughput time* of a case (i.e. the time a token needs to go from the initial to the final place) analytically in case of DTPN with (*In-zero, Out-fint*). Models of this class can be built by *transition refinement*, using the patterns displayed in Fig. 5. Pattern 1 is a *sequence* construction. Pattern 2 is an *iteration* where we have arc weights q and $1-q$ for the probability of continuing or ending the loop. Pattern 3 is the *parallel* construction. Pattern 4 is the *choice*, which has also arc weights q and $1-q$ representing the probabilities for the choices. In Fig. 5 the intervals $[a,b],[c,d]$ indicate the finite probability distributions. In order to be a model of this class, it must be possible to construct it as follows. We start with an initial net and we may replace all transitions t with $|{}^\bullet t| = |t^\bullet| = 1$ using one of the four rules. There should be a proper parse tree for a net of this class. We associate to all transitions with output delay sets a random variable; for the initial net the random variable U with distribution on $[a,b]$ and similarly random variable Y and Z for the patterns.

If we have such a net, we can apply the rules in the reversed order. If we have at some stage a subnet satisfying to one of the four patterns, with the finite distributions as indicated, we can replace it by an initial subnet with a "suitable" distribution on the output delay interval. For the initial subnet we have a random output variable U. For the sequential construction (rule 1) we have two independent random variables Y and Z with discrete distributions on $[a,b]$ and $[c,d]$ respectively. So $U = Y + Z$ and the distribution of U is the *convolution* of the distributions of Y and Z. For the parallel construction (rule 3) we have $U = \max(Y,Z)$ which is the product distribution, i.e. $\mathbb{P}[U \leq x] = \mathbb{P}[Y \leq x].\mathbb{P}[Z \leq x]$. For the choice (rule 4) it is a *mixture* of two distributions, $\mathbb{P}[U \leq x] = \mathbb{P}[Y \leq x].q + \mathbb{P}[Z \leq x].(1-q)$. The most difficult one is the iteration (rule 2), since here we have the distribution of $U := \sum_{n=0}^{N} (Y_n + Z_n)$ where N is a geometrically distributed random variable with distribution $\mathbb{P}[N = n] = q^{n-1}.(1-q)]$ indicating the number of iterations and Y_n and Z_n are random variables from the distributions on $[a,b]$ and $[c,d]$ respectively. All these random variables are independent. The distribution of U can be derived using the Fourier transform (see [1]). This is an approximation, since the domain of U is infinite, even if Y_n and Z_n have finite domains. However, we can cut the infinite domain with a controllable error.

Thus, we are able to reduce a complex DSPN. This method is only applicable if the original net is safe, otherwise different cases can influence each other and so the independency assumptions are violated.

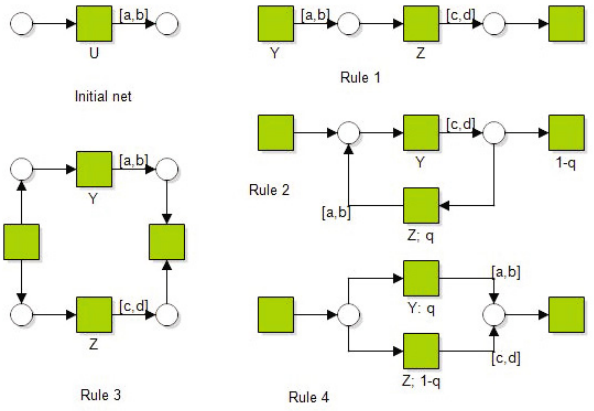

Fig. 5. Refinement rules

6 Conclusions

In this paper we considered Petri nets with time and stochastics, and in particular, one class of them that did not get much attention in literature: the class with the timestamps for tokens. We call this class DTPN and show how we can analyse the behaviour of the nets from this class in case the underlying Petri net is bounded. We did not consider complexity but only computability, since history has shown that methods can become feasible due to increase in computing power and smart heuristics. We considered several subclasses of DTPN and we showed that they all are Turing complete, because they can express inhibitor arcs, but that some have better *modeling comfort*, i.e. they are easier for modeling. The DTPN class can easily be extended to deal with stochastics, as we have shown. Here, we have the advantage above the GSPN model that we can use arbitrary finite distributions, while only exponential distributions can be used in GSPN. The analysis of stochastic behavior is based on Markov chains and so it is similar to the approach in GSPN. We also showed how analytical methods for stochastic analysis can be incorporated in model checking.

References

1. van der Aalst, W.M.P., van Hee, K.M., Reijers, H.A.: Analysis of discrete-time stochastic Petri nets. Statistica Neerlandica 54(2), 237–255 (2000)
2. van der Aalst, W.M.P.: Interval Timed Coloured Petri Nets and their Analysis. PhD thesis, Eindhoven University of Technology (1993)
3. Abdulla, P.A., Nylén, A.: Timed Petri nets and BQOs. In: Colom, J.-M., Koutny, M. (eds.) ICATPN 2001. LNCS, vol. 2075, pp. 53–70. Springer, Heidelberg (2001)
4. Marsan, M.A., Conte, G., Balbo, G.: A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. ACM Trans. Comput. Syst. 2(2), 93–122 (1984)

5. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. 126(2), 183–235 (1994)
6. Bengtsson, J., Larsen, K.G., Larsson, F., Pettersson, P., Yi, W.: Uppaal - a tool suite for automatic verification of real-time systems. In: Alur, R., Sontag, E.D., Henzinger, T.A. (eds.) HS 1995. LNCS, vol. 1066, pp. 232–243. Springer, Heidelberg (1996)
7. Bera, D., van Hee, K.M., Sidorova, N.: Discrete timed Petri nets. Computer Science Report 13-03, Technische Universiteit Eindhoven, P.O. Box 513, 5600 MB Eindhoven, The Netherlands (April 2013)
8. Berthelot, G., Boucheneb, H.: Occurrence graphs for interval timed coloured nets. In: Valette, R. (ed.) ICATPN 1994. LNCS, vol. 815, pp. 79–98. Springer, Heidelberg (1994)
9. Berthomieu, B., Diaz, M.: Modeling and verification of time dependent systems using time Petri nets. IEEE Trans. Softw. Eng. 17(3), 259–273 (1991)
10. Berthomieu, B., Menasche, M.: An enumerative approach for analyzing time Petri nets. In: Proceedings IFIP, pp. 41–46. Elsevier Science Publishers (1983)
11. Boucheneb, H.: Interval timed coloured Petri net: efficient construction of its state class space preserving linear properties. Form. Asp. Comput. 20(2), 225–238 (2008)
12. Bouyer, P., Haddad, S., Reynier, P.-A.: Undecidability results for timed automata with silent transitions. Fundam. Inf. 92(1-2), 1–25 (2009)
13. Boyer, M., Roux, O.H.: On the compared expressiveness of arc, place and transition time Petri nets. Fundam. Inf. 88(3), 225–249 (2008)
14. Cassez, F., Roux, O.-H.: Structural translation from time Petri nets to timed automata. Electron. Notes Theor. Comput. Sci. 128(6), 145–160 (2005)
15. Cerone, A., Maggiolio-Schettini, A.: Time-based expressivity of timed Petri nets for system specification. Theor. Comput. Sci. 216(1-2), 1–53 (1999)
16. Ghezzi, C., Mandrioli, D., Morasca, S., Pezze, M.: A unified high-level Petri net formalism for time-critical systems. IEEE Trans. Softw. Eng. 17(2), 160–172 (1991)
17. van Glabbeek, R.: The linear time-branching time spectrum. In: Baeten, J.C.M., Klop, J.W. (eds.) CONCUR 1990. LNCS, vol. 458, pp. 278–297. Springer, Heidelberg (1990)
18. Haddad, S., Moreaux, P.: Sub-stochastic matrix analysis for bounds computation - theoretical results. European Journal of Operational Research 176(2), 999–1015 (2007)
19. van Hee, K.M., Somers, L.J., Voorhoeve, M.: Executable specifications for distributed information systems. In: Proceedings of the IFIP TC 8/WG 8.1, pp. 139–156. Elsevier (1989)
20. Jantzen, M.: Language theory of Petri nets. In: Brauer, W., Reisig, W., Rozenberg, G. (eds.) APN 1986. LNCS, vol. 254, pp. 397–412. Springer, Heidelberg (1987)
21. Jensen, K.: An introduction to the theoretical aspects of coloured Petri nets. In: de Bakker, J.W., de Roever, W.-P., Rozenberg, G. (eds.) REX 1993. LNCS, vol. 803, pp. 230–272. Springer, Heidelberg (1994)
22. Knapik, M., Penczek, W., Szreter, M., Polrola, A.: Bounded parametric verification for distributed time Petri nets with discrete-time semantics. Fundam. Inf. 101(1-2), 9–27 (2010)
23. Christensen, S., Kristensen, L.M., Mailund, T.: Condensed state spaces for timed petri nets. In: Colom, J.-M., Koutny, M. (eds.) ICATPN 2001. LNCS, vol. 2075, pp. 101–120. Springer, Heidelberg (2001)
24. Lime, D., Roux, O.H.: Model checking of time Petri nets using the state class timed automaton. Discrete Event Dynamic Systems 16(2), 179–205 (2006)

25. Lime, D., Roux, O.H.: Model checking of time Petri nets using the state class timed automaton. Discrete Event Dynamic Systems 16(2), 179–205 (2006)
26. Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modelling with generalized stochastic Petri nets. SIGMETRICS Perform. Eval. Rev. 26(2) (August 1998)
27. Merlin, P.M., Farber, D.J.: Recoverability of communication protocols: Implications of a theoretical study. IEEE Trans. Comm. (September 1976)
28. Pnueli, A.: The temporal logic of programs. In: Proceedings of the 18th Annual Symposium on Foundations of Computer Science, SFCS 1977, pp. 46–57. IEEE Computer Society, Washington, DC (1977)
29. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st edn. John Wiley and Sons, Inc., New York (1994)
30. Ramachandani, C.: Analysis of asynchronous concurrent systems by timed Petri nets. PhD thesis, Massachusetts Institute of Tenchnology, Cambridge, MA (1974)
31. Ramamoorthy, C.V., Ho, G.S.: Performance evaluation of asynchronous concurrent systems using Petri nets. IEEE Transactions on Software Engineering 6(5), 440–449 (1980)
32. Recalde, L., Haddad, S., Silva, M.: Continuous Petri nets: Expressive power and decidability issues. Int. J. Found. Comput. Sci. 21(2), 235–256 (2010)
33. Ross, S.M.: Introduction to Probability Models, 9th edn. Academic Press, Inc., Orlando (2006)
34. Valero Ruiz, V., de Frutos Escrig, D., Cuartero Gomez, F.: On non-decidability of reachability for timed-arc Petri nets. In: Proc. 8th. International Workshop on Petri Nets and Performance Models, pp. 188–196 (1999)
35. Sifakis, J.: Use of Petri nets for performance evaluation. In: Proceedings of the Third International Symposium on Measuring, Modelling and Evaluating Computer Systems, Bonn - Bad Godesberg, Germany, October 3-5, pp. 75–93. North-Holland (1977)
36. Starke, P.H.: Some properties of timed nets under the earliest firing rule. In: Rozenberg, G. (ed.) APN 1989. LNCS, vol. 424, pp. 418–432. Springer, Heidelberg (1990)
37. CPN website, `http://www.cpntools.org/`
38. ExSpecT website, `http://www.exspect.com/`
39. Great SPN website, `http://www.di.unito.it/~greatspn/index.html`
40. UPPAAL website, `http://www.uppaal.org/`