Expressiveness and analysis of Delayable Timed Petri Net

Rémi Parrot, Hanifa Boucheneb, Mikaël Briday, Olivier H Roux

Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 55-28 (2022) 284–290

Expressiveness and analysis of Delayable Timed Petri Net ⋆

Rémi Parrot * **Hanifa Boucheneb** ** **Mikaël Briday** *
Olivier H. Roux *

* *Nantes Université, Ecole Centrale Nantes, CNRS, LS2N UMR 6004, F-44000 Nantes, France (e-mail: firstname.surname@ec-nantes.fr),*
** *École Polytechnique de Montréal, Québec, Canada, (e-mail: hanifa.boucheneb@polymtl.ca)*

Abstract: We consider an extension of Timed Petri Nets "à la Ramchandani" where the transitions are partitioned into delayable and non-delayable transitions which has proven to be suitable for the design of synchronous circuits. For this model called Delayable Timed Petri Net (DTPN), the firing delay of a non-delayable transition is strict whereas a delayable transition can miss its firing delay. Since the delays are natural numbers, this model can be studied as a discrete time model.
We deal with the expressiveness of DTPN by a comparison with the well known Merlin's *Time Petri Net* model for which transitions can fire in a time interval. We show that DTPN are strictly more expressive w.r.t. weak timed bisimilarity than Merlin's model under the discrete-time semantics.
We then deal with the symbolic reachability analysis of DTPN, for which we show the complexity of the successor symbolic state computation to be $O(n)$. In addition, we propose a reduction of the number of edges to explore that preserves the markings and the firing sequences.
The symbolic state space exploration is implemented in a prototype tool, which is evaluated on a classical TPN problem and a circuit design application.

Keywords: Timed Petri Net, Time Petri Net, Expressivity, Reachability analysis, Complexity

1. INTRODUCTION

The concept of quantitative time is not explicitly given in the original definition of Petri Nets and has been introduced for performance evaluation and scheduling problems of dynamic systems. Time has been associated with transitions or places by assigning time delays, either using a fixed and deterministic value or choosing it from a probability distribution.

Petri nets and time The two main extensions of Petri Nets with non-stochastic time are Timed Petri Nets (TdPN) (Ramchandani, 1974) and Time Petri Nets (TPN) (Merlin, 1974). The first one associated a firing duration with each transition of the net and the second one associated a time interval with each transition in order to model an uncertain occurrence of the transition. In TdPNs, delays were first associated with transitions (T-TdPN) and then to places (P-TdPN). The two corresponding subclasses are expressively equivalent (Ramchandani, 1974; Sifakis, 1977). Thus, a time delay can represent a minimum duration of firing or a minimum residence time of a token in a place.

State space exploration For interval time model, as long as dense-time semantics is considered, the state space is generally infinite. Verification algorithms then compute finite abstractions, e.g. state class graph or zone graph, with dedicated and data structures like Difference Bounded Matrices (Dill, 1989) (DBM).

In (Popova, 1991), Popova proposes to analyse the whole behavior of a TPN by using only its so-called "integer-states" (*i.e.* states where the current local time for all enabled transitions are integers). The relationship between discrete and dense time semantics have been discussed for various timed models, showing that in many cases, the discrete time analysis allows capturing reachability or time-bounded properties (Henzinger et al., 1992; Popova, 1991, 2006). However, discrete-time based approaches suffer from a combinatorial explosion of the state space and reach their limits as soon as there are transitions with a large time interval in the model.

Petri nets and circuit design An important step in logic circuit design is the placement of the pipeline stages. The circuit composed of atomic operators is divided into several stages physically implemented with memories (flip-flops), allowing the concurrent execution of the stages and the synchronisation of their inputs/outputs. Timed Petri Nets "à la Ramchandani" have been defined for the modeling of circuits. An extension of Timed Petri Nets with a *reset* action (RTPN) has been proposed in (Parrot et al., 2021b) for pipelined synchronous circuit design. It integrates the impacts of registers on the circuit delay with a special reset

⋆ This work is supported by the Renault-Centrale Nantes chair dedicated to the propulsion performance of electric vehicles.

10.1016/j.ifacol.2022.10.355

operation, and allows some time constraints to be relaxed with delayable transitions. This model has been shown to be effective for circuit design in (Parrot et al., 2021a).

Our contribution In this article we consider the class of RTPN in which we keep the timed features of transition but without the *reset* action. The *reset* can be expressed with the other features of the model, which makes this class a superclass of RTPN. This model, that we call Delayable Timed Petri Net (DTPN), can then be defined as an extension of Timed Petri nets "à la Ramchandani" where the transitions are partitioned into delayable and non-delayable transitions. The firing delay of a non-delayable transition is strict. A delayable transition can however miss its firing delay.

Firstly, we give the semantics of the DTPN model which is a maximal-step semantics. Then, we provide a simple DTPN model for which there is no weakly timed bisimilar Merlin's model, and we show that, under the discrete-time semantics, any Merlin's model can be translated into a DTPN model that preserves the weak timed bisimilarity. These two results allow concluding that DTPN are strictly more expressive *w.r.t.* weak timed bisimilarity.

From a symbolic reachability analysis point of view, the computation complexity of the time part of a successor symbolic state is $O(n)$ for the DTPN model but $O(n^2)$ for the Merlin's model, n being the number of transitions in the model. In this context, for the DTPN model under the maximal-step semantics, we show how to reduce the number of maximal-steps to be explored, from a symbolic state, without affecting the set of reachable markings.

Outline of the paper Section 2 defines the DTPN model and its (maximal-step) semantics. Section 3 is devoted to the expressiveness of the DTPN model relatively to the Merlin's model. Sections 4 and 5 deal with the symbolic reachability analysis and its complexity. Section 6 evaluates an implementation of the symbolic state space exploration with two applications. Finally, Section 7 concludes this paper.

2. DEFINITION

$\mathbb{N}$ and $\mathbb{R}_{\geq 0}$ are respectively the sets of integers and non-negative real numbers. For vectors of size n, the usual operators $+, -, \times, <, \leq, >, \geq$ and $=$ are used on vectors of $\mathbb{N}^n$ and $\mathbb{R}^n_{\geq 0}$ and are the point-wise extensions of their counterparts in $\mathbb{N}$ and $\mathbb{R}_{\geq 0}$. Let $\bar{\mathbf{0}}$ be the null vector of size n. The operator $\ominus$ over elements of $\mathbb{R}_{\geq 0}$ and taking values in $\mathbb{R}_{\geq 0} \cup \{-\infty\}$, is defined by $\forall a, b \in \mathbb{R}_{\geq 0}$, $a \ominus b = a - b$ if $a \geq b$ and $= -\infty$ otherwise.

2.1 Delayable Timed Petri Net

The authors of (Parrot et al., 2021b) defined a semantics of DTPN classically associating with each transition a clock which increases with time. In this work, we propose an equivalent semantics using instead decreasing delays. This will come in handy with the state exploration, in particular for the delayable transitions that missed their firing date (delayed transitions).

Informally, with each transition of the net is associated a dynamic delay. Time elapsing decreases its value. The dynamic delay sets a firing condition: the transition may and must fire when it is equal to zero. Moreover, some transitions are delayable and may fire when the dynamic delay is equal to zero but can fire later. In this case, its value is noted $-\infty$.

Formally:

Definition 1. (DTPN). A Delayable Timed Petri Net is a tuple $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, \delta, M_0)$ defined by:

- $P = \{p_1, p_2, \ldots, p_m\}$ is a non-empty set of places;
- $T = \{t_1, t_2, \ldots, t_n\}$ is a non-empty set of transitions;
- $T_D \subseteq T$ is the set of delayable transitions;
- ${}^\bullet(.) : T \to \mathbb{N}^P$ is the backward incidence function;
- $(.)^\bullet : T \to \mathbb{N}^P$ is the forward incidence function;
- $M_0 \in \mathbb{N}^P$ is the initial marking of the Petri Net;
- $\delta : T \to \mathbb{N}$ is the function giving the firing times (delays) of transitions.

This paper considers the DTPN model, in the context of the maximal-step firing semantics, which means that, from each state, the largest possible sets of transitions are fired simultaneously.

Let $\mathcal{N}$ be DTPN . A marking M of $\mathcal{N}$ is an element of $\mathbb{N}^P$ such that $\forall p \in P$, $M(p)$ is the number of tokens in place p. A marking M enables a transition $t \in T$ if $M \geq {}^\bullet t$. The set of transitions enabled at M is denoted by $enab(M)$. A set of transitions $\tau \subseteq T$ is an *enabled step* at M, if all its transitions are enabled at M and not in conflict: $M \geq \Sigma_{t \in \tau}({}^\bullet t)$. The simultaneous firing of the step τ leads to the marking $M' = M + \Sigma_{t \in \tau}(t^\bullet - {}^\bullet t)$. A transition t' is said to be *newly* enabled by the firing of an enabled step τ from M, if $M + \Sigma_{t \in \tau}(t^\bullet - {}^\bullet t)$ enables t' and $(M - \Sigma_{t \in \tau}\ {}^\bullet t)$ does not enable t'. If t remains enabled after its firing then t is newly enabled. The set of transitions newly enabled by a step τ from a marking M is noted $\uparrow enab(M, \tau)$.

A state of $\mathcal{N}$ is a pair $q = (M, \Delta)$, where M is a marking and $\Delta \in (\mathbb{R}_{\geq 0} \cup \{-\infty\})^T$ is the vector of dynamic delays.

When a transition $t \in T$ is newly enabled, its dynamic delay $\Delta(t)$ is set to $\delta(t)$ then it decreases synchronously with time. Thus, $\delta(t) - \Delta(t)$ gives the time elapsed since the transition has been newly enabled. A non-delayable transition must fire immediately, when its dynamic delay reaches 0, unless it is disabled by another firing. The firing of a delayable transition $t \in T_D$ is however not mandatory, when its dynamic delay reaches 0. Its delay may be negative. All the negative delays are represented by $-\infty$.

Definition 2. (Timed Transition System). A *timed transition system (TTS)* over the set of actions $\mathcal{A}$ is a tuple $S = (Q, q_0, \mathcal{A}, \to)$ where Q is a set of states, $q_0 \in Q$ is the initial state, $\mathcal{A}$ is a finite set of actions disjoint from $\mathbb{R}_{\geq 0}$, $\to \subseteq Q \times (\mathcal{A} \cup \mathbb{R}_{\geq 0}) \times Q$ is a set of edges. If $(q, e, q') \in \to$, we also write $q \xrightarrow{e} q'$.

The semantics of a DTPN $\mathcal{N} = (P, T, {}^\bullet(.), (.)^\bullet, \delta, M_0)$ is a TTS $S_\mathcal{N} = (Q, q_0, \mathcal{A}, \to)$, where $Q = \mathbb{N}^P \times \mathbb{R}^T$, $q_0 = (M_0, \Delta_0)$, where M_0 is the initial marking and Δ_0 is the valuation assigning $\delta(t)$ to every transition t, and $\mathcal{A} = 2^T$.

Let $q = (M, \Delta)$ be a state of $\mathcal{N}$, $t \in T$ and $\tau \subseteq T$ a step enabled at M. A non-delayable (resp. delayable) transition t is firable at q if it is enabled and its dynamic delay is equal to 0 (resp. equal to 0 or $-\infty$). The step τ is a maximal-step at q, if for every non-delayable transition t firable at state q, either $t \in \tau$ or $\tau \cup \{t\}$ is not an enabled step at M. Note that a maximal-step may contain some delayable transitions, but it must contain at least one transition t, which has not yet missed its deadline, *i.e.* such that $\Delta(t) = 0$. The firing of a maximal-step $\tau \subseteq T$ from state (M, Δ) is denoted $(M, \Delta) \xrightarrow{\tau} (M', \Delta')$. It leads to the new marking $M' = M + \Sigma_{t\in\tau}(t^\bullet - {}^\bullet t)$, and sets to $\delta(t)$ the delay of all newly enabled transitions. In a Marked Graph where every place has one incoming arc, and one outgoing arc, there can not be conflict and the firing of a transition cannot disable another transition. But in the general case, there can be conflicts, and thus there can be several maximal-steps τ from a given state.

Waiting for a duration $d \in \mathbb{R}_{\geq 0}$ from a state (M, Δ) is denoted $(M, \Delta) \xrightarrow{d} (M, \Delta')$. It leads to the new valuation Δ' such that for all enabled transitions t, $\Delta'(t) = \Delta(t) \ominus d$. A delay transition is possible only if the clock of all non-delayable transition $t \in T \setminus T_D$ are greater or equal than the duration $\Delta(t) \geq d$.

2.2 Graphical representation and example

A Petri net is a directed bipartite graph, in which the transitions are represented by boxes (or bars), places are represented by circles and backward forward incidence functions (pre and post conditions) of transitions are represented by arrows. Moreover, we use the following notations: the delay of a transition is in red and a delayable transition is in grey.

Let us consider the DTPN of Fig. 1. To simplify the notation we note a marking as a set of marked places (instead of a vector), and we give the valuation Δ only for the enabled transitions. For example, in the initial state, we have tokens in places p_0 and p_1, and only two enabled transitions t_0 and t_1 with a delay of 1, then the initial state is noted $q_0 = \begin{matrix}\{p_0, p_1\}\\ \Delta(t_0) = 1\\ \Delta(t_1) = 1\end{matrix}$

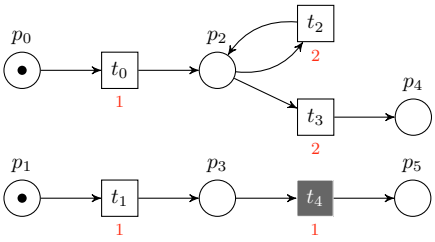

Fig. 1. A Delayable Timed Petri Net example

Some runs of the DTPN of Fig. 1 from the initial state q_0 are given in Fig. 2.

Notice that, as the transition t_4 is delayable, it can fire either alone when $\Delta(t_4) = 0$ (in q_3), or it can wait to fire with t_2 and t_3 when $\Delta(t_4) = -\infty$ (from q_4). But when its delay is missed $\Delta(t_4) = -\infty$, and no other transition can fire at its delay, then t_4 can no more fire (in q_5).

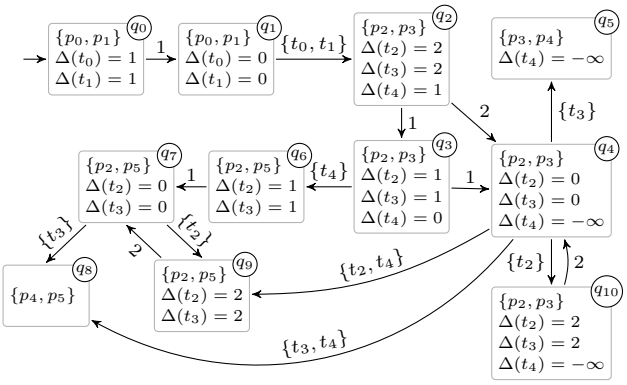

Fig. 2. Part of the state graph of DTPN of Fig. 1

3. EXPRESSIVENESS

In this section, we will compare DTPN with different classes of Petri Nets with time:

- Time Petri Net "à la Merlin" (TPN) where a dense time interval is associated with each transition. Since a transition is enabled, it can fire after elapsing a duration in the associated interval and must fire before the upper bound. The formal semantics is given in (Berthomieu and Diaz, 1991).
- TPN with discrete time semantics (TPN^{dt}) where transition can fire only for integer values of the elapsing time as studied in (Popova, 1991).
- TPN with a maximal-step semantics (TPN_{ms}) where, in a single execution step, a largest possible set of enabled transitions, not in conflict, will fire simultaneously.
- TPN with discrete time and maximal-step semantics ($\text{TPN}^{\text{dt}}_{\text{ms}}$).

3.1 Weak Timed bisimulation

Two TTS S_1 and S_2 are *strongly (timed) bisimilar* if there exists a strong (timed) bisimulation relation between S_1 and S_2. We use the classical definition of Strong Timed bisimulation as in (Bérard et al., 2005).

Let $\mathcal{A}$ be a set of actions (or alphabet). We denote $\mathcal{A}_\varepsilon = \mathcal{A} \cup \{\varepsilon\}$ with $\varepsilon \notin \mathcal{A}$, where ε is an invisible action (empty word).

Let $S = (Q, q_0, \mathcal{A}_\varepsilon, \rightarrow)$ be a TTS. We define the ε-abstract TTS $S^\varepsilon = (Q, q_0, \mathcal{A}, \rightarrow_\varepsilon)$ (with no ε-transitions) by:

- $q \xrightarrow{d}_\varepsilon q'$ with $d \in \mathbb{R}_{\geq 0}$ iff there is a run in S from q to q' of duration d (possibly 0) with only ε actions,
- $q \xrightarrow{a}_\varepsilon q'$ with $a \in \mathcal{A}$ iff there is a run in S from q to q' of duration 0 with a unique visible action a.

Let $S_1 = (Q_1, q_0^1, \mathcal{A}_\varepsilon, \rightarrow_1)$ and $S_2 = (Q_2, q_0^2, \mathcal{A}_\varepsilon, \rightarrow_2)$ be two TTS. S_1 and S_2 are *weakly (timed) bisimilar* if there exists a strong (timed) bisimulation relation between S_1^ε and S_2^ε.

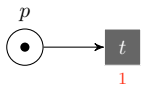

Fig. 3. The DTPN $\mathcal{N}_0$

3.2 TPN vs DTPN

First we prove that bounded TPN, TPN_{ms}, TPN^{dt} and $\text{TPN}^{\text{dt}}_{\text{ms}}$ are not more expressive than bounded DTPN *w.r.t.* weak timed bisimilation.

For all these models, their semantics are TTS, and their state are pairs (m, ν) where m is a marking and ν is a representation of the enabling time of transitions.

We first recall the lemma of (Bérard et al., 2005) stating that *Waiting Cannot Disable Transitions* in TPN and the same result applies for TPN_{ms}, TPN^{dt} and $\text{TPN}^{\text{dt}}_{\text{ms}}$.

Lemma 3. (Waiting Cannot Disable Transitions).
Let (m, ν) be a state of a TPN. If $(m, \nu) \xrightarrow{t_1 t_2 \cdots t_k}$ with $t_1 t_2 \cdots t_k$ an instantaneous firing sequence and $(m, \nu) \xrightarrow{d} (m_d, \nu_d)$ for some $d \geq 0$, then $(m_d, \nu_d) \xrightarrow{t_1 t_2 \cdots t_k}$.

Lemma 4. There is no TPN, TPN_{ms}, TPN^{dt} or $\text{TPN}^{\text{dt}}_{\text{ms}}$ weakly timed bisimilar to the DTPN $\mathcal{N}_0$ (Fig. 3).

Proof. Assume there is a TPN $\mathcal{N}$ that is weakly timed bisimilar to $\mathcal{N}_0$ and let $\approx$ be a weak timed bisimulation between their semantics $S_{\mathcal{N}}$ and $S_{\mathcal{N}_0}$. Let (m_0, ν_0) be the initial state of $S_{\mathcal{N}}$ and $(M_0, 1)$ the initial state of $S_{\mathcal{N}_0}$. We have $(M_0, 1) \xrightarrow{1} (M_0, 0)$ and $(m_0, \nu_0) \xrightarrow{1}_{\varepsilon} (m_1, \nu_1)$ with $(M_0, 1) \approx (m_0, \nu_0)$ and $(M_0, 0) \approx (m_1, \nu_1)$. As t can be fired from $(M_0, 0)$ all the states (m'_1, ν'_1) reachable from (m_1, ν_1) in null duration (ε transitions) can fire an instantaneous sequence labelled t. Since t is a delayable transition, there must be one such state (m'_1, ν'_1) s.t. some duration $d > 0$ can elapse from (m'_1, ν'_1) reaching (m'', ν'') weakly timed bisimilar to $(M_0, 0-d)$ (denoted $(M_0, -\infty)$) which prevents a t to be fired. However, by Lemma 3, some instantaneous sequence labelled by t can be fired from (m'', ν'') which is a contradiction. The reasoning is the same for TPN_{ms}, TPN^{dt} and $\text{TPN}^{\text{dt}}_{\text{ms}}$. □

We now study the relationship in the other direction.

For DTPN, a discrete transition can occur only when at least one transition has been enabled for a duration $\delta \in \mathbb{N}$, then all discrete transitions occur at an integer value of time, and we have the following lemma.

Lemma 5. Let $\mathcal{N}$ be a DTPN and $\rho = q_1 \xrightarrow{\tau_1} q_2 \xrightarrow{d} q_3 \xrightarrow{\tau_2} q_4$ be a sequence in its semantics $S_{\mathcal{N}}$ with $\tau_1 \subseteq T$, $d \in \mathbb{R}$ and $\tau_2 \subseteq T$, then $d \in \mathbb{N}$.

We can then state the following theorem.

Theorem 6. DTPN and TPN are incomparable *w.r.t.* weak timed bisimularity, as are DTPN and TPN_{ms}.

Proof. In TPN and TPN_{ms}, a transition can be fired at any time in dense time interval which, from Lemma 5, cannot happen in a DTPN. Lemma 4 ends the proof. □

We now focus on TPN^{dt} and $\text{TPN}^{\text{dt}}_{\text{ms}}$ that have discrete-time semantics.

The Timed State Graph of bounded TPN^{dt} and $\text{TPN}^{\text{dt}}_{\text{ms}}$ can be written as a tight Durational Kripke structure (DKS) (Alur and Henzinger, 1994; Popova, 1998) where the set of labels is the set T of transitions for a TPN^{dt} or is in 2^T for a $\text{TPN}^{\text{dt}}_{\text{ms}}$. This DKS can be translated into a DTPN (with labels and with a unique token) weakly timed bisimilar to the initial net. For lack of space, we do not formalise the translation nor the bisimulation relationship, but we illustrate the translation in Figure 4 from the TPN^{dt} $\mathcal{N}_1$ into the DTPN on the right where all transitions are delayable except those corresponding to an upper bound of $\mathcal{N}_2$ (as t_1 from m_1 in figure 4). Moreover, each transition (such as t_0) in the initial net with an infinite upper bound leads to an ε loop in one time unit associated with an immediate delayable transition labelled with t_0 (as from m_4 in figure 4).

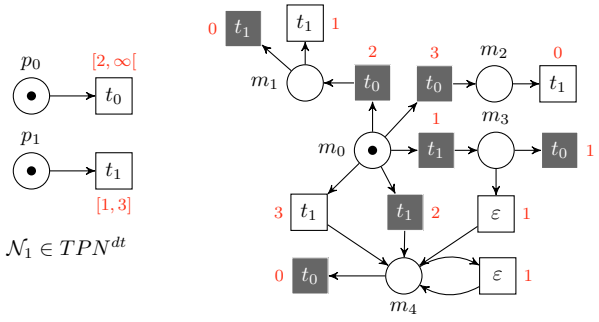

Fig. 4. A TPN^{dt} $\mathcal{N}_1$ and its translation into DTPN

Theorem 7. Bounded DTPN are strictly more expressive than bounded TPN^{dt} and $\text{TPN}^{\text{dt}}_{\text{ms}}$ *w.r.t.* weak timed bisimularity.

Proof. The previous translation shows that every bounded TPN^{dt} or $\text{TPN}^{\text{dt}}_{\text{ms}}$ can be translated into a weak timed bisimilar DTPN. Lemma 4 completes the strict relation. □

4. SYMBOLIC STATE SPACE

4.1 Symbolic semantics

The semantics of DTPN is a time transition system where the states are pairs of a marking and a dynamics value of delays. A symbolic abstraction is proposed in this section in order to regroup the states. All states reachable from a given state by time progression can be grouped together to constitute a symbolic state.

Let $q = (M, \Delta)$ and $q' = (M', \Delta')$ be two states such that q' is reachable from q by d time units. According to the DTPN semantics, it holds that $M = M'$ and for each transition $t \in enab(M), \Delta'(t) = \Delta(t) \ominus d$. It follows that q simulates q', which means that every discrete run feasible from q' is also feasible from q. Consequently, a symbolic state can be represented by the greatest state it contains, in terms of dynamics value of delays.

Definition 8. (Symbolic state). A symbolic state is a pair $(M, \widehat{\Delta})$ where M is a marking and $\widehat{\Delta}$ is a set of dynamic delays defined by:

$$(M,\widehat{\Delta}) = \{(M,\Delta') \mid \forall t \in enab(M),\ \Delta'(t) = \Delta(t) \ominus d, \text{ with } d \in \mathbb{R}_{\geq 0} \text{ and if } enab(M) \setminus T_D \neq \emptyset, d \leq \min_{t \in enab(M)\setminus T_D}(\Delta(t))\}$$

The symbolic state $(M,\widehat{\Delta})$ holds all the states reachable from the state (M,Δ) by elapsing time.

Definition 9. (Successor symbolic states). Let $(M,\widehat{\Delta})$ be a symbolic state and τ be a maximal-step of $(M,\widehat{\Delta})$. The successor symbolic state of $(M,\widehat{\Delta})$ by τ is the symbolic state $(M',\widehat{\Delta'})$ s.t.

$$(M,\Delta) \xrightarrow{d} (M,\Delta_d) \text{ and } (M,\Delta_d) \xrightarrow{\tau} (M',\Delta'),$$

where $d = \max_{t\in\tau}(\Delta(t))$. This successor is denoted by $\mathsf{Succ}((M,\widehat{\Delta}),\tau)$.

For a DTPN $\mathcal{N}$ with an initial state (M_0,Δ_0) we build the symbolic state graph by applying iteratively the operator Succ starting from the initial symbolic state $(M_0,\widehat{\Delta_0})$, and converging by equality of symbolic state. The set of reachable symbolic states from the initial symbolic state is $\mathsf{Reach}(\mathcal{N})$.

Lemma 10. Let $\mathcal{N}$ be a bounded DTPN. $\mathsf{Reach}(\mathcal{N})$ is finite.

Proof. First, the number of reachable marking is bounded. Second, from Lemma 5 we deduce that there are a finite number of states from which maximal-steps can be fired in a symbolic state. Finally, there are a finite number of transitions, and thus finite number of maximal-steps fireable from any state. □

Building the symbolic state graph by converging by inclusion allows regrouping the states even further. For this purpose we define the following subsumption relation over symbolic states.

Definition 11. (Subsumption). Let $(M,\widehat{\Delta})$ and $(M',\widehat{\Delta'})$ be two symbolic states. We say that $(M,\widehat{\Delta})$ is subsumed by $(M',\widehat{\Delta'})$, and we write $(M,\widehat{\Delta}) \sqsubseteq (M',\widehat{\Delta'})$ iff all the states of $(M,\widehat{\Delta})$ are reachable from $(M',\widehat{\Delta'})$ by elapsing time.

The subsumption relation defines a simulation: if $(M,\widehat{\Delta}) \sqsubseteq (M',\widehat{\Delta'})$ then all the states in $(M,\widehat{\Delta})$ are simulated by $(M',\widehat{\Delta'})$.

4.2 Covering steps

The aim of this section is to reduce the number of maximal-steps to be explored from each symbolic state while preserving markings and firing sequences of steps. The following theorem establishes that according to the semantics of the DTPN, a maximal-step τ_1 covers any larger maximal-step τ_2. Consequently, there is no need to explore the maximal-steps larger than τ_1.

Theorem 12. Let $q = (M,\Delta)$ be a state, τ_1 and τ_2 two maximal-steps firable from q such that $\tau_1 \subset \tau_2$. According to the semantics, the transitions within $\tau_2 \setminus \tau_1$ are all delayable. Let $q_1 = (M_1,\Delta_1)$ and $q_2 = (M_2,\Delta_2)$ be the successors of $q = (M,\Delta)$ by τ_1 and τ_2, respectively.
Then the step $\tau_2 \setminus \tau_1$ is firable from $q_1 = (M_1,\Delta_1)$ and this firing leads to a state $q_1' = (M_1',\Delta_1')$ such that $q_1' = (M_1',\Delta_1')$ simulates q_2, in fact $q_1' = q_2$.

Figure 5 depicts this situation.

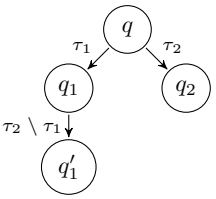

Fig. 5. Covering steps

Proof. To begin with, the associativity of addition over vectors gives $M_1' = M_2$.
Firstly, if a transition t is disabled in q_1', it is also the case in q_2, and thus $\Delta_1'(t) = \Delta(t) = \Delta_2(t)$.
Secondly, if a transition is newly enabled from M by τ_2, then it is newly enabled from M by τ_1 or from M_1 by $\tau_2 \setminus \tau_1$, formally: $\uparrow enab(M,\tau_2) \subseteq \uparrow enab(M,\tau_1) \cup \uparrow enab(M_1,\tau_2\setminus\tau_1)$. Indeed, let $t \in \uparrow enab(M,\tau_2)$. If $t \in \uparrow enab(M,\tau_1)$ then $\Delta_1(t) = \delta(t)$, and as t stays enabled in q_1' then $\Delta_1'(t) = \delta(t) = \Delta_2(t)$. If $t \in \uparrow enab(M_1,\tau_2\setminus\tau_1)$ then $\Delta_1'(t) = \delta(t)$ then $\Delta_1'(t) = \delta(t) = \Delta_2(t)$.
Finally, if a transition is enabled in q but not newly enabled from M by τ_2 then it stays enabled in q_1, and it is neither newly enabled from M by τ_1 nor from M_1 by $\tau_2 \setminus \tau_1$, formally: $enab(M) \cap \overline{\uparrow enab(M,\tau_2)} \subseteq enab(M_1) \cap \overline{\uparrow enab(M,\tau_1)} \cap \overline{\uparrow enab(M_1,\tau_2\setminus\tau_1)}$. Let $t \in enab(M) \cap \overline{\uparrow enab(M,\tau_2)}$, then $\Delta_1(t) = \Delta_1'(t) = \Delta(t) = \Delta_2(t)$. □

Remark 13. The step $\tau_2 \setminus \tau_1$ may not be a maximal-step from q_1, because it can contain only delayed transitions, which is forbidden by the semantics. However, firing it leads to a reachable state. Thus, this new discrete transition can be added to the semantics without altering the reachability.

5. COMPLEXITY

This section focuses on establishing the complexity of exploring the symbolic state graph of Delayable Timed Petri Net.

Symbolic state space of timed models are usually based on difference bound matrix (DBM) representing the union of timed domains as in state class (Berthomieu and Diaz, 1991) for Merlin's TPN. The computation of a successor state class, based on the Floyd-Warshall's algorithm, is, in general, of complexity $O(n^3)$ with regard to the number n of transitions in the net. In (Boucheneb and Rakkay, 2007), the authors have reduced this complexity to $O(n^2)$.

In the case of the DTPNs, DBMs are not necessary because as shown in Section 4, only one value per transition is sufficient to describe the symbolic state. Thus, the data structure of the vector of dynamic delays Δ greatly simplifies the computation of successor.

Theorem 14. Let $\mathcal{N}$ be an DTPN, and $(M,\widehat{\Delta})$ a symbolic state of $\mathcal{N}$. The successor of $(M,\widehat{\Delta})$ by a maximal-step τ, $\mathsf{Succ}((M,\widehat{\Delta}),\tau)$ can be computed in $O(n)$ with n the number of transitions.

Proof. In Definition 9, the delay d can be computed in $|\tau|$ steps (calculation of a maximum), the delay transition can be computed in $|enab(M)|$ steps, and the discrete transition can be computed in $|P| \times |\tau|$ steps.

Remark 15. In fact, the computation of the maximal-steps doable from a symbolic state $(M, \widehat{\Delta})$ is very complex. In the real implementation, we first compute the delays d that lead to states where maximal-steps can be fired: basically we calculate the minimum of $\Delta(t)$ among the non-delayable enabled transitions, then all the $\Delta(t)$ among delayable enabled transitions that are inferior to it. Then we compute the states obtained by elapsing the delays, and for each one we compute the maximal-steps. Yet, the computation of symbolic successor remains of complexity $O(n)$.

Eventhough the successors can be computed efficiently, the size of the symbolic state graph remains exponential in the number of transitions, as for Timed Petri Net. The complexity of the algorithm building the symbolic state graph is then clearly not linear.

However, thanks to the data structure of the vector of dynamic delays, some required algorithms are of complexity $O(n)$ with n the number of transitions. This is the case of the computation of subsumption relation between symbolic states of Definition 11.

6. EXPERIMENTS

We have implemented the symbolic abstraction proposed in Section 4.1 and the approach minimizing the explored edges proposed in Section 4.2. We compare the simple symbolic abstraction with the symbolic abstraction with edge minimisation on two examples.

First we look at the classical level-crossing problem proposed in (Berthomieu and Vernadat, 2003). The net is adapted to the DTPN model by using two parallel transitions instead of one transition with the interval semantics: one delayable for the minimum bound, and one non-delayable for the maximum bound. The example can be easily scaled up by adding more trains. The results are presented in Table 1.

Second we look at the pipeline optimisation approach proposed in (Parrot et al., 2021a). The DTPN model is used to explore various pipeline configurations of arithmetic operators, and a cost allows representing the resources consumed by the pipeline. Note that in this experiment we choose to explore the whole symbolic state graph, but the real goal is to find the run minimizing the cost. The circuits are generated using FloPoCo (de Dinechin and Pasca, 2011), a tool generating floating point arithmetic operators. They are then modeled with a DTPN. The results are displayed in Table 2.

The two experiments show good improvement in the number of edges in the symbolic state graph, and thus in the number of successors computation. The explored edges are reduced up to 24.6%. Note that in some case, the two approaches can produce a different number of symbolic states. This happens when the order of computation of successors is changed, and thus a bigger state (in terms of subsumption) is found sooner. Also note that the dedicated tool is a non-optimal prototype, so the computation time and memory usage are not representative of the complexity. In this implementation, the "Min-edges" approach does not show great gains compared to the simple symbolic exploration.

		Nb trains				
		6	7	8	9	10
Symbolic	Nb States	213	283	363	453	553
	Nb Edges	11 913	33 666	129 547	365 556	1 396 313
Symbolic	Nb States	213	283	363	453	553
Min-edges	Nb Edges	10 159	30 955	114 329	341 506	1 192 241
	Improvement	**14.7%**	**8.0%**	**11.7%**	**6.6%**	**14.6%**

Table 1. Level crossing

		FPAdd(8,23) 500 MHz	FPDiv(8,23) 500 MHz	FPSqrt(8,23) 500 MHz
Symbolic	Nb States	1695	329	540
	Nb Edges	11 150	901	2 095
Symbolic	Nb States	1693	328	540
Min-edges	Nb Edges	8 411	831	1 728
	Improvement	**24.6%**	**7.8%**	**17.5%**

Table 2. Arithmetic operators pipelining

7. CONCLUSION

We have studied an extension of Timed Petri Nets well suited to solve synchronous circuit design problems, in which the transitions are divided into two disjoint sets, corresponding to delayable and non-delayable transitions.

After recalling its semantics, we are interested in its expressiveness compared to the Merlin's model, in which intervals are associated with transitions. We proved that the DTPN is strictly more expressive w.r.t. weak timed bisimilarity than Merlin's model under the discrete-time semantics.

Our semantics of DTPNs is based on dynamic delays decreasing with time allowing us to build a symbolic abstraction that turns out to be efficient. Indeed, the basic algorithms of the symbolic state graph computation such as the successor or the subsumption algorithms have a $O(n)$ complexity. Moreover, we have proposed a reduction of the successor number to be computed that preserves the set of reachable states.

Finally, an implementation of the exploration of the symbolic state space has been developed. It is evaluated on a classical example from the literature, and also on a concrete case of synchronous circuit design.

REFERENCES

Alur, R. and Henzinger, T.A. (1994). A really temporal logic. *Journal of the ACM*, 41(1), 181–203.

Bérard, B., Cassez, F., Haddad, S., Lime, D., and Roux, O.H. (2005). Comparison of the expressiveness of timed automata and time Petri nets. In *3rd International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS 05)*, volume 3829 of *Lecture Notes in Computer Science*, 211–225. Springer.

Berthomieu, B. and Diaz, M. (1991). Modeling and verification of time dependent systems using time Petri nets. *IEEE transactions on software engineering*, 17(3), 259–273.

Berthomieu, B. and Vernadat, F. (2003). State class constructions for branching analysis of time petri nets. In H. Garavel and J. Hatcliff (eds.), *Tools and Algorithms for the Construction and Analysis of Systems*, 442–457. Springer Berlin Heidelberg, Berlin, Heidelberg.

Boucheneb, H. and Rakkay, H. (2007). A more efficient time petri net state space abstraction preserving linear properties. In T. Basten, G. Juhás, and S.K. Shukla (eds.), *Seventh International Conference on Application of Concurrency to System Design (ACSD 2007), 10-13 July 2007, Bratislava, Slovak Republic*, 61–70. IEEE Computer Society.

de Dinechin, F. and Pasca, B. (2011). Designing custom arithmetic data paths with FloPoCo. *IEEE Design & Test of Computers*, 28(4), 18–27.

Dill, D.L. (1989). Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems*, volume 407 of *LNCS*, 197–212. Springer.

Henzinger, T.A., Manna, Z., and Pnueli, A. (1992). What good are digital clocks? In *Proc. 19th Int. Coll. Automata, Languages, and Programming (ICALP'92)*, number 623 in LNCS, 545–558. Springer.

Merlin, P.M. (1974). *A study of the recoverability of computing systems*. Ph.D. thesis, Dep. of Information and Computer Science, University of California, Irvine, CA.

Parrot, R., Briday, M., and Roux, O.H. (2021a). Pipeline Optimization using a Cost Extension of Timed Petri Nets. In *The 28th IEEE International Symposium on Computer Arithmetic (ARITH 2021)*. IEEE.

Parrot, R., Briday, M., and Roux, O.H. (2021b). Timed Petri Nets with Reset for Pipelined Synchronous Circuit Design. In *The 42th International Conference on Application and Theory of Petri Nets and Concurrency (Petri Nets 2021)*, volume 12734 of *Lecture Notes in Computer Science*. Springer.

Popova, L. (1998). Essential States in Time Petri Nets. *Informatik-Berichte*, 96.

Popova, L. (2006). Time Petri Nets State Space Reduction using Dynamic Programming. *Journal of Control and Cynernetics*, 35(3), 721–748.

Popova, L. (1991). On time petri nets. *Journal Inform. Process. Cybern., EIK (formerly: Elektron. Inform. verarb. Kybern.)*, 27(4), 227–244.

Ramchandani, C. (1974). *Analysis of asynchronous concurrent systems by timed Petri nets*. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Sifakis, J. (1977). Use of petri nets for performance evaluation. In *Proceedings of the Third International Symposium on Measuring, Modelling and Evaluating Computer Systems*, 75–93. North-Holland Publishing Co.