

# Symbolic Reachability Analysis of Integer Timed Petri Nets

Min Wan and Gianfranco Ciardo

Department of Computer Science and Engineering  
University of California, Riverside  
{mwan,ciardo}@cs.ucr.edu

**Abstract.** Petri nets are an effective formalism to model discrete event systems, and several variants have been defined to explicitly include real time in the model. We consider two fundamental reachability problems for Timed Petri Nets with positive integer firing times: *timed reachability* (find all markings where the model can be at a given finite time) and *earliest reachability* (find the minimum time when each reachable marking is entered). For these two problems, we define efficient symbolic algorithms that make use of both ordinary and edge-valued decision diagrams, and provide runtime results on an extensive suite of models.

## 1 Introduction

Petri nets are very effective at modeling distributed systems, and have been extensively researched [17]. For systems where timing is an integral part of the dynamics and can affect the logical evolution, various time-related extension of Petri nets have been proposed. Of particular relevance to our present work are the so-called *time Petri nets* [15] and *timed Petri nets* [21], where the durations of events are either known to lie in a given interval, or to be constants.

We consider a class of timed Petri nets where the durations of the transition firing times are integer-valued, but can be chosen from an arbitrary finite set of not necessarily contiguous values. For this class, we explore two fundamental reachability problems: *timed reachability* (find the set of markings where the Petri net can be at a given finite point in time) and *earliest reachability* (find the first instant of time when the Petri net might enter each reachable marking).

These problems can be tackled in principle with explicit methods that explore states one by one, where a “state” describes both the marking of the Petri net and the remaining firing time of each enabled transition. However, even more than for untimed nets, the size of the state space is a formidable obstacle in practice. Following the trend in logical reachability and model checking, where efficient *symbolic* algorithms based on decision diagrams have been widely adopted [8, 18], we develop novel symbolic algorithms based on both ordinary and edge-valued decision diagrams, and demonstrate their effectiveness on a suite of models.

## 2 Background

We now briefly describe the class of timed Petri net we adopt in our work, and recall the definition of the decision diagrams used by our algorithms.

## 2.1 Integer Timed Petri Nets

Several extensions of standard Petri nets [17] have been proposed to explicitly represent a notion of time. Usually, a firing time is associated with each transition of the net. In *time Petri nets* [15], the firing times lie in a given interval  $[t_{min}, t_{max}]$ , where  $0 \leq t_{min} < \infty$  and  $t_{min} \leq t_{max} \leq \infty$ . If  $t_{min} = t_{max}$  for all transitions, i.e., all firing times are constants, as in *timed Petri nets* [21], non-determinism can only arise from the resolution of conflicts among transitions.

The class of nets we consider is a restriction of time and timed Petri nets, as we require firings to occur only at integer times, and an extension, as we allow the firing time to be nondeterministically chosen among a finite set of non-necessarily contiguous values. Such nets have been shown to give rise to discrete-time Markov chains if choices are resolved probabilistically [4]. Formally, *integer timed Petri nets* (ITPNs) are specified by a tuple  $(\mathcal{P}, \mathcal{T}, \mathcal{A}, \mathcal{M}_0, \mathcal{F}, w)$  where:

- $\mathcal{P}$  is a finite, nonempty set of *places*.
- $\mathcal{T}$  is a finite, nonempty set of *transitions*, with  $\mathcal{P} \cup \mathcal{T} \neq \emptyset$  and  $\mathcal{P} \cap \mathcal{T} = \emptyset$ .
- $\mathcal{A} \subseteq \mathcal{P} \times \mathcal{T} \cup \mathcal{T} \times \mathcal{P}$  is a set of directed arcs which connect places to transitions (*input arcs*) and transitions to places (*output arcs*).
- $w : \mathcal{A} \rightarrow \mathbb{N}^+$  specifies a constant positive integer *cardinality* for each arc.
- $\mathcal{F} : \mathcal{T} \rightarrow 2^{\mathbb{N}^+}$  specifies a finite set of integer *firing times* for each transition.
- $\mathcal{M}_0 \subset \mathbb{N}^{\mathcal{P}}$  specifies a finite set of *initial markings*.

It is convenient to define the *input flow matrix* of the net as  $D^- : \mathbb{N}^{\mathcal{P} \times \mathcal{T}}$ , where, for  $p \in \mathcal{P}$  and  $t \in \mathcal{T}$ ,  $D^-[p, t] = w(p, t)$ , if  $(p, t) \in \mathcal{A}$ , and 0 otherwise. If we treat the marking  $\mu \in \mathbb{N}^{\mathcal{P}}$  of the net as a (column) vector, we can then say that a transition  $t \in \mathcal{T}$  is *marking-enabled* in  $\mu$  if  $\mu \geq D^-[\cdot, t]$ . Let  $\mathcal{E}(\mu) \subseteq \mathcal{T}$  be the set of marking-enabled transitions in  $\mu$ . Analogously, we can define the *output flow matrix* as  $D^+ : \mathbb{N}^{\mathcal{P} \times \mathcal{T}}$ , where, for  $p \in \mathcal{P}$  and  $t \in \mathcal{T}$ ,  $D^+[p, t] = w(t, p)$ , if  $(t, p) \in \mathcal{A}$ , and 0 otherwise. Then, the *incidence matrix* of the net is given by  $D = D^+ - D^-$ , and the vector  $D[\cdot, t]$  represents the net effect on the marking when transition  $t$  fires. More precisely, the *firing* of transition  $t \in \mathcal{E}(\mu)$  in marking  $\mu$  changes the marking to  $\mu' = \mu + D[\cdot, t]$ , we write this as  $\mu \xrightarrow{[t]} \mu'$ .

The *state* of an ITPN is a pair  $\langle \mu, \tau \rangle \in \mathbb{N}^{\mathcal{P}} \times (\mathbb{N} \cup \{\infty\})^{\mathcal{T}}$ , where  $\mu$  is the marking and  $\tau$  is the *remaining firing times*, which, for each transition  $t \in \mathcal{T}$ , must satisfy  $\tau[t] \leq \max \mathcal{F}(t)$  if  $t \in \mathcal{E}(\mu)$ , and  $\tau[t] = \infty$  otherwise. We say that a transition  $t \in \mathcal{T}$  is *state-enabled* in  $\langle \mu, \tau \rangle$  if it is marking-enabled in  $\mu$  and its firing time has elapsed,  $\tau[t] = 0$ . Let  $\mathcal{E}(\langle \mu, \tau \rangle) \subseteq \mathcal{E}(\mu)$  be the set of state-enabled transitions in  $\langle \mu, \tau \rangle$ . If the ITPN is in  $\langle \mu, \tau \rangle$  at time  $\theta$ , it evolves as follows:

- If  $\mathcal{E}(\mu) = \emptyset$ , thus  $\tau = \{\infty\}^{\mathcal{T}}$ , the marking  $\mu$ , as well as the state  $\langle \mu, \tau \rangle$ , is *dead*. The net will remain forever in that state.
- If  $\mathcal{E}(\mu) \neq \emptyset$  but  $\mathcal{E}(\langle \mu, \tau \rangle) = \emptyset$ , there are marking-enabled transitions but none of their firing times has elapsed, the state is *tangible*. The net remains in marking  $\mu$  for  $\tau_* = \min_{t \in \mathcal{T}} \{\tau[t]\} > 0$  time units, while the remaining firing times elapse at unit rate. At time  $\theta + \tau_*$ , the state is  $\langle \mu, \tau - \tau_* \rangle$ , where the subtraction of scalar  $\tau_*$  from vector  $\tau$  is interpreted elementwise, i.e.,  $(\tau - \tau_*)[t] = \tau[t] - \tau_*$  for  $t \in \mathcal{T}$ . We write  $\langle \mu, \tau \rangle \xrightarrow{[\tau_*]} \langle \mu, \tau - \tau_* \rangle$ , for  $0 \leq \tau_* \leq \tau_*$ .
- If  $\mathcal{E}(\langle \mu, \tau \rangle) \neq \emptyset$ , there are marking-enabled transitions with elapsed firing times, the state is *vanishing*. The state then immediately changes by firing a nondeterministically chosen *maximally serializable* set  $\mathcal{E}_f \subseteq \mathcal{E}(\langle \mu, \tau \rangle)$  as a single operation, we write  $\langle \mu, \tau \rangle \xrightarrow{[\mathcal{E}_f]} \langle \mu', \tau' \rangle$ , where the new marking is  $\mu' = \mu + \sum_{t \in \mathcal{E}_f} D[\cdot, t]$ , while the remaining firing times  $\tau'$  are chosen so that

- $\tau'[t] = \infty$  if  $t \notin \mathcal{E}(\mu')$ , i.e., if  $t$  is disabled in the new marking,
- $\tau'[t] \in \mathcal{F}(t)$  if  $t \in (\mathcal{E}_f \cap \mathcal{E}(\mu')) \cup (\mathcal{E}(\mu') \setminus \mathcal{E}(\mu))$ , i.e., if  $t$  is newly (re)enabled in the new marking, a firing time is nondeterministically chosen for it.
- $\tau'[t] = \tau[t]$  otherwise, i.e., if the firing time of  $t$  continues to elapse undisturbed in the new marking, it remains unchanged.

A set of  $n$  transitions  $\mathcal{E}_f \subseteq \mathcal{E}(\langle \mu, \tau \rangle)$  is maximally serializable if it is fireable, i.e., its elements can be ordered as  $(t_1, \dots, t_n)$  s.t.  $\mu[t_1] \mapsto \mu_1[t_2] \mapsto \dots \mapsto \mu_n$ , and maximal, i.e., any other transition that was state-enabled in  $\langle \mu, \tau \rangle$  becomes marking disabled in  $\mu_n$ , i.e.,  $(\mathcal{E}(\langle \mu, \tau \rangle) \setminus \mathcal{E}_f) \cap \mathcal{E}(\mu_n) = \emptyset$ .

To make explicit the time evolution of the state, we write  $\langle \mu, \tau @ \theta \rangle$  to signify that the ITPN is in vanishing state  $\langle \mu, \tau \rangle$  at time  $\theta$ , just before the firing of a maximally serializable set  $\mathcal{E}_f$  of transitions, and  $\langle \mu, \tau @ \theta \rangle$  to signify that the ITPN is in tangible state  $\langle \mu, \tau \rangle$  at time  $\theta$ . Thus, a possible sequence of “snapshots” for the evolution of the ITPN from tangible (non-dead) state  $\langle \mu, \tau \rangle$  at time  $\theta$  is

$$\langle \mu, \tau @ \theta \rangle [\tau_1] \mapsto \langle \mu, \tau - \tau_1 @ \theta + \tau_1 \rangle [\tau_2] \mapsto \langle \mu, \tau - \tau_* @ \theta + \tau_* \rangle [\mathcal{E}_f] \mapsto \langle \mu', \tau' @ \theta + \tau_* \rangle,$$

where  $\langle \mu', \tau' \rangle$  is tangible and  $\tau_1 + \tau_2 = \tau_* = \min_{t \in \mathcal{T}} \{\tau[t]\}$ . Then, we write  $\langle \mu_0, \tau_0 @ \theta \rangle [*] \mapsto \langle \mu_n, \tau_n @ \theta' \rangle$  iff, from tangible state  $\langle \mu_0, \tau_0 \rangle$  at time  $\theta$ , the ITPN can reach tangible state  $\langle \mu_n, \tau_n \rangle$  at time  $\theta' = \theta + \sum_{i=1}^n \tau_{i*}$  visiting vanishing states  $\langle \mu_i, \tau_i - \tau_{i*} \rangle$  and tangible states  $\langle \mu_i, \tau_i \rangle$ , where  $\tau_{i*} = \min_{t \in \mathcal{T}} \{\tau_i[t]\}$ , and  $\mathcal{E}_i$  is a maximally serializable set of transitions in  $\langle \mu_i, \tau_i - \tau_{i*} \rangle$  whose firing causes the state change  $\langle \mu_i, \tau_i - \tau_{i*} \rangle [\mathcal{E}_i] \mapsto \langle \mu_{i+1}, \tau_{i+1} \rangle$ , for  $i = 0, \dots, n-1$ .

We stress that the *maximal serializability semantics* we have adopted is similar to but differs from the *maximal step semantics* of [3], which further requires  $\mu \geq \sum_{t \in \mathcal{E}_f} D^-[ \cdot, t ]$ , so that any permutation of all transitions in  $\mathcal{E}_f$  is fireable. It is also similar to, but different from, the *maximal non-blocking semantics* of [20], which simply requires that  $\mu + \sum_{t \in \mathcal{E}_f} D[ \cdot, t ] \geq 0$ , i.e., that the cumulative effect on  $\mu$  of all firings in  $\mathcal{E}_f$  be such that the result is a legal marking. Our semantic is suitable to model concurrent systems which require serializability, e.g., transaction processing systems [1]. Maximal and interleaving semantic differ in a fundamental way: in the former, the firing time of a transition that is disabled for a zero amount of time (if we sequentialize the transitions in  $\mathcal{E}_f$ ) is not reset.

For simplicity, we defined ITPNs with an initial set of markings  $\mathcal{M}_0$  but their analysis requires an initial set of states  $\mathcal{S}_0$ , which we can then define as  $\mathcal{S}_0 = \{ \langle \mu, \tau \rangle : \mu \in \mathcal{M}_0 \text{ and } \tau[t] = \infty \text{ if } t \notin \mathcal{E}(\mu'), \tau[t] \in \mathcal{F}(t) \text{ otherwise} \}$ . Indeed, we could even allow an arbitrary initial setting for the remaining firing times associated to any initial marking. The techniques we present would still be applicable, as long as the set of initial states  $\mathcal{S}_0$  is finite.

The upper left part of Figure 5 shows our ITPN running example, with places  $a$ ,  $b$ , and  $c$  and transitions  $\alpha$ ,  $\beta$ , and  $\gamma$ . The set of initial markings  $\mathcal{M}_0$  contains a single marking,  $a^2$  (we use *bag notation*, i.e.,  $a^2 = a^2 b^0 c^0$  means that  $a$  contains two tokens while  $b$  and  $c$  are empty). The firing time of all transitions can be either 1 or 2. So  $\mathcal{S}_0 = \{ \langle a^2, \alpha^1 \beta^1 \rangle, \langle a^2, \alpha^1 \beta^2 \rangle, \langle a^2, \alpha^2 \beta^1 \rangle, \langle a^2, \alpha^2 \beta^2 \rangle \}$ . where the notation for the remaining firing times is analogous to the one for markings, except that we omit disabled transitions instead of empty places.

<pre> mdd Union(mdd s, mdd u) 1 if s = 0 or s = u then return u; 2 if u = 0 then return s; 3 if CacheHit(UNI, s, u, r) then return r; 4 r ← NewNode(s<sub>v</sub>); 5 foreach i ∈ V<sub>s<sub>v</sub></sub> do r[i] ← Union(p[i], u[i]); 6 CacheAdd(UNI, s, u, Unique(r)); 7 return r; </pre>	<pre> 3 if CacheHit(DIF, s, u, r) then return r; 4 r ← NewNode(p<sub>v</sub>); 5 foreach i ∈ V<sub>p<sub>v</sub></sub> do 6   r[i] ← Difference(s[i], u[i]); 7 CacheAdd(DIF, s, u, Unique(r)); 8 return r; </pre>
<pre> mdd Intersection(mdd s, mdd u) 1 if s = u or u = 0 then return u; 2 if s = 0 then return s; 3 if CacheHit(INT, s, u, r) then return r; 4 r ← NewNode(s<sub>v</sub>); 5 foreach i ∈ V<sub>s<sub>v</sub></sub> do 6   r[i] ← Intersection(s[i], u[i]); 7 CacheAdd(INT, s, u, Unique(r)); 8 return r; </pre>	<pre> evmdd Minimum(evmdd ⟨α, s⟩, ⟨β, u⟩) 1 if α = ∞ then return ⟨β, u⟩; 2 if β = ∞ then return ⟨α, s⟩; 3 γ ← min{α, β}; 4 if s = u then return ⟨γ, s⟩; 5 if α &lt; β then Swap(⟨α, s⟩, ⟨β, u⟩) 6 if CacheHit(MIN, α - β, s, u, r) then 7   return ⟨γ, r⟩; 8 r ← NewNode(s<sub>v</sub>); 9 foreach i ∈ V<sub>s<sub>v</sub></sub> do 10  α' ← α - γ + s[i].val; β' ← u[i].val; 11  r[i] ← Minimum(⟨α', s[i]⟩, ⟨β', u[i]⟩); 12 CacheAdd(MIN, α - β, s, u, Unique(r)); 13 return ⟨γ, r⟩; </pre>
<pre> mdd Difference(mdd s, mdd u) 1 if s = 0 or s = u then return 0; 2 if u = 0 then return s; </pre>	

Fig. 1. Standard MDD and EV<sup>+</sup>MDD operations.

## 2.2 Decision Diagrams

Since the introduction of reduced ordered binary decision diagrams (BDDs) [2], further variations of decision diagrams have been proposed in the literature to compactly encode and efficiently compute functions on structured sets. We now outline the two types of decision diagrams used in our work.

**Multiway decision diagrams [14].** Given  $N$  variables  $\mathbf{v} = (v_N, \dots, v_1)$  with an order  $v_N \succ \dots \succ v_1$ , each  $v_k$  taking value in a finite set  $\mathcal{V}_{v_k} = \{0, 1, \dots, n_{v_k}\} \subset \mathbb{N}$ , a (quasi-reduced) *multiway decision diagram (MDD)* defined on  $\mathbf{v}$  is a directed acyclic edge-labeled multi-graph where:

- Each nonterminal node  $s$  is associated to a variable  $s_v = v_k \in \{v_N, \dots, v_1\}$ .
- $\mathbf{0}$  and  $\mathbf{1}$  are the *terminal* nodes. Let  $\mathbf{0}_v = \mathbf{1}_v = v_0$  and  $v_k \succ v_0$ , for  $N \geq k \geq 1$ .
- Each nonterminal node  $s$  associated with  $v_k$  has  $|\mathcal{V}_{v_k}|$  edges, labeled with a different index  $i \in \mathcal{V}_{v_k}$  and pointing to a node  $u$  with  $u_v = v_{k-1}$  or  $u = \mathbf{0}$ , we write  $s[i] = u$ . At least one edge does not point to  $\mathbf{0}$ .
- *Duplicate* nodes are not allowed, i.e., given two distinct nonterminal nodes  $s$  and  $u$  with  $s_v = u_v$ , there must be an index  $i \in \mathcal{V}_{s_v}$  such that  $s[i] \neq u[i]$ .

Each node  $s$ , with  $s_v = v_k$ , encodes a set of tuples:  $\mathcal{B}(\mathbf{0}) = \emptyset$ ,  $\mathcal{B}(\mathbf{1}) = \{()\}$ , the set containing only the empty tuple, and, for  $N \geq k \geq 1$ ,  $\mathcal{B}(s) = \bigcup_{i=0}^{n_{v_k}} \{i\} \times \mathcal{B}(s[i])$ . These MDDs are canonical [8, 14]: given a set  $\mathcal{X} \subseteq \mathcal{V}_{v_N} \times \dots \times \mathcal{V}_{v_1}$ , there is a unique MDD  $p = \text{enc}(\mathcal{X})$  satisfying  $\mathcal{X} = \mathcal{B}(p)$  and  $p_v = v_N$  (except for  $\mathcal{X} = \emptyset$ , where  $p = \mathbf{0}$ , thus  $p_v = v_0$ ). In Fig. 5, at position  $(0, \mathcal{S}_\theta)$  is the MDD encoding the set of initial states of our running model, where only edges not pointing to  $\mathbf{0}$  are displayed.

**Additive edge-valued multiway decision diagrams [9].** A (quasi-reduced) *additive edge-valued multiway decision diagram (EV<sup>+</sup>MDD)* defined on  $\mathbf{v}$  is a directed acyclic edge-labeled multi-graph where:

- Each nonterminal node  $s$  is associated to a variable  $s_v = v_k \in \{v_N, \dots, v_1\}$ .

<pre> mdd GenerateByBFS(mdd s) 1 repeat s ← Union(s, RelProd(s, r<sub>all</sub>)); 2 until s does not change; 3 return s; </pre>	<pre> mdd RelProd(mdd s, mdd r) 1 if s = 1 and r = 1 then return 1; 2 if CacheHit(RPR, s, r, u) then return u; 3 u ← NewNode(s<sub>v</sub>); 4 foreach i ∈ V<sub>s<sub>v</sub></sub> s.t. s[i], r[i] ≠ 0 do 5   foreach i' ∈ V<sub>r[i]<sub>v</sub></sub> s.t. r[i][i'] ≠ 0 do 6     d ← RelProd(s[i], r[i][i']); 7     u[i'] ← Union(u[i'], d); 8 CacheAdd(RPR, s, r, Unique(u)); 9 return u; </pre>
<pre> mdd Saturate(mdd s) 1 if CacheHit(SAT, s, u) then return u; 2 u ← NewNode(s<sub>v</sub>); 3 foreach i ∈ V<sub>s<sub>v</sub></sub> s.t. s[i] ≠ 0 do 4   u[i] ← Saturate(s[i]); • saturate   nodes below first 5 u ← DoFixPoint(u); 6 CacheAdd(SAT, s, Unique(u)); 7 return u; </pre>	<pre> mdd RelProdSat(mdd s, mdd r) 1 if s = 1 and r = 1 then return 1; 2 if CacheHit(RPRS, s, r, u) then return u; 3 u ← NewNode(s<sub>v</sub>); 4 foreach i ∈ V<sub>s<sub>v</sub></sub> s.t. s[i], r[i] ≠ 0 do 5   foreach i' ∈ V<sub>r[i]<sub>v</sub></sub> s.t. r[i][i'] ≠ 0 do 6     d ← RelProdSat(s[i], r[i][i']); 7     u[i'] ← Union(u[i'], d); 8 u ← Saturate(u); 9 CacheAdd(RPRS, s, r, u); 10 return u; </pre>
<pre> mdd DoFixPoint(mdd u) 1 repeat 2   foreach t ∈ T s.t. top(t) = u<sub>v</sub> 3     foreach i ∈ V<sub>u<sub>v</sub></sub> s.t. u[i], r<sub>t</sub>[i] ≠ 0 do 4       foreach i' ∈ V<sub>r<sub>t</sub>[i]<sub>v</sub></sub> s.t. r<sub>t</sub>[i][i'] ≠ 0 do 5         d ← RelProdSat(u[i], r<sub>t</sub>[i][i']); 6         u[i'] ← Union(u[i'], d); 7 until u does not change; </pre>	

**Fig. 2.** State-space generation for untimed Petri nets: BFS and saturation.

- $\Omega$  is the only *terminal* node. Let  $\Omega_v = v_0$  and  $v_k \succ v_0$ , for  $N \geq k \geq 1$ .
- Each nonterminal node  $s$  associated with  $v_k$  has  $|\mathcal{V}_{v_k}|$  edges, labeled with a different index  $i \in \mathcal{V}_{v_k}$  and associated to a value  $\mathbb{N} \cup \{\infty\}$ . We write  $s[i] = \langle \rho, u \rangle = \langle s[i].val, s[i].node \rangle$  if the edge labeled by  $i$  points to node  $u$  and is associated to  $\rho$ , and require that  $u = \Omega$  if  $\rho = \infty$ , and  $u_v = v_{k-1}$  otherwise. Also, at least one edge must have an associated value equal to 0.
- *Duplicate* nodes are not allowed: given two distinct nonterminal nodes  $s$  and  $u$  with  $s_v = u_v = v_k$ , there must be an index  $i \in \mathcal{V}_{v_k}$  such that  $s[i] \neq u[i]$ .

The function encoded by edge  $\langle \rho, s \rangle$ , with  $s_v = v_k$ , is recursively defined by

$$\forall (i_k, \dots, i_1) \in \mathcal{V}_{v_k} \times \dots \times \mathcal{V}_{v_1}, f_{\langle \rho, s \rangle}(i_k, \dots, i_1) = \begin{cases} \rho + f_{s[i_k]}(i_{k-1}, \dots, i_1) & \text{if } \rho \in \mathbb{N} \\ \infty & \text{if } \rho = \infty. \end{cases}$$

EV<sup>+</sup>MDDs are canonical [9]. Given a function  $f : \mathcal{V}_{v_k} \times \dots \times \mathcal{V}_{v_1} \rightarrow \mathbb{N} \cup \{\infty\}$ , there is a unique value  $\rho \in \mathbb{N}$  and a unique node  $r$  with  $r_v = v_k$  such that  $f_{\langle \rho, r \rangle} = f$ , with the exception of the constant function  $f \equiv \infty$ , for which we have the special case  $\rho = \infty$  and  $r = \Omega$ . In Fig. 5, at position  $(0, \mathcal{M}_\theta)$  is the EV<sup>+</sup>MDD encoding the function  $f(c^0, b^0, a^2) = 0$  for our running model, omitting edges valued  $\infty$ .

**State-space generation for untimed Petri nets.** We recall symbolic state-space generation approaches for untimed Petri nets under interleaving semantics. These nets correspond to ignoring the  $\mathcal{F}$  component in the ITPN, thus identifying the state with the marking. The *transition relation* due to transition  $t \in \mathcal{T}$  is  $\mathcal{R}_t = \{(\mu, \mu') : \mu [t] \Rightarrow \mu'\}$  and the *overall* transition relation is  $\mathcal{R} = \bigcup_{t \in \mathcal{T}} \mathcal{R}_t$ . The set of reachable markings  $\mathcal{M} \subseteq \mathbb{N}^P$  is the minimal set satisfying  $\mathcal{M} \supseteq \mathcal{M}_0$  and  $\mathcal{M} \supseteq \text{Img}(\mathcal{M}, \mathcal{R})$ , where  $\text{Img}(\mathcal{X}, \mathcal{Z}) = \{\mu' : \exists \mu \in \mathcal{X}, (\mu, \mu') \in \mathcal{Z}\}$  denotes the *image* of the set of markings  $\mathcal{X}$  under the binary relation  $\mathcal{Z}$ . We can start

from  $\mathcal{M}_0$  and repeatedly perform image computations under  $\mathcal{R}$ , or  $\mathcal{R}_t$  as long as each of them is applied often enough, until reaching a fixpoint.

We use an MDD on  $\mathbf{x} = (x_L, \dots, x_1)$ , with  $L = |\mathcal{P}|$  to encode  $\mathcal{X}$ , where each variable encodes number of tokens in a different place. The transition relation  $\mathcal{R}_t$  is instead encoded by a  $2L$ -variable MDD defined on  $(\mathbf{x}, \mathbf{x}') = (x_L, x'_L, \dots, x_1, x'_1)$ , where the unprimed variables refer to the “from” markings, the primed variables refer to the “to” markings, and two are *interleaved*. As firing transition  $t$  only affects or is affected by its input and output places, we denote with  $top(t)$  the maximum variable according to “ $\succ$ ” on which  $\mathcal{R}_t$  depends. Once  $\mathcal{X}$  and  $\mathcal{Z}$  are encoded as MDDs, the MDD encoding  $Img(\mathcal{X}, \mathcal{Z})$  is obtained as the *relational product* of these two MDDs.

Two classes of symbolic state-space generation algorithms exist: *breadth-first* [18] and *saturation* [6–8, 10]. Fig. 2 shows pseudocode for both, assuming that the  $2L$ -variable MDDs  $r_{all}$  and  $r_t$ , for  $t \in \mathcal{T}$ , encode the overall transition relation and the transition relation for each transition of the net, respectively. *GenerateByBFS* is the breadth-first algorithm while *Saturate* shows the saturation approach. The latter differs from the former in that it recursively computes a fixpoint at each node, in a low-to-high order with respect to the variables. Saturation uses *RelProdSat* instead of *RelProd*, which differ in that *RelProd*( $s, r$ ) only computes a *one-step* image of the set encoded by  $s$  and the relation encoded by  $r$ , while *RelProdSat*( $s, r$ ) returns the *fixpoint* of the computed image with respect to any transition  $t$  with  $top(t) = x_h$ , for  $s_v \succeq x_h$ . Experimentally, saturation-based algorithms perform far better than breadth-first methods.

## 3 Reachability analysis of ITPNs

### 3.1 Timed reachability

The timed reachability problem aims at finding all markings where the model can be at a finite time  $\theta_f \in \mathbb{N}$ . Formally, we seek the set of markings

$$\{\mu' : \exists \langle \mu, \tau \rangle \in \mathcal{S}_0, \langle \mu, \tau @ 0 \rangle [*] \Rightarrow \langle \mu', \tau' @ \theta \rangle, \theta \leq \theta_f < \theta + \min_{t \in \mathcal{T}} \{\tau'[t]\}\}.$$

For ITPN models, this section presents an efficient solution that uses symbolic manipulations. We stress that, while the problem statement focuses on *markings*, the reachability algorithms must process *states*, since the remaining firing time information is essential in determining the future evolution of the ITPN.

**Preliminaries.** We use a global variable  $\theta$  to keep track of time. If the ITPN is in tangible state  $\langle \mu, \tau \rangle$  at time  $\theta = 0$  and  $\tau_* = \min_{t \in \mathcal{T}} \{\tau(t)\}$ , the set  $\mathcal{S}_{\theta_f}$  of tangible states in which the ITPN can be at time  $\theta_f$  is determined as follows:

- If  $\theta_f < \tau_*$ , the ITPN will still be in the same marking at time  $\theta_f$ .
- If  $\theta_f = \tau_*$ , the ITPN reaches vanishing state  $\langle \mu, \tau - \tau_* @ \tau_* \rangle$  at time  $\theta_f$ , then immediately moves to a tangible state by firing a nondeterministically chosen  $\mathcal{E}_f$ , i.e.,  $\mathcal{S}_{\theta_f} = \{\langle \mu', \tau' \rangle : \langle \mu, \tau @ 0 \rangle [\tau_*] \Rightarrow \langle \mu, \tau - \tau_* @ \tau_* \rangle [\mathcal{E}_f] \Rightarrow \langle \mu', \tau' \rangle\}$ .
- If  $\theta_f > \tau_*$ , repeat these steps starting from each state in  $\mathcal{S}_{\tau_*}$  at time  $\theta = \tau_*$ .

The first case is trivial, as it generates no new markings. For the second case, we need to generate the set of tangible states  $\langle \mu', \tau' \rangle$  reachable from the vanishing state  $\langle \mu, \tau - \tau_* \rangle$  by firing all possible  $\mathcal{E}_f$ . This is a fixpoint iteration analogous to the one for state-space generation of untimed nets, but restricted to applying each transition in  $\mathcal{E}(\langle \mu, \tau - \tau_* \rangle)$  at most once. Furthermore, unlike ordinary state-space generation, we are only interested in collecting the *tangible*

*frontier*, i.e., the states where any transition in  $\mathcal{E}(\langle\mu, \tau - \tau_*\rangle)$  that has not yet been fired is now marking-disabled. The key ideas to apply the symbolic state-space generation algorithms of Sec. 2.2, in particular saturation, are to treat a remaining firing time of 0 as part of the enabling condition, so that only transitions in  $\mathcal{E}(\langle\mu, \tau - \tau_*\rangle)$  can be enabled, and to temporarily set the remaining firing time of any fired transitions to  $\infty$ , thus ensuring they cannot fire twice.

**Algorithm.** From the above analysis, when  $\theta = 0$ , the model cannot change to a new marking before  $\theta = \tau_b$ , where we call  $\tau_b = \min_{t \in \mathcal{T}, \langle\mu, \tau\rangle \in \mathcal{S}_0} \{\tau(t)\}$  a *breakpoint*. At  $\theta = \tau_b$ , the model moves to a vanishing state, and we collect the set of possible states at  $\theta = \tau_b$  immediately before the firing, which includes those vanishing states  $\mathcal{S}_{\tau_b}^- = \{\langle\mu^-, \tau^-\rangle : \exists \langle\mu, \tau\rangle \in \mathcal{S}_0, \mu^- = \mu, \tau^- = \tau - \tau_b\}$ . Then, we apply a fixpoint image computation  $FixImg$  on  $\mathcal{S}_{\tau_b}^-$  using a modified version of the transition relations  $\mathcal{R}_t$ , so that  $(\langle\mu, \tau\rangle, \langle\mu', \tau'\rangle) \in \mathcal{R}_t$  iff:

- $t \in \mathcal{E}(\langle\mu, \tau\rangle)$ , i.e.,  $t$  is marking-enabled in  $\mu$  and  $\tau(t) = 0$ .
- $\mu [t] \Rightarrow \mu'$ , i.e, the firing of  $t$  in  $\mu$  leads to  $\mu'$ .
- $\tau'(t) = \infty$  and  $\tau'(t') = \tau(t')$  for  $t' \neq t$ , which ensures that  $t$  can only fire once and its firing does affect the remaining time of other transitions.

Consider a state  $\langle\mu^-, \tau^-\rangle \in \mathcal{S}_{\tau_b}^-$ ; if it is tangible, it is unaffected by the above fixpoint image computation, since no transition is state-enabled in it, i.e.,  $FixImg(\langle\mu^-, \tau^-\rangle) = \{\langle\mu^-, \tau^-\rangle\}$ ; if it is vanishing,  $FixImg(\langle\mu^-, \tau^-\rangle)$  instead contains all states  $\langle\mu', \tau'\rangle$  satisfying: (1) there is a subset  $\mathcal{T}_f \subseteq \mathcal{E}(\langle\mu^-, \tau^-\rangle)$  such that  $\mu^- [\mathcal{T}_f] \Rightarrow \mu'$ , (2)  $\tau'[t] = \tau^-[t]$  if  $t \notin \mathcal{T}_f$ , (3)  $\tau'[t] = \infty$  if  $t \in \mathcal{T}_f$ . Thus,  $FixImg(\langle\mu^-, \tau^-\rangle)$  “almost” contains our desired tangible frontier, except for two problems. First, it also contains all intermediate vanishing states, recognized by having at least one zero component in the remaining firing time for a marking-enabled transition. Second, the remaining firing times of a tangible frontier state  $\langle\mu', \tau'\rangle$  might be incorrect, as some transition marking-enabled in  $\mu^-$  might be disabled in  $\mu'$ , or vice versa; these must be updated in a separate step.

**Symbolic implementation.** Following Sec. 2.2, we encode a set of states using an MDD over the  $L$  variables  $(y_{|\mathcal{T}|}, \dots, y_1, x_{|\mathcal{P}|}, \dots, x_1)$ , where  $y_t$  is used to encode the remaining firing time of transition  $t$  and  $x_p$  is used to encode the number of tokens in place  $p$ . The special value  $\infty$  is stored in practice as the value  $\max \mathcal{F} + 1$ , although in the pseudocode we still write  $\infty$  for clarity. The transition relation  $\mathcal{R}_t$  is then encoded by a 2L-variable MDD.

Fig. 3 shows the pseudocode to solve the timed reachability problem. Procedure *TimedReach* computes  $enc(\mathcal{S}_{\theta_f})$  from  $\mathcal{S}_0$  by iteratively advancing to the next breakpoint, given the MDD encoding the current set of states encoded by MDD  $s$  at time  $\theta$ . It first computes the minimal remaining time  $\tau_b$  using procedure *MinNonzero*, then subtracts it from all remaining firing times  $\tau$  of the states in  $\mathcal{B}(s)$  to obtain  $enc(\mathcal{S}_{\tau_b}^-)$  using procedure *Elapse*, and increases  $\theta$  by the same amount. If  $\tau_b$  is  $\infty$ , this means that the model can only be in dead states by time  $\theta$ . Next, it builds  $enc(\mathcal{S}_{\tau_b}^+)$  through a fixpoint image computation, using procedure *Saturate* discussed in Sec. 2.2. Then, it calls procedure *Reset* to reset the remaining firing times for all states in  $\mathcal{S}_{\tau_b}^+$  and, finally, it calls procedure *ElimVan* to eliminate all vanishing states. The global computation ends when the (next) value of  $\theta$  exceeds  $\theta_f$ . At that point,  $\mathcal{B}(s)$  encodes all the *states* in which the ITPN can be at time  $\theta_f$ ; the call *GetMarkings(s)* strips the remaining

<pre> mdd TimedReach(integer <math>\theta_f</math>) 1 <math>\theta \leftarrow 0</math>; <math>s \leftarrow enc(\mathcal{S}_0)</math>; 2 repeat forever 3   <math>\tau_b \leftarrow MinNonzero(s)</math>; 4   if <math>\tau_b = \infty</math> then return <math>\mathbf{0}</math>; 5   <math>\theta \leftarrow \theta + \tau_b</math>; <math>s \leftarrow Elapse(s, \tau_b)</math>; 6   if <math>\theta &gt; \theta_f</math> then return <math>GetMarkings(s)</math>; 7   <math>s \leftarrow Saturate(s)</math>; 8   <math>s \leftarrow Reset(s)</math>; 9   <math>s \leftarrow ElimVan(s)</math>; </pre>	<pre> 2 if <math>CacheHit(ELS, s, b, u)</math> then return <math>u</math>; 3 <math>u \leftarrow NewNode(s_v)</math>; 4 foreach <math>i \in \mathcal{V}_{s_v}</math> s.t. <math>s[i] \neq \mathbf{0}</math> do 5   <math>u[i - b] \leftarrow Elapse(s[i], b)</math>; 6 <math>CacheAdd(ELS, s, b, Unique(u))</math>; 7 return <math>q</math>; </pre>
<pre> integer MinNonzero(mdd <math>s</math>) 1 <math>m \leftarrow \min\{i \in \mathcal{V}_{s_v} : s[i] \neq \mathbf{0}\}</math>; 2 if <math>s_v = y_1</math> then return <math>m</math>; 3 if <math>CacheHit(MNI, s, m)</math> then return <math>m</math>; 4 foreach <math>i \in \mathcal{V}_{s_v}</math> s.t. <math>s[i] \neq \mathbf{0}</math> do 5   <math>u \leftarrow MinNonzero(s[i])</math>; 6   <math>m \leftarrow \min\{m, u\}</math>; 7 <math>CacheAdd(MNI, s, m)</math>; 8 return <math>m</math>; </pre>	<pre> mdd Reset(mdd <math>s</math>) 1 foreach <math>t \in \mathcal{T}</math> do 2   <math>s_{enb} \leftarrow Intersect(s, s_{enabled}(t))</math>; 3   <math>s_{dis} \leftarrow Difference(s, s_{enb})</math>; 4   <math>u_1 \leftarrow RelProd(s_{enb}, r_{resample}(t))</math>; 5   <math>u_2 \leftarrow RelProd(s_{dis}, r_{reset}(t))</math>; 6 return <math>Union(u_1, u_2)</math>; </pre>
<pre> mdd Elapse(mdd <math>s</math>, integer <math>b</math>) 1 if <math>s_v = x_{ \mathcal{P} }</math> then return <math>s</math>; </pre>	<pre> mdd ElimVan(mdd <math>s</math>) 1 if <math>s_v = x_{ \mathcal{P} }</math> then return <math>s</math>; 2 if <math>CacheHit(ELV, s, u)</math> then return <math>u</math>; 3 <math>u \leftarrow NewNode(s_v)</math>; 4 foreach <math>i \in \mathcal{V}_{s_v} \setminus \{0\}</math> do 5   <math>u[i] \leftarrow ElimVan(s[i])</math>; 6 <math>CacheAdd(ELV, s, Unique(u))</math>; 7 return <math>u</math>; </pre>

**Fig. 3.** Timed reachability algorithm.

firing time information from  $s$ , and returns the MDD encoding the *markings* in which the ITPN can be at time  $\theta_f$ .

The pseudocode assumes that the following MDDs have been generated prior to calling *TimedReach*, for  $t \in \mathcal{T}$ : the  $L$ -variable MDDs  $s_{enabled}(t)$ , encoding the set of states where  $t$  is marking-enabled, the  $2L$ -level MDD  $r_{resample}(t)$ , which changes  $y_t = \infty$  into  $y'_t \in \mathcal{F}_t$ , and the  $2L$ -level MDD  $r_{reset}(t)$ , which sets  $y'_t = \infty$  regardless of the value of  $y_t$ .

### 3.2 Earliest Reachability

The earliest reachability problem aims at finding the minimum time at which each reachable marking of the ITPN is entered. Formally, we seek the function  $\epsilon : \mathbb{N}^{\mathcal{P}} \rightarrow \mathbb{N} \cup \{\infty\}$  satisfying  $\epsilon(\mu') = \min \{\theta : \exists \langle \mu, \tau \rangle \in \mathcal{S}_0, \langle \mu, \tau @ 0 \rangle [*] \Rightarrow \langle \mu', \tau' @ \theta \rangle\}$ , where the entries of  $\tau'$  are strictly positive (possibly infinite), and  $\epsilon(\mu') = \infty$  if the marking is unreachable (thus, this also gives us the reachable markings).

This is similar to finding the shortest path in reachability graphs and can be solved by exploring all markings and keeping track of the minimum time at which we first saw each reachable state, by comparing the original minimum time and the new time when it is reached. Following Sec. 3.1, we know the model can only reach a new marking at some breakpoint  $\theta = \tau_b$ . Thus we let the model evolve as for timed reachability and update the minimum time for each marking reached at every breakpoint. Unlike timed reachability, however, we need to generate *all* reachable markings, and stop only when no new marking is found. To do so, we accumulate the states  $\mathcal{S}_{\tau_b}^-$  at every breakpoint, including vanishing states, until we reach a fixpoint, ensuring that no new marking can be reached in the future evolution of the ITPN.

EV<sup>+</sup>MDDs are very efficient for the joint computation of reachable states and their distance, using the *Minimum* operation of Sec. 2.2. We adopt them to

<pre> evmdd EarliestReach() 1 <math>\langle \rho, u \rangle \leftarrow \text{MakeEvmdd}(0, \text{enc}(\mathcal{M}_0))</math> 2 <math>s \leftarrow \text{enc}(\mathcal{S}_0)</math>; <math>\tau_b \leftarrow \text{MinNonzero}(s)</math>; 3 <math>s \leftarrow \text{Elapse}(s, \tau_b)</math>; <math>g \leftarrow s</math>; 4 repeat 5   <math>s \leftarrow \text{Saturate}(s)</math>; <math>s \leftarrow \text{Reset}(s)</math>; 6   <math>s \leftarrow \text{ElimVan}(s)</math>; </pre>	<pre> 7 <math>m \leftarrow \text{GetMarkings}(s)</math>; 8 <math>\langle \rho_c, u_c \rangle \leftarrow \text{MakeEvmdd}(\theta, m)</math>; 9 <math>\langle \rho, u \rangle \leftarrow \text{Minimum}(\langle \rho, u \rangle, \langle \rho_c, u_c \rangle)</math>; 10 <math>\tau_b \leftarrow \text{MinNonzero}(s)</math>; 11 <math>s \leftarrow \text{Elapse}(s, \tau_b)</math>; <math>g \leftarrow \text{Union}(g, s)</math>; 12 until <math>g</math> does not change; 13 return <math>\langle \rho, u \rangle</math>; </pre>
---	--

**Fig. 4.** Earliest reachability algorithm.

encode the function  $\epsilon$  we seek, so that  $\epsilon(\mu)$  records the first time marking  $\mu$  was encountered; by default,  $\epsilon(\mu) = \infty$  for any  $\mu$  not yet found in the exploration.

Fig. 4 shows the procedure to compute the EV<sup>+</sup>MDD  $\langle \rho, u \rangle$  encoding the desired earliest reachability function  $\epsilon$ . It proceeds as for timed reachability, except that we extract the set  $\mathcal{M}$  of markings that are part of tangible states after advancing to each breakpoint, with a call to *GetMarkings*. Then,  $\mathcal{M}$ , encoded in an MDD that only refers to the variables  $x_{|\mathcal{P}|}, \dots, x_1$ , is used to build the EV<sup>+</sup>MDD encoding the function defined by  $f_{\langle \rho_c, u_c \rangle}(\mu) = \theta$ , the current time, if  $\mu \in \mathcal{M}$ , and  $f_{\langle \rho_c, u_c \rangle}(\mu) = \infty$  otherwise. After that, a call to *Minimum* ensures that, if some new marking  $\mu$  was found, its earliest reachability time is recorded in the EV<sup>+</sup>MDD  $\langle \rho, u \rangle$ , overwriting the old value  $f_{\langle \rho, u \rangle}(\mu) = \infty$ . The MDD  $g$  is used to accumulate all states before fixpoint image computation at every breakpoint. The global computation halts when  $g$  reaches a fixpoint or if the model has reached only dead states.

**Running model.** Fig. 5 shows the MDDs and EV<sup>+</sup>MDDs built during timed and earliest reachability computation of our running model. Each row corresponds to a different time point  $\theta$  and each column corresponds to a different stage of the iteration. Sets of states are encoded by a 6-variable MDD and the earliest reachability function is encoded by a 3-variable EV<sup>+</sup>MDD. We start reachability computation from  $\text{enc}(\mathcal{S}_0)$ , at time  $\theta = 0$ , then  $\text{enc}(\mathcal{S}_\theta^-)$  is obtained after *Elapse* is called to move to the breakpoint  $\tau_d = 1$ , then *Saturation* is called on it to obtain  $\text{enc}(\mathcal{S}_\theta^+)$ , then *Reset* and *ElimVan* are called on the MDD encoding  $\mathcal{S}_\theta$ , the set of tangible states at time  $\theta$ . Then *GetMarkings*, *MakeEvmdd*, and *Minimum* from  $\text{enc}(\mathcal{S}_\theta)$  generate the EV<sup>+</sup>MDD encoding the earliest reachability time of each reachable marking found up to time  $\theta$ . For  $\mathcal{S}_\theta^-$ , we use a white box in the node to emphasize when a transition has remaining firing time 0, i.e., when it is state-enabled. Our ITPN model can evolve up to  $\theta = 6$ , at which time it can only be in a dead state. The fixpoint for earliest reachability is obtained at time  $\theta = 4$ , not shown in the figure. Thus, at time  $\theta = 3$ , i.e., the last EV<sup>+</sup>MDD gives the encoding of the earliest reachability times for *all* markings.

### 3.3 Decidability, Complexity, Extensions and Related Research

**Decidability.** It is well known that the *marking reachability problem*: “given marking  $\mu, \exists \langle \mu, \tau \rangle \in \mathcal{S}$ ?” is undecidable for arbitrary time Petri nets [13] but decidable for token-bounded TPNs [22], although boundedness is itself undecidable for TPNs. For ITPNs, the problem is different, given their essentially discrete nature. Nevertheless, we can show that their integer firing times can be used to enforce priorities between transitions, hence achieve the “test-for-zero” of counter machines, and Turing-equivalence. Thus, (earliest) reachability is undecidable for ITPNs (it is of course decidable for bounded ITPNs but, again,

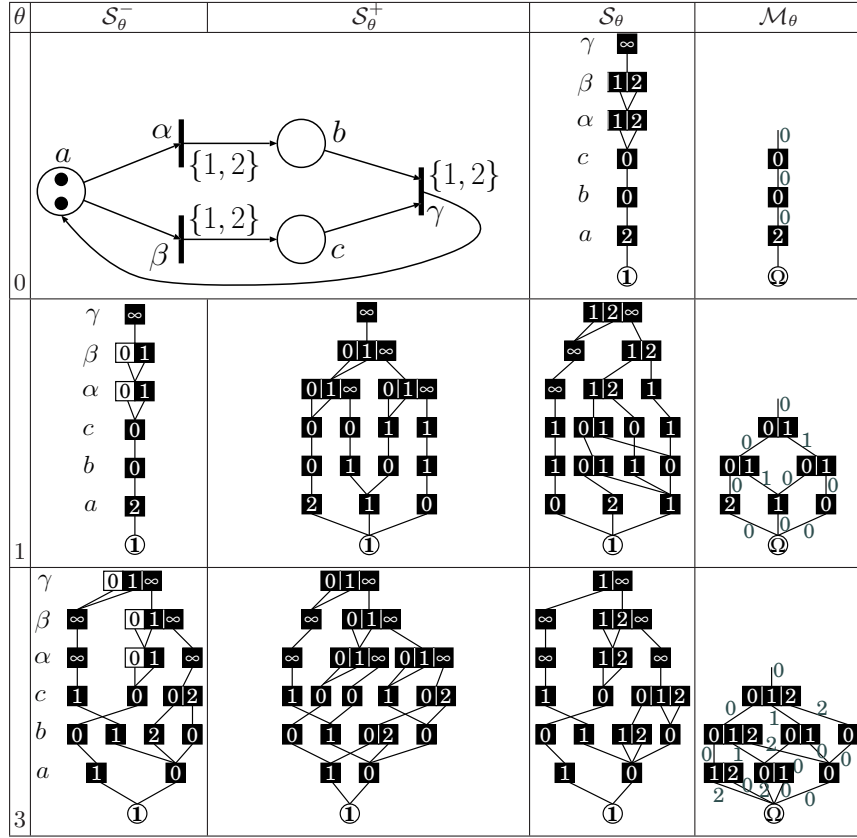


Fig. 5. MDDs and EV<sup>+</sup>MDDs for timed and earliest reachability of our running model.

boundedness itself is not). Of course, timed reachability is instead decidable, since only a finite number of markings can be reached in a finite time horizon, given that all firing times are positive integers.

**Complexity.** The time complexity is  $N_{\tau_b} \cdot T_s$ , where  $N_{\tau_b}$  counts the breakpoints and  $T_s$  is the average complexity for the work performed at each breakpoint. When  $\mathcal{F}$  contains more firing times, the number of markings  $|\mathcal{M}|$  tends to grow and approach the number of reachable markings in the underlying untimed PN.

**Related work.** [23] proposes a symbolic state-space generation for FIHTPNs, a type of TPNs with different semantics from our ITPNs. They translate the TPN into an untimed PN by defining a subnet for each timed transition, combine these subnets, and perform breadth-first symbolic state-space generation on the resulting PN. Our approach does not require this translation step and, more importantly, does not increment a counter (a PN place) one unit at a time; instead, we advance the global clock by the minimal remaining firing time  $\tau_b$ .

## 4 Experimental Results

We implemented our algorithms in our SMART model checker [5] and ran them on an Intel Xeon 3.0Ghz workstation with 16GB RAM under SuSE Linux 9.1.

Model	$N$	TR $\theta_{fin} = 5$			TR $\theta_{fin} = 100$			ER		
		$ \mathcal{S} $	time	mem	$ \mathcal{S} $	time	mem	$ \mathcal{M} $	time	mem
<i>bittoggle</i>	10	$4.14 \cdot 10^7$	0.13	4.36	$1.00 \cdot 10^8$	1.42	34.28	$1.02 \cdot 10^3$	1.46	35.46
	[12] 15	$2.27 \cdot 10^{11}$	0.91	21.90	$7.83 \cdot 10^{11}$	91.57	1415.63	-	-	-
<i>bitshift</i>	20	$1.56 \cdot 10^2$	0.01	0.11	$4.08 \cdot 10^7$	0.01	0.50	$4.19 \cdot 10^6$	0.02	0.54
	[10] 70	$1.56 \cdot 10^2$	0.02	0.26	$3.60 \cdot 10^{16}$	0.09	3.37	$4.72 \cdot 10^{21}$	0.14	4.15
<i>bqueue</i>	5	$2.37 \cdot 10^7$	1.08	24.53	$1.82 \cdot 10^8$	22.02	408.38	$1.58 \cdot 10^4$	22.47	414.68
	[19] 10	$2.99 \cdot 10^7$	1.13	24.31	-	-	-	-	-	-
<i>bubsort</i>	5	$7.50 \cdot 10^4$	0.01	0.37	$7.50 \cdot 10^4$	0.01	0.37	$1.20 \cdot 10^2$	0.01	0.37
	[10] 8	$3.15 \cdot 10^9$	5.72	51.29	$3.15 \cdot 10^9$	5.70	51.29	$4.03 \cdot 10^4$	6.06	29.89
<i>fms</i>	2	$3.42 \cdot 10^7$	2.78	58.41	$3.97 \cdot 10^7$	10.20	193.06	$3.44 \cdot 10^3$	10.29	195.95
	[16] 4	$1.12 \cdot 10^{11}$	103.04	919.49	-	-	-	-	-	-
<i>kanban</i>	2	$1.86 \cdot 10^7$	0.33	8.44	$3.87 \cdot 10^9$	33.50	555.02	$4.60 \cdot 10^3$	62.19	353.39
	[16] 4	$1.50 \cdot 10^9$	2.56	49.98	-	-	-	-	-	-
<i>intshift</i>	20	$5.30 \cdot 10^2$	0.01	0.23	$3.83 \cdot 10^{17}$	0.33	6.12	$4.02 \cdot 10^{16}$	0.37	6.51
	[10] 70	$5.30 \cdot 10^2$	0.02	0.54	$2.84 \cdot 10^{26}$	0.72	17.43	$3.25 \cdot 10^{55}$	10.09	82.57
<i>phils</i>	5	$3.75 \cdot 10^7$	4.76	95.54	$3.77 \cdot 10^7$	7.33	146.35	$1.36 \cdot 10^3$	7.47	149.22
	[16] 6	$1.22 \cdot 10^9$	28.89	503.04	$1.23 \cdot 10^9$	52.16	873.54	-	-	-
<i>queen</i>	10	$2.61 \cdot 10^6$	13.89	248.98	$1.27 \cdot 10^4$	30.71	519.94	$3.55 \cdot 10^4$	27.63	460.29
	[11] 11	$1.41 \cdot 10^7$	35.08	594.04	-	-	-	$1.66 \cdot 10^9$	81.74	1300.12
<i>robin</i>	20	$8.07 \cdot 10^2$	0.08	2.84	$1.59 \cdot 10^4$	14.40	263.12	$4.00 \cdot 10^2$	13.95	259.11
	[10] 30	$8.07 \cdot 10^2$	0.15	5.01	$2.38 \cdot 10^4$	57.22	894.75	$6.00 \cdot 10^2$	56.54	883.04
<i>slot</i>	4	$2.20 \cdot 10^7$	5.18	112.20	$2.55 \cdot 10^7$	13.71	282.11	$5.13 \cdot 10^3$	13.95	288.50
	[10] 5	$1.51 \cdot 10^9$	80.21	1316.79	-	-	-	-	-	-
<i>swapper</i>	10	$2.20 \cdot 10^{18}$	1.25	26.25	$3.20 \cdot 10^{18}$	1.96	39.79	$1.67 \cdot 10^5$	2.04	28.01
	[10] 20	$4.75 \cdot 10^{35}$	49.82	250.60	$2.38 \cdot 10^{38}$	126.43	250.60	$1.31 \cdot 10^{11}$	123.62	127.51

**Table 1.** Experimental results (time in sec, mem in Mbytes); “-” means time > 600sec.

Table. 1 shows the running time and peak memory consumption to compute timed reachability (TR) at  $\theta_f = 5$  or 100 and earliest reachability (ER), on a variety of models (referenced in the table). The possible firing times of each transition in each model are  $\{1, 2, 3, 4, 5\}$ . Parameter  $N$  denotes the initial number of tokens in certain places or the number of repeated subnets, and affects the size of state space. We record the number of reachable *states* at time  $\theta_f$  (for TR) and of overall reachable *markings* (for ER). From the table, we can see that our algorithms can generate large set of states (up to  $10^{38}$ ) for TR and markings (up to  $10^{55}$ ) for ER with a small time and memory requirements, which demonstrates the applicability and efficiency of symbolic methods on state-space generation of synchronous timed systems, not just to untimed asynchronous systems, where they have already been extensively applied.

## 5 Conclusion

We considered two fundamental problems for a class of timed Petri net models where events have integer but not necessarily constant durations: timed reachability and earliest reachability. For these, we provided detailed symbolic algorithms that, through the use of both ordinary and edge-valued decision diagrams, can quickly explore very large state spaces. These algorithms are non-trivial extensions of known symbolic reachability algorithms for untimed nets, showing that the potential of these approaches goes well beyond strict logical analysis.

As a first extension of the present work, we envision exploring decision diagrams variants that can better tackle models with large ranges of durations.

## References

1. P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.
2. R. E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, Aug. 1986.
3. H.-D. Burhard. On priorities of parallelism: Petri nets under the maximum firing strategy. In *Logics of Programs and Their Appl.*, LNCS 148, pages 86–97. 1983.
4. G. Ciardo. Discrete-time Markovian stochastic Petri nets. In *Computations with Markov Chains*, pages 339–358. Kluwer, 1995.
5. G. Ciardo, R. L. Jones, A. S. Miner, and R. Siminiceanu. Logical and stochastic modeling with SMART. *Perf. Eval.*, 63:578–608, 2006.
6. G. Ciardo et al. Saturation: An efficient iteration strategy for symbolic state space generation. In *TACAS*, LNCS 2031, pages 328–342, 2001.
7. G. Ciardo, R. Marmorstein, and R. Siminiceanu. Saturation unbound. In *TACAS*, LNCS 2619, pages 379–393, 2003.
8. G. Ciardo, R. Marmorstein, and R. Siminiceanu. The saturation algorithm for symbolic state space exploration. *STTT*, 8(1):4–25, 2006.
9. G. Ciardo and R. Siminiceanu. Using edge-valued decision diagrams for symbolic generation of shortest paths. In *FMCAD*, LNCS 2517, pages 256–273, 2002.
10. G. Ciardo and A. J. Yu. Saturation-based symbolic reachability analysis using conjunctive and disjunctive partitioning. In *CHARME*, LNCS 3725, 146–161, 2005.
11. H. E. Dudeney. *Amusements in Mathematics*. Dover, 1970.
12. G. J. Holzmann. *The SPIN Model Checker*. Addison-Wesley, 2003.
13. N. D. Jones, L. H. Landweber, and Y. E. Lien. Complexity of some problems in Petri nets. *Theoretical Computer Science*, 4:277–299, 1977.
14. T. Kam et al. Multi-valued decision diagrams: theory and applications. *Multiple-Valued Logic*, 4(1–2):9–62, 1998.
15. P. M. Merlin. *A study of the recoverability of computing systems*. PhD thesis, Department of Information and Computer Science, U. C. Irvine, 1974.
16. A. S. Miner. Implicit GSPN reachability set generation using decision diagrams. *Perf. Eval.*, 56(1-4):145–165, Mar. 2004.
17. T. Murata. Petri nets: properties, analysis and applications. *Proc. of the IEEE*, 77(4):541–579, Apr. 1989.
18. E. Pastor, O. Roig, J. Cortadella, and R. M. Badia. Petri net analysis using boolean manipulation. In *ICATPN*, LNCS 815, pages 416–435, 1994.
19. B. Plateau, P. Fernandes, and W. Stewart. The PEPS software tool. In *Modelling Techniques and Tools for Computer Perf. Eval.*, LNCS 2794, 98–115, 2003.
20. Y. Zhang. *Non-blocking Synchronization: Algorithms and Performance Evaluation*. PhD thesis, Chalmers Univ. of Technology, 2003.
21. W. L. Zuberek. Timed Petri nets definitions, properties, and applications. *Microelectronics and Reliability*, 31:627–644, 1991.
22. B. Berthomieu and M. Menasche. An enumerative approach for analyzing time Petri nets. *Proceedings IFIP 1983*, 41–46.
23. M. Magnin, P. Molinaro, and O. Roux. Decidability, expressivity and state-space computation of stopwatch Petri nets with discrete-time semantics. In *8th International Workshop on Discrete Event Systems (WODES)*, 33–38, 2006.