Computing a Finite Prefix of a Time Petri Net*

Hans Fleischhack and Christian Stehno

Fachbereich Informatik
Carl von Ossietzky Universität Oldenburg
D-26111 Oldenburg
{fleischhack,stehno}@informatik.uni-oldenburg.de

Abstract. Recently, model checking of Petri nets based on partial order semantics w.r.t. temporal logic formulae has been extended to time Petri nets. In this paper, we present an improved algorithm for computing the McMillan-unfolding of a time Petri net which gives a finite representation of the partial order semantics and some experimental results of its implementation within the PEP tool.

1 Introduction

Model checking is a widely accepted method for proving properties of distributed systems but faces the problem of 'state explosion'. To tackle this problem, besides partial order reductions [12] or BDD-based techniques [7, 19] also methods based on partial order semantics have been applied. The latter have proven especially successful when applied to systems with a high degree of asynchronous parallelism [5, 9].

Safety critical applications often require verification of real time constraints, in addition to functional or qualitative temporal properties. For this task, model checking algorithms have been developed based on interleaving semantics (cf. e.g. [2, 22]), but much less work has been done starting from partial order semantics. By extending McMillan's [20] technique of unfolding safe Petri nets to the class of safe time Petri nets a considerable step in this direction has been taken in [6].

In this paper, we present a simplified definition of the McMillan-unfolding of a time Petri net from which we derive an efficient algorithm for its actual computation, thereby also removing the restrictions of diverging time and of finite upper time bounds for transitions (cf. [6]) .

The key idea of the approach presented here consists in transforming time restrictions into net structure, i.e. representing them by additional places, transitions, and arcs. Following this approach, for a given time Petri net TN the time-expansion $EX(TN)$ is constructed as an ordinary P/T-net. Moreover, it is shown that a formula ϕ of a suitable temporal logic for time Petri nets is satisfied by TN iff it is satisfied by $EX(TN)$. Subsequently, the McMillan-unfolding $McM(TN)$ of TN is defined as (an abstraction of) $McM(EX(TN))$. As it turns

* This work has been partially supported by the the Procope projects BAT (Box Algebra with Time) and PORTA (Partial Order Real Time Semantics).

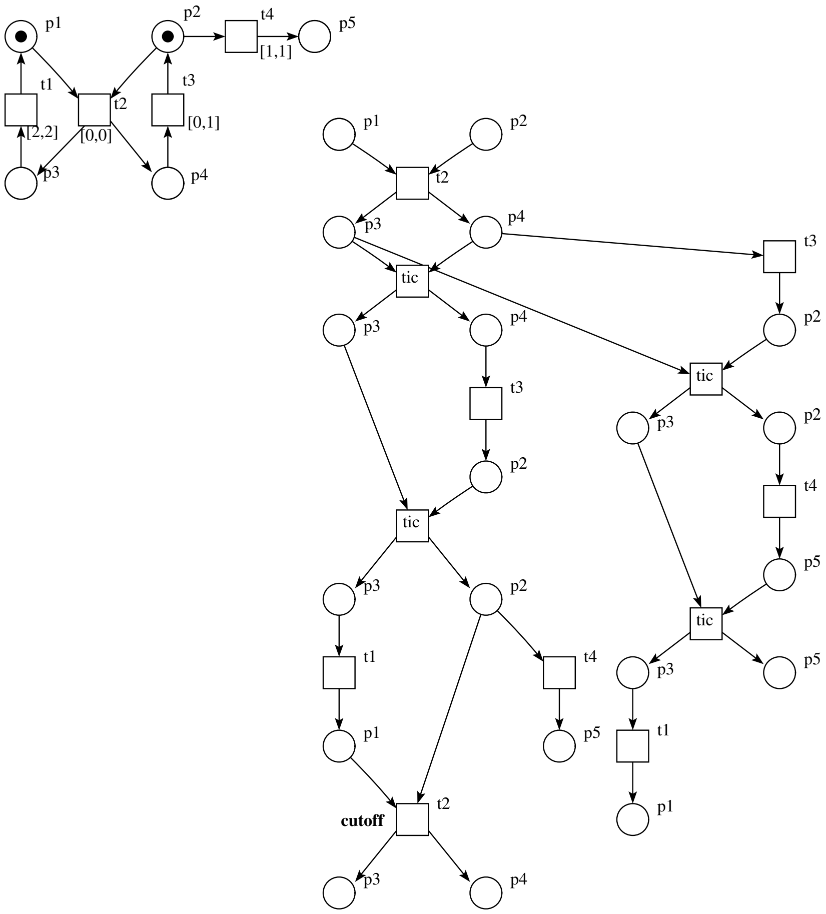

Fig. 1. A time Petri net and its timed behaviour

out, for the actual computation of $McM(TN)$ the explicit construction of the (complex) net $EX(TN)$ is not necessary.

Following this approach corresponding tools for P/T-nets may be reused for the causal analysis of time Petri nets. This applies especially to the model checking component of the PEP tool [5, 23].

The paper is structured as follows: Section 2 provides the basic notions used throughout the paper. In Sect. 3 a new notion of bisimulation for time Petri nets and P/T-nets is introduced. Section 4 contains the definition of time expansion of time Petri nets. The new method for the actual computation of the McMillan-unfolding of a safe time Petri net is given in Sect. 5. Based on the implementation of this method we give some experimental results in Sect. 6. We conclude with discussing some related work (Sect. 7) and with remarks on further research (Sect. 8).

2 Basic Notions

In this section, partial order semantics and temporal logics for safe Petri nets is defined. A notion of time is added to P/T-nets and the temporal logics is extended to time Petri nets.

2.1 Petri Nets

A *net* is a triple $N = (P, T, F)$ consisting of a set P of *places*, a set T of *transitions*, a set $F \subseteq ((P \times T) \cup (T \times P))$, called *flow relation*, and such that $P \cap T = \emptyset$, ${}^\bullet p \neq \emptyset$ or $p^\bullet \neq \emptyset$ for all $p \in P$, and ${}^\bullet t \neq \emptyset \neq t^\bullet$ for all $t \in T$, where, for $x \in (P \cup T)$, ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (x, y) \in F\}$. For $X \subseteq (P \cup T)$, ${}^\bullet X$ and $X^\bullet$ are defined elementwise.

A *marking* of N is a multi set $M : P \to \mathbb{N}$. A transition t is *enabled under* M in N ($M \to^t$) iff ${}^\bullet t \leq M$ [1]. In this case, $M' = M - {}^\bullet t + t^\bullet$ is called *successor marking* of M w.r.t. t ($M \to^t M'$). A set U of transitions is *enabled under* M in N ($M \to^U$) iff $(\bigcup_{t \in U} {}^\bullet t) \leq M$. In this case the successor marking is given by $M' = M - (\bigcup_{t \in U} {}^\bullet t) + (\bigcup_{t \in U} t^\bullet)$.

A *net system* or *P/T-net* or *marked net* (N, M_0) consists of a net and an (initial) marking. A marking M is *reachable* in (N, M_0) if, for some $n \in \mathbb{N}$, there are $t_1, \ldots, t_n \in T$ and markings $M_1, \ldots, M_n$ such that $M_n = M$ and $M_0 \to^{t_1} M_1 \ldots \to^{t_n} M_n$. The set of reachable markings is denoted by $Reach(N, M_0)$. N is *finite* iff $P \cup T$ is finite and *safe* iff $M(p) \in \{0, 1\}$ for all $p \in P$ holds for any reachable marking M. For a safe net N, marking M is written as a subset $M \subseteq P$. The firing rule is reformulated accordingly. During the rest of the paper, only safe nets are considered.

Nodes $x_1, x_2 \in (P \cup T)$ of a net N are *in conflict* ($x_1 \# x_2$) iff there exist distinct transitions $t_1, t_2 \in T$ such that ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$ and $(t_1, x_1), (t_2, x_2) \in F^*$, the reflexive and transitive closure of F. $x \in (P \cup T)$ is in *self-conflict* iff $x \# x$.

2.2 Branching Processes

The maximal branching process associates a partial order semantics to each safe net P/T-net N [9]. A *causal process* of a safe P/T-net N describes a possible run of N, displaying the causal dependencies of the events that take place during the run. A *branching process* can represent several alternative runs of N in one structure and hence may be seen as the union of some causal processes. It consists of an occurrence net and a homomorphism.

An *occurrence net* $ON = (B, E, G)$ is an acyclic net such that $\mid {}^\bullet b \mid \leq 1$ for all $b \in B$, no event $e \in E$ is in self conflict, and for all $x \in (B \cup E)$, the set of elements $y \in (B \cup E)$ such that $(y, x) \in G^*$ is finite. The elements of B are called *conditions* and the elements of E are called *events*. $\leq$ denotes the partial order induced by G on $B \cup E$; $<$ denotes the corresponding strict partial order;

[1] Here and in the following sets are sometimes interpreted as special multi sets and operations on multi sets rather than on sets are applied.

$Min(ON)$ denotes the set of minimal elements of ON w.r.t. $\leq$. A *causal net* is an occurrence net which also satisfies $|b^\bullet| \leq 1$ for every condition b.

A *homomorphism* from an occurrence net ON to a P/T-net (N, M_0) is a mapping $\pi : (B \cup E) \rightarrow (P \cup T)$ such that $\pi(B) \subseteq P$ and $\pi(E) \subseteq T$; for all $e \in E$, the restriction of π to ${}^\bullet e$ is a bijection between ${}^\bullet e$ and ${}^\bullet \pi(e)$; the restriction of π to $e^\bullet$ is a bijection between $e^\bullet$ and $\pi(e)^\bullet$; the restriction of π to $Min(ON)$ is a bijection between $Min(ON)$ and M_0; for all $e_1, e_2 \in E$ it holds that if ${}^\bullet e_1 = {}^\bullet e_2$ and $\pi(e_1) = \pi(e_2)$ then $e_1 = e_2$.

Let (N, M_0) be a P/T-net. A *branching process* of (N, M_0) is a pair $\beta = (ON, \pi)$, consisting of an occurrence net ON, and a homomorphism π from ON to (N, M_0). β is a *causal process* of (N, M_0) iff ON is a causal net. Let (N_1, π_1) and (N_2, π_2) be two branching processes of (N, M_0). (N_1, π_1) and (N_2, π_2) are *isomorphic* iff, for some bijective homomorphism $\pi : N_1 \rightarrow N_2$, $\pi_2 \circ \pi = \pi_1$.

For each P/T-net (N, M_0), there exists a unique (up to isomorphism) maximal (w.r.t. prefix ordering; cf. [8]) branching process β_m. This is called the *unfolding of* (N, M_0).

An initial part of a causal process may be represented uniquely by the set of events contained in that part: A *configuration* C of a process β is a downward closed conflict free set of events, i.e., a set $C \subseteq E$ such that $e \in C$ implies $e' \in C$ for all $e' \leq e$, and, for all $e, e' \in C$, $e \# e'$ does not hold. $\emptyset$ denotes the empty configuration and, for each event e, $[e] = \{e' \in E \mid e' \leq e\}$ denotes the configuration generated by e. Note, that each configuration C of a branching process $\beta = (ON, \pi)$ uniquely determines a causal process, having C as set of events and all conditions connected to elements of C in ON as set of conditions. Arcs and inscriptions are also inherited from ON. Hence, notions defined for (causal) processes may also be applied to configurations and vice versa.

A configuration C defines a unique marking, consisting of exactly all conditions of ON which are marked after occurrence of all events of C: A *co-set* is a set B' of conditions of an occurrence net ON such that, for all $b \neq b' \in B'$, neither $b < b'$ nor $b' < b$ nor $b \# b'$. A *cut* is a maximal co-set B' (w.r.t. to set inclusion). The cut of a finite configuration C is defined by $Cut(C) = (Min(ON) \cup C^\bullet) \setminus {}^\bullet C$. C defines a marking of ON by $Mark(C) = \pi(Cut(C))$.

We have the following

Proposition 1. *(i) If Co is a co-set in the branching process $\beta = (ON, \pi)$ and $x_1, x_2 \in Co$ then $\pi(x_1) \neq \pi(x_2)$.*
(ii) A marking M is reachable in N iff the maximal branching process β_m contains a configuration C such that $M = Mark(C)$.

Figure 1 shows a causal branching process of the P/T-net underlying the time Petri net TN_1. Names of conditions and events are omitted; the image of a vertex x under π (also called *label of* x) is written beside it.

2.3 Temporal Logics for Safe P/T-Nets

In this section we introduce the branching time logic $\mathcal{L}$ for safe P/T-nets defined in [9]. Later on we will extend this logic to time Petri nets and show how the model checking algorithm of [9] can be applied. Properties of the current marking of a P/T-net (N, M_0) are expressed using *place assertions*. E.g., a formula '$s_1 \wedge s_2$' expresses the fact that places s_1 and s_2 are marked. '$\Diamond\phi$' (*"possibly ϕ"*) means that a marking satisfying 'ϕ' is reachable. The derived operator '$\Box\phi$' (*"always ϕ"*) signifies that 'ϕ' is satisfied at all reachable markings.

Let N be a net. The syntax of $\mathcal{L}$ over N is presented given by the following grammar:

$$\phi ::= \texttt{true} \mid s \in P_N \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \Diamond\phi.$$

Additionally, the following abbreviations are allowed:

$$\begin{array}{ll} \texttt{false} & = \neg\texttt{true} \\ (\phi_1 \vee \phi_2) & = \neg(\neg\phi_1 \wedge \neg\phi_2) \\ (\phi_1 \Rightarrow \phi_2) & = \neg\phi_1 \vee \phi_2 \\ \Box\phi & = \neg\Diamond\neg\phi \end{array}$$

Satisfaction of a formula ϕ w.r.t. a P/T-net (N, M_0) is defined inductively as follows. Let C be a finite configuration of a branching process β of (N, M_0). The superscript N is dropped if it is clear from the context.

$$\begin{array}{lll} (\beta, C) & \models^N \texttt{true} & \\ (\beta, C) & \models^N p & \text{iff } p \in \mathit{Mark}(C) \\ (\beta, C) & \models^N \neg\phi & \text{iff not } (\beta, C) \models^N \phi \\ (\beta, C) & \models^N \phi_1 \wedge \phi_2 & \text{iff } (\beta, C) \models^N \phi_1 \text{ and } (\beta, C) \models^N \phi_2 \\ (\beta, C) & \models^N \Diamond\phi & \text{iff } (\beta, C') \models^N \phi \text{ for some finite configuration } C' \supseteq C \\ \beta & \models^N \phi & \text{iff } (\beta, \emptyset) \models^N \phi \\ N & \models^N \phi & \text{iff } \beta_m \models^N \phi. \end{array}$$

E.g., consider the following property Q: It always holds that, if p_1 and p_2 are marked, then possibly p_3 becomes marked. Q is expressed by the formula $\phi = \Box((p_1 \wedge p_2) \Rightarrow (\Diamond p_3))$, which is not satisfied by the P/T-net underlying TN_1.

2.4 McMillan-unfoldings

The McMillan-unfolding of a P/T-net N was defined in [19] as a finite prefix β_f of the maximal branching process β_m of N such that each reachable marking is represented in β_f and, moreover, β_f contains an event labeled t for any transition t that can occur in N. A branching process with these properties will be called *complete* or a *McMillan-unfolding* of N.

The construction of a McMillan-unfolding is based on the notion of a *cut-off event*:

Let $\prec$ denote the partial order defined on the finite configurations of a branching process β by

$$[e] \prec [e'] \text{ iff } |[e]| < |[e']|.$$

An event e of β is a *cut-off event* (w.r.t. $\prec$) iff there exists an event e' of β such that $[e'] \prec [e]$ and $Mark([e]) = Mark([e'])$. Based on $\prec$ a complete finite prefix $\beta_\prec$ of a net system N can be defined as follows. Let E_f be the set of events of β_m given by: $e \in E_f$ iff no event $e' < e$ is a cut-off event and define $\beta_f = (B_f, E_f, F_f, \pi_f)$, the prefix of β_m having E_f as set of events. Note, that if in addition $\prec$ is total, there are at most as many non-cut-off events as there are reachable markings in N, i.e. in the worst case, the size of a McMillan-unfolding is linear in the size of the reachability graph.

Let $Sat(\phi)_N$ and $Sat_f(\phi)_N$ denote, respectively, the set of finite configurations of the maximal branching process and the set of configurations of a McMillan-unfolding of the P/T-net N that satisfy ϕ. Since $Sat(\phi)_N = \emptyset$ implies (not $N \models \phi$) which is equivalent to $N \models \neg\phi$, the model checking problem reduces to checking emptiness of $Sat(\phi)_N$. Esparza's theorem [9] states that for the latter it is sufficient to inspect $Sat_f(\phi)_N$:

Theorem 1. *Let N be a P/T-net and ϕ a formula in $\mathcal{L}$. Then:*

$$Sat(\phi)_N \neq \emptyset \textit{ iff } Sat_f(\phi)_N \neq \emptyset.$$

In [10] an improved algorithm for computing a complete finite prefix of a P/T-net N is presented. All results presented in this paper also apply to the improved algorithm.

2.5 Time Petri Nets

Time Petri nets were introduced in [21] and further investigated e.g. in [24, 26]. A *time Petri net TN* consists of a P/T-net together with an additional inscription of the transitions by closed time intervals with nonnegative rational bounds: $TN = (P, T, F, M_0, \chi)$ such that $\chi : T \to \mathcal{T}$, where $\mathcal{T} = \mathbb{Q}_{\geq 0} \times (\mathbb{Q}_{\geq 0} \cup \{\infty\})$. For $\chi(t) = (\mathit{eft}(t), \mathit{lft}(t))$ we call $\mathit{eft}(t)$ and $\mathit{lft}(t)$ the *earliest firing time* and, respectively, the *latest firing time* of t. The intended meaning is that $\mathit{eft}(t)$ and $\mathit{lft}(t)$ denote the minimal and, respectively, the maximal amount of time which may pass between enabling and successive occurrence of t. A time Petri net TN is called safe if its underlying P/T-net N is safe.

A *state* (M, I) of a time Petri net is given by a marking M of the underlying P/T-net together with a clock vector $I : T \to \mathbb{R}_{\geq 0}$. The clock vector associates a clock to each transition, showing the amount of time that has elapsed since the last enabling of the transition. State (M, I) is called *consistent* iff $I(t) \leq \mathit{lft}(t)$ for all $t \in Enabled(M)$.

For a clock vector I and a time delay $\theta \in \mathbb{R}_{\geq 0}$ such that $I(t) + \theta \leq \mathit{lft}(t)$ for all $t \in Enabled(M)$, $(I + \theta)$ is defined by $(I + \theta)(t) = I(t) + \theta$ for all $t \in T$. The *initial state* (M_0, I_0) is given by the initial marking M_0 of the underlying P/T-net and the *initial clock vector* I_0 defined by $I_0(t) = 0$ for all $t \in T$.

Two types of events are considered for time Petri nets, namely events for passing of time (*time-events*) and events for occurring of transitions (*occur-events*):

1. *Time-events*: A time-event $\theta \in \mathbb{R}_{\geq 0}$ is fireable at state (M, I) iff no transition is forced to occur in between, i.e. iff for all $t \in \mathit{Enabled}(M)$, $I(t) + \theta \leq \mathit{lft}(t)$. In this case, the successor state (M', I') is given by $M' = M$ and $I' = (I + \theta)$. Time-event θ is denoted by $(M, I) \rightarrow^{\theta} (M', I')$.
2. *Occur-events*: An occur-event $t \in T$ is fireable at state (M, I) iff
   - $t \in \mathit{Enabled}(M)$ and
   - $\mathit{eft}(t) \leq I(t) \leq \mathit{lft}(t)$.

   In this case, the successor state (M', I') w.r.t. (M, I) and t is given by $M' = (M \setminus {}^{\bullet}t) \cup t^{\bullet}$ and for all $t' \in T$

   $$I'(t') = \begin{cases} 0 & \text{if } ({}^{\bullet}t' \cap {}^{\bullet}t) \neq \emptyset \\ I(t') & \text{otherwise.} \end{cases}$$

   Occur-event t is denoted by $(M, I) \rightarrow^{t} (M', I')$.

An *occur-step* consists of a nonempty set of concurrently fireable occur-events. A set $\{t_1, \ldots, t_n\} \subseteq T$ of occur-events is *concurrently fireable* at state (M, I) iff, for $1 \leq i \leq n$, t_i is fireable at state (M, I) and, for all $1 \leq i < j \leq n$, ${}^{\bullet}t_i \cap {}^{\bullet}t_j = \emptyset$. The successor state (M', I') is given by element wise application and the occur-step is denoted by $(M, I) \rightarrow^{\{t_1, \ldots, t_n\}} (M', I')$.

Note, that concurrent firing of a time-event and an occur-event is impossible, as is concurrent firing of two different time-events. To unify notation, we will sometimes denote time-events as *time-steps*.

A *firing schedule* of a time Petri net *TN* is a finite or infinite sequence $\sigma = (\overline{t_0}\ \overline{t_1}\ \overline{t_2}\ \ldots)$ such that, for all i, $\overline{t_i}$ is either an occur-step or a time-step. σ is called

- *fireable* from state $S = S_0'$ iff there exist states $S_1', S_2', S_3', \ldots$ such that $\overline{t_i}$ is fireable at S_i'.
- *integral* iff all time steps occurring in σ are integral, i.e. iff $\overline{t_i} \in \mathbb{R}_{\geq 0}$ implies $\overline{t_i} \in \mathbb{N}$ for all i.

State S is called *reachable* from S_0 via σ iff some fireable firing schedule σ leads from S_0 to S. $\mathit{Reach}(\mathit{TN})$ denotes the set of reachable states. A marking M is *reachable via* σ iff, for some clock vector I, there is a state (M, I) reachable via σ.

Based on the following theorem of Popova-Zeugmann [24] only time Petri nets with integral earliest and latest firing times and only time steps of length 1 – denoted by $\surd$ – are considered throughout the rest of the paper.

Theorem 2 (Popova-Zeugmann). *For each time Petri net TN there is a time Petri net TN′ such that*

1. $\mathit{eft}(t) \in \mathbb{N}$ *and* $\mathit{lft}(t) \in (\mathbb{N} \cup \{\infty\})$ *for all* $t \in T$.
2. *A marking M is reachable in TN iff it is reachable via some integral firing schedule σ in TN′.*

Consider the net TN_1 of Fig. 1. The initial clock vector is given by $I_0 = (0,0,0,0)$. For example, the firing schedule $\sigma = (t_2 \surd t_3 \surd t_1\ t_2 \surd t_3)$ is fireable at (M_0, I_0).

The behaviour of a time Petri net TN may be described by the reachability graph [24]. Note that this has the same property as for P/T-nets of not reflecting the concurrent behaviour, but instead representing all interleavings of concurrent events. To describe the concurrent behaviour of time Petri nets, in this section we adapt the notion of a branching process as well as some related notions to time Petri nets.

Definition 1 (Branching process of a time Petri net).

(i) A *homomorphism* from an occurrence net $ON = (B, E, G)$ to a (safe) time Petri net $TN = (P, T, F, M_0, \chi)$ is a mapping

$$\pi : (B \cup E) \to (P \cup T \cup \{\surd\})$$

such that

1. $\pi(B) \subseteq P$ and $\pi(E) \subseteq (T \cup \{\surd\})$.
2. for all $e \in E$ the following holds:
   - If $\pi(e) \in T$, then the restriction of π to ${}^\bullet e$ is a bijection between ${}^\bullet e$ and ${}^\bullet\pi(e)$ and the restriction of π to $e^\bullet$ is a bijection between $e^\bullet$ and $\pi(e)^\bullet$.
   - If $\pi(e) = \surd$, then ${}^\bullet e$ and $e^\bullet$ are cuts of ON and $\pi({}^\bullet e) = \pi(e^\bullet)$.
3. the restriction of π to $Min(ON)$ is a bijection between $Min(ON)$ and M_0.
4. for all $e_1, e_2 \in E$, if ${}^\bullet e_1 = {}^\bullet e_2$ and $\pi(e_1) = \pi(e_2)$ then $e_1 = e_2$.

(ii) A *branching process* of a time Petri net N is a pair $\beta = (ON, \pi)$, consisting of an occurrence net ON and a homomorphism π from ON to TN. β is a *causal process* of N iff ON is a causal net.

Branching process $\beta = (B, E, G, \pi)$ of TN defines a set of transition sequences of TN: Let $e_1 \ldots e_n$ be a sequence of events of β such that, for $1 \leq i < n$, $(e_i, e_{i+1}) \in G \circ G$. Then $\pi(e_1) \ldots \pi(e_n)$ is a transition sequence in N. $OC(\beta)$ denotes the set of these sequences.

Definition 2. *Let TN be a time Petri net and β a branching process of TN. β is called admissible iff every $\sigma \in OC(\beta)$ is a firing sequence of TN.*

The following theorem may be proven by adapting the proof of Engelfriet's corresponding theorem for P/T-nets:

Theorem 3. *For each (finite) time Petri net TN, there exists a unique (up to renaming of conditions and events) admissible branching process β_m called unfolding of TN which is maximal w.r.t. the prefix ordering on the set of all admissible branching processes of TN.*

The notions of a configuration and of a base configuration are defined as for P/T-nets, but w.r.t. admissible processes.

Definition 3. *Let TN be a time Petri net and $\beta = (B, E, G, \pi)$ an admissible branching process of TN.*

(i) A configuration C of β is a downward closed conflict free set of events, i.e., a set $C \subseteq E$ such that $e \in C$ implies $e' \in C$ for all $e' \leq e$, and, for all $e, e' \in C$, $e \# e'$ does not hold. The empty configuration is denoted by $\emptyset$.

(ii) For any event $e \in E$, the basic configuration of e is given by

$$[e] = \{e' \in E \mid e' \leq e\}.$$

The notions of cut and of the marking of a configuration carry over from P/T-nets directly. A new notion relevant for time Petri nets is that of the state of a configuration.

Definition 4 (State of a configuration). *Let TN be a time Petri net, $\beta = (B, E, G, \pi)$ an admissible branching process of TN, and C a configuration of β.*

(i) The clock vector $\mathit{Clocks}(C) : T \to \mathbb{N}$ of C *is defined inductively as follows:*
Base:

$$\mathit{Clocks}(\emptyset)(t) = 0$$

Step: *Let $C = C' + \{e\}$ for some configuration C' of β and some $e \in E$.*
Case 1: $\pi(e) \in T$. Then:

$$\mathit{Clocks}(C)(t) = \begin{cases} \mathit{Clocks}(C')(t) & \text{if } t \in (\mathit{Enabled}(\mathit{Mark}(C)) \\ & \cap \mathit{Enabled}(\mathit{Mark}(C'))) \text{ and} \\ & {}^\bullet t \cap {}^\bullet \pi(e) = \emptyset \\ 0 & \text{otherwise, } t \in \mathit{Enabled}(\mathit{Mark}(C)) \end{cases}$$

Case 2: $\pi(e) = \surd$. Then:

$$\mathit{Clocks}(C)(t) = \begin{cases} \mathit{Clocks}(C')(t) + 1 & \text{if } t \in \mathit{Enabled}(\mathit{Mark}(C)) \\ \mathit{Clocks}(C')(t) & \text{otherwise.} \end{cases}$$

(ii) The state of C is defined as $\mathit{State}(C) = (\mathit{Mark}(C), \mathit{Clocks}(C))$.

A different notion of the change of time is possible when storing time with tokens (resp. conditions). This way it becomes easier to keep the information within the branching process.

Definition 5 (Age of a token). *Let TN be a time Petri net, $\beta = (B, E, G, \pi)$ an admissible branching process of TN and C a configuration of β.*
The age of a token $\mathit{Age} : B \to \mathbb{N}$ is defined as follows:

$$\mathit{Age}(b) = \begin{cases} 0 & \text{if } \pi({}^\bullet(b)) \neq \surd \\ \mathit{Age}(b' + 1) & \text{otherwise, with } b' \in {}^\bullet({}^\bullet(b)) \text{ and } \pi(b') = \pi(b) \end{cases}$$

With this, one can define a state function

$$\mathit{State}'(C) = (\mathit{Mark}(C), \mathit{Age}(\mathit{Cut}(C))).$$

Lemma 1. *Let TN be a time Petri net, $\beta = (B, E, G, \pi)$ an admissible branching process of TN and C a configuration of β.*

$$Clocks(C)(t) = \min_{b \in {}^\bullet(t)} (Age(b))$$

The logic $\mathcal{L}$ is now extended to time Petri nets. The syntax of formulae will not be changed. The satisfaction relation is defined as follows:

Definition 6. *Let TN be a time Petri net and ϕ a formula over TN, i.e. over the underlying P/T-net N. Satisfaction of a formula ϕ w.r.t. a time Petri net TN is defined inductively as follows. Let β denote an admissible branching process of TN and C, C' finite configurations of β. The superscript TN may be dropped.*

$$
\begin{array}{lll}
(\beta, C) \models_t^{TN} \texttt{true} & & \\
(\beta, C) \models_t^{TN} p & \textit{iff } p \in Mark(C) \\
(\beta, C) \models_t^{TN} \neg\phi & \textit{iff not } (\beta, C) \models_t^{TN} \phi \\
(\beta, C) \models_t^{TN} \phi_1 \wedge \phi_2 & \textit{iff } (\beta, C) \models_t^{TN} \phi_1 \textit{ and } (\beta, C) \models_t^{TN} \phi_2 \\
(\beta, C) \models_t^{TN} \Diamond\phi & \textit{iff } (\beta, C') \models_t^{TN} \phi \textit{ for some finite configuration } C' \supseteq C \\
\beta \models_t^{TN} \phi & \textit{iff } (\beta, \emptyset) \models_t^{TN} \phi \\
TN \models_t^{TN} \phi & \textit{iff } \beta_m \models_t^{TN} \phi.
\end{array}
$$

As an example, property Q of the introductory example is expressed by the formula $\phi = \Box((p_3 \wedge p_4) \Rightarrow (\Diamond p_1))$, which is not satisfied by TN_1.

Suppose that we are given a time Petri net TN together with an $\mathcal{L}$-formula ϕ over TN and that we have to check whether TN satisfies ϕ. From the considerations of Sect. 1 it follows that it is not sufficient to check ϕ against the intersection of the maximal branching process β_m of TN and the McMillan-unfolding $McM(N)$ of the underlying P/T-net N. The reason for this is that the definition of a cut-off event does not consider the clock vector part of states of TN. This will be done by the following definition.

Definition 7 (Time cut-off event). *Let $\beta = (B, E, G, \pi)$ be an admissible process of a time Petri net TN and $\prec$ the partial order on configurations defined by $[e] \prec [e']$ iff $|[e]| < |[e']|$. An event $e \in E$ is called (time) cut-off event iff there is some event $e' \in E$ such that*

$$[e'] \prec [e] \textit{ and } State([e']) = State([e]).$$

In the next section we will define the time expansion $X(TN)$ of TN as a safe finite P/T-net embodying the clock vector as part of the structure and its possible changes as part of the behaviour. It will then turn out that the McMillan-unfolding $McM(X(TN))$ of the time expansion is sufficient for checking satisfiability of the formula ϕ.

3 Bisimulation of Time Petri Nets and P/T-nets

In Sect. 4, to each time Petri net TN an 'equivalent' ordinary P/T-net N is associated as its time expansion. Intuitively, TN and N are called equivalent, if both can perform the 'same' set of sequences of (concurrent) steps. In this section, the intuitive relation between a time Petri net and its time expansion is formalized by the notion of place distributive step-bisimulation equivalence. Based on this notion, it is shown that bisimular nets essentially satisfy the same temporal logic formulae.

Definition 8 (step-bisimulation). Let TN be a time Petri net and N a P/T-net. Let $\sim_r \subseteq Reach(TN) \times Reach(N)$ and $\sim_t \subseteq (T_{TN} \cup \{\surd\}) \times T_N$.

(i) $(\sim_r, \sim_t)$ is called *bisimulation from TN to N* iff
  1. $\sim_t^{-1}$ is a mapping from T_N onto $(T_{TN} \cup \{\surd\})$.
  2. $S_{0TN} \ \sim_r \ M_{0N}$.
  3. If $S \rightarrow^{\{t_1,\ldots,t_n\}} S'$ in TN and $S \ \sim_r \ \overline{M}$ for some marking $\overline{M}$ of N, then there exist transitions $t_1',\ldots,t_n'$ of N and a marking $\overline{M'}$ of N such that
     (a) $t_i \ \sim_t \ t_i'$ for $1 \leq i \leq n$,
     (b) $S' \ \sim_r \ \overline{M'}$, and
     (c) $\overline{M} \rightarrow^{\{t_1',\ldots,t_n'\}} \overline{M'}$ in N.
  4. If $\overline{M} \rightarrow^{\{t_1',\ldots,t_n'\}} \overline{M'}$ in N and $S \ \sim_r \ \overline{M}$ for some state S of TN, then there exist transitions $t_1,\ldots,t_n$ of TN and a state S' of TN such that
     (a) $t_i \ \sim_t \ t_i'$ for $1 \leq i \leq n$,
     (b) $S' \ \sim_r \ \overline{M'}$, and
     (c) $S \rightarrow^{\{t_1',\ldots,t_n'\}} S'$ in TN.

(ii) A bisimulation $(\sim_r, \sim_t)$ is called *concurrency preserving* or *step-bisimulation from TN to N* iff, for all $t_1', t_2' \in T_N$ it holds that

$$\text{If } t_1' \neq t_2' \text{ and } \overline{M} \rightarrow^{\{t_1',t_2'\}} \text{ in } N \text{ and } t_i \sim_t t_i', i = 1,2, \text{ then } t_1 \neq t_2.$$

(iii) A bisimulation $(\sim_r, \sim_t)$ is called *place-distributive* (*pd* for short) iff, for all $p \in P_{TN}$, there are $k \in \mathbb{N}$ and $p^1,\ldots,p^k \in P_N$ such that
  - $(M \cup \{p\}, I) \ \sim_r \ \overline{M}$ implies $|\overline{M} \cap \{p^1,\ldots,p^k\}| = 1$ and
  - $(M - \{p\}, I) \ \sim_r \ \overline{M}$ implies $\overline{M} \cap \{p^1,\ldots,p^k\} = \emptyset$.

  For a pd bisimulation $(\sim_r, \sim_t)$ we write $p \sim_r p'$ iff $p' \in \{p^1,\ldots,p^k\}$ with $p^1,\ldots,p^k$ as above.

Proposition 2. *Let N and TN as above and $(\sim_r, \sim_t)$ a step-bisimulation from TN to N. If $(M, I) \rightarrow^{\sigma} (M', I')$ for a step sequence σ in TN and $(M, I) \sim_r \overline{M}$ then there exist $\overline{M'}$ and a step sequence $\overline{\sigma}$ in N such that $(M', I') \sim_r \overline{M'}$, $\sigma \sim_t \overline{\sigma}$*[2] *and $\overline{M} \rightarrow^{\overline{\sigma}} \overline{M'}$.*

Definition 9 (place expansion).
Let N and TN as above and $(\sim_r, \sim_t)$ a pd step-bisimulation from TN to N. The *place expansion* $PX(\phi)$ of a formula ϕ over TN is given by

[2] Application of a relation to a set or a sequence of elements is understood pointwise.

1. $PX(\texttt{true}) = \texttt{true}$
2. $PX(p) = (p^1 \vee \ldots \vee p^k)$ for all $p \in P$, where $\{p^1, \ldots, p^k\} = \{p' \mid p \sim_r p'\}$.
3. $PX(\neg\phi) = \neg(PX(\phi))$.
4. $PX(\phi_1 \wedge \phi_2) = PX(\phi_1) \wedge PX(\phi_2)$.
5. $PX(\Diamond\phi) = \Diamond(PX(\phi)$.

Lemma 2. *Let $(\sim_r, \sim_t)$ a pd step-bisimulation from TN to N. Let $(M, I) \sim_r \overline{M}$ and ϕ a formula over TN. Then:*

$$(M, I) \models_t \phi \textit{ iff } \overline{M} \models PX(\phi)$$

Proof. The proof is done by induction on the structure of ϕ.

Theorem 4. *Let TN be a time Petri net, N a P/T-net, $(\sim_r, \sim_t)$ a place-distributive step-bisimulation from TN to N, and ϕ a formula over TN. Then*

$$TN \models_t \phi \textit{ iff } N \models PX(\phi).$$

4 Expanding Time Restrictions

The key idea of the approach presented here consists of transforming time restrictions into net structure, i.e. representing them by additional places, transitions, and arcs. Following this approach, for a given time Petri net TN the time-expansion $EX(TN)$ is constructed as an ordinary P/T-net. Later on, it will be shown that a formula ϕ is satisfied by TN iff it is satisfied by the time-expansion $EX(TN)$. Subsequently, the McMillan-unfolding $McM(TN)$ of TN is defined by (an abstraction of) $McM(EX(TN))$. As it turns out, for the actual computation of $McM(TN)$ the explicit construction of the (complex) net $EX(TN)$ is not necessary.

The new notion of time-expansion presented here improves the method of [6] in several aspects:

- The restriction of *divergent time* (cf. [6]) is removed.
- Transitions may have an infinite upper time bound.
- TN and $EX(TN)$ are bisimular.
- The finite prefix of $EX(TN)$ may be computed directly and efficiently.

Consider a time Petri net TN. Its behaviour differs from the behaviour of the underlying P/T-net in that firing of transitions depends also on the current clock vector, and that time steps are possible. Hence both aspects have to be included in the time expansion.

Clock vectors will not directly be represented in the time expansion. Instead, for each place and each token we will keep track of the number of time units elapsing while the token resides in the place. This is done by providing appropriate instances of each place storing the token age. Fireability of a transition then depends on the minimal token age in its preset. Hence, for each transition, we have to provide one copy for each instance of the preset at which the transition may fire.

In general, a token could reside in a place forever. Unfortunately this would lead to an infinite number of copies needed for such a place. This is avoided by giving a finite upper bound for each place such that we do not need to distinguish different token ages above this bound.

Following these considerations, a time step at a marking M is represented by a transition which increments the token age for each place marked under M. Here the following problem arises: If $M' \subseteq M$ in the time expansion, and the time step for M is enabled, then the time step for M' is also enabled. But occurrence of the time step for M' would increase the age of some tokens, leaving some tokens untouched. This would lead to a wrong time distribution. This problem is fixed by adding one more instance for each place indicating that the place is empty. Moreover, for each place which is not marked at M, this instance is added to both the preset and the postset of the time step. Hence, for each time step, exactly one instance of each place is contained in the preset as well as in the postset.

For the formal definition of time expansion we need some more notation. So let $TN = (P, T, F, M_0, \chi)$ be a time Petri net. Then:

– For each $p \in P$,

$$max(p) = \begin{cases} 0 & \text{if } p^\bullet = \emptyset \\ max(\{eft(t) \mid t \in p^\bullet\}) & \text{otherwise and} \\ & lft(t) = \infty \text{ for all } t \in p^\bullet \\ max(\{lft(t) \mid t \in p^\bullet \wedge lft(t) \neq \infty\}) & \text{otherwise.} \end{cases}$$

– Let $Q \subseteq P$. Then $\mu : P \to (\mathbb{N} \cup \{\perp\})$ is called *Q-instance* iff
  - $\mu(p) \in \mathbb{N}$ iff $p \in Q$ and
  - $\mu(p) \leq max(p)$

  for all $p \in P$.
– Let $Q \neq \emptyset$ and μ a Q-instance. Then $min(\mu) = min\{\mu(p) \mid \mu(p) \in \mathbb{N}\}$.
– For a Q-instance μ we define the Q-instance $\mu \oplus 1$ by

$$(\mu \oplus 1)(p) = \begin{cases} \perp & \text{if } \mu(p) = \perp \\ max(p) & \text{if } \mu(p) = max(p) \\ \mu(p) + 1 & \text{otherwise.} \end{cases}$$

Definition 10 (Time-Expansion).
Let $TN = (P, T, F, M_0, \chi)$ be a time Petri net. The *time-expansion* of TN is given by the P/T-net

$$EX(TN) = (P_{EX}, T_{EX}, F_{EX}, M_{EX_0})$$

defined as follows:

– $P_{EX} = \bigcup_{p \in P}\{p^\perp, p^0, \ldots, p^{max(p)}\}$.

- $T_{EX} = \{t^\mu \mid t \in T$ and μ is a ${}^\bullet t$-instance s.t. $min(\mu) \geq eft(t)\}$
  $\cup \{\surd^\mu \mid \mu(p) < max(p)$ for all $p \in P\}$.
- F_{EX} is given by
  - ${}^\bullet(t^\mu) = \{p^{\mu(p)} \mid p \in {}^\bullet t\}$
  - $(t^\mu)^\bullet = \{p^0 \mid p \in t^\bullet\}$
  - ${}^\bullet(\surd^\mu) = \{p^{\mu(p)} \mid p \in P\}$
  - $(\surd^\mu)^\bullet = \{p^{\mu(p)\oplus 1} \mid p \in P\}$.
- $M_{0_{EX}} = \{p^0 \mid p \in M_0\} \cup \{p^\perp \mid p \notin M_0\}$.

Theorem 5. *Let TN be a time Petri net and EX(TN) its time expansion. Then TN and EX(TN) are place-distributive step-bisimular.*

Corollary 1. *Let TN be a time Petri net and ϕ a formula over TN. Then:*

$$TN \models_t \phi \textit{ iff } EX(TN) \models PX(\phi) \textit{ iff } McM(EX(TN)) \models PX(\phi).$$

5 Computation of McMillan-unfoldings

Now we have reduced the model checking problem of a time Petri net TN to the model checking problem of the P/T-net $EX(TN)$, but at the expense of a larger formula. Since each reachable marking of $EX(TN)$ contains exactly one instance of each place p of TN, the same holds true for each cut of the maximal branching process of $EX(TN)$. Hence, we may as well remove the $\perp$-instances of places, forget about the 'labels' of the remaining places, and check the resulting process against the original formula:

Definition 11 (Reduced branching process). Let TN be a time Petri net, $EX(TN)$ its time expansion, and β a process of $EX(TN)$. The *reduction $Red(\beta)$ of β* is constructed from β by

- removing all conditions labeled $p^\perp$ for some $p \in P_{TN}$ and
- replacing
  - each label of the form p^n for some $p \in P_{TN}$ by p,
  - each label of the form t^μ for some $t \in T_{TN}$ by t, and
  - each label of the form $\surd^\mu$ by $\surd$.

Theorem 6. *Let TN be a time Petri net, EX(TN) its time expansion, and β a process of EX(TN). Then*

$$McM(EX(TN)) \models PX(\phi) \textit{ iff } Red(McM(EX(TN))) \models \phi.$$

Hence, it is reasonable to define the McMillan-unfolding of a time Petri net in the following way:

Definition 12 (McMillan-unfolding). Let TN be a time Petri net. The McMillan-unfolding $McM(TN)$ of TN is defined by

$$McM(TN) = Red(McM(EX(TN))).$$

Corollary 2. *The McMillan-unfolding $McM(TN)$ of a time Petri net TN is sufficient for model checking of formulae over TN.*

Corollary 3. *The finite prefix may also be computed directly using the State′ function. Given the original algorithm using the cut-off condition*

$$[e'] \prec [e] \textit{ and } State'([e']) = State'([e]).$$

Remark 1. Starting from the order $\prec$ defined in [10] we are able to define an adequate total order $\prec_t$ for time Petri nets. As a consequence, we are able to show that the size of $\beta_{\prec_t}$ is linearly bounded in the number of reachable states of *TN*.

6 Experimental Results

This section provides the results of the algorithm on some small but yet complete examples. The algorithm has been implemented as an extension of the already existing model checker from the PEP tool[23]. This implementation is available from the Internet, but does not yet offer the improved total order from [10]. Nevertheless, it proves the proposed theory to be feasible and useable on a range of problems. All computations were made on a dual processors Linux system with 2x450 MHz and a total of 1GByte of RAM.

The first example modelled with time Petri nets is a simple version of the alternating bit protocol. It is a well-known transmission protocol consisting of a server that sends messages and a client who receives them replying an acknowledge message in case of success. Messages are sent via lossy channel so data and ack packets may be lost. In both cases the message is resent after some time out. Time intervals are set for all actions concerning the channel, i.e. sending and receiving, and also for some delays during transmission as we have an asynchronous protocol. The size of the net and its unfolding are given in tab. 2 together with its unfolding time and deadlock checking times in seconds.

Places	Transitions	Conditions	Events	Unfolding (s)	Checking(s)
25	25	2946	932	.2	.3

Fig. 2. Alternating bit protocol

The second example given is an implementation of Fischer's protocol ([1]). It is a very basic mutual exclusion protocol only allowing one process to reach the criticial section and then stopping, though it can be extended for an arbitrary number of rounds. The protocol itself is very simple and scalable for any number of processes. It consists of a global variable and special timing parameter on every round of the algorithm. Provided the interval for writing ends before the one for reading starts the protocol guarantees letting only one process enter the critical section. Due to its simplicity and scalability this protocol is likely to be used as a benchmark for timed systems (cf. [17]). Figure 3 shows a Petri net model for 2 processes and a global variable v. A process enters its critical section after firing the last transition with attached cs. The other transitions have inscriptions stating their influence on v.

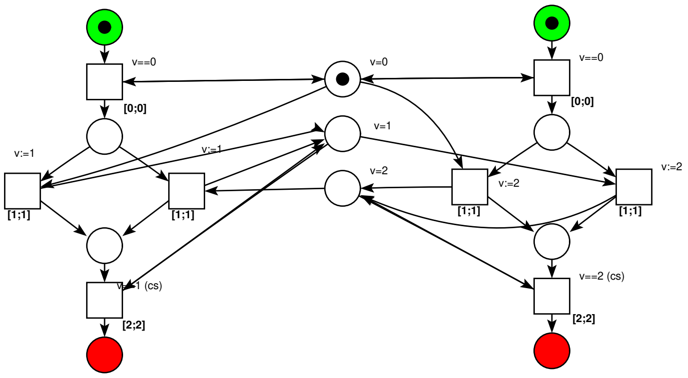

Fig. 3. Fischer's protocol for 2 processes.

As shown in Table 4 the size of the unfolding grows exponentially with the number of processes. This limits the scope of the proposed algorithm, though the table also shows its benefits. The time for checking of the mutual exclusion property is done in almost no time and does not change much with the growing unfolding. So the crucial part for further investigations will be the efficiency of the unfolder. As the current implementation does not include the proposed technique from [10] and has not been optimized in its implementation, the algorithm itself may yield some promising improvements, besides e.g. parallel programming benefits like [15].

Processes	Places	Transitions	Conditions	Events	Unfolding (s)	Checking(s)
2	11	8	127	47	<.1	<.1
3	16	15	3616	1080	0.6	.1
4	21	24	245025	63451	4572.9	.4

Fig. 4. Fischer's protocol

7 Related Work

Approaches to model checking of time Petri nets w.r.t. (timed) temporal logics are either based on the timed state graph (TSG) [28, 29], or on the timed branching process [3, 18, 28] of a time Petri net.

In [29], model checking of safe time Petri nets with rational firing bounds w.r.t. a real time extension of linear time temporal logic is considered, which is based on differences between timing variables rather than absolute time points. A finite graph representing the state space is computed whose nodes consist of reachable markings augmented by additional sets of inequalities describing timing conditions. Since each reachable marking is directly represented by at least one node, this approach faces the state explosion problem.

The time state graph construction of [29] is improved in [18] in several ways. First, a coarser semantics which does not take into account implicit ordering of events is considered. Second, the finite prefix of the underlying net is used to speed-up searching the time state graph.

Model checking against TCTL is explored in [28]. Here, a region graph construction is used to compute a finite representation of the timed state space, cf. the construction for timed automata in [2].

A notion of timed process of a time Petri net, denoting a causal process which is realizable under the time constraints of the net, is introduced in [3]. A method is developed to compute all valid timings for a causal process of a time Petri net with divergent time.

In [25], the class of safe time Petri nets is restricted to *time independent choice* nets and analysed w.r.t. reachability. The reachability checking algorithm is based on a finite prefix of the underlying net. Like in [29], timing conditions are represented explicitely by sets of inequalities as additional inscriptions of events. Using the prefix construction, an asynchronous circuit with bounded delay is verified w.r.t. hazard-freedom.

Building a complete finite prefix is still a bottleneck in partial order verifications. Although [10] gave a major improvement in terms of the size of the unfolding, the construction is still too slow for large systems. Some new algorithms try to cope with this. In [16] the search for possible new events in the unfolding is changed. With information from the structure of the net it is possible to reduce the number of calls to this search. Furthermore the search may be improved if transition's presets are kept in a tree structure to avoid redundancy.

In [14] the cutoff-criterion is refined. With the use of logic programs it is shown that a much smaller prefix may suffice to check reachability and deadlock properties. The approach was implemented as a minimizer after actual unfolding but may also be integrated into the unfolding algorithm. As the question if this minimized prefices are useable for model checking is left open, its benefits for a general unfolder are harder to predict.

The approach presented in this paper may also be compared to timed automata (cf. e.g. [2]). E.g. checking reachability in a time Petri net TN can be reduced to reachability analysis in a timed automaton TA in the following way: Given a safe time Petri net TN, first compute its reachability graph $G = (V, E)$, giving the states and the transitions of TA. Introduce a clock c_t for each transition t of TN. The inscription of a transition $M \xrightarrow{t} M' \in E$ consists of the guard $\mathit{eft}(t) \leq c_1 \leq \mathit{lft}(t)$ and the set of clocks to be reset is given by $\{t' \mid t' \in \mathit{Enabled}(M - {}^\bullet t)\}$. Moreover, for each state M, the invariant is given by `true`. Unfortunately, this construction is very expensive: First, the size of G may be exponential in the size of TN; second, in order to check reachability in TA, the region graph construction is needed, bringing along an additional exponential blow up.

On the other hand, simulation of general timed automata by time Petri nets seems to be unfeasible.

8 Conclusion

We have defined the finite prefix of a time Petri net TN, based on the time-expansion of TN. By showing that this is sufficient for applying Esparza's model checking algorithm for safe P/T-nets, we have established a first – but still substantial – step towards quantitative analysis of distributed real-time systems based on partial order semantics. The next steps will consist in extending our approach to *real-time temporal logic*.

Acknowledgements

The authors would like to thank the members of the BAT/PORTA group for helpful comments and discussions on the paper.

References

1. M. Abadi and L. Lamport: An Old-Fashioned Recipe for Real-Time. In *Real Time: Theory in Practice*. Volume 600 of LNCS. Springer-Verlag (1992) 1–27
2. R. Alur, C. Courcoubetis, and D. Dill: Model Checking for Real-time Systems. In *Proc. 5th Symp. LICS.* (1990) 414–425
3. T. Aura and J. Lilius: A causal semantics for time Petri nets. In *Theoretical Computer Science*, Vol 243, No. 1–2 (2000) 409–447
4. B. Berthomieu, M. Diaz: Modelling and Verification of Time Dependent Systems Using Time Petri Nets. In *IEEE Transactions on Software Engineering*, Volume 17/3 (1991) 259–273
5. E. Best: Partial Order Verification with PEP. In G. Holzmann, D. Peled, and V. Pratt (eds.): *Proc. of POMIV'96*, American Mathematical Society (1996) 305–328
6. B. Bieber and H. Fleischhack: Model Checking of Time Petri Nets Based on Partial Order Semantics. In *Proceedings of CONCUR'99.* Volume 1664 of LNCS. Springer-Verlag (1999) 210–225
7. E.M. Clarke, E. A. Emerson, and A. P. Sistla: Automatic Verification of Finite-state Concurrent Systems Using Temporal Logic Specifications. *ACM TOPLAS* 8 (1986) 244–263

8. J. Engelfriet: Branching Processes of Petri Nets. *Acta Informatica* **28**. (1991) 575–591
9. J. Esparza: Model Checking Using Net Unfoldings. *Science of Computer Programming* **23**. Elsevier (1994) 151–195
10. J. Esparza, S. Römer, and W. Vogler: An Improvement of McMillan's Unfolding Algorithm. In *Proc. of TACAS'96*. Volume 1055 of LNCS. Springer-Verlag (1996) 87–106
11. H. Fleischhack, and J. Tapken: An M-Net Semantics for a Real-Time Extension of μSDL,. In *Proc. FME'97*. Graz (1997).
12. P. Godefroid: Using partial orders to improve automatic verification methods. In *Proc. Workshop on Computer Aided Verification* (1990) 176–185
13. B. Graves: Computing Reachability Properties Hidden in Finite Net Unfoldings. In *Proc. of FST&TCS'97*. Volume 1346 of LNCS. Springer-Verlag (1997) 327–341
14. K. Heljanko: Minimizing Finite Complete Prefixes. In *Proceedings of CS&P'99*. Warsaw University (1999) 83–95
15. K. Heljanko, V. Khomenko and M. Koutny: Parallelisation of the Petri Net Unfolding Algorithm. To appear in *Proc. of TACAS'2002*. Grenoble, France (2002)
16. V. Khomenko, and M. Koutny: Towards An Efficient Algorithm for Unfolding Petri Nets. In *Proc. of CONCUR'2001*, K.G.Larsen, M.Nielsen (Eds.). Volume 2154 of LNCS. Springer-Verlag (2001) 366–380.
17. K. Larsen, P. Petterson and W. Yi: Model-Checking for Real-Time Systems. In *Proc. of Fundamentals of Computation Theory*. Volume 965 of LNCS. Springer-Verlag (1995) 62–88.
18. J. Lilius: Efficient State Space Search for Time Petri Nets. In *MFCS Workshop on Concurrency*. Volume 18 of ENTCS. Elsevier (1998).
19. K. McMillan: *Symbolic Model Checking – an Approach to the State Explosion Problem*. PhD Thesis, SCS, Carnegie Mellon University (1992).
20. K.L. McMillan: Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In *Proc. Workshop on Computer Aided Verification* Volume 663 of LNCS. Springer-Verlag (1992) 164–174
21. P. Merlin, D. Farber: Recoverability of Communication Protocols – Implication of a Theoretical Study. *IEEE Transactions on Software Communications* **24** (1976) 1036–1043
22. X. Nicollin, J. Sifakis, and S. Yovine: From ATP to Timed Graphs and Hybrid Systems. *Acta Informatica* **30**. Springer-Verlag (1993) 181–202
23. *PEP homepage* http://parsys.informatik.uni-oldenburg.de/~pep
24. L. Popova: On Time Petri Nets. *Journal of Information Processing and Cybernetics*. EIK 27:4 (1991) 227–244.
25. A. Semenov, A. Yakovlev, and A. Koelmans: *Time Petri net unfoldings and hardware verification*. University of Newcastle (1998) (Draft)
26. P. Starke: *Analyse von Petri-Netz-Modellen*. Teubner Verlag, Stuttgart (1990) (in German)
27. C. Stehno: *Entfaltung von Zeit-Petrinetzen, Integration in das PEP-System und Optimierungen*. University of Oldenburg (2000) (Master thesis, in german)
28. I. Virbitskaite, E. Pokozy: *On Partial Order Verification of Time Petri Nets*. University of Novosibirsk (1998) (Draft)
29. T. Yoneda, B.-H. Schlingloff: Efficient Verification of Parallel Real-Time Systems. *Journal of Formal Methods in System Design*, Vol. 11-2. (1997) 187–215