Discrete timed Petri nets

Citation for published version (APA):

Document status and date:
Published: 01/01/2013

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

- A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
- The final author version and the galley proof are versions of the publication after peer review.
- The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.

Download date: 20. Jun. 2019
CS-Report 13-03

Discrete Timed Petri nets

D. Bera, K.M. van Hee and N. Sidorova
Discrete Timed Petrinets

D. Bera, K.M. van Hee and N. Sidorova

ISSN 0926-4515
All rights reserved
editors: prof.dr. P.M.E. De Bra
            prof.dr.ir. J.J. van Wijk

Reports are available at:
http://library.tue.nl/catalog/TUEPublication.csp?Language=dut&Type=ComputerScienceReports&Sort=Author&level=1 and
http://library.tue.nl/catalog/TUEPublication.csp?Language=dut&Type=ComputerScienceReports&Sort=Year&Level=1

Computer Science Reports 13-03
Eindhoven, May 2013
Abstract
In this paper we first give an overview of the different approaches to time in Petri nets. If time is considered for performance analysis then we need also probability distributions for choices. So then we have Petri nets with time and stochastics. In literature most attention is paid to models where the time is connected to the transitions and for the stochastic case to continuous time models with exponential enabling distributions, none by its software tools as GSPN. Here we focus on discrete models where the time is discrete and the probability distributions are discrete, because we believe that this model class has some advantages. We study the expressive power of different discrete models and we show how several approaches can be unified by transformations from one into the other, which is also useful for the application of software tools. We also consider timed automata and compare them with the Petri net models. We show how model checking methods can be applied for the non-stochastic case. We show when the tools for timed automata, UPPAAL, can be used for this purpose. For the stochastic case we show how Markov techniques can be used. We also consider structural analysis techniques, that do not need the state space, for the class of workflow nets and we show how these techniques can be applied to reduce the computational effort.

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, since timing can influence behavioral properties of the system, and thus time dimension is highly relevant in the verification context. 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 [26] which introduces delaying of transitions and [29] 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. [12, 14, 19], and on model checking [9, 8, 33, 3, 21]. Next to the verification questions, performance analysis (c.f. [34, 34, 30, 15]) 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.

Extending tokens with timestamps is studied in a number of works, see [18, 2, 20, 7, 22, 10]. There is an overwhelming literature on timed automata and their relationship to Petri nets with time, e.g. [5, 23, 13, 11]. Most of these papers refer to timed automata with clocks as incorporated in the UPPAAL-tool [38] described in [6].

Note that all Petri net models with time are extensions of classical Petri nets, so if we discard the time aspects we obtain a classical Petri net. Petri nets with time can be considered without stochastics, but stochastic Petri nets

Email addresses: d.bera@tue.nl (Debjyoti Bera), k.m.v.hee@tue.nl (Kees van Hee), n.sidorova@tue.nl (Natalia Sidorova)
will have time. There is 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, 25, 17]. For the class of Generalized Stochastic Petri Nets, having also instantaneous transitions, there is a famous software tool for modeling and analysis GSPN [37]. This class is in fact the stochastic extension of Merlin time, so the approach with delaying transitions.

Approaches proposed 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. Also the application domain has impact on the right choice of the way to model time. Two extreme application domains are logistic systems (e.g. the business chain from cows to the dairy products) and embedded control systems (e.g. robots). Not only the time units are different but also the type of questions is different. In this paper we focus on the approach with token timestamps since we have the feeling that this class did not obtained enough attention although it is a powerful class both for modeling and for analysis. We call this class Discrete Timed Petri nets (DTPN). The popular software tools CPN-tools [35] and ExSpect [36] are using it. Next to that, we will consider a stochastic variant of DTPN called Discrete Stochastic Petri nets (DSPN) that can be seen as an alternative to the GSPN model. We feel that this class is very important for practical applications and it is not studied enough yet. This class has discrete time, i.e. finite or countable time domain and discrete probability distributions over the choices and it encompasses several well-known subclasses. We show how these subclasses are related and we show that the reachability graph of such a net is branching bisimilar to a finite transition system if the underlying untimed Petri net is bounded.

We start with preliminaries in Section 2. In Section 3, we give an overview of the five important time models. We do not consider coloring of tokens, although most of the results can be extended, with some technical effort, to colored Petri nets. We also do not consider continuous Petri nets (cf [31]) because the underlying untimed net is not a classical Petri net any more for this class of nets. In Section 4, we present the model class DTPN and a state space reduction method to analyze DTPN models in a finite way. In Section 5, we consider Timed Automata, in particular the standard UPPAAL model and we show the relationship with a DTPN. This is important because UPPAAL is a powerful toolset that can be applied to a large subclass of DTPN. In Section 6, we show by modeling inhibitor arcs, that several subclasses of DTPN are Turing complete. We also show how we can transform models from one subclass into the other. Further, we discuss their weaknesses and their strength from the modeling point of view, without claiming to be complete. In Section 7 we consider the stochastic version of DSPN 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 of reaching a set of states and the equilibrium distribution. For these methods we need the whole state space, like in model checking. However, for workflow nets there are also structural techniques that do not need the whole state space. We will show how these techniques 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 \( A \prec B \) if and only if \( A \subseteq \mathbb{Q}^+ \wedge A \subseteq B \wedge \sup(A) = \sup(B) \wedge \inf(A) = \inf(B) \). Given a grid distance \( 1/d \in \mathbb{N} \), a lower bound \( l \in \mathbb{Z} \) and an upper bound \( u \in \mathbb{Z} \), an equidistant interval is a finite rational set \( \text{eint}(d, l, u) = \{ i/d \mid i \in \mathbb{Z} : l/d \leq i/d \leq u/d \} \).

A function \( f \) is a set of pairs with unique first element such that \( \text{dom}(f) = \{ x \mid (x, y) \in f \} \) and \( \text{rng}(f) = \{ y \mid (x, y) \in f \} \). The generalized cartesian product of a set-valued function \( F \) is denoted by \( \Pi_F = \{ f \mid f \text{ is function} \wedge \text{dom}(f) = \text{dom}(F) \wedge \forall t \in \text{dom}(F) : f(t) \in F(t) \} \).

A finite sequence \( \sigma \) of length \( n \in \mathbb{N} : n > 0 \) over some set \( S \) is a function \( \sigma : \{ 1, \ldots, n \} \to S \). The length of a sequence is denoted by \( |\sigma| \). The set of all finite sequences over \( S \) is denoted by \( S^* \). The presence of an element \( x \in S \) is asserted by the predicate \( x \in \sigma \). We denote the empty sequence by \( \epsilon \). A bag (multiset) \( B \) over a set \( S \) is a function \( S \to \mathbb{N} \). If \( x \in S \) then \( B(x) \) gives the number of occurrences of \( x \) in bag \( B \). The empty bag is denoted by \( \emptyset B \). The set of all bags over set \( S \) is denoted by \( \mathbb{N}^S \). We use + and \( \cdot \) for the sum of two bags and \( =, \leq, \geq, >, < \) to element wise
compare bags, which are defined in the standard way. Note that a set is a bag in each of the elements of the set occurs exactly once.

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. For \(s, s' \in S\) and \(a \in A\), \(s \xrightarrow{a} s'\) if and only if \((s, a, s') \in \rightarrow\). 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 actions to sequences.

We say that a non-empty finite sequence \(\sigma \in A^*\) of length \(n \in \mathbb{N}\) is a firing sequence of net \(N\), 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 \(0 \leq i < n\). We write \(s \xrightarrow{\sigma} s'\) 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 \(s'_2 \in S_2\) and \(a \in A_2\) such that \(s_2 \xrightarrow{a} s'_2\) and \((s'_1, s'_2) \in R\). We write \(N_1 \preceq N_2\) 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 [16] and denote it by the symbol \(\approx_b\).

A Petri net is a tuple \(N = (P, T, F)\), where \(P\) is the set of places; \(T\) is the set of transitions such that \(P \cap T = \emptyset\); \(F\) is the flow relation \(F \subseteq (P \times T) \cup (T \times P)\). An inhibitor net is a tuple \((P, T, F, i)\), where \((P, T, F)\) is a Petri net and \(i : T \rightarrow \mathcal{P}(P)\) specifies the inhibitor arcs, with \(\mathcal{P}(P)\) denoting the powerset of \(P\). We refer to elements of \(P \cup T\) as nodes and elements from \(F\) as arcs. Nets can be depicted graphically. Places and transitions are represented as circles and squares, respectively, an arc \((n, m)\) is depicted as a directed arc from node \(n\) to node \(m\). A dot-headed arc is drawn from place \(p\) to transition \(t\) if \(p \in i(t)\). We denote the places of net \(N\) by \(P_N\), transitions as \(T_N\) and similarly for other elements of the tuple. If the context is clear, we omit \(N\) in the subscript. We define the preset of a node \(n\) as \(*n = \{m | (m, n) \in F\}\) and the postset as \(n^* = \{m | (n, m) \in F\}\). We extend the notion of postset and preset of a node to a set of nodes. The postset of a set of nodes \(A \subseteq P \cup T\) is denoted as \(A^* = \bigcup_{a \in A} a^*\). The preset of a set of nodes is defined in a similar way.

The state of a Petri net \(N = (P, T, F)\) is determined by its marking which represents the distribution of tokens over 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 \(*t \subseteq m\). If \(N\) is an inhibitor net then for the enabling of transition \(t\) from marking \(m\), we also require that \(m(p) = 0\) for all places \(p \in i(t)\). An enabled transition may fire which results in a new marking \(m' = m - *t + t^*\), denoted by \(m \xrightarrow{t} m'\). We lift the notion of transition firing and enabledness to sequences in a standard way. A sequence \(\sigma\) of length \(n \in \mathbb{N}\) is a firing sequence of net \(N\) if there exist markings \(m_{i-1}, m_i \in \mathbb{N}^P : m_{i-1} \xrightarrow{\sigma(i)} m_i\) for all \(1 \leq i \leq n\) and is denoted by \(m_0 \xrightarrow{\sigma} m_n\). We denote the set of all firing sequences from a marking \(m\) by \(FS(N, m) = \{\sigma \in T^* | \exists m' : m \xrightarrow{\sigma} m'\}\) and the set of all reachable markings by \(R(N, m) = \{m' \mid \sigma \in FS(N, m) : m \xrightarrow{\sigma} m'\}\). We define the net system of a Petri net \(N\) as a pair \(M = (N, m_0)\), where \(m_0 \in \mathbb{N}^P\) is the initial marking.

### 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 (Syntax of a timed Petri net).** A timed Petri net (TPN) is a tuple \((P, T, F, \delta)\), 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:

3

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. \(\forall t \in T, (p_1, t), (p_2, t) \in F, \delta(p_1, t) = \delta(p_2, t)\)

Pl-equal For every place, the delays on its input arcs are equal, i.e. \(\forall p \in P, (p, t_1), (p, t_2) \in F, \delta(p, t_1) = \delta(p, t_2)\).

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 time stamps to indicate at which moment of time these tokens become consumable or 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 time stamp. The time stamps have the meaning that a token cannot be consumed by a transition before this time.\(^1\)

\(^1\)By extending the time domain to \(\text{Time} \times \text{Time}\) with \(\text{Time} \subseteq \mathbb{R}\) we could talk about “usability” of tokens, with tokens having the earliest and the latest consumption times.
**Definition 2 (Marking of a TPN).** Let \( I \) denote a countably infinite set of identifiers. A marking of a TPN is a partial function \( m : I \rightarrow P \times \bar{Q} \) with a finite domain. For \( i \in \text{dom}(m) \) with \( m(i) = (p, q) \) we say that \((i, p, q)\) is a token on place \( p \) with time stamp \( q \). We denote the set of all markings of a TPN by \( M \) and we define the projection functions \( \pi, \tau \) 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 \( \rightarrow \subseteq M \times T \times M \) of a TPN satisfies the following property: if \( m \rightarrow m' \) then:

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

Actually there are five main classes of TPN, with several subclasses:

- **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 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 to 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, 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’ or Out-fint, In-single). This is the DTPN (Discrete Timed Petri net) model where a transition can fire at the time of the maximal time stamp of its input tokens. One of the earliest enabled transitions will fire. The input delay is added to the time stamp in order to determine if it is consumable or not.

- **Generalized Stochastic Petri nets (M4)**
  This model class is a stochastic extension of model class \( M2 \) with eager semantics and exponentially distributed delays connected to transitions.

- **Discrete Stochastic Petri nets (M5)**
  This model class is a stochastic extension of model class \( M3 \) with arbitrary discrete delay distributions for the output delays of transitions. Competitions between enabled transitions are solved by a weight function over the transitions. Note that this class has not been studied much.

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 \([24, 13]\). Some models of class \( M3 \) (DTPN) can also be translated to timed automata, but because of eagerness the translation is only possible for the sub-class (Tr-equal, In-single, Out-rint) (see section 5).

Generalized Stochastic Petri Nets (GSPN) is the famous class of stochastic Petri nets. It belongs to the class of \( M2 \), i.e. 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, we will introduce a stochastic version of the class of \( M3 \), which we call \( DSPN \) in the Section 7.

To compare model classes we will define an equivalence between classes.

**Definition 4 (Equivalence of model classes).** A model class \( I \) of timed Petri nets is included in model class \( J \), denoted by \( I \subseteq J \), if for each net system \((X, x)\) of class \( I \), there exists a net system \((Y, y)\) of class \( J \) such that \((X, x) \cong_y (Y, y)\).

Model classes \( I \) and \( J \) are equivalent, denoted by \( I \sim J \) if \( I \subseteq J \land J \subseteq I \).

Note that if for each net system \((X, x)\) of class \( I \), there exists a net system \((Y, y)\) of class \( J \) such that \( FS(X, x) \subseteq FS(Y, y) \) then we denote it by \( I \subseteq_{fs} J \).

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.

**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 5 (Activator).** Consider a TPN with the set of all markings \( M \). A marking \( a \in M \) is called an activator of transition \( t \in T \) in a marking \( m \in M \) if (1) \( a \subseteq m \), (2) \( \pi(m(i)) \mid i \in dom(a) \) = \( ^*t \) and \( |a| = |^*t| \) (\( t \) is classically enabled), and (3) for any \( i \in dom(a) : t \pi(m(i)) = \min(t \pi(m(j)) \mid j \in dom(a) \land \pi(a(j)) = \pi(a(i))) \) (\( i \) 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 6 (Enabling time, firing time).** For a DTPN let \( a \) be an activator of \( t \in T \) in marking \( m \). The enabling time of transition \( t \) by an activator \( a \) is defined as

\[
et(a, t) = \max\{\tau(a(i)) + \delta(\tau(a(i)), t) \mid i \in dom(a)\}
\]

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 completely determined by the marking itself!

The firing of a transition and its effect on a marking of a DTPN is described by the transition relation. Produced tokens are "fresh", i.e. they have new identities.
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.
The round-off
definition 11 (Round-off). A DTPN \( N = (P, T, F, δ) \) with an initial marking \( m_0 \) has a proxy fDTPN \( ϕ(N) \) with the same initial marking such that \( ϕ(N) = (P, T, F, δ') \) where:

- \( A = \{τ(m(i)) \mid i ∈ dom(m_0)\} \cup \{\min(δ(x, y), \max(δ(x, y)) \mid (x, y) ∈ F\}
- \( ∀(x, y) ∈ F : δ'(x, y) = \{i/d \mid i ∈ \mathbb{Z} ∧ \min(δ(x, y))/d ≤ i/d ≤ \max(δ(x, y))/d\} \) where \( d = \min\{k ∈ \mathbb{N}[k.a | a ∈ A] \subseteq \mathbb{N}\} \) 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 11 (Round-off Relation). For the set of all markings \( \mathbb{M} \) of a DTPN: \( ∀m, \bar{m} ∈ \mathbb{M} : m \sim \bar{m} \) iff \( dom(m) = dom(\bar{m}) \) \& \( ∀i ∈ dom(m) : π(m(i)) = π(\bar{m}(i)) \) and \( ∀i ∈ dom(m) : ∃k ∈ \mathbb{Z}, d ∈ \mathbb{N} : \)

\[
(τ(m(i)) ∈ [k/d, k/d + 1/2d] ∧ τ(\bar{m}(i)) = k/d) \lor \\
(τ(m(i)) ∈ (k/d + 1/2d, (k + 1)/d) ∧ τ(\bar{m}(i)) = (k + 1)/d)
\]

Note that we will assert that a marking \( \bar{m} ∈ \mathbb{M} \) is a round-off of a marking \( m ∈ \mathbb{M} \) by the predicate \( ro(m, \bar{m}) \).

The round-off relation preserves the order of timestamps.

\[\begin{array}{c}
\text{Current Marking: } m' \\
\text{Two executable firing sequences} \\
m' \xrightarrow{f5} m'' \quad \text{Creation time of marking } m'' \text{ is 8} \\
m' \xrightarrow{f3, f4} m'' \quad \text{Creation time of marking } m'' \text{ is 13}
\end{array}\]

Figure 2: Example of creation time
Corollary 3. Let $N$ be a DTPN with the set of all reachable markings $\mathbb{M}$ from an initial marking $m_0$. Let $\tilde{N} = \varphi(N)$ be its proxy fDTPN with the set of all reachable markings $\tilde{\mathbb{M}}$ from the same initial marking. If two markings $m \in \mathbb{M}, \tilde{m} \in \tilde{\mathbb{M}} : m \sim \tilde{m}$ then $\forall i, j \in \text{dom}(m)$:

\[
\tau(m(i)) = \tau(m(j)) \Rightarrow \tau(\tilde{m}(i)) = \tau(\tilde{m}(j))
\]
\[
\tau(m(i)) < \tau(m(j)) \Rightarrow \tau(\tilde{m}(i)) \leq \tau(\tilde{m}(j))
\]
\[
\tau(m(i)) > \tau(m(j)) \Rightarrow \tau(\tilde{m}(i)) \geq \tau(\tilde{m}(j))
\]

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

Theorem 4 (Simulation by proxy). Let $N$ be a DTPN with the set of all reachable markings $\mathbb{M}$ from an initial marking $m_0$. Let $\tilde{N} = \varphi(N)$ be its proxy fDTPN with the set of all reachable markings $\tilde{\mathbb{M}}$ from the same initial marking. Then $(\tilde{N}, m_0) \leq (N, m_0)$ with respect to the round-off relation.

Proof. Consider two markings $m, \tilde{m} \in \mathbb{M}, \tilde{m} \in \tilde{\mathbb{M}} : m \sim \tilde{m}$. Suppose $m \xrightarrow{t} m'$. We will prove that $(1) \exists \tilde{m'} \in \tilde{\mathbb{M}} : \tilde{m} \xrightarrow{\tilde{t}} \tilde{m}'$ and $(2) m' \sim \tilde{m}'$.

(1) Since $m \sim \tilde{m}$ by Corollary 3, the order of timestamps is preserved in marking $\tilde{m}$ and hence the same transition $t$ is enabled and $\exists \tilde{m}' \in \tilde{\mathbb{M}} : \tilde{m} \xrightarrow{\tilde{t}} \tilde{m}'$.

(2) Since $m \xrightarrow{t} m'$ and $m \sim \tilde{m}$, there exists a token with identity $i \in \text{dom}(m)$ such that $\tau(m(i)) + \delta(\pi(m(i)), t) = \text{ft}(m)$ and $\tau(\tilde{m}(i)) + \delta(\pi(\tilde{m}(i)), \tilde{t}) = \text{ft}(\tilde{m})$ and $\text{ro}(\pi(m(i)), \tau(m(i)))$. Hence $\text{ro}(\pi(\tilde{m}(i)), \tau(\tilde{m}(i)))$ holds, i.e. $\text{ro}(\text{ft}(m), \text{ft}(\tilde{m}))$. Note that $|\text{ft}(m) - \text{ft}(\tilde{m})| \leq 1/2d$.

Since $\text{ro}(\text{ft}(m), \text{ft}(\tilde{m}))$, $\exists k \in \mathbb{Z} : \text{ft}(m) \in [k/d, (k + 1)/d]$ and we may write $\text{ft}(m) = k/d + v$, where $0 \leq v \leq 1/d$.

For the production of tokens (see Def. 6), each fresh token is assigned a timestamp $\text{ft}(m) + x$, where the value $x$ is chosen from an associated delay set $[l, u]$. So on the grid, there exists $i, w \in \mathbb{Z}$ such that $\text{ft}(m) + x = (k + i)/d + w$.

Since $\text{ft}(m) = k/d + v$, we may rewrite as $x = i/d + w - v$.

We will show that $\exists \tilde{x} \in [l, u] : \text{ro}(\text{ft}(m) + x, \text{ft}(\tilde{m}) + \tilde{x})$.

- Suppose $\text{ft}(m) < \text{ft}(\tilde{m})$ and $0 \leq w < 1/2d$. Then $1/2d < v \leq 1/d$ and $-1/2d \leq w - v < 0$. Hence $l \leq (i - 1)/d \leq x < i/d \leq u$. Since $|\text{ft}(m) - \text{ft}(\tilde{m})| \leq 1/2d$, we may choose $\tilde{x} = (i - 1)/d$ and then $\text{ro}(\text{ft}(m) + x, \text{ft}(\tilde{m}) + \tilde{x})$.

- Suppose $\text{ft}(m) < \text{ft}(\tilde{m})$ and $1/2d < w < 1/2d$. Then $1/2d < v \leq 1/d$ and $-1/2d \leq w - v < 1/2d$. Hence $l \leq (i + 1)/d + 1/2d < x < i/d + 1/2d \leq u$. Since $|\text{ft}(m) - \text{ft}(\tilde{m})| \leq 1/2d$, we may choose $\tilde{x} = i/d$ and then $\text{ro}(\text{ft}(m) + x, \text{ft}(\tilde{m}) + \tilde{x})$.

- Suppose $\text{ft}(m) > \text{ft}(\tilde{m})$ and $1/2d < w < 1/2d$. Then $0 < v \leq 1/2d$ and $0 < w - v < 1/2d$. Hence $l \leq i/d < x < (i + 1)/d \leq u$. Since $|\text{ft}(m) - \text{ft}(\tilde{m})| \leq 1/2d$, we may choose $\tilde{x} = (i + 1)/d$ and then $\text{ro}(\text{ft}(m) + x, \text{ft}(\tilde{m}) + \tilde{x})$.

- Suppose $\text{ft}(m) > \text{ft}(\tilde{m})$ and $0 < w \leq 1/2d$. Then $0 < v \leq 1/2d$ and $-1/2d < w - v < 1/2d$. Hence $l \leq i/d - 1/2d < x < i/d + 1/2d \leq u$. Since $|\text{ft}(m) - \text{ft}(\tilde{m})| \leq 1/2d$, we may choose $\tilde{x} = i/d$ and then $\text{ro}(\text{ft}(m) + x, \text{ft}(\tilde{m}) + \tilde{x})$.

Hence $m' \sim \tilde{m}'$. \hfill \Box

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. 3. The grid distance $1/d = 1/2$. From the initial marking with tokens in places $P_1$ and $P_2$ with zero timestamps, a marking $m$ with tokens in places $P_3$ and $P_4$ with timestamps $11/16$ and $10/16$, respectively is reachable. In the proxy fDTPN, a marking $\tilde{m}$ is reachable such that $m \sim \tilde{m}$, i.e. with tokens in places $P_3$ and $P_4$ with timestamps equal to one. From marking $m$ only transition $b$ is enabled but from marking $\tilde{m}$ both transitions $a$ and $b$ are enabled. However, by theorem 2 we know that a DTPN simulates its proxy w.r.t. the identity relation.

Corollary 5 (Simulation Equivalence). A DTPN $N$ and its proxy fDTPN $\tilde{N} = \varphi(N)$ are simulation equivalent and therefore $FS(N, m_0) = FS(\tilde{N}, m_0)$.
The final step in the reduction process is to show that any fDTPN is 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. 4.

![Figure 4: Construction of a sDTPN from an fDTPN](image)

**Definition 12 (Reduction of fDTPN to sDTPN).** Let \( N = (P, T, F, \delta) \) be a fDTPN. For each \( t \in T \) let \( A_t = \prod_{p \in t} \delta(t, p) \) be the generalized cartesian product of all its delay sets. Then, the corresponding sDTPN is the tuple \( \text{construct}(N) = (P, T', F', \delta') \), where

\[
egin{align*}
T' &= \{ t_x \mid t \in T \land x \in A_t \} \\
F' &= \{(p, t_x) \mid (p, t) \in F \land x \in A_t \} \cup \{(t_x, p) \mid (t, p) \in F \land x \in A_t \} \\
\forall p \in P : \forall t \in T : \forall x \in A_t : \delta'(p, t_x) = x(p) \\
\forall p \in P : \forall t \in T : \forall x \in A_t : \delta'(t_x, p) = x(p)
\end{align*}
\]

**Theorem 6 (Bisimulation of fDTPN and sDTPN).** Let \( N \) be an arbitrary fDTPN. Then, \( N \simeq \text{construct}(N) \).

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

Note that if we extend the definition of a DTPN to outgoing arc delays having open rational intervals then the proxy fDTPN does not preserve the simulation property. We show this with an example in the Fig. 5.

**4.3. Analysis of sDTPN**

To analyze the behavior of DTPN’s it is sufficient to consider sDTPN’s. However, since time is non-decreasing, the reachability graph of a sDTPN is usually infinite. But there is a finite time window that contains all relevant behavior. This is 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
Consider a sDTPN with the set of all markings \( M \).

Let \( N \) be a sDTPN with the set of all markings \( M \) and timestamps less than or equal to \( o \), call this labeled transition system rDTPN, although it formally is not a DTPN.

We introduce a labeled transition system for an sDTPN that is strongly bisimilar to it. Therefore we can call this labeled transition system rDTPN, although it formally is not a DTPN.

\[ \delta_{\text{in}} = \max\{\delta(p,t) | (p,t) \in (P \times T) \cap F\} \]

\[ \delta_{\text{out}} = \max\{\delta(t,p) | (t,p) \in (T \times P) \cap F\} \]

A reduced marking is obtained by subtracting the firing time from the timestamp of each token in the marking of a sDTPN, with a lower bound of \(-\delta_{\text{in}}\).

**Definition 13 (Reduction function).** Consider a sDTPN with the set of all markings \( \mathbb{M} \). The reduction function \( \alpha : \mathbb{M} \rightarrow \mathbb{M} \) is defined \( \forall m \in \mathbb{M} : \text{dom}(\alpha(m)) = \text{dom}(m) \) and \( \forall i \in \text{dom}(m) : \pi(\alpha(m))(i) = \pi(m)(i) \land \tau(\alpha(m))(i) = \max\{-\delta_{\text{in}}, \tau(m)(i) - ft(m)\} \).

Note that the reduction function is either (1) reducing the timestamp of tokens by the firing time, or (2) mapping timestamps less than or equal to \(-\delta_{\text{in}}\) to \(-\delta_{\text{in}}\).

**Corollary 7.** Let \( N \) be a sDTPN with the set of all markings \( \mathbb{M} \). Then \( \forall m \in \mathbb{M}, t \in T : \)

- \( \forall i \in \text{dom}(m) : \tau(\alpha(m))(i) \geq \tau(m)(i) - ft(m) \)
- \( A(\alpha(m), t) = \{a(a) | a \in A(m, t)\} \)

The firing time of a reduced marking is zero.

**Lemma 8.** Consider a sDTPN with the set of all markings \( \mathbb{M} \). Then \( \forall m \in \mathbb{M} : ft(\alpha(m)) = 0. \)

**Proof.** Note that \( \alpha(m) \in \mathbb{M} \).

\[ \text{ft}(\alpha(m)) = \min_{t \in T} \min_{i \in \text{dom}(\alpha(m), t)} \max \{\tau(\alpha(m))(i) + \delta(\pi(\alpha(m))(i)), t\} \]

By Def. 13 and Cor. 7, we may write

\[ = \min_{t \in T} \max_{i \in \text{dom}(\alpha(m))} \{\min\{\max\{-\delta_{\text{in}}, \tau(\alpha(m))(i) - ft(m)\} + \delta(\pi(\alpha(m))(i), t)\} \}
\]

\[ = \min_{t \in T} \max_{i \in \text{dom}(\alpha(m))} \{\min\{\max\{-\delta_{\text{in}}, \tau(\alpha(m))(i) - ft(m) + \delta(\pi(\alpha(m))(i), t)\} \}
\]

\[ = \min_{t \in T} \max_{i \in \text{dom}(\alpha(m))} \{\min\{\max\{-\delta_{\text{in}}, \tau(\alpha(m))(i) - ft(m) + \delta(\pi(\alpha(m))(i), t)\} \}
\]

\[ = \min_{t \in T} \max_{i \in \text{dom}(\alpha(m))} \{\min\{\max\{-\delta_{\text{in}}, \tau(\alpha(m))(i) - ft(m) + \delta(\pi(\alpha(m))(i), t)\} \}
\]
By definition of $\delta^+_i$ the term
\[
\max_{i \in \text{dom}(a)} \{-\delta^+_i + \text{ft}(m) + \delta(\pi(a(i)), t)\} \leq \text{ft}(m)
\]
and for all $t \in T$ and $a \in A(m,t)$ the term
\[
\max_{i \in \text{dom}(a)} \{\tau(a(i)) + \delta(\pi(a(i)), t)\} \geq \text{ft}(m)
\]
So we may write
\[
= \min_{t \in T} \min_{a \in A(m,t)} \max_{i \in \text{dom}(a)} \{\tau(a(i)) + \delta(\pi(a(i)), t)\} - \text{ft}(m) = 0
\]

As a consequence of Lemma 8, the reduction function $\alpha$ is idempotent.

**Corollary 9.** Let $N$ be a sDTPN. Then $\forall m \in M : \alpha(\alpha(m)) = \alpha(m)$.

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 original system.

![Figure 6: Markings reachable from an abstract marking](image)

**Lemma 10.** Consider a sDTPN with the set of markings $M$. Let $m, m' \in M : m \rightarrow m'$. Then $\exists \tilde{m} \in M, \tilde{a} \in A(\alpha(m), t) : \alpha(m) \rightarrow \tilde{m}$ and $\text{ft}(\tilde{m}) = \text{ft}(m') - \text{ft}(m)$ and $\alpha(m') = \alpha(\tilde{m})$.

**Proof.** For $\alpha(m) = m$ the lemma holds trivially. Suppose $\alpha(m) \neq m$.

- We will first prove $\exists \tilde{m} \in M, \tilde{a} \in A(\alpha(m), t) : \alpha(m) \rightarrow \tilde{m}$.

  \[
  \text{ft}(m) = \min_{t \in T} \min_{a \in A(m,t)} \max_{i \in \text{dom}(a)} \{\tau(a(i)) + \delta(\pi(a(i)), t)\}
  \]
  \[
  0 = \min_{t \in T} \min_{a \in A(m,t)} \max_{i \in \text{dom}(a)} \{\tau(a(i)) - \text{ft}(m) + \delta(\pi(a(i)), t)\}
  \]

  By the Def. of $\delta^+_i$ the term
  \[
  -\delta^+_i + \delta(\pi(a(i)), t) \leq 0
  \]

  Furthermore, for all $t \in T$ and $a \in A(m,t)$ the term
  \[
  \max_{i \in \text{dom}(a)} \{\tau(a(i)) - \text{ft}(m) + \delta(\pi(a(i)), t)\} \geq 0
  \]
Hence,

\[ 0 = \min_{i \in T} \min_{\delta(i)} \max_{\alpha(i) \in \text{dom}(\alpha)} \{ \max \{ \tau(\alpha(i)) - ft(m), -\delta_i \} + \delta(\alpha(i), t) \} \]

By the Def. 13 and Cor. 7 we may write

\[ = \min_{i \in T} \min_{\delta(i)} \max_{\alpha(i) \in \text{dom}(\alpha)} \{ \tau(\alpha(i)) + \delta(\alpha(i), t) \} = ft(\alpha(m)) \]

So there exists an activator \( \tilde{a} \in A(\alpha(m), t) : \text{dom}(\tilde{a}) = \text{dom}(a) \wedge \pi(\alpha(i)) = \pi(\tilde{a}(i)) \wedge \tau(\tilde{a}(i)) = \max \{ -\delta_i, \tau(\alpha(i)) - ft(m) \} \) and a marking \( \tilde{m} \in M : \alpha(m) \rightarrow \tilde{m} \).

Consider the effect of firing transition \( t \) from markings \( m \) and \( \alpha(m) \).

Consumption: Since \( \text{dom}(a) = \text{dom}(\tilde{a}) \), the same tokens are consumed from marking \( m \) and \( \alpha(m) \).

Production: In marking \( m' \), tokens are produced in each \( p \in T^* \) with a timestamp \( x = ft(m) + \delta(t, p) \). In marking \( \tilde{m} \), tokens are produced in corresponding places but with a timestamp \( \delta(t, p) = \max \{ x - ft(m), -\delta_i \} \). Note that the identities of the produced tokens can be chosen the same in both systems

For the remaining tokens, \( \forall i \in \text{dom}(m) \wedge \text{dom}(a) : \tau(\tilde{m}(i)) = \max \{ \tau(m'(i)) - ft(m), -\delta_i \} \), holds. Hence we have proven \( \forall i \in \text{dom}(\tilde{m}) : \tau(\tilde{m}(i)) = \max \{ \tau(m'(i)) - ft(m), -\delta_i \} \).

- Next we will prove \( ft(\tilde{m}) = ft(m') - ft(m) \).

\[ \begin{align*}
ft(\tilde{m}) &= \min_{i \in T} \min_{\delta(i)} \max_{\alpha(i) \in \text{dom}(\alpha)} \{ \tau(\alpha(i)) + \delta(\alpha(i), t) \} \\
&= \min_{i \in T} \min_{\delta(i)} \max_{\alpha(i) \in \text{dom}(\alpha)} \{ \max \{ \tau(\alpha(i)) - ft(m), -\delta_i \} + \delta(\alpha(i), t) \} \\
&= \min_{i \in T} \min_{\delta(i)} \max_{\alpha(i) \in \text{dom}(\alpha)} \{ \max \{ \tau(\alpha(i)) + \delta(\alpha(i), t) \} - ft(m) \} \\
&= \min_{i \in T} \min_{\delta(i)} \max_{\alpha(i) \in \text{dom}(\alpha)} \{ \tau(\alpha(i)) + \delta(\alpha(i), t) \} - ft(m) \\
&= \max_{\alpha(i) \in \text{dom}(\alpha)} \{ \tau(\alpha(i)) + \delta(\alpha(i), t) \} - ft(m) \\
\end{align*} \]

Since \( \delta_i \geq \delta(\alpha(i), t) \),

\[ \max_{\alpha(i) \in \text{dom}(\alpha)} \{ ft(m) - \delta_i + \delta(\alpha(i), t) \} \leq ft(m) \]

Since \( ft(m') \geq ft(m) \), we have for all \( t \in T \) and \( \alpha' \in A(m', t) \)

\[ \max_{\alpha' \in A(m', t)} \tau(\alpha'(i)) + \delta(\alpha'(i), t) \geq ft(m) \]

Hence \( ft(\tilde{m}) = ft(m') - ft(m) \).

- Finally, we prove \( \alpha(\tilde{m}) = \alpha(m') \). By the definition of reduction function, \( \text{dom}(\alpha(\tilde{m})) = \text{dom}(\alpha(m')) \) and \( \forall i \in \text{dom}(\alpha(\tilde{m})) : \pi(\alpha(\tilde{m})) = \pi(\alpha(m')) \) and

\[ \forall i \in \text{dom}(\alpha(m')) : \tau(\alpha(m')(i)) = \max \{ \tau(m'(i)) - ft(m'), -\delta_i \}, \text{ and} \]

\[ \forall i \in \text{dom}(\alpha(\tilde{m})) : \tau(\alpha(\tilde{m})(i)) = \max \{ \tau(\tilde{m}(i)) - ft(\tilde{m}), -\delta_i \} \]

Since \( \forall i \in \text{dom}(\tilde{m}) : \tau(\tilde{m}(i)) = \max \{ \tau(m'(i)) - ft(m), -\delta_i \} \),

We may write

\[ \forall i \in \text{dom}(\alpha(\tilde{m})) : \tau(\alpha(\tilde{m})(i)) = \max \{ \max \{ \tau(m'(i)) - ft(m), -\delta_i \} - ft(\tilde{m}), -\delta_i \} \]
Since we know already that \( ft(\tilde{m}) = ft(m') - ft(m) \), we may write
\[
\forall i \in dom(\alpha(\tilde{m})) : \tau(\alpha(\tilde{m}))(i) = \max\{\max\{\tau(m'(i)) - ft(m'), -\delta_i^1\} - ft(m)\}, -\delta_i^1\}
\]
Since \( ft(m') \geq ft(m) \), we have \(-\delta_i^1 = ft(m') + ft(m) \leq -\delta_i^1\). Hence \( \forall i \in dom(\alpha(\tilde{m})) : \tau(\alpha(\tilde{m}))(i) = \max\{\tau(m'(i)) - ft(m'), -\delta_i^1\} \). So we have proven \( \alpha(\tilde{m}) = \alpha(m') \).

\[\square\]

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

**Corollary 11.** Let \( N \) be an sDTPN and \((N, M, \to, m_0)\) be its labeled transition system. Let \( n \in \mathbb{N} : n > 0 \) and markings \( m_0, m_1, \ldots, m_n \in M \). Let \( \sigma = (t_1; \ldots; t_n) \) be an executable firing sequence of net \( N \) such that \( m_0 \xrightarrow{t_1} m_1 \xrightarrow{t_2} \ldots \xrightarrow{t_n} m_n \).
Then there exist markings \( \tilde{m}_0, \tilde{m}_1, \ldots, \tilde{m}_n \in \tilde{M} : \tilde{m}_0 = m_0 \land \forall i \in [1, n] : \alpha(m_{i-1}) \xrightarrow{t_i} \tilde{m}_i \) and
\[
ft(m_n) = \sum_{j=0}^{n} ft(\tilde{m}_j)
\]

Note that in this way, given a firing sequence, we are able to compute the time a marking is reachable using only reduced markings.

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

**Definition 14 (rDTPN).** Let \( N = (P, T, F, \delta) \) be an sDTPN with the set of all reachable markings \( \tilde{M} \) and the set of all reduced markings \( \tilde{M} \). Its **reduced DTPN** called rDTPN is a labeled transition system \( \rho(N) = (\tilde{M}, T, \rightsquigarrow, \tilde{m}_0) \), where \( \rightsquigarrow \subseteq \tilde{M} \times T \times \tilde{M} \) is the **reduced transition relation** defined as \( \forall \tilde{m}, \tilde{m}' \in \tilde{M} : \tilde{m} \rightsquigarrow \tilde{m}' \leftrightarrow \exists \tilde{m} \in \tilde{M} : \tilde{a} \in A(\tilde{m}, t) \land \tilde{m} \xrightarrow{t} \tilde{m} \land \alpha(\tilde{m}) = \tilde{m}' \) and \( \tilde{m}_0 = \alpha(m_0) \) is an initial marking.

For a given sDTPN, its labeled transition system and rDTPN are bisimilar w.r.t reduction relation. Due to Lemma 10, the time relation in the bisimulation is implicit.

**Theorem 12 (Bisimulation sDTPN and rDTPN).** Consider an sDTPN with a labeled transition system \( N = (M, T, \to, m_0) \) and its rDTPN \( \rho(N) = (\tilde{M}, T, \rightsquigarrow, \tilde{m}_0) \). Then \( (N, m_0) \simeq (\tilde{N}, \tilde{m}_0) \).

**Proof.** Let \( R \subseteq M \times \tilde{M} \) be a relation defined as
\[
\forall m \in M, \tilde{m} \in \tilde{M} : (m, \tilde{m}) \in R \iff \alpha(m) = \tilde{m}
\]
1. Let \( m \xrightarrow{t} m' \) and \( (m, \alpha(m)) \in R \). Then, by Lemma 10, a marking \( \tilde{m} \in \tilde{M} \) exists such that \( \alpha(m) \xrightarrow{\tilde{t}} \tilde{m} \) and \( \alpha(m') = \alpha(\tilde{m}) \). By the Def. 14, we get \( \alpha(m) \xrightarrow{\tilde{t}} \alpha(m') \).
2. Let \( \tilde{m} \xrightarrow{\tilde{t}} \tilde{m}' \) and \( (m, \tilde{m}) \in R \), i.a. \( \alpha(m) = \tilde{m} \). By the Def. 14, there exists a marking \( \tilde{m} \in \tilde{M} : \tilde{m} \xrightarrow{\tilde{t}} \tilde{m} \) and \( \alpha(\tilde{m}) = \tilde{m}' \). We will first prove that transition \( \tilde{t} \) is enabled in marking \( \tilde{m} \). Note that since \( \alpha(m) \xrightarrow{\tilde{t}} \tilde{m} \), we have by Corollary 7 that an activator \( \tilde{a} \in \tilde{M} \) exists such that \( \alpha(a) = \tilde{a} \). Hence,
\[
ft(\alpha(m)) = \max_{i \in dom(\tilde{a})} \{\tau(\tilde{a}(i)) + \delta(\pi(\tilde{a}(i)), t)\}
\]
\[
0 = \max_{i \in dom(a)} \{\max\{\tau(\alpha(i)) - ft(m), -\delta_i^1\} + \delta(\pi(\alpha(i)), t)\}
\]
\[
0 = \max_{i \in dom(a)} \{\max\{\tau(\alpha(i)) + \delta(\pi(\alpha(i)), t), -\delta_i^1\} + ft(m)\}
\]
Since $-\delta^i_t + ft(m) + \delta(\pi(a(i)), t) \leq ft(m)$ and for all $a \in A(m, t)$ the term

$$\max_{t \in dom(a)} \tau(a(i)) + \delta(\pi(a(i)), t) \geq ft(m)$$

Hence,

$$ft(m) = \max_{t \in dom(a)} [\tau(a(i)) + \delta(\pi(a(i)), t)]$$

So transition $t$ is enabled in marking $m$ and a marking $m' \in M$ exists such that $m \xrightarrow{t} m'$. By Lemma 10, we know $\alpha(m) = \alpha(m')$. Hence we have proven $\alpha(m') = \tilde{m}'$.

\[\square\]

The number of different timestamps in the reachability graph of the rDTPN is finite. This is observed in several papers (see [7]). 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 11 and then we are in the former case.

**Theorem 13.** 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.

**Proof.** Consider the first case where all time stamps and delays are in $\mathbb{Z}$. Note that all operations we perform are:

- taking the maxima or minima of time stamps, i.e. selecting one of them.
- adding one of the finite set of delays.
- subtracting a selected time stamp from others with a min. delay of $-\delta^i_t$.

With induction we will show that all time stamps stay in $[-\delta^i_t, e]$, where $e$ is the maximum of $\delta^i_t, \delta^i_t$ and the maximal time stamp is in the initial marking. (We assumed in the initial marking there are no negative time stamps).

So in the initial marking all time stamps are in $[0, e]$. Suppose in an arbitrary marking $m$ they are in $[-\delta^i_t, e]$, then we compute $ft(m)$ which is the selection of one of the time stamp, possibly increased with a delay. This time is subtracted from the remaining time stamps but with a lower bound of $-\delta^i_t$. So they stay in $[-\delta^i_t, e]$. Now consider the newly produced tokens. They have a time stamp in the interval $[0, \delta^i_t]$. So all time stamps are in $[-\delta^i_t, e]$. So we have finitely different time stamps.

Now consider the case where the initial time stamps and the delays are in $\mathbb{Q}$. This is a finite set, so the set of denominators has a least common multiple say $l$. Now we may multiply all time stamps of the initial marking and all delays with $l$ to obtain natural numerators with a common denominator $l$. In all operations we may extract division by $l$, for e.g. $\max(\frac{a}{l}, \frac{b}{l}) = \frac{\max(a, b)}{l}$ or $\frac{a}{l} - \frac{b}{l} = \frac{a-b}{l}$. So we have in fact only computations with integers between two bounds. Only at the end of a computation we divide by $l$. So we are in the former case: a finite set of time stamps and delays, say $E$.

Now consider the underlying Petri net system $(P, T, F, \bar{m}_0)$, where $\bar{m}_0$ is derived from $m_0$ by deleting the time stamps. The maximal number of tokens in all reachable markings is bounded by some $b \in \mathbb{N}$. Since the firing rules of transitions in DTPN’s are restrictions, the set of reachable markings in the reduced DTPN’s can never have more tokens than the underlying Petri net system. Hence it is bounded. Since the number of possible time stamps is bounded

$$\forall m \in M : |\{i \in dom(m) | \tau(m(i)) \in E\}| = |dom(\bar{m})| \leq b.$$  

Furthermore, using Corollary 11, 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.
5. Relationship between DTPN and Timed Automata

There exist many tools to model and verify properties of a timed system, most notably UPPAAL. In UPPAAL, a timed system is modeled as a timed automaton but whose clock valuations range over natural numbers. In this section, we will transform an DTPN into a timed automaton in order to be able to analyze them. However, not all DTPN’s can be expressed as a timed automaton. In fact, only a subclass of DTPN’s that are safe and whose synchronizing transitions have the same delay on each incoming arc (as in the Merlin time model (Tr-equal)), can be expressed as a timed automata. Note that we have only trace equivalence and not branching bisimilarity, since DTPN’s make choices of token timestamps (for tokens being produced) earlier than timed automata by specifying timestamps in the future. In order to make the transformation to UPPAAL understandable, we introduce a sub-class of DTPN called a clock DTPN (cDTPN). We will show that an arbitrary DTPN satisfying the delay requirement on synchronizing transitions can be transformed into a cDTPN by the arc refinement operation and by adding a set of marked places called timer places connected with bi-directional arcs to a subset of transitions. Note that the criteria for safe nets can be relaxed but we do not consider it. We will first briefly introduce the syntax and semantics of a timed automaton (c.f. [6]).

A Timed Automaton, is a finite automaton (a graph consisting of a finite set of locations and labeled edges) extended with a set of clocks modeled as real-valued variables. All clocks of the system are initialized to zero and then increase synchronously at the same rate. The behavior of a timed automaton is restricted by adding constraints on clock variables (guards on edges). Location invariants are used to force an automata to progress, i.e. leave the location before the invariant becomes false. An edge e between a location l₁ and l₂ is denoted by e = (l₁, C, t₁, R, l₂), where C is specifying the clock constraint that must be satisfied for the edge to be enabled, t₁ is the label associated with edge e and R is the set of clocks to reset when this edge is taken.

Definition 15 (Clock Constraints). Consider a finite set C of real valued variables. A clock constraint is a conjunction of predicates of the form x ≡ n or x − y ≡ n for x, y ∈ C, ≡∈ {≤, <, =, >, ≥} and n ∈ N. The set of clock constraints is denoted by B(C).

Definition 16 (Syntax of a Timed Automaton). A timed automaton is a tuple (L, T, C, E, I), where L is a finite set of locations, T is a finite set of transition labels, C is a finite set of clocks, E ⊆ L × B(C) × T × P(C) × L is the set of edges, and I : L → B(C) assigns clock constraints to locations called invariants. We restrict location invariants to clock constraints of the form x ≤ n or x < n, where x ∈ C and n ∈ N.

A location along with a valuation of all clocks of the system defines a state of a timed automaton.

Definition 17 (State). Let A = (L, T, C, E, I) be a timed automaton. The state of a timed automaton is a pair (l, v), where l ∈ L and v : C → R⁺ is a valuation function that assigns to each clock a non-negative real number. If the valuation of all clocks satisfies a clock constraint g ∈ B(C), then we write v ∈ g. The set of all states of a timed automaton is denoted by S and the initial state by s₀ = (l₀, v₀) satisfying ∀c ∈ C : v₀(c) = 0.

The state of a timed automaton changes when an enabled transition is fired. There are two types of transitions between states. A delay transition does not cause a change in location but allows the time in clocks to increase. A discrete transition causes a change in location if the clock valuations satisfy the guard condition associated with an edge corresponding the discrete transition. Once a discrete transition has fired, a subset of clocks will be reset to zero.

Definition 18 (Semantics of a Timed Automaton). The semantics of a timed automaton A = (L, T, C, E, I) is defined as a transition system having two types of transitions between states defined in the following way.

- A delay transition relation d → S × N × S is defined as ∀(l, v) ∈ S, d ∈ N : (l, v) → (l, v) d+ v(c) + d and v ∈ I(l) and v(c) + d' ∈ I(l) for all d' ≤ d.

- A discrete transition relation d → S × T × S is defined as ∀(l, v), (l', v') ∈ S, t ∈ T : (l, v) →t (l', v') if ∃l, g, t, y, l' ∈ E, v' ∈ I(l'), v ∈ g and v'(x) = 0, if x ∈ y and v'(x) = v(x), otherwise.
Throughout this section, we will require safe DTPN’s. Hence, we simplify the definition of a marking a bit. The timed marking of a safe DTPN assigns to each place, the timestamp of a token in that place. If a place has no token then we indicate it with infinity.

**Definition 19 (Simplification of Marking Definition).** The marking of a safe DTPN $N$ is a function $\tilde{m} : P \rightarrow \mathbb{N}^\infty$ defined for all $p \in P$ such that

$\tilde{m}(p) = \infty$, if $\neg\exists i \in \text{dom}(m) : \pi(m(i)) = p$

$\tilde{m}(p) = \tau(m(i))$, if $\exists i \in \text{dom}(m) : \pi(m(i)) = p$

Note that $\tilde{m}$ is defined for all places since the net is safe. When the context is clear, we will denote a marking $\tilde{m}$ by $m$.

A cDTPN is a safe DTPN with constraints on the syntax: (a) all incoming arcs belonging to a synchronizing transition (more than one pre-place) must have the same delay, (b) if a place has more than one incoming arc then all incoming arcs have zero delays, and (c) if a place has at least one incoming arc with a non-zero delay then there exists exactly one outgoing arc and this arc is connected to a transition with zero delay and this transition has exactly one pre-place. Note that the requirements (b) and (c) can be induced on an arbitrary DTPN by the arc refinement operation (see Fig. 7). It is easy to check that arc refinement preserves branching bisimulation.

**Definition 20 (cDTPN).** A cDTPN is a tuple $N = (P, T, F, \delta, m_0)$, where

![Figure 7: An example of a cDTPN](image-url)

Furthermore, a subset of marked places called timer places are associated with synchronizing transitions and transitions having at least one non-zero delay on outgoing arcs (called interval transitions). A timer place associated with a synchronizing transition is connected with bi-directional arcs to each transition that produces a token in the pre-place of the synchronizing transition. In this way, the token timestamp of the timer place is capturing the timestamp of the latest token in the pre-place of a synchronizing transition. A timer place associated with an interval transition is connected to itself with bi-directional arcs. In a similar way, the timestamp of the token in the timer place is reflecting the time of firing of an interval transition. We give an example of the transformation of a DTPN to cDTPN in Fig. 7.

**Definition 20 (cDTPN).** A cDTPN is a tuple $N = (P, T, F, \delta, m_0)$, where
Consider a cDTPN N with an initial marking m. Let the set of all reachable timed markings from the initial marking be $M$. Then for all $m' \in M$, there exists a transition $t \in T_{sync}$ such that $m(p) = \max(m(p) | p \in T_{sync})$ and $t \in T_{preint}$ such that $m' = m(\hat{\lambda}(t)) = \beta(t(m))$.

A cDTPN can be expressed as a timed automaton. The untimed reachability graph of a cDTPN defines the underlying automaton where markings correspond to locations and transitions between markings correspond to edges between locations labeled by the corresponding transition. Note that more than one edge can be labeled with the same transition. Each place of a clock DTPN is modeled as a clock variable in a timed automaton.

By the definition of a cDTPN, the latest token arriving in a pre-place of a synchronizing transition is a safe net and the set of interval places.

Eagerness in a DTPN means that as soon as a transition is enabled in a marking, it must fire. In a timed automaton, eagerness can be expressed by a location invariant that restricts the maximum delay that can occur in this location (delay transition of a timed automaton). This enforces a location to be left before its invariant evaluates to false. The invariant of a location is computed by taking the conjunction of the negation of guards belonging to outgoing edges from this location, with the following exception: If a guard has a clock constraint containing a minimum over a delay set then we replace it with a maximum over the same delay set. The set of clocks to be reset on each edge are the
clocks that correspond post-places of the transition (labeled on this edge) in the cDTPN. In this way, we are able to transfer the notion of transition enabling and urgency to a timed automaton. Note that eagerness is not expressed for edges labeled with delay transitions of a cDTPN.

Note that we will write $\neg(X \leq Y)$ to denote $(X \geq Y)$. Furthermore, we extend the notion of postset and preset of a node in a Petri net to locations and edges in a timed automata. We will sometimes refer to an edge of a timed automaton by its label (corresponding a transition in a cDTPN). Note that we identify clocks in a timed automaton with places in a cDTPN.

**Definition 21 (Timed Automaton of a cDTPN).** Let $N = (P, T, F, \delta, m_0)$ be a cDTPN. Let $\bar{M}$ be the set of all reachable untimed markings from marking $m_0$ (ignoring timestamps) of the underlying Petri net $(P, T, F)$. Two guard functions $G : T \rightarrow \mathcal{B}(P)$ and $\bar{G} : T_{\text{postint}} \rightarrow \mathcal{B}(P)$ satisfy $t \in T$ and $p \in \pi(t)$ such that

- $G(t) = (p \geq \delta(p, t))$, if $t \in T \setminus (T_{\text{postint}} \cup T_{\text{sync}})$,
- $G(t) = (\lambda(t) \geq \max_{p \in \pi(t)}(\delta(p, t)))$, if $t \in T_{\text{sync}}$,
- $G(t) = (\lambda(\text{int}(t)) \geq \min(\delta(\text{int}(t), p)))$, if $t \in T_{\text{postint}}$,
- $\bar{G}(t) = (\lambda(\text{int}(t)) \geq \max(\delta(\text{int}(t), p)))$, if $t \in T_{\text{postint}}$.

The timed automaton of clock DTPN $N$ is denoted by $\text{TA}(N) = (\bar{M}, T, \pi, E, I)$, where

- Edges: $(m, G(t), t, t', m') \in E \Leftrightarrow \exists m, m' \in \bar{M}, t \in T : m' = m - t + t'$
- Invariants: $\forall m \in \bar{M} : I(m) = \bigwedge_{t \in T \setminus T_{\text{postint}}} \neg G(t) \land \bigwedge_{t \in T_{\text{postint}}} \neg \bar{G}(t)$
We will first explain the intuition behind a timer place with an example described in the Fig. 8. Consider the state with tokens in places $P_3, P_4, U_1, U_2$ and $U_3$. From this state two transitions are enabled of which one transition has two pre-places, namely $P_3$ and $P_4$. So the invariant of this state becomes $(P_3 \leq 3 \lor P_4 \leq 3) \land P_3 \leq 7$. However, as disjunctions are not allowed in clock constraints, this strategy does not work. Note that the problem arises because the guard belonging to an edge of a synchronizing transition $t$ is a conjunction of $|\tau|$ terms. To solve this problem, we must model the guard belonging to an edge with a synchronizing transition using a single clock. This is only possible if we restrict each synchronizing transition to incoming arcs having equal delays. Note that in a cDTPN, we require for each synchronizing transition, the corresponding timer place is connected with bi-directional arcs to all transitions that produce a token in the pre-place of the synchronizing transition. Hence the timestamp of token in the timer place now represents the maximal timestamp of tokens in the pre-places of the synchronizing transition. So the guard of a synchronizing transition can be expressed as a predicate over a clock corresponding the timer place. In the Fig. 8, the invariant of state with tokens in places $P_3, P_4, U_1, U_2$ and $U_3$ translates into $P_3 \leq 7 \land U_1 \leq 3$.

Note that in a cDTPN, if a non-timer place has a preset larger than one, then all incoming arcs have zero delays. Due to this requirement, we are able to compute locally, the delay incurred by a token in a place before it becomes available. Furthermore, if a non-timer place has an interval transition in its preset then the only outgoing arc from this place to post-interval transition has a zero delay. Due to this requirement, we are able to express the guard of a post-interval transition $t$ as a clock constraint over a single clock $\lambda(\text{int}(t))$.

The relation between a timed marking of a cDTPN and a state of a timed automaton is defined by the state mapping relation.

**Definition 22 (State Relation).** Let $\mathbb{M}$ be the set of all timed markings of a cDTPN $N$. Let $S$ be the set of all states of a timed automaton $\text{TA}(N)$. Let $\text{mct} : \mathbb{M} \rightarrow \mathbb{N}$ be a function defined as $\text{mct}(m) = \max\{m(p) \mid p \in P \setminus P_{\text{int}}\}$, i.e. the maximal consumable token of marking $m$. The state relation $\beta \subseteq \mathbb{M} \times S$ is defined as $(m, (l, v)) \in \beta$ if and only if

$$l = \{p \mid m(p) \neq \infty\} \land \exists b \in [\text{mct}(m), \text{ft}(m)] : \forall p \in l \setminus P_{\text{int}} : v(p) = b - m(p)$$

Note that in a DTPN, choices about the timestamps of produced tokens are made when a transition fires, but in a timed automaton these choices are made in the state reachable after taking the edge labeled with this transition because of the guard and invariant associated with this state. So a DTPN is making a choice earlier than a timed automaton. Since transitions with delays on outgoing arcs are producing tokens in interval places and these tokens may have a timestamp in the future, we discard the clocks corresponding interval places in the state relation. Furthermore, if all incoming arcs of a cDTPN have zero delays then for all markings $m$ reachable from an initial marking $m_0$, we have $\text{mct}(m) = \text{ft}(m)$.

Given a set of reachable markings of a cDTPN, the state space of its timed automaton is the set of all related states.

**Definition 23 (State Space).** Let $N = (P, T, F, \delta, m_0)$ be a cDTPN. Let $\mathbb{M}$ be the set of all reachable markings from initial marking $m_0$. The state space of a timed automaton $\text{TA}(N)$ is the set

$$S = \{(l, v) \in S \mid \exists m \in \mathbb{M} : (m, (l, v)) \in \beta\}$$

We treat a delay transition of a timed automaton as a silent ($\tau$-labeled) step.

**Lemma 15 (Enabling of Delay Transitions).** Let $N$ be a cDTPN with the set of all reachable markings $\mathbb{M}$ from the initial marking $m_0$. Let $S$ be the state space of timed automaton $\text{TA}(N)$. Then $\forall m \in \mathbb{M}, (l, v) \in S : (m, (l, v)) \in \beta$ for some $b \in [\text{mct}(m), \text{ft}(m)]$ implies

$$(l, v) \xrightarrow{d} (l, v + d) \land (m, (l, v + d)) \in \beta, \text{ where } d = \text{ft}(m) - b$$

**Proof.** Suppose a delay transition $d = \text{ft}(m) - b$ cannot occur from state $(l, v)$. This implies that for some delay $d' < d : (l, v) \xrightarrow{d'} (l, v + d')$ and $\exists t \in T$ that is enabled in state $(l, v + d')$. 

20
• Suppose \( t \in T \setminus (T_{\text{Sync}} \cup T_{\text{Postint}}) \). Then \( \exists ! p \in ^*t \) and by the Def. 21
\[
\nu(p) + d > \nu(p) + d' = \delta(p, t)
\leq m(p) + \delta(p, t) - m(p)
\geq et(m, t) - m(p)
\geq ft(m) - m(p)
\]
Hence \( \nu(p) + d > ft(m) - m(p) \). Since \( d = ft(m) - b \), we may write \( \nu(p) + ft(m) - b > ft(m) - m(p) \). This implies \( \nu(p) + m(p) > b \). But \( \nu(p) + m(p) = b \). This is a contradiction.

• Suppose \( t \in T_{\text{Sync}} \). Then we replace \( p \) by \( \lambda(t) \) and arc delay \( \delta(p, t) \) by \( \max \{ \delta(p, t) \mid p \in ^*t \} \) and then the arguments are the same as for the preceding case.

• Suppose \( t \in T_{\text{Postint}} \). Then we replace \( p \) by \( \lambda(\text{int}(t)) \) and arc delay \( \delta(p, t) \) by \( \max \{ \delta(\text{int}(t), p) \} \) and then the arguments are the same as for the preceding case (using the lemma 14: \( m(\lambda(\text{int}(t))) + \max \{ \delta(\text{int}(t), p) \} = et(m, t) \)).

\[ \square \]

**Theorem 16 (TA simulates cDTPN).** Let \( N \) be a cDTPN with the set of all reachable marking \( \mathbb{M} \) from an initial marking \( m_0 \). Let \( S \) be the set of all states of TA(N). Then TA(N) simulates (branching) \( N \) w.r.t. the state relation.

**Proof.** Let \( m \in \mathbb{M} \) and \( (l, v) \in S \) such that \( (m, (l, v)) \in \beta \) for some \( b \in \{ mct(m), ft(m) \} \) (see Def. 22). Suppose \( \exists m' \in \mathbb{M}, t \in T : m \rightarrow m' \).

• First assume \( b = ft(m) \).
  
  – Suppose \( t \in T \setminus (T_{\text{Sync}} \cup T_{\text{Postint}}) \). Then \( \exists ! p \in ^*t : ft(m) = m(p) + \delta(p, t) \land \nu(p) = ft(m) - m(p) \). This implies \( \nu(p) = \delta(p, t) \).

  By the def. 21, the guard \( G(t) \) is the predicate \( p \geq \delta(p, t) \) and the invariant \( I(t) \) is a conjunction of predicates containing the predicate \( p \leq \delta(p, t) \). Hence there exists an enabled edge labeled with transition \( t \) and a state \( (l', v') \in S \) such that \( (l, v) \rightarrow_t (l', v') \).

  – Suppose \( t \in T_{\text{Sync}} \). Then we replace \( p \) by \( \lambda(t) \) and arc delay \( \delta(p, t) \) by \( \max \{ \delta(p, t) \mid p \in ^*t \} \) and then the arguments are the same as for the preceding case.

  – Suppose \( t \in T_{\text{Postint}} \). Then, by the state mapping relation \( \nu(\lambda(\text{int}(t))) = ft(m) - m(\lambda(\text{int}(t))) \). By the def. 21, both the guard and invariant are true, so there exists an enabled edge labeled with transition \( t \) and a state \( (l', v') \in S \) such that \( (l, v) \rightarrow_t (l', v') \).

By the transition relation of a timed automaton, \( \forall p \in t' \setminus P_{\text{int}} : v'(p) = 0 \) and \( \forall p \notin t' \setminus P_{\text{int}} : v'(p) = \nu(p) \). By the transition relation of a cDTPN \( \forall p \in t' \setminus P_{\text{int}} : m'(p) = ft(m) = mct(m') \) and \( \forall p \notin t' \setminus P_{\text{int}} : m'(p) = m(p) \). This means \( \forall p \in t' \setminus P_{\text{int}} : v'(p) + m'(p) = mct(m') \) and \( \forall p \notin t' \setminus P_{\text{int}} : v'(p) + m'(p) = \nu(p) + m(p) = ft(m) = mct(m') \).

So \( \forall p \in P \setminus P_{\text{int}} : v'(p) = mct(m') - m'(p) \) and therefore \( m', (l', v') \in \beta \).

• Now assume \( b < ft(m) \). Then by the lemma 15, for \( d = ft(m) - b \), we have \( (l, v) \rightarrow_d (l, v + d) \) and \( (m, (l, v + d)) \in \beta \) and then we are in the former case.

\[ \square \]

**Corollary 17 (Trace Inclusion).** Let \( N \) be a cDTPN with the set of all reachable markings \( \mathbb{M} \) from an initial marking \( m_0 \). Let \( (l_0, v_0) \) be the initial state of a timed automaton \( \text{TA}(N) \) with state space \( S \). Then
\[
\forall \sigma \in T^* : m_0 \xrightarrow{\sigma} ; \exists \tilde{\sigma} \in (T \cup \mathbb{N})^* : (l_0, v_0) \rightarrow_{\tilde{\sigma}} \land \tilde{\sigma} |_{T} = \sigma
\]

21
Lemma 18 (cDTPN simulates TA). Let $N$ be a cDTPN such that $P_{\text{int}} = \emptyset$. Then $N$ can simulate (branching) TA($N$) with respect to the state relation.

**Proof.** Note that $P_{\text{int}} = \emptyset \Rightarrow T_{\text{preint}} = \emptyset \land T_{\text{postint}} = \emptyset$. Let $M$ be the set of all reachable timed markings from an initial marking $m_0$ and $S$ be the state space of TA($N$). Let $m \in M$ and $(l,v) \in S$ such that $(m,(l,v)) \in \beta$.

- Suppose $\exists (l',v') \in S, t \in T : (l,v) \rightarrow_t (l',v')$. Then by the Def. 21, transition $t$ is classically enabled.

- If $t \notin T_{\text{SYNC}}$ then by the state invariant and guards of TA($N$), $\exists p \in *t$ and $v(p) = \delta(p,t)$. Note that the state invariant guarantees that no edge corresponding a transition can fire earlier. Hence by the state relation $v(p) = b - m(p)$, where $b \in [mct(m),ft(m)]$. So $b = m(p) + \delta(p,t)$, but this means that $b \geq ft(m)$. Hence $b = ft(m)$ and $t$ is also enabled in the marking $m$. If $t \in T_{\text{SYNC}}$, then we reuse the preceding arguments after replacing place $p$ by $\lambda(t)$ and arc delay $\delta(p,t)$ by $\text{max}\{\delta(p,t) \mid p \in *t\}$.

Hence $\forall p \in P : m(p) \neq \infty \Rightarrow v(p) = ft(m) - m(p)$ and $\exists m' \in \bar{M} : m \parallel m'$.

By the transition relation of a timed automaton, $\forall p \in *t^* : v'(p) = 0$ and $\forall p \notin *t^* : v'(p) = v(p)$. By the transition relation of a cDTPN $\forall p \in *t^* : v'(p) = m'(p) = mct(m')$ and $\forall p \notin *t^* : m'(p) = m(p)$. This means $\forall p \in *t^* : v'(p) + m'(p) = mct(m')$ and $\forall p \notin *t^* : v'(p) + m'(p) = v(p) + m(p) = ft(m) = mct(m')$. So $\forall p \in P : v'(p) = mct(m') - m(p)$ and therefore $(m',(l',v')) \in \beta$.

- Suppose $\exists d \in \mathbb{N} : (l,v) \nrightarrow (l,v+d)$. Then by the lemma 15, for $d = ft(m) - b$, we have $(l,v) \nrightarrow (l,v + d)$ and $(m,(l,v + d)) \in \beta$ and then we are in the former case.

□

Corollary 19 (Branching Bisimulation). Let $N$ be a cDTPN such that $P_{\text{int}} = \emptyset$. Then $N$ and TA($N$) are branching bisimilar with respect to the state relation.

In the fig. 9, we show why in general there is no bisimulation between a cDTPN and its timed automaton. This is because the choices of timestamps of produced tokens (lying possibly in the future) are made by the transition producing them, while in the corresponding timed automaton, the choice is made in the state reachable after taking an edge labeled with this transition.

However, we still have trace equivalence between a cDTPN and its timed automaton for finite traces.

Theorem 20 (Trace Equivalence). Let $N$ be a cDTPN with the set of all reachable timed markings $M$ from the initial marking $m_0$. Let $(l_0,v_0)$ be the initial state of a timed automaton TA($N$) such that $(m_0,(l_0,v_0)) \in \beta$ and mct($m_0$) = 0 and $\forall p \in l_0 : v_0(p) = 0$. Then the following holds:

(a) $\forall \sigma \in T^* : m_0 \overset{\sigma}{\rightarrow} : \exists \tilde{\sigma} \in (T \cup \mathbb{N})^* : (l_0,v_0) \rightarrow_{\tilde{\sigma}} \sigma$

(b) $\forall \tilde{\sigma} \in (T \cup \mathbb{N})^* : (l_0,v_0) \rightarrow_{\tilde{\sigma}} : \exists \sigma \in T^* : m \overset{\sigma}{\rightarrow} \land \sigma = \tilde{\sigma}$

**Proof.** By the Corollary 19, (a) holds.

Consider a trace $\tilde{\sigma} \in (T \cup \mathbb{N})^*$ such that $\tilde{\sigma} = (d_0,t_1,d_1,t_2,\ldots,t_n)$ and

$$(l_0,v_0) \overset{d_0}{\rightarrow} (l_0,v_0 + d_0) \rightarrow_{t_1} (l_1,v_1) \overset{d_1}{\rightarrow} (l_1,v_1 + d_1) \rightarrow_{t_2} \cdots \rightarrow_{t_n} (l_n,v_n)$$
We will prove by induction that

\[
\exists m_1, \ldots, m_n \in M : \forall k \in \{1, \ldots, n\} : \\
(m_k, (l_k, v_k)) \in \beta \land d_0 = ft(m_0) \land d_k = ft(m_k) - ft(m_{k-1}) \land m_{k-1} \xrightarrow{t_k} m_k \land \\
t_k \in T_{\text{preint}} \Rightarrow \forall p \in t_k^* : m_k(p) = \sum_{i=0}^{j} d_i \\
\]

where, \( j \geq k \) is the first index in \( \sigma \) such that \( t_j \in p^* \) or \( j = n \).

Note that the choice of the value to be picked from a delay set corresponding an outgoing arc in a cDTPN is made by looking ahead in the firing sequence \( \hat{\sigma} \) of TA(N) to find out the time the token must be consumed in the cDTPN. If this cannot be determined by looking ahead in the firing sequence \( \sigma \), then we choose the value to be picked as equal to the sum of all delay transitions in the firing sequence \( \hat{\sigma} \).

Consider \( k = 1 \). Note \( (m_0, (l_0, v_0)) \in \beta \) and \( mct(m_0) = 0 \) and \( \forall p \in l_0 : v_0(p) = 0 \). By the lemma 15, \( (m_0, (l_0, v_0 + d_0)) \in \beta \), where \( d_0 = ft(m_0) \).

We have in the timed automaton \( (l_0, v_0 + d_0) \xrightarrow{\tau_1} (l_1, v_1) \). Since \( \forall p \in P_{\text{int}} : m_0(p) = \infty \), transition \( \tau_1 \notin T_{\text{postint}} \). Hence, we may use the arguments for transition enabling as in the lemma 18 and then transition \( \tau_1 \) is enabled in the marking \( m_0 \).

If \( \tau_1 \in T \setminus T_{\text{preint}} \) then \( \exists m_1 \in M : m_0 \xrightarrow{\tau_1} m_1 \). If \( \tau_1 \in T_{\text{preint}} \) then we apply the construct of the induction hypothesis to determine for each \( p \in t_1^* : m_1(p) = \sum_{i=0}^{j} d_i \), where \( j \geq 1 \) is the first index such that \( t_j \in p^* \) or \( j = n \).

By the arguments of the lemma 18, \( (m_1, (l_1, v_1)) \in \beta \) holds.

Suppose the statement holds for all \( k \leq i \). Consider \( k = i + 1 \). Then by the induction hypothesis \( (m_i, (l_i, v_i)) \in \beta \) and \( d_i = ft(m_i) - ft(m_{i-1}) \) and \( ft(m_i) = \sum_{j=0}^{i} d_j \). By the lemma 15, the relation \( (m_i, (l_i, v_i + d_i)) \in \beta \) holds.

We have in the timed automaton \( (l_i, v_i + d_i) \xrightarrow{\tau_{i+1}} (l_{i+1}, (l_{i+1}, v_{i+1})) \).

If \( \tau_{i+1} \in T \setminus T_{\text{postint}} \) we may use the arguments for transition enabling as in the lemma 18 and then transition \( \tau_{i+1} \) is enabled in the marking \( m_i \). If \( \tau_{i+1} \in T_{\text{postint}} \) then \( m_i(p) \neq \infty \), where \( p \in \ast \tau_{i+1} \cap P_{\text{int}} \). By the induction hypothesis there is a greatest \( x \leq i \) such that \( \tau_x \in p^* \) and so \( \tau_x \in T_{\text{preint}} \) and \( m_x(p) = \sum_{j=0}^{x} d_j \). Since \( ft(m_i) = \sum_{j=0}^{i} d_j = m_i(p) \), transition \( \tau_{i+1} \) is enabled in the marking \( m_i \).

Next we show that the state relation holds. If \( \tau_{i+1} \notin T_{\text{preint}} \) then \( \exists m_{i+1} \in M : m_i \xrightarrow{\tau_{i+1}} m_{i+1} \). If \( \tau_{i+1} \in T_{\text{preint}} \) then
we apply the construct of the induction hypothesis to determine for each \( p \in t_{i+1}^* : m_{i+1}(p) = \sum_{j=0}^{i} d_i \), where \( j \geq i + 1 \) is the first index such that \( t_j \in p^* \) or \( j = n \).

By the arguments as in the lemma 18, \( (m_{i+1}, (t_{i+1}, v_{i+1})) \in \beta \) holds.

\[\square\]

6. Expressiveness of DTPN

In this section, we use two variants of sDTPN, one satisfying In-single, Out-zero (input delays only (IDO)) and the other satisfying In-zero, Out-single (output delays only (ODO)). We have seen already that sDTPN is simulation equivalent with a general DTPN. First, we show that sDTPN with IDO and sDTPN with ODO are capable of modeling inhibitor arcs, which means that they are Turing complete. Hence the model class DTPN is Turing complete. We prove this claim by constructing for an arbitrary Petri net with inhibitor arcs, an sDTPN model with IDO and an sDTPN model with ODO that is branching bisimilar to it. Thus, we show that the model classes of DTPN with IDO and ODO have the same expressive power. That does not mean however that they have the same modeling comfort, i.e. the ease of modeling. In order to explore this, we construct for an sDTPN with IDO, an sDTPN model with ODO that is bisimilar and vice versa. Finally, we also show how different model classes (i.e. \( M_1 \) and \( M_2 \)) can be expressed using model class \( M_3 \).

6.1. Expressing inhibitor nets with sDTPN

We will first consider the case of sDTPN’s with IDO. We start with a construction. Given an inhibitor net \( N \) we construct an sDTPN \( N' \) with IDO as in Fig. 10. We add a place called Tick and we replace each inhibitor arc of \( N \) by a simple sub-net consisting of one inhibitor place \( S \) and one inhibitor transition \( T \) (which will be the silent transition). The extra place Tick is connected to all original transitions with input and output arcs. Tick always contains one token. \( S \) is connected with input and output arcs to inhibitor transition \( A \) and also with \( T \). Further \( T \) is connected with an input and an output arc to \( Q \) the inhibitor place. All original arcs between a place and a transition get input delay 0. All input arcs from place Tick get a delay 2, the input arc from \( S \) to \( T \) gets a delay 1 and from \( S \) to \( A \) a delay 2. In the initial marking there are tokens in \( S \) and Tick with timestamps 0. Let the initial marking of \( N \) be \( m_0 \), then the initial marking \( m'_0 \) of \( N' \) has for each untimed token in \( m_0 \) a timed token in \( m'_0 \) in the same place with a timestamp 0. Note that all transitions, except for the ones that were added during the construction may have more input and output arcs. We will formally define the construction as a refinement of inhibitor arcs.

Given a DTPN \((P,T,F,\delta)\) with a timed marking \( m \in \mathbb{M} \), we will denote the set of all enabled transitions by \( \text{enabled}(m) = \{ t \in T \mid \text{en}(m,t) \leq \text{ft}(m) \} \) and assert the existence of at least one token in a place \( p \in P \) in a marking \( m \in \mathbb{M} \) by the predicate \( \text{marked}(m,p) \). To retrieve the number of tokens in a place having a particular timestamp in a given marking, we will define a function, \( \nu : \mathbb{M} \times P \times \mathbb{Q} \to \mathbb{N} \) that satisfies \( \nu(m,p,q) = | \{ i \in \text{dom}(m) \mid \pi(m(i)) = p \land \tau(m(i)) = q \} | \). To retrieve the number of tokens in a place, irrespective of their timestamps, we overload this function as follows: \( \nu : \mathbb{M} \times P \to \mathbb{N} \) is defined as \( \nu(m,p) = \sum_{q \in \mathbb{Q}} \nu(m,p,q) \).

Note that since all places \( p \in P, \text{ and tick are safe, we will use the notion of marking as in the Def. 19. A transition } t \in T \setminus T_r \text{ is called an inhibitor transition if } \exists p \in P : (t,p) \in I, \text{ otherwise, we call it a non-inhibitor transition. Furthermore, given a DTPN with a marking } m \in \mathbb{M}, \text{ and a transition } t \in T, \text{ we will assert that transition } t \text{ is classically enabled by the predicate } \text{ce}(m,t) \text{ iff } \forall p \in \ast t : \exists i \in \text{dom}(m) : \pi(m(i)) = p. \)

**Definition 24 (Inhibitor Arc Refinement with Input Delays).** Let \( N = (P,T,F,\delta) \) be an inhibitor net with an initial marking \( m_0 \). Let \( I = \{(t,p) \in T \times P \mid p \in t(t)\} \). Let tick \( \notin P \) be a tick place. Let \( T_s = \{ \tau_{tp} \mid (t,p) \in I \} \) be the set of silent transitions such that \( T \cap T_s = \emptyset \). Let \( P_s = \{ \tau_{tp} \mid (t,p) \in I \} \) such that \( P \cap P_s = \emptyset \). The inhibitor arc refinement with input delays only of net \( N \) is a DTPN with IDO denoted by \( \psi_t(N) = (P',T',F',\delta) \) with an initial marking \( m_0, \)
where

\[ P' = P \cup \{\text{tick}\} \cup P_s \]
\[ T' = T \cup T_s \]
\[ F' = F \cup \{(t, \pi tp) \mid (t, p) \in I\} \cup \{(\pi tp, t) \mid (t, p) \in I\} \]
\[ \{(\pi tp, \tau tp) \mid (t, p) \in I\} \cup \{\tau tp, \pi tp \mid (t, p) \in I\} \]
\[ \{\text{tick}, t \mid t \in T\} \cup \{(t, \text{tick}) \mid t \in T\} \]

The function \( \delta \) is defined as \( \forall (t, p) \in I : \delta(\pi tp, t) = 2 \wedge \delta(\pi tp, \tau tp) = 1 \) and \( \forall t \in T : \delta(\text{tick}, t) = 2 \) and all other arcs in \( F' \) have a zero delay. Furthermore, the initial marking \( m_0 \) satisfies \( \forall p \in P : \overline{m_0}(p) = \nu(m_0, p) \) and \( \forall (t, p) \in I : \nu(m_0, \pi tp) = \nu(m_0, \text{tick}) = 1 \) and \( \forall i \in \text{dom}(m_0) : \tau(m_0(i)) = 0 \).

Note that since all places \( p \in P_s \) and tick are safe, we will use the notion of marking as in the Def. 19.

We make a few observations:

- \( \forall m \in M : m(\text{tick}) \) is even, since in the initial marking the token in tick has a timestamp zero and all outgoing arcs have a fixed delay of value two. Note that incoming arcs to tick have a zero delay.

- \( \forall m \in M, t \in T \setminus T_i : t \in \text{enabled}(m) \Rightarrow \text{ft}(m) = 2k \) for some \( k \in \mathbb{N} \), i.e. non-silent transitions fire only at even time points. This is because

  - all transitions \( t \in T \setminus T_s \) such that \( \exists p \in P : (t, p) \in I \) are delayed only by the token in place tick whose timestamp is always even.

  - all transitions \( t \in T \setminus T_s \) such that \( \exists p \in P : (t, p) \in I \) are delayed by the tokens in places tick and \( \pi tp \). The only way transition \( t \) can fire at \( 2k + 1 \) is if \( m(\pi tp) = 2k + 1 > m(\text{tick}) \).

So if transition \( t \) is delayed by a token in \( \pi tp \) until \( 2k + 1 \) then place \( p \) was marked at \( 2k - 1 \) and only at time \( 2k \) place \( p \) could become unmarked, thereby updating the token in the tick place to \( 2k \). Hence transition \( t \) can only fire at \( 2k + 2 \) and not at \( 2k + 1 \).
If a marking $m \in M$ and an inhibitor transition $t \in T \setminus T_s$ such that $(t, p) \in I$ for some $p$ satisfy $ce(m, t)$ and $\neg \text{marked}(m, p)$ and $m(\text{tick}) = 2k$, then $m(\pi_{tp}) \leq 2k$. Since if $m(\pi_{tp}) = 2k + 1$ then transition $\tau_{tp}$ fired at $2k + 1 - 2 = 2k - 1$ and transition $t$ will fire at $2k + 1$. But the latest time that place $p$ could become unmarked by a non-silent transition is at $2k - 2$ and then $m(\pi_{tp}) = 2k$ which means $t$ will fire at $2k$, which is a contradiction to the conclusion that $t$ will fire at $2k + 1$.

Next, we will consider the case of sDTPN’s with ODO, which is slightly more complex. We start with a construction shown in Fig. 11. For each inhibitor arc, like the arc between an inhibitor transition $A$ and a place $Q$, we replace this arc by a sub-net with four places $S_1, S_2, S_3$ and $S_4$, and two transitions $T_1$ and $T_2$ with double arcs (input and output) between $S_1$ and $A$, $S_2$ and $A$, $S_1$ and $T_1$, $S_2$ and $T_2$ and between $T_1$ and $Q$ and $T_2$ and $Q$. Further $T_1$ is the only input transition and $T_2$ the only output transition for $S_4$. Similarly $T_2$ is the only input transition and $T_1$ the only output transition for $S_3$. We call $T_1$ and $T_2$ the inhibitor transitions and $S_1, \ldots, S_4$ the inhibitor places. Like in the former case, all transitions of the original model are connected with double arcs with a new place called $\text{tick}$. All original output arcs get a delay 0, the output arcs from $T_1$ to $S_1$ and from $T_2$ to $S_2$ and all non-inhibitor transitions connected to place $\text{tick}$ get an output delay of value 2. The output arcs from $T_1$ to $S_4$ and from $T_2$ to $S_3$ get a delay 1. Places $\text{tick}$, $S_1$, $S_2$ and one of the places $S_3$ or $S_4$ are marked with one token in the initial marking. All tokens have initially a timestamp 0 except for the token in place $\text{tick}$ having a timestamp 2.

**Definition 25 (Inhibitor Arc Refinement with Output Delays).** Let $N = (P, T, F, i)$ be an inhibitor net with an initial marking $\bar{m}_0$. Let $I = \{(t, p) \in T \times P \mid p \in \nu(t)\}$. Let $\text{tick} \notin P$ be a tick place. Let $T_s = \{\tau_{tp}^j \mid (t, p) \in I \land j \in [1, 2]\}$ be the set of silent transitions such that $T \cap T_s = \emptyset$. Let $P_s = \{\pi_{tp}^j \mid (t, p) \in I \land j \in [1, 4]\}$ such that $P \cap P_s = \emptyset$. The inhibitor arc refinement with output delays only of net $N$ is a DTPN with ODO denoted by $\phi_o(N) = (P', T', F', \delta)$.
with an initial marking \( m_0 \), where
\[
P' = P \cup \{\text{tick}\} \cup P_s
\]
\[
T' = T \cup T_s
\]
\[
F' = F \cup \{\tau^1_{ip}, \pi^1_{ip}\} \cup \{\pi^4_{ip}, \tau^2_{ip}\} \cup \{\tau^2_{ip}, \pi^3_{ip}\} \cup \{\pi^3_{ip}, \tau^1_{ip}\} \cup
\[
\{(t, \pi^1_{ip}) : (t, p) \in I \land j \in [1, 2] \}\cup
\{(\pi^4_{ip}, \tau^2_{ip}) : (t, p) \in I \land j \in [1, 2] \}\cup
\{(\tau^2_{ip}, \pi^3_{ip}) : (t, p) \in I \land j \in [1, 2] \}\cup
\{(\pi^3_{ip}, \tau^1_{ip}) : (t, p) \in I \land j \in [1, 2] \}\cup
\{(\text{tick}, t) : t \in T \}\cup \{(t, \text{tick}) : t \in T\}.
\]
Function \( \delta \) is defined as follows: \( \forall (t, p) \in I : \delta(\pi^1_{ip}, \pi^4_{ip}) = \delta(\pi^2_{ip}, \pi^3_{ip}) = 2 \land \delta(\pi^1_{ip}, \pi^3_{ip}) = \delta(\pi^2_{ip}, \pi^1_{ip}) = 1 \); \( \forall t \in T : \delta(t, \text{tick}) = 2 \), and all other arcs in \( F' \) have a zero delay. Furthermore, the initial marking \( m_0 \) satisfies \( \forall p \in P : \hat{m}_0(p) = \nu(m_0, p) \) and \( \forall (t, p) \in I : \nu(m_0, \pi^4_{ip}) = \nu(m_0, \text{tick}) = 1 \) and all tokens have a timestamp zero except for the token in place tick having a timestamp two.

We make a few observations:

- \( \forall m \in M : m(\text{tick}) \) is even, since in the initial marking the token in tick has a timestamp two and all incoming arcs have a fixed delay of value two. Note that outgoing arcs from tick have a zero delay.
- \( \forall m \in M, t \in T \setminus T_s : t \in \text{enabled}(m) \Rightarrow \text{fit}(m) = 2k \) for some \( k \in \mathbb{N} \), i.e. non-silent transitions fire only at even time points. This is because
  - all non-inhibitor transitions \( t \in T \setminus T_s \) are delayed only by the token in place tick whose timestamp is always even.
  - all inhibitor transitions \( t \in T \setminus T_s \) are delayed by the tokens in places tick, \( \pi^1_{ip} \) and \( \pi^2_{ip} \). The only way transition \( t \) can fire at \( 2k + 1 \) is if \( \text{max}(m(\pi^1_{ip}), m(\pi^2_{ip})) = 2k + 1 > m(\text{tick}) \). So if transition \( t \) is delayed either by the token in \( \pi^1_{ip} \) or \( \pi^2_{ip} \) until \( 2k + 1 \) then place \( p \) was marked at \( 2k - 1 \) and at time \( 2k \) place \( p \) could become unmarked, thereby updating the token in the tick place to \( 2k + 2 \). Hence transition \( t \) can only fire at \( 2k + 2 \) and not at \( 2k + 1 \).
- If a marking \( m \in M \) and an inhibitor transition \( t \in T \setminus T_s \) for some \( (t, p) \in I \) satisfies \( \text{ce}(m, t) \) and \( \neg\text{marked}(m, p) \) and \( m(\text{tick}) = 2k \) then \( \text{max}(m(\pi^1_{ip}), m(\pi^2_{ip})) \leq 2k \). Since if \( \text{max}(m(\pi^1_{ip}), m(\pi^2_{ip})) = 2k + 1 \) then transition \( \tau^1_{ip} \) or \( \tau^2_{ip} \) fired at \( 2k + 1 - 2 = 2k - 1 \) and transition \( t \) will fire at \( 2k + 1 \). But the latest time that place \( p \) could become unmarked by a non-silent transition is at \( 2k - 2 \) and then \( \text{max}(m(\pi^1_{ip}), m(\pi^2_{ip})) = 2k \) which means \( t \) will fire at \( 2k \). This is a contradiction.

Since the case of ODO is more complex than IDO, we will first prove the inhibitor property for the case of ODO.

**Theorem 21.** Let \( N \) be an inhibitor net and \( \varphi_o(N) \) be its DTPN with ODO having an initial marking \( m_0 \).
Then \( \forall m \in \mathcal{R}(N, m_0), (t, p) \in I \) satisfies

\[(i) \text{marked}(m, p) \Rightarrow t \notin \text{enabled}(m)\]
\[(ii) \neg\text{marked}(m, p) \land \text{ce}(m, t) \Rightarrow \text{et}(m, t) = \text{fit}(m) \lor \text{enabled}(m) \cap (T \setminus T_s) = \emptyset\]

**Proof.** Consider a marking \( m \in \mathcal{R}(N, m_0) \) and \( (t, p) \in I \). There are three cases to consider.

- Suppose \( \text{fit}(m) = \infty \). Then \( t \notin \text{enabled}(m) \). If \( \neg\text{marked}(m, p) \) then silent transitions \( \tau^1_{ip} \) or \( \tau^2_{ip} \) cannot fire, so transition \( t \) cannot be delayed. Hence \( t \) is not classically enabled.
- Suppose \( \text{fit}(m) = 2k + 1 \) for some \( k \in \mathbb{Z} \). Since transition \( t \) can fire only at even time points, \( t \notin \text{enabled}(m) \). However transition \( \tau^1_{ip} \) or \( \tau^2_{ip} \) can fire, in which case \( \text{marked}(m, p) \).
• Suppose \( ft(m) = 2k \) and \( t \) is classical enabled. We consider four cases:
  
  - If marked\((m, p)\) and \( p \) became marked at time \( 2k \). Then only silent transitions \( t' \in T_s \) can fire.
  
  - If marked\((m, p)\) and \( p \) became marked latest at time \( 2k - 2 \). Then one of the silent transitions \( t^1_{ip} \) or \( t^2_{ip} \) might have fired is at \( 2k - 2 \) and at \( 2k - 1 \), thereby delaying transition \( t \) until \( 2k + 1 \).
  
  - If \( \neg\text{marked}(m, p) \) and ce\((m, t)\) and the place \( p \) became unmarked at time \( 2k \) by firing a transition \( t' \in T \setminus T_s \). Then the place tick has a token timestamp \( 2k + 2 \), so only silent transitions that are enabled may fire at time \( 2k \).
  
  - \( \neg\text{marked}(m, p) \) and ce\((m, t)\) and the place \( p \) became unmarked latest at time \( 2k - 2 \) by firing a transition \( t' \in T \setminus T_s \). This means that the latest firing time of a transition \( t^1_{ip} \) or \( t^2_{ip} \) is at most \( 2k - 2 \). So \( t \) is delayed until time \( 2k \).

\[\square\]

For simulating inhibitor nets with sDTPN satisfying IDO we have a similar result.

**Theorem 22.** Let \( N \) be an inhibitor net and \( \phi(N) \) be its DTPN with IDO having an initial marking \( m_0 \). Then \( \forall m \in R(N, m_0), (t, p) \in I \) satisfies

\[
\begin{align*}
(i) & \quad \text{marked}(m, p) \Rightarrow t \not\in \text{enabled}(m); \\
(ii) & \quad \neg\text{marked}(m, p) \land \text{ce}(m, t) \Rightarrow \text{et}(m, t) = ft(m) \lor \text{enabled}(m) \cap (T \setminus T_s) = \emptyset.
\end{align*}
\]

The proof is similar to the proof of theorem 21, since the only difference is that the token in the places \( P_s \) are always ready for consumption, because tokens in these places have a timestamp at most equal to the current time of the system. So for each arc \( (t, p) \in I \), one silent transition \( t_{ip} \) and a place \( p_{ip} \) is sufficient to delay transition \( t \) by two time units in each time step, if the place \( p \) is marked. So we replace occurrences of silent transitions \( t^1_{ip} \) and \( t^2_{ip} \) with \( t_{ip} \) in the proof of theorem 21.

**Definition 26 (Marking relation for ODO).** Let \( N \) be an inhibitor net with an initial marking \( m_0 \) and \( N' = \psi_o(N) \) be its DTPN with IDO having an initial marking \( m_0' \). Let \( M = R(N, m_0) \) be the set of all reachable untimed markings of \( N \) and \( M' = R(N', m_0) \) be the set of all reachable timed markings of \( N' \). The marking relation for ODO denoted by \( \sim_o \subseteq M \times M' \) satisfies \( m \sim_o m' \) iff \( m' \in M' \lor \forall p \in P_N : m(p) = v(m', p) \).

By replacing \( N' = \psi_o(N) \) in the Def. 26, we define the marking relation for IDO denoted by \( \sim_i \).

**Theorem 23 (Branching Bisimulation - ODO).** Let \( N \) be an inhibitor net. Then \( N \) and \( \psi_o(N) \) are branching bisimilar with respect to \( \sim_o \).

**Proof.** (sketch) If a transition \( t \) in \( N \) such that \( \sigma(t) = \emptyset \) is enabled then the same transition is also enabled in \( \psi_o(N) \) because of the marking relation but with a possible delay induced by the token in the tick place.

If a transition \( t \) in \( N \) such that \( \sigma(t) \neq \emptyset \) is enabled then all places in \( \sigma(t) \) are unmarked, and due to the theorem 21, transition \( t \) can also fire in the net \( \psi_o(N) \) (possibly delayed).

If an inhibitor transition \( t \) in the net \( \psi_o(N) \) is enabled, then due to the theorem 21, all inhibitor places of \( t \) are unmarked and so \( t \) can fire in the net \( N \) as well. For all other enabled non-silent transitions in the net \( \psi_o(N) \), the same transitions are also enabled in the net \( N \) because of the marking relation. If a silent transition \( \tau \in T_s \) is enabled in the net \( \psi_o(N) \) then the firing of such a transition is not changing the distribution of tokens in places but only delaying its corresponding inhibitor transition.

**Theorem 24 (Branching Bisimulation - IDO).** Let \( N \) be an inhibitor net. Then \( N \) and \( \psi_i(N) \) are branching bisimilar with respect to \( \sim_i \).
The proof is similar to the theorem 23, since by the theorem 22, the net $\psi_i(N)$ is having the desired inhibitor behavior and firing silent transitions is not changing the distribution of tokens in the net.

Since it is possible to express inhibitor arcs, using an sDTPN, model class $M3$ is Turing complete.

**Corollary 25.** The model class $M3$ is Turing complete.

We will now extend the class of DTPN with a timed variant of inhibitor arcs. The timed inhibitor arc is allowing a transition to be enabled only if the connected place is empty or has tokens with timestamps greater than the current time. We will call this class **timed inhibitor nets**.

**Definition 27 (Timed Inhibitor nets).** A **timed inhibitor net** is a tuple $(P, T, F, \delta, \iota)$, where $(P, T, F, \delta)$ is a DTPN and $\iota : T \rightarrow P$ is the set of inhibitor arcs.

**Definition 28 (Transition Relation of a Timed Inhibitor net).** The transition relation of a timed inhibitor net satisfies Def. 7 and moreover, for any inhibitor transition $t \in T$ and markings $m, m' \in \mathcal{M}$ such that $m \xrightarrow{t} m'$ is satisfying $\forall p \in \iota(t) : \neg \exists i \in \text{dom}(m) : \pi(m(i)) = p \land \tau(m(i)) \leq \tau(t(m))$.

![Figure 12: Counter Example for simulating inhibitors in DTPN](image)

An obvious question arises: Using the construction techniques described above, is it possible to simulate an inhibitor arc in a timed inhibitor net? The answer is no! This is because the interleaving of non-silent transitions caused by the token the place tick is destroying the simulation property. We give an example in the Fig. 12, where the firing sequence $\langle A; C; D; B \rangle$ is not possible in the system with place tick.

However for two cases we are able to simulate timed inhibitor arcs.

For inhibitor transitions connected to a safe place, we use the well-known construct as in the Fig. 13. All delays on arcs with $Q$ are 0. (In case $P$ is bounded by some $n$ we need multiple arcs or arc weights).

In case place $P$ is not bounded and the net system satisfies in each step of every executable firing sequence that the time is strictly increasing, then we may reuse the construction defined for simulating untimed inhibitor arcs after dropping the place tick and choosing a value for $\varepsilon$ such that $\varepsilon < \Delta$, where $\Delta$ is the smallest arc delay of the system as shown in the Fig. 13. Note that (a) if place $R$ at time $X$ has a token with timestamp less than or equal to $X$ then transition $A$ is inhibited, (b) if transition $A$ is enabled at $X$ then it might be delayed until $X + \varepsilon$.

In the next section, we will compare the two variants of sDTPN, namely input delays only and output delays only, by giving a construction to translate one variant into another while preserving branching bisimilarity.

### 6.2. Modeling ODO with IDO

In the Fig. 14, we present an example transformation of output delays into input delays. The transformation is in fact just a refinement of outgoing arcs, where for each outgoing arc from a transition, the corresponding output delay becomes the input delay of the newly introduce silent transition. We will formally define the refinement of outgoing arcs.
Definition 29 (Output Arc Refinement of a DTPN). Let $N = (P, T, F, \delta)$ be a DTPN. Let $\bar{F} = F \cap (T \times P)$ be the set of all outgoing arcs. Let $\bar{P}$ be the set of intermediate places such that $\bar{P} \cap P = \emptyset$ and $|\bar{P}| = |\bar{F}|$. Let $\bar{T}$ be the set of $\tau$-labeled transitions such that $\bar{T} \cap T = \emptyset$ and $|\bar{T}| = |\bar{F}|$. Let $\lambda : \bar{F} \to \bar{P} \times \bar{T}$ be a function that assigns to every outgoing arc, an intermediate place and a $\tau$-labeled transition. We define two standard projection functions $\pi_1$ and $\pi_2$ over $\bar{P} \times \bar{T}$ such that $\pi_1(q, \tau) = q$ and $\pi_2(q, \tau) = \tau$. The output arc refinement of a DTPN $N$ is denoted by $\gamma_o(N) = (P', T', F', \delta')$, where

- $P' = P \cup \bar{P}$,
- $T' = T \cup \bar{T}$,
- $F' = (F \setminus \bar{F}) \cup \{(t, \pi_1(\lambda(t, p))) \mid (t, p) \in \bar{F}\} \cup \{(\pi_2(\lambda(t, p)), \pi_2(\lambda(t, p))) \mid (t, p) \in \bar{F}\}$,
- $\forall (x, y) \in F \setminus \bar{F} : \delta'(x, y) = \delta(x, y)$, i.e. original arcs have the same delay,
- $\forall (t, p) \in \bar{F} : \delta'(\pi_1(\lambda(t, p)), \pi_2(\lambda(t, p))) = \delta(t, p)$, i.e. move the delay from output arc to input arc of the silent transition,
- $\forall (t, p) \in \bar{F} : \delta'(t, \pi_1(\lambda(t, p))) = \delta'(\pi_2(\lambda(t, p)), p) = 0$, i.e. all other arc delays in the newly introduced subnet are equal to zero.

Definition 30 (Marking Relation ODO-IDO). Let $N = (P, T, F, \delta)$ be a DTPN with the set of all reachable timed markings $\bar{M}$ from an initial marking $m_0$. Let DTPN $\gamma_o(N)$ have the set of all reachable timed markings $\bar{M}$ from an initial marking $m'_0$. A Marking Relation ODO-IDO, $\phi \subseteq \bar{M} \times \bar{M}$ is defined $\forall m \in \bar{M}, \bar{m} \in \bar{M} : (m, \bar{m}) \in \phi$ if and only if $\forall p \in P, x \in Q$ such that

$$v(m, p, x) = v(\bar{m}, p, x) + \sum_{t \in T} v(\bar{m}, \pi_1(\lambda(t, p)), x - \delta(t, p))$$

Theorem 26. Let $N$ be a DTPN. Then $\gamma_o(N)$ and $N$ are branching bisimilar with respect to $\phi$.  

Figure 13: Simulating timed inhibitors
We sketch the proof of branching bisimilarity with respect to the relation $\phi$: For each token in a place of the original model, there exists a token in the arc refined net, either in the corresponding place having the same timestamp, or in the pre-place of a silent transition that is producing a token in the corresponding place and having a timestamp less by a factor of the input delay of that silent transition. This means, when a token in a place becomes available in the original model, a token with the same timestamp is also available in the corresponding place of the arc refined net, otherwise a silent transition is enabled, firing which leads us to the previous case. Hence the same transitions can be fired in both markings.

6.3. **Modeling IDO as ODO**

We use Fig. 15 to explain the construction. Consider a subnet consisting of a place $P$ having transitions $A_1, \ldots, A_n$ in
its postset with incoming delays $x_1, \ldots, x_n$. Note that place $P$ may have arbitrary additional input arcs and transitions $A_1, \ldots, A_n$ may have arbitrary additional input and output arcs. We order them, if possible, such that their input delays $x_1, \ldots, x_n$ are strictly non-decreasing. We replace all the arcs between place $P$ and transitions $A_1, \ldots, A_n$ by a subnet having one transition $T_0$ (silent transition) that consumes the tokens from $P$ and puts them in $a_1, \ldots, a_n$ with corresponding output delays $x_1, \ldots, x_n$, respectively. The rest of the subnet is about garbage collection, i.e., as soon as one of the transitions $A_1 \ldots A_n$ can fire the others should become disabled. If $A_n$ fires it just consumes the tokens for all the others because they are ready for consumption. But if $A_1$ fires then it can’t consume the tokens for the others, so it triggers silent transitions $T_2, \ldots, T_n$ by putting tokens in the places $S_2, \ldots, S_n$. Note that transitions $T_2, \ldots, T_n$ are able to fire without delay induced by arcs. Note that the net derived in this way is a timed inhibitor net and that timed inhibitors can be simulated by a DTPN for the two cases of bounded and unbounded places as presented before.

It is straightforward to formalize this construct and we omit a formal proof of the following theorem.

**Theorem 27.** Let $N$ be a DTPN. Then $\gamma(N)$ and $N$ are branching bisimilar.

We will sketch the proof using the Fig. 15. For each token in the place $P$ of the original net, either there exists a token in place $P$ of the constructed net having the same timestamp, or there exists one token in each place $a_1, \ldots, a_n$ having timestamps increased by delays $x_1, \ldots, x_n$, respectively (by firing silent transition $T_0$). If a transition $A_k$ for some $k < n$, is firing, then it is consuming one available token from each place $a_1, \ldots, a_k$ and producing one token each in the place $S_{k+1}, \ldots, S_n$, thereby disabling transitions $A_{k+1}, \ldots, A_n$ (due to inhibitor arcs) until silent transitions (garbage collectors) $T_{k+1}, \ldots, T_n$ have fired. If silent transition $T_0$ is firing again while transitions $A_{k+1}, \ldots, A_n$ are inhibited, then one token is produced in each place $a_k, \ldots, a_n$ having timestamps greater than equal to the maximal timestamp of tokens in that place. As garbage collector $T_k$ is delayed only by the token in the place $a_k$, transition $T_k$ is consuming the earliest token in the place $a_k$. So the correct tokens are collected. In case $A_n$ has fired, then it is able to consume all tokens from places $a_1, \ldots, a_n$, so does not need to enable any garbage collector. This means for every token produced by the transition $T_0$, one token is consumed by a transition $A_k$ and the rest are consumed by their respective garbage collectors as soon as the token becomes available. So if a transition $A_k$ for some $k \in \{1, \ldots, n\}$ is enabled in the original net, then the same transition must be enabled in the constructed net as well, because the token in place $a_k$ is available for consumption and the rest marking is the same. Furthermore, the garbage collectors are ensuring that the extra tokens produced by transition $T_0$ for each token in place $P$ are eventually consumed.

**Modeling time-outs** Finally, note that singleton input delays are handy for modeling of time outs, which are an essential construct in distributed systems, c.f. Fig. 16. The transition $y$ (receiver) is supposed to fire if $z$ (sender) has sent a message. If the delay $\delta(z, q)$ incurred due to sending is taking more than $d_1$ time units then transition $x$ (time-out) will fire. Input delays are also handy for garbage collection. Suppose transition $x$ has fired and afterwards transition $z$ then $w$ (garbage-collector) will remove the token produced by $z$ after $d_2$ time units.

On the other hand, output delays, particularly with finite delay sets are handy for modeling stochastic behavior as seen in the section 7. For these reasons we consider M3 as the best model for practice.
6.4. Simulating timed Petri nets and time Petri nets (Merlin time)

In this section we will establish the following relationship between model classes: \( M_1 \subseteq M_3 \) and \( M_2 \subseteq s M_3 \), i.e. (a) for every model of type timed Petri nets, there exists a model of class DTPN that is branching bisimilar, and (b) for every model of class Merlin time, there exists a model of class DTPN that is trace equivalent.

**Simulating Timed Petri nets \( M_1 \) with Timed inhibitor nets \( M_3 \).**

For model class \( M_1 \), we have the property \( M_1 \subseteq M_3 \). To verify this we construct for each model of \( M_1 \) a model of \( M_3 \) that is branching bisimilar. There are two cases, one where a transition may be firing concurrently with itself, and one where this is excluded. The constructions are displayed in Fig. 17. Transition \( t \) is refined by a silent transition \( x \) and place \( y \). The transition delay for \( t \) in \( M_1 \) is transformed into an output delay of \( x \) in \( M_3 \). Place \( z \) with initially one token is used for the second case. It is straightforward to verify that the models are branching bisimilar.

**Simulating Merlin time \( M_2 \) with Timed inhibitor nets \( M_3 \).**

For model class \( M_2 \) we have \( M_2 \subseteq s M_3 \). To verify this, we construct for a model of class \( M_2 \), a model of class \( M_3 \) having ODO that is trace equivalent.

**Assumption.** Like most Merlin time proposals, we will assume that transition \( T \) cannot be enabled more than once at the same time, i.e. for all reachable markings, there is at least one place in the preset of a timed transition with at most one token. So it suffices to assume that at least one pre-place of a timed transition is safe.

**Construction.** We will explain the construction using Fig. 18. We replace the incoming arcs to transition \( T \) in the original net by a subnet consisting of four silent transitions: silent transition \( T_0 \), two cancel transitions and one garbage collector. Only the token in place \( S_4 \) is delayed by a selection of delay from the interval \( [x, y] \). All other arcs have zero delays.

Note that transition \( T_0 \) is enabled only when the current time equals the maximum of the timestamp of tokens in \( P \) and \( Q \). So when transition \( T_0 \) fires the earliest tokens in the places \( P \) and \( Q \) are updated with the enabling time of \( T_0 \). As only the tokens older than the current time are updated with the current time, the enabling of transitions in the postset of places \( P \) and \( Q \) remain unchanged.

Furthermore, when transition \( T_0 \) is firing, a token with a delay (chosen from interval \( [x, y] \)) is produced in the place \( S_4 \) and a token without this additional delay is produced in the place \( S_2 \). The token in place \( S_3 \) is ensuring that only one enabling of transition \( T \) is considered at a given time. If transition \( T_0 \) has fired already then \( T_0 \) cannot fire again before either \( T \) has fired or one of the cancel transitions have fired. A cancel transition will fire if either the place \( P \) or \( Q \) (input places of \( T \)) becomes empty while there is still a token in the place \( S_2 \). Firing a cancel transition will return the token back to the place \( S_3 \) (allowing \( T_0 \) to be enabled again in the next time step) and will produce a token also in the place \( S_1 \) in order to trigger the garbage collector to consume a
Figure 18: Transforming Merlin time ($M_2$) into $M_3$

Figure 19: Modeling task execution times

delayed token from place $S_4$. When there are more than one delayed token in the place $S_4$, then the garbage collector will fire until the place $S_1$ has become empty and then transition $T$ is enabled.

There is an underlying order: If there is more than one token in the place $S_4$, then by the eagerness of transition firing, the garbage collector will consume the tokens that have the lowest time stamp in place $S_4$, so transition $T$ will fire with the token having the highest timestamp. So it is possible that transition $T$ may fire with a token from an earlier enabling as sketched by the following example:

Let $[x, y]$ be the interval $[3, 9]$. Suppose tokens arrive in place $S_4$ at times: 1, 2, 3 (i.e. creation time of tokens) with timestamps: 10, 5, 12, respectively. Suppose that transition $T$ is still enabled and there are two tokens in place $S_1$. Then the garbage collector will fire at 5 and 10, so $T$ fires at 12, which corresponds to the last enabling!

Now suppose tokens arrive at 1, 2, 3 with timestamps: 4, 11, 6. Suppose again that transition $T$ is still enabled and there are two tokens in place $S_1$. Then the garbage collector fires at 4 and 6 and transition $T$ at 11 which is not the last enabling!

**Observation.** Suppose tokens arrive in place $S_4$ at times $X_1$ and $X_2$ such that $X_1 < X_2$ with chosen delays $Y_1$ and $Y_2$ such that $Y_1 > Y_2$ (from interval $[X, Y]$), respectively. Then the enabling time of transition $T$ is $\max\{X_1 + Y_1, X_2 + Y_2\}$. Consider the two possibilities: (a) If $X_1 + Y_1 < X_2 + Y_2$ then the garbage collector fires at $X_1 + Y_1$ and transition $T$ fires at $X_2 + Y_2$, and (b) If $X_1 + Y_1 > X_2 + Y_2$ then $T$ fires at $X_1 + Y_1$. Observe that $X_2 + Y_2 < X_1 + Y_1 < X_2 + Y_1$, which means that transition $T$ fires at $X_2 + d$, where $d = Y_1 - X_2 + X_1$, and $Y_2 < d < Y_1$. Hence $d$ is a value in the
interval of \([X, Y]\) and the property \(Y_1 - X_2 + X_1 > Y_2\) iff \(Y_1 + X_1 > X_2 + Y_2\) holds.

Since in the Merlin time semantics, delay is an arbitrary value chosen from an interval, it is straightforward to check that a Merlin time model is able to simulate any choice of delays made by its constructed DTPN model. However, when more than one enabling is considered at different time points, the constructed DTPN model is not able to simulate its Merlin time model as choices made by silent transition \(T_0\) during a previous enabling may interfere with the choice made by \(T_0\) in the current enabling (as described in the case (b)). However using a lookahead strategy on firing sequences similar to the proof of theorem 20, we are able to prove trace equivalence.

**Theorem 28.** A time Petri net (Merlin time (MT)) and its constructed ODO model are trace equivalent.

**Proof.** (sketch only). We will use the notations of Fig. 19.

Consider a sequence of transitions \(\sigma = (e_1, d_1, e_2, d_2, \ldots, e_n, d_n, e_{n+1})\) that are causing a timed transition \(T\) to be enabled and disabled \(n\) times before eventually transition \(T\) fires. Let \(t_1, t_1 + s_1, t_2, t_2 + s_2, \ldots, t_n + s_n, t_{n+1}\) be the absolute time points corresponding each occurrence of a transition in the sequence \(\sigma\).

- If the sequence \(\sigma\) belongs to the MT model. Then we are able to execute the sequence \(\sigma\) in the ODO model in the following way: In each enabling of transition \(T\) caused by a transition \(e_i\), for some \(i \in \{1, \ldots, n\}\), silent transition \(T_0\) is choosing a delay \(\delta_i \in [X, Y]\) such that \(\delta_i = \min(X, s_i)\) and \(\max(t_i + \delta_i \mid i \in \{1, \ldots, n\}) \le t_{n+1} + \delta_{n+1}\), where \(\delta_{n+1}\) is chosen as \(t_{n+2} - t_n\). These choices are feasible since \(s_j \in [0, Y]\), because otherwise the sequence could not have occurred in the MT model and since \(\forall i \in \{1, \ldots, n\} : t_i + s_i \le t_{n+1}\), so \(\forall i \in \{1, \ldots, n\} : t_i + s_i \le t_{n+1}\) and since \(\delta_i = \min(X, s_i)\), we may write \(\forall i \in \{1, \ldots, n\} : t_i + \delta_i \le t_i + \min(X, s_i) = \min(t_i + X, t_i + s_i) = \min(t_i + X, t_{i+1}) \le t_{i+1} + X \le t_{n+1} + X \le t_{n+1} + \delta_{n+1}\).

- If the sequence \(\sigma\) belongs to the ODO model. Then the timed transition \(T\) will fire in the ODO model at \(\max(t_i + \delta_i \mid i \in \{1, \ldots, n\})\) and in the MT model the delay for the last enabling should be \(\max(t_i + \delta_i \mid i \in \{1, \ldots, n+1\}) - t_{n+1}\) which is possible since \(\delta_{n+1} \le \max(t_i + \delta_i \mid i \in \{1, \ldots, n+1\}) - t_{n+1} \le t_{n+1} + Y - t_{n+1} = Y\).

\(\square\)

Note that for modeling of 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. 19. 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 \(b \in \delta(P, Y)\).

### 7. 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 \(w(x)/\sum_{y \in \text{enabled}} w(y)\). 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 31 (DSPN).**

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

- \((P, T, F, \delta)\) is a DTPN with (In-single, Out-fint).
- \(w : T \rightarrow \mathbb{R}^+\) a weight function, used to choose one of the simultaneously enabled transitions,
• \( \phi \) 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 \( \sum_{c \in \delta(t, p)} \phi(t, p) = 1 \), 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. 12, where for each value of a finite output delay interval, a transition is introduced with a one point output delay, as in Fig. 4. Here transition \( r \) 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 \( \phi(\cdot, \cdot) \) because all output delays are singletons. So we only have to deal with the weight function \( w \). By Theorem 6, 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 \((P, T, F, \delta, w)\).

The next transformation concerns this sDTPN model into the reduced labeled transition system (rDTPN) as in Theorem 12 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 \( m, m' \in M : m \rightarrow m' \): (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 10.

**Definition 32 (Transition probability and sojourn time).** The transition probability \( Q : M \times M \rightarrow [0, 1] \) satisfies:

\[
Q_{m,m'} = \sum_{x : m \rightarrow m'} w(x) / \sum_{y : m' \rightarrow m'} w(y).
\]

For \( m, m' \in M : r(m, m') = f(t(m')) - f(t(m)) \) 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 33 (Markov chain).** Let a rDSPN \((P, T, F, \delta, w)\) 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 \( \{X_n\}_{n = 0, 1, \ldots} \), where \( X_0 = m_0 \) the initial marking and \( X_n \) is marking after \( n \) steps, such that:

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

for arbitrary \( m_0, ..., m_{n-1} \in M \).

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 \( m_0 \rightarrow t_1 \rightarrow m_1, ..., t_n \rightarrow m_n \) we have

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

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:

\[
\mathbb{E} \left[ \sum_{n=0}^{N} c(X_n, X_{n+1}) | X_0 = m \right].
\]
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 $\{X_n | n = 0, 1, \ldots\}$ is then the embedded Markov chain of the semi-Markov process [32]. We will use the function $v : \mathbb{M} \rightarrow \mathbb{R}$, which is usually called the value function for Markov processes (see [28]): $v(m) = \mathbb{E}\left[\sum_{n=0}^{N} c(X_n, X_{n+1}) | X_0 = m\right]$ 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 [32] and [28]).

- 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 [27]).

**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 $m \in B$ and $m' \in A$ and $c(m, m') = 0$ otherwise. Further we stop as soon as we reach $A$. Then

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

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

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

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: $\pi_{m}(m') := \lim_{N \to \infty} \sum_{n=0}^{N} Q_{m,m'}^n$ and it satisfies

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

We now assume the system is a strongly connected component (i.e. the Markov chain is irreducible), which implies that the limit distribution $\pi$ 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:

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

which can be rewritten using Bayes rule to:

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

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

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

\[
\sum_{m \in M} r(m, m') Q_{m,m'} \cdot \pi(m).
\]

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:

\[
\forall m \in B : v_0(m) = 0 \quad \text{and} \quad \forall m \in B : v_{n+1}(m) = \sum_{m' \in A} Q_{m,m'} + \sum_{m' \in B} Q_{m,m'} \cdot v_n(m').
\]

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. 20. 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. 20 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 \([t] = [t'] = 1\) 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. \( \mathbb{P}[U \leq x] = \mathbb{P}[Y \leq x] \cdot \mathbb{P}[Z \leq x] \). For the choice (rule 4) it is a mixture of two distributions, \( \mathbb{P}[U \leq x] = \mathbb{P}[Y \leq x] \cdot q + \mathbb{P}[Z \leq x] (1 - q) \). The most difficult one is the iteration (rule 2), since here we have the distribution of \( U := \sum_{n=1}^{N} (Y_n + Z_n) \) where \( N \) is a geometrically distributed random variable with distribution \( \mathbb{P}[N = n] = q^{n-1} \cdot (1 - q) \) 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.

8. Conclusions

In this paper we reviewed the most studied Petri nets with time and stochastics. There are many model classes that have been proposed and it is difficult for a model engineer to understand the differences and to decide the strong and weak points of each model class. Some are strong for theoretical purposes others for modeling. One class did not get much attention in literature: the class with timestamps for tokens. We call this class DTPN and showed how we can analyse this class with model checking, in case the underlying Petri net is bounded. Since model checking of timed automata is supported already by the popular and mature toolset of UPPAAL, we identify a subclass of DTPN that can be transformed into a timed automaton. However, the transformation is preserving language equivalence when output delays are considered and branching bisimulation when output delays are omitted. We considered several subclasses of DTPN and showed that they all have the same expressive power (Turing completeness, because they can express inhibitor arcs) but that some have better modeling comfort, i.e. they are easier for modeling. We also extended model class DTPN with a timed variant of inhibitor arcs (called the timed inhibitor net) that is also taking into account the
availability of tokens and show that only in two cases, we may replace a timed inhibitor arc by a subnet modeled as a DTPN and yet preserve the simulation property. For modeling output delays seem to be the natural way to express that some activity takes time while input delays are handy for modeling time outs. If model engineers stick to the convention that synchronization actions should not take time and that time consuming activities should be modeled with one start event and one stop event, then most different time models boil down to the same! The DTPN class can easily be extended to deal with stochastics as we have shown. Here we have the advantage that we can apply arbitrary finite distributions which has the advantage above the GSPN model that requires exponential distributions only. The analysis of stochastic behavior is based on Markov chains and so it is similar to the approach in GSPN.

References


# Science Reports

Department of Mathematics and Computer Science  
Technische Universiteit Eindhoven

If you want to receive reports, send an email to: [wsinsan@tue.nl](mailto:wsinsan@tue.nl) (we cannot guarantee the availability of the requested reports).

**In this series appeared (from 2009):**

<table>
<thead>
<tr>
<th>Date</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/01</td>
<td>Wil M.P. van der Aalst, Kees M. van Hee, Peter Massuthe, Natalia Sidorova and Jan Martijn van der Werf</td>
<td>Compositional Service Trees</td>
</tr>
<tr>
<td>09/03</td>
<td>Maarten G. Meulen, Frank P.M. Slappers and Tim A.C. Willemsen</td>
<td>Breadth-Bounded Model Checking</td>
</tr>
<tr>
<td>09/04</td>
<td>Muhammad Atif and MohammadReza Mousavi</td>
<td>Formal Specification and Analysis of Accelerated Heartbeat Protocols</td>
</tr>
<tr>
<td>09/05</td>
<td>Michael Franssen</td>
<td>Placeholder Calculus for First-Order logic</td>
</tr>
<tr>
<td>09/06</td>
<td>Daniel Trivellato, Fred Spiessens, Nicola Zannone and Sandro Etalle</td>
<td>POLIPO: Policies &amp; Ontologies for the Interoperability, Portability, and autOnomy</td>
</tr>
<tr>
<td>09/07</td>
<td>Marco Zapletal, Wil M.P. van der Aalst, Nick Russell, Philipp Liegl and Hannes Werthner</td>
<td>Pattern-based Analysis of Windows Workflow</td>
</tr>
<tr>
<td>09/08</td>
<td>Mike Holenderski, Reinder J. Bril and Johan J. Lukkien</td>
<td>Swift mode changes in memory constrained real-time systems</td>
</tr>
<tr>
<td>09/09</td>
<td>Dragan Bošnački, Aad Mathijsen and Yaroslav S. Usenko</td>
<td>Behavioural analysis of an FC Linux Driver</td>
</tr>
<tr>
<td>09/10</td>
<td>Ugur Keskin</td>
<td>In-Vehicle Communication Networks: A Literature Survey</td>
</tr>
<tr>
<td>09/11</td>
<td>Bas Ploeger</td>
<td>Analysis of ACS using mCRL2</td>
</tr>
<tr>
<td>09/12</td>
<td>Wolfgang Boehmer, Christoph Brandt and Jan Friso Groote</td>
<td>Evaluation of a Business Continuity Plan using Process Algebra and Modal Logic</td>
</tr>
<tr>
<td>09/13</td>
<td>Luca Aceto, Anna Ingolfsdottir, MohammadReza Mousavi and Michel A. Reniers</td>
<td>A Rule Format for Unit Elements</td>
</tr>
<tr>
<td>09/14</td>
<td>Maja Pešić, Dragan Bošnački and Wil M.P. van der Aalst</td>
<td>Enacting Declarative Languages using LTL: Avoiding Errors and Improving Performance</td>
</tr>
<tr>
<td>09/16</td>
<td>Mohammad Atif</td>
<td>Formal Analysis of Consensus Protocols in Asynchronous Distributed Systems</td>
</tr>
<tr>
<td>09/17</td>
<td>Jeroen Keiren and Tim A.C. Willemsse</td>
<td>Bisimulation Minimisations for Boolean Equation Systems</td>
</tr>
<tr>
<td>09/18</td>
<td>Kees van Hee, Jan Hidders, Geert-Jan Houben, Jan Paredaens, Philippe Thiran</td>
<td>On-the-fly Auditing of Business Processes</td>
</tr>
<tr>
<td>10/01</td>
<td>Ammar Osaiweran, Marcel Boosten, MohammadReza Mousavi</td>
<td>Analytical Software Design: Introduction and Industrial Experience Report</td>
</tr>
<tr>
<td>10/02</td>
<td>F.E.J. Kruseman Aretz</td>
<td>Design and correctness proof of an emulation of the floating-point operations of the Electrologica X8. A case study</td>
</tr>
<tr>
<td>Date</td>
<td>Title</td>
<td>Authors</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>10/03</td>
<td>On Rule Formats for Zero and Unit Elements</td>
<td>Luca Aceto, Matteo Cimini, Anna Ingolfsdottir, Mohammad Reza Mousavi and Michel A. Reniers</td>
</tr>
<tr>
<td>10/04</td>
<td>Towards Model-Based Testing of Electronic Funds Transfer Systems</td>
<td>Hamid Reza Asaadi, Ramtin Khosravi, Mohammad Reza Mousavi, Neda Noroozi</td>
</tr>
<tr>
<td>10/05</td>
<td>Schedulability analysis of synchronization protocols based on overrun without payback for hierarchical scheduling frameworks revisited</td>
<td>Reinder J. Bril, Uğur Keskin, Moris Beinham, Thomas Nolte</td>
</tr>
<tr>
<td>10/06</td>
<td>Locally unique labeling of model elements for state-based model differences</td>
<td>Zvezdan Protić</td>
</tr>
<tr>
<td>10/07</td>
<td>Converting existing analysis to the EDP resource model</td>
<td>C.G.U. Okwudire and R.J. Bril</td>
</tr>
<tr>
<td>10/08</td>
<td>Reconstruction and verification of group membership protocols</td>
<td>Mohammed Atif, Sjoerd Cranen, Mohammad Reza Mousavi</td>
</tr>
<tr>
<td>10/09</td>
<td>A linear translation from LTL to the first-order modal μ-calculus</td>
<td>Sjoerd Cranen, Jan Friso Groote, Michel Reniers</td>
</tr>
<tr>
<td>10/10</td>
<td>Extending an Open-source Real-time Operating System with Hierarchical Scheduling</td>
<td>Mike Holenderski, Wim Cools, Reinder J. Bril, Johan J. Lukkien</td>
</tr>
<tr>
<td>10/11</td>
<td>1st Doctoral Symposium of the International Conference on Software Language Engineering (SLE)</td>
<td>Eric van Wyk and Steffen Zschaler</td>
</tr>
<tr>
<td>10/12</td>
<td>3rd International Software Language Engineering Conference</td>
<td>Pre-Proceedings</td>
</tr>
<tr>
<td>10/13</td>
<td>Discrimination Aware Decision Tree Learning</td>
<td>Faisal Kamiran, Toon Calders and Mykola Pechenizkiy</td>
</tr>
<tr>
<td>10/14</td>
<td>Specification Guidelines to avoid the State Space Explosion Problem</td>
<td>J.F. Groote, T.W.D.M. Kouters and A.A.H. Osaiweran</td>
</tr>
<tr>
<td>10/15</td>
<td>GEM: a Distributed Goal Evaluation Algorithm for Trust Management</td>
<td>Daniel Trivellato, Nicola Zannone and Sandro Etalle</td>
</tr>
<tr>
<td>10/17</td>
<td>Decompositional Reasoning about the History of Parallel Processes</td>
<td>L. Aceto, A. Birgisson, A. Ingolfsdottir, and M.R. Mousavi</td>
</tr>
<tr>
<td>10/18</td>
<td>Robustness on Behavioral Equivalence on Open Terms</td>
<td>P.D. Mosses, M.R. Mousavi and M.A. Reniers</td>
</tr>
<tr>
<td>10/19</td>
<td>Desynchronisability of (partial) closed loop systems</td>
<td>Harsh Beohar and Pieter Cuijpers</td>
</tr>
<tr>
<td>11/01</td>
<td>Refinement of Synchronizable Places with Multi-workflow Nets - Weak termination preserved!</td>
<td>Kees M. van Hee, Natalia Sidorova and Jan Martijn van der Werf</td>
</tr>
<tr>
<td>11/02</td>
<td>Using a DSL and Fine-grained Model Transformations to Explore the boundaries of Model Verification</td>
<td>M.F. van Arnstel, M.G.J. van den Brand and L.J.P. Engelen</td>
</tr>
<tr>
<td>11/05</td>
<td>Semantics, bisimulation and congruence results for a general stochastic process operator</td>
<td>Jan Friso Groote and Jan Lanik</td>
</tr>
<tr>
<td>11/06</td>
<td>Moore-Smith theory for Uniform Spaces through Asymptotic Equivalence</td>
<td>P.J.L. Cuijpers</td>
</tr>
<tr>
<td>11/07</td>
<td>Transforming SOS Specifications to Linear Processes</td>
<td>F.P.M. Stappers, M.A. Reniers and S. Weber</td>
</tr>
<tr>
<td>11/08</td>
<td>A Component Framework where Port Compatibility Implies Weak Termination</td>
<td>Debyjoti Bera, Kees M. van Hee, Michiel van Osch and Jan Martijn van der Werf</td>
</tr>
<tr>
<td>11/09</td>
<td>Model, analysis, and improvements for inter-vehicle communication using one-hop periodic broadcasting based on the 802.11p protocol</td>
<td>Tsesuren Batsuuri, Reinder J. Bril and Johan Lukkien</td>
</tr>
</tbody>
</table>
11/10 Neda Noroozi, Ramin Khosravi, Mohammad Reza Mousavi and Tim A.C. Willemse
Synchronizing Asynchronous Conformance Testing

11/11 Jeroen J.A. Keiren and Michel A. Reniers
Type checking mCRL2

11/12 Muhammad Atif, Mohammad Reza Mousavi and Ammar Osaiweran
Formal Verification of Unreliable Failure Detectors in Partially Synchronous Systems

11/13 J.F. Groote, A.A.H. Osaiweran and J.H. Wesselius
Experience report on developing the Front-end Client unit under the control of formal methods

11/14 J.F. Groote, A.A.H. Osaiweran and J.H. Wesselius
Ananlyzing a Controller of a Power Distribution Unit Using Formal Methods

11/15 John Businge, Alexander Serebrenik and Mark van den Brand
Eclipse API Usage: The Good and The Bad

Investigating the Effects of Designing Control Software using Push and Poll Strategies

11/17 M.F. van Amstel, A. Serebrenik and M.G.J. van den Brand
Visualizing Traceability in Model Transformation Compositions

11/18 F.P.M. Stappers, M.A. Reniers, J.F. Groote and S. Weber
Dogfooding the Structural Operational Semantics of mCRL2

12/01 S. Cranen
Model checking the FlexRay startup phase

12/02 U. Khadim and P.J.L. Cuijpers
Appendix C / G of the paper: Repairing Time-Determinism in the Process Algebra for Hybrid Systems ACP

12/03 M.M.H.P. van den Heuvel, P.J.L. Cuijpers, J.J. Lukkien and N.W. Fisher
Revised budget allocations for fixed-priority-scheduled periodic resources

12/04 Ammar Osaiweran, Tom Fransen, Jan Friso Groote and Bart van Rijnsoever
Experience Report on Designing and Developing Control Components using Formal Methods

12/05 Sjoerd Cranen, Jeroen J.A. Keiren and Tim A.C. Willemse
A cure for stuttering parity games

12/06 A.P. van der Meer
CIF MSOS type system

12/07 Dirk Fahland and Robert Prüfer
Data and Abstraction for Scenario-Based Modeling with Petri Nets

12/08 Luc Engelen and Anton Wijs
Checking Property Preservation of Refining Transformations for Model-Driven Development

12/09 M.M.H.P. van den Heuvel, M. Behnam, R.J. Bril, J.J. Lukkien and T. Nolte
Opaque analysis for resource-sharing components in hierarchical real-time systems - extended version –

12/10 Milosh Stolikj, Pieter J. L. Cuijpers and Johan J. Lukkien
Efficient reprogramming of sensor networks using incremental updates and data compression

12/11 John Businge, Alexander Serebrenik and Mark van den Brand
Survival of Eclipse Third-party Plug-ins

12/12 Jeroen J.A. Keiren and Martijn D. Klabbers
Modelling and verifying IEEE Std 11073-20601 session setup using mCRL2

12/13 Ammar Osaiweran, Jan Friso Groote, Mathijs Schuts, Jozef Hooman and Bart van Rijnsoever
Evaluating the Effect of Formal Techniques in Industry

12/14 Ammar Osaiweran, Mathijs Schuts, and Jozef Hooman
Incorporating Formal Techniques into Industrial Practice

13/01 S. Cranen, M.W. Gazda, J.W. Wesselin and T.A.C. Willemse
Abstraction in Parameterised Boolean Equation Systems

13/02 Neda Noroozi, Mohammad Reza Mousavi and Tim A.C. Willemse
Decomposability in Formal Conformance Testing
13/03  D. Bera, K.M. van Hee and N. Sidorova  Discrete Timed Petri nets

13/04  A. Kota Gopalakrishna, T. Ozcelebi, A. Liotta and J.J. Lukkien  Relevance as a Metric for Evaluating Machine Learning Algorithms