Model Checking of Time Petri Nets Based on Partial Order Semantics

Burkhard Bieber, Hans Fleischhack
Fachbereich Informatik
Carl-von-Ossietzky – Universität Oldenburg
D-26111 Oldenburg
{bieber,fleischhack}@informatik.uni-oldenburg.de

Abstract

Model checking of place/transition-nets based on partial order semantics has been applied successfully to the analysis of causal behaviour of distributed systems. Here, this approach is extended to the causal behaviour of time Petri nets. Expansion of a time Petri net to an equivalent P/T-net is defined, and it is shown that (an abstraction of) the McMillan unfolding of the expanded net is sufficient for model checking w.r.t. formulae of a simple branching time temporal logic $\mathcal{L}$.

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 [13] or BDD-based techniques [7, 16] also methods based on partial order semantics have been applied. The latter have proven especially successful with 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. [1, 19]), but much less work has been done starting from partial order semantics (cf. section 5 for some exceptions). By extending McMillan's [17] technique of unfolding safe Petri nets to the class of safe time Petri nets [18] this paper takes a step in this direction.

In [9], a branching time temporal logic $\mathcal{L}$ for safe nets is introduced. A model checking algorithm is given based on the finite prefix of the maximal branching process of a net N, the McMillan-unfolding $McM(N)$ of N (cf. also [10]). One might naively assume that in order to extend this approach to safe time Petri nets, it is sufficient to take the McMillan-unfolding of the underlying P/T-net and reduce it to that part which is not prohibited by the time restrictions. The following example of Figure 1 shows that this is not true.

First, consider the net TN_1 (Figure 1 (i)) without time restrictions. Its possible (concurrent) behaviour is described in Figure 1 (ii). All behaviour happens to remain realizable under the time requirements in the sense that for each concurrent run of the system there is a timing schedule respecting the requirements.

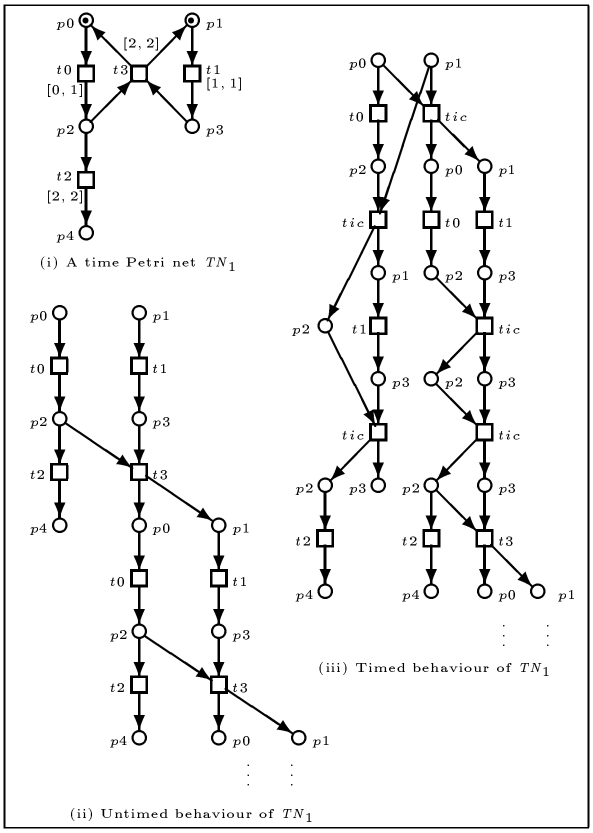

Fig. 1. A time Petri net TN_1 and its behaviour

The untimed behaviour of TN_1 (cf. Figure 1(ii)) satisfies the following property Q: It always holds that, if p_2 and p_3 are marked, then possibly p_0 becomes marked.

But, if t_0 occurs in TN_1 at time 0, then t_2 becomes enabled at time 0 and has to occur at time 2. On the other hand, t_1 is forced to occur at time 1, so t_3 is enabled at time 1. However, t_3 cannot occur before time 3, and, since p_2 is not marked after time 2, t_3 can never occur at all. Hence, this property is not satisfied by TN_1. This is reflected by the net's behaviour if we represent lapse of time by inserting special events called *tic-events* (cf. Figure 1 (iii)): If t_0 occurs at time 0 (i.e. before any tic-event), then t_3 never becomes enabled.

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 $X(TN)$ is constructed as an ordinary P/T-net. Moreover, it is shown that an $\mathcal{L}$-formula ϕ is satisfied by TN iff it is satisfied by the time-expansion $X(TN)$. Subsequently, the McMillan-unfolding $McM(TN)$ of TN is defined by (an abstraction of) $McM(X(TN))$. As will be shown in [11], for the actual computation of $McM(TN)$ the explicit construction of the (complex) net $X(TN)$ is not necessary.

An additional benefit of this approach is given by the fact that corresponding tools for P/T-nets may be reused for the causal analysis of time Petri nets, which applies especially to the model checking component of the PEP tool [5].

We proceed as follows: In section 2 the notions used throughout the paper are introduced. Section 3 contains the definition of time expansion of time Petri nets. The new method for model checking of time Petri nets is given in section 4, based on the 'finite prefix' of the time expansion. We conclude with discussing some related work (section 5) and with remarks on further research (section 6).

An extended version of this paper is presented in [6].

2 Basic Notions

In this section, a partial order semantics as well as temporal logics for safe Place/Transition-nets (P/T-nets) is defined and a notion of time is added to P/T-nets. Moreover, the temporal logics is extended to time Petri nets. Note that most notions carry over to bounded P/T-nets.

2.1 Processes of Place/Transition-nets

Let $N = (P, T, F, M_0)$ be a P/T-net. N is called *safe* iff $M(p) \in \{0, 1\}$ for all $p \in P$ holds for any reachable marking. For the rest of the paper, only safe nets are considered. Nodes $x_1, x_2 \in (P \cup T)$ 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 $x \# x$.

Often, the behaviour of a P/T-net N is described via the *reachability graph*. Its vertices consist of the set of all reachable markings of N and there is an edge leading from M to M', which is labeled by t iff $M \rightarrow^t M'$. However, using the reachability graph the concurrent behaviour of N cannot be retrieved easily and all interleavings of concurrent behaviour have to be represented explicitly, which often leads to an unnecessary exponential blow up.

In part, these problems are avoided by the maximal branching process, which associates a partial order semantics to each safe 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* may represent different alternative runs of N in one structure and hence may seen as the union of some causal processes. It consists of an occurrence net and a homomorphism.

An *occurrence net* $CN = (B, E, G)$ is an acyclic net such that $| {}^\bullet b | \leq 1$ 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 (E, respectively) are called *conditions* (*events*, respectively). $\leq$ denotes the partial order induced by G on $B \cup E$; $<$ denotes the corresponding strict partial order;

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

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

A *branching process* of a P/T-net N is a pair $\beta = (CN, \pi)$, consisting of an occurrence net CN, and a homomorphism π from CN to N. β is a *causal process* of N iff CN is a causal net. For each P/T-net N, there exists a unique (up to renaming of conditions and events) maximal branching process β_m, where 'maximal' is related to the prefix ordering (cf. [8]). 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. 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 $\beta = (CN, \pi)$ uniquely determines a causal process, containing C as set of events and all conditions connected to elements of C in CN as set of conditions. Arcs and labels are also inherited from CN. 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 CN which are marked after occurrence of all events of C: A *co-set* is a set B' of conditions of an occurrence net CN 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 $Cut(C) = (Min(C) \cup C^\bullet) \setminus {^\bullet C}$. A configuration C of a process $\beta = (CN, \pi)$ defines a marking of CN by $Mark(C) = p(Cut(C))$. Note that a marking M is reachable in a P/T-net N iff the maximal branching process β_m of N contains some configuration C such that $M = Mark(C)$.

Figure 1(ii) 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.

2.2 A Temporal Logic for Safe P/T-Nets

In this section we introduce the temporal 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.

Let N be a P/T-net. The syntax of $\mathcal{L}$ over N is presented in Figure 2.

Properties of the current marking of a P/T-net N are expressed using place assertions. E. g., a formula '$s_1 \wedge s_2$' expresses that places s_1 and s_2 are marked. '$\Diamond\phi$' means that a marking satisfying 'ϕ' is reachable. The derived operator '$\Box\phi$' signifies that 'ϕ' is satisfied at all reachable markings.

$$
\begin{array}{rll}
\phi ::= & \texttt{true} & \text{(Truth)} \\
 & \mid s \in P_N & \text{(Place Assertion)} \\
 & \mid \neg\phi & \text{(Negation)} \\
 & \mid \phi_1 \wedge \phi_2 & \text{(Conjunction)} \\
 & \mid \Diamond\phi & \text{(Possibly } \phi\text{)} \\
 & & \\
\texttt{false} & = \neg\texttt{true} & \text{(Falsehood)} \\
(\phi_1 \vee \phi_2) & = \neg(\neg\phi_1 \wedge \neg\phi_2) & \text{(Disjunction)} \\
(\phi_1 \Rightarrow \phi_2) & = \neg\phi_1 \vee \phi_2 & \text{(Implication)} \\
\Box\phi & = \neg\Diamond\neg\phi & \text{(Always } \phi\text{)}
\end{array}
$$

Fig. 2. Esparza's Temporal Logic

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

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

The McMillan-unfolding of a P/T-net N was defined in [16] as a finite prefix of the maximal branching process β_m of N such that each reachable marking of N occurs as an image of some cut of this prefix. Let N be a P/T-net and β_m its maximal branching process. Let $\perp$ be a new event ('pseudo-event') and define $[\perp] = \emptyset$. An event $e \in E_m \cup \{\perp\}$ is a *cut-off event* iff there exists an event e' such that $| [e'] | < | [e] |$ and $\mathit{Mark}([e]) = \mathit{Mark}([e'])$. 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. $\beta_f = (B_f, E_f, F_f, \pi_f)$, the prefix of β_m having E_f as set of events, is called *McMillan-unfolding* of N and denoted by $\mathit{McM}(N)$. It can be shown to be unique and finite.

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

To simplify notation, a different but equivalent definition of the satisfaction relation will be applied in this paper. Let N be a P/T-net, ϕ an $\mathcal{L}$-formula, and M a marking of N. Then:

$$\begin{array}{lll}
M \models' \texttt{true} & & \\
M \models' p & \text{iff} & p \in M \\
M \models' \neg\phi & \text{iff not} & M \models' \phi \\
M \models' \phi_1 \wedge \phi_2 & \text{iff} & M \models' \phi_1 \text{ and } M \models' \phi_2 \\
M \models' \Diamond\phi & \text{iff} & M' \models' \phi \text{ for some marking } M' \text{ reachable from } M \\
N \models' \phi & \text{iff} & M_0 \models' \phi.
\end{array}$$

Proposition 1. *Let N be a P/T-net and ϕ a formula over N. Then:*

$$N \models \phi \textit{ iff } N \models' \phi.$$

2.3 Time Petri Nets

Time Petri nets were introduced in [18]. Following the reasoning of [20, 22], we consider time Petri nets with discrete time (also cf. [6]).

A *safe time Petri net TN* net consists of a safe P/T-net (P_{TN}, T_{TN}, F_{TN}) and a transition inscription $\chi_{TN} : T \to \mathcal{T}$ (where $\mathcal{T} = \mathbf{N} \times \mathbf{N}$) by closed time intervals with nonnegative integer bounds. The subscripts are omitted if clear from the context.

For $\chi(t) = (\mathit{eft}(t), \mathit{lft}(t))$ we call $\mathit{eft}(t)$ ($\mathit{lft}(t)$, respectively) the *earliest firing time* (*latest firing time*, respectively) of t. The intended meaning is that $\mathit{eft}(t)$ ($\mathit{lft}(t)$, respectively) denotes the minimal (maximal, respectively) number of time units which may pass between enabling and occurrence of t. $\chi(t)$ is denoted by $(\mathit{eft}, \mathit{lft})$ if t is clear from the context. 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 $I : T \to (\mathbf{N} \cup \{\$\})$ such that $I(t) = \$$ or $I(t) \leq \mathit{lft}(t)$ for all $t \in T$. The clock vector associates a clock to each transition, showing the number of time units that have elapsed since enabling of the transition, the \$-symbol indicating that the corresponding transition is not enabled at all. State (M, I) is called *consistent* iff, for all $t \in T$, $I(t) \neq \$ \Leftrightarrow t \in \mathit{Enabled}(M)$. Only consistent states will be considered.

For a clock vector I and a time delay $\theta \in \mathbf{N}$ such that $I(t) + \theta \leq \mathit{lft}(t)$ for all $t \in \mathit{Enabled}(M)$, $(I + \theta)$ is defined by

$$(I + \theta)(t) = \begin{cases} I(t) + \theta & \text{if } t \in \mathit{Enabled}(M) \\ \$ & \text{otherwise.} \end{cases}$$

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

$$I_0(t) = \begin{cases} 0 & \text{if } t \in \mathit{Enabled}(M) \\ \$ & \text{otherwise.} \end{cases}$$

Two types of events are considered for a time Petri net, namely events during which time passes (called *tic-events*) and events during which a transition occurs (called *occur-events*):

1. *Tic-event*: The tic-event *tic* is *fireable at state* (M, I) iff no transition is forced to occur in between, i.e. iff for all $t \in \mathit{Enabled}(M)$, $I(t) < \mathit{lft}(t)$. In this case, the successor state (M', I') is given by $M' = M$ and $I' = (I + 1)$. The tic-event is denoted by $(M, I) \to^{tic} (M', I')$.
2. *Occur-events*: An occur-event is *fireable at state* (M, I) iff some transition t may occur, i.e. if $t \in \mathit{Enabled}(M)$ and $\mathit{eft}(t) \leq I(t) \leq \mathit{lft}(t)$. In this case, the successor state (M', I') is given by $M' = (M \setminus {}^\bullet t) \cup t^\bullet$ and

$$I'(t') = \begin{cases} \$ & \text{if } t' \notin \mathit{Enabled}(M') \\ 0 & \text{if } t' \notin \mathit{Enabled}(M \setminus {}^\bullet t) \wedge t' \in \mathit{Enabled}(M') \\ I(t') & \text{otherwise.} \end{cases}$$

   An occur-event is denoted by $(M, I) \to^t (M', I')$.

A set $\{t_1, ... t_n\}$ of transitions is *concurrently fireable* from state (M, I) iff, for $1 \leq i \leq n$, t_i is fireable from state (M, I) and, for all $1 \leq i < j \leq n$, ${}^\bullet t_i \cap {}^\bullet t_j = \emptyset$. A *firing schedule* of a time Petri net TN is a finite or infinite sequence $\sigma = (\overline{t_0}\ \overline{t_1}\ \overline{t_2}\ ...)$ such that, for all i, $\overline{t_i} \in T$ or $\overline{t_i} = tic$. σ is *fireable at state* $S = S_0'$ iff there exist states $S_1', S_2', S_3', ...$ such that $\overline{t_i}$ is fireable at S_i'. State S is called *reachable* from S_0 via σ iff some fireable firing schedule σ leads from S_0 to S. A marking M is *reachable via* σ iff, for some clock vector I, there is a state (M, I) reachable via σ.

Consider the net TN_1 of Figure 1. The initial clock vector is given by $I_0(t_0) = I_0(t_1) = 0$ and $I_0(t_2) = I_0(t_3) = \$$. For example, the firing schedule $\sigma = (tic\ t_0\ t_1\ tic\ tic\ t_3\ t_0\ tic\ t_1)$ is fireable at (M_0, I_0).

The behaviour of a time Petri net TN can be described by the reachability graph [20], which – in the case of P/T-nets – faces the same problems of not reflecting the concurrent behaviour, but instead representing all interleavings of concurrent events.

The logic $\mathcal{L}$ may be extended to time Petri nets. Let TN be a time Petri net, ϕ a formula, and (M, I) a marking of TN. The superscript TN is dropped if it is clear from the context.

$$\begin{array}{lll}
(M,I) \models_t^{TN} \texttt{true} & & \\
(M,I) \models_t^{TN} p & \text{iff} & p \in M \\
(M,I) \models_t^{TN} \neg\phi & \text{iff} & \text{not } (M,I) \models_t^{TN} \phi \\
(M,I) \models_t^{TN} \phi_1 \wedge \phi_2 & \text{iff} & (M,I) \models_t^{TN} \phi_1 \text{ and } (M,I) \models_t^{TN} \phi_2 \\
(M,I) \models_t^{TN} \Diamond\phi & \text{iff} & (M',I') \models_t^{TN} \phi \text{ for some state} \\
 & & (M',I') \text{ reachable from } (M,I) \\
TN \quad \models_t^{TN} \phi & \text{iff} & (M_0,I_0) \models_t^{TN} \phi.
\end{array}$$

E.g. property Q of the introductory example is expressed by the formula $\phi = \Box((p_2 \wedge p_3) \Rightarrow (\Diamond p_0))$, which is not satisfied by TN_1.

3 Expanding Time Restrictions

In this section, the notion of time-expansion of a time Petri net TN into a P/T-net $N = X(TN)$ is introduced. It is shown that N captures the behaviour of TN

in the sense that satisfaction w.r.t. the temporal logic $\mathcal{L}$ is preserved. In general, the size of $X(TN)$ may be exponential in the size of TN, but the unfolding of TN may be generated without explicitly constructing $X(TN)$ (cf. [11]). Still, the definition of $X(TN)$ is needed to obtain a proper definition of the finite prefix of TN.

For the construction of N we will start with the P/T-net underlying TN and then add parts representing the components of TN related to time. First, clock vectors have to be represented in N. This is done by adding new places for the possible clock positions for each transition t.

Next, new transitions are introduced modelling tic-events. Since we would like to represent time dependence of conditions and events in TN by causal dependence in the time-expansion, these have to take into account also $TN's$ 'original' places. The obvious solution is to introduce, for each state (M, I) of TN which enables a tic-event, a new transition, having M and the places representing I as preset and M and the places representing $(I+1)$ as postset. But, this solution causes a problem: If $M' \subseteq M$, then the tic-transition for (M', I) instead of the tic-transition for (M, I) may fire, possibly leaving some places untouched. To avoid this problem, for each place p of TN, we introduce a new 'complementary' place p^c. Consequently, the marking part of (M, I) is represented by $M \cup M^c$ where $M^c = \{p^c \mid p \in P \wedge p \notin M\}$ in N.

At last, we have to re-adjust the transitions of TN, since these have to respect the complementary places and the clock places (fire-transitions). The first task is fulfilled by having ${}^\bullet t \cup \{p^c \mid p \in t^\bullet\}$ as part of the preset and $t^\bullet \cup \{p^c \mid p \in {}^\bullet t\}$ as part of the postset (except for those places which are contained in both preset and postset of t). The second task raises new problems (see figure 3).

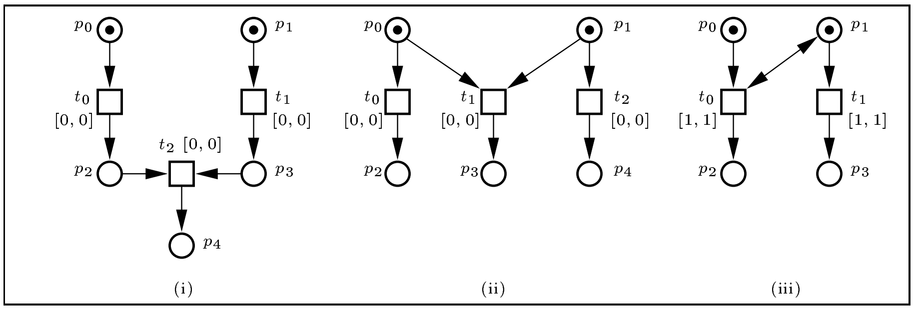

Fig. 3. Problems with updating of clocks

First, consider the time Petri net in part (i) of figure 3. If both t_0 and t_1 fire, then the clock for t_2 has to be started. But this can be decided neither by t_0 nor by t_1 alone. We will solve this problem by allowing fire-transitions to produce an intermediate marking representing an inconsistent state and by introducing additional repair-transitions called update-transitions. Subsequently, also the tic-transitions have to be re-adjusted to act properly on inconsistent states. Otherwise, faulty time distributions would be generated.

Second, consider (ii) of figure 3. Note that t_0 and t_2 are concurrently enabled. Upon firing, any of t_0 and t_2 has to switch off the clock for t_1. Hence, transitions modelling t_0 and t_2 in N have to synchronize on the clock-places for t_1. This introduces a conflict between t_0 and t_2, i.e. concurrency gets lost. This problem may also be fixed by leaving switching off the clock for t_1 to some update- or tic-transition.

Update- and tic-transitions are not sufficient to treat correctly the net of part (iii) of figure 3. Consider e.g. the state $(\{p_0, p_1\}, 11)$. Here, t_0 may occur, leading to $(\{p_1, p_2\}, \$0)$. But in the approach sketched here, the result would be $(\{p_1, p_2\}, \$1)$, and there is no inconsistency between the marking part and the clock part of this state. Thus a repair-transition cannot help. The problem relies on the fact that t_2 becomes disabled and newly enabled at the same instance of time or equivalently, that a token is removed from p_1 and a token is put on p_1 at the same instance of time.

In this special case the problem could be solved by introducing a private set of clock places w.r.t. t_1 for both t_0 and t_1, and by allowing firing of t_1 only if both its clocks are running. This solution also works in the general case, but leads to a much more involved definition of time-expansion and to a bigger time-expanded net $X(TN)$. Instead, we will restrict the class of nets under consideration by introducing a requirement of (local) time-divergence: If a place p loses its token at time θ, then no token will arrive at p before time $\theta + 1$[1].

Places of Petri nets denote (local) state. Hence, the divergent-time condition demands that if a local state is left it may not be entered again at the same instance of time. The latter property is fulfilled in many practical applications. Consider e.g. a resource with exclusive access: Here a local state (i.e. a place) may serve as a guard for the resource, a token in the place indicating availability of the resource. In this case, the time elapsing between the token leaving the place and a new token entering the place corresponds to the time for one allocation cycle of the resource. If time requirements are at all taken into account it is very reasonable to assume that this needs some non-zero amount of time.

These considerations lead to the following definitions:

Definition 1. *[Divergent-Time Property]*

Let $TN = (P, T, F, \chi, M_0)$ be a time Petri net. TN satisfies the *divergent-time property* (and is called *DT-net*) iff the following holds: If $(t_1 ... t_n)$ is enabled at (M, I) in TN such that $\bigcup_{1 \leq i \leq n} {}^{\bullet}t_i \cap \bigcup_{1 \leq i \leq n} t_i^{\bullet} \neq \emptyset$, then, for some $i \in \{1, ..., n\}$, $t_i = tic$. $\diamond$

Definition 2. *[Time-Expansion]*

Let $TN = (P, T, F, \chi, M_0)$ be a DT-net. The *time-expansion* of TN is given by the P/T-net $X(TN) = (P_X, T_X, F_X, M_{X_0})$ defined as follows:

[1] Note, that the requirement of locally divergent time is slightly stronger than that given in [3], but may be fully released if the 'multiple clock solution' is chosen.

1. For each place $p \in P$, a new (complementary) place p^c is introduced. The set of these places is denoted by P^c.
2. For each transition $t \in T$ new places $p_t^{\$}, p_t^0, ..., p_t^{lft(t)}$ are introduced. The intended meaning of these places is to describe all possible clock positions for the corresponding transition t. The set of the new places is denoted as *Clock*, and the places are referred to as *clock-places*.
3. For a clock function I of TN, let $PL(I) = \{p_t^{I(t)} \mid t \in T\}$. The time-expansion of a state (M, I) of TN is defined by $X(M, I) = (M \cup M^c \cup PL(I))$, where $M^c = \{p^c \mid p \in (P \setminus M)\}$.
4. For each state (M, I) of TN such that a time-event may occur a new transition $tic(M, I)$ is introduced, having $M \cup M^c \cup PL(I)$ as its preset, and $M \cup M^c \cup PL(I')$ as its postset, where, for all $v \in T$,

$$I'(v) = \begin{cases} I(v)+1 & I(v) \neq \$ \wedge v \in Enabled(M) \\ 1 & I(v) = \$ \wedge v \in Enabled(M) \\ \$ & \text{otherwise.} \end{cases}$$

   These transitions are intended to model time-events and, in addition, updating clock vectors. The set of these transitions is denoted as *Tic*, and the transitions are referred to as *tic-transitions*.
5. For each transition t in TN and for $eft(t) \leq i \leq lft(t)$, a new transition $t(i)$ is introduced. The preset of $t(i)$ is given by ${}^\bullet t \cup \{p^c \mid p \in (t^\bullet \setminus {}^\bullet t)\} \cup \{p_t^i\}$ and the postset by $t^\bullet \cup \{p^c \mid p \in ({}^\bullet t \setminus t^\bullet)\} \cup \{p_t^{\$}\}$.
   These transitions are intended to model changing of the marking. Moreover, the clock of the firing transition is switched off. The set of these transitions is denoted as *Fire*, and the transitions are referred to as *fire-transitions*.
6. For each inconsistent state (M, I) of TN a new transition $update(M, I)$ is generated. The preset of the transition $update(M, I)$ is given by $M \cup M^c \cup PL(I)$ and the postset by $M \cup M^c \cup PL(I')$, where, for all $v \in T$,

$$I'(v) = \begin{cases} I(v) & I(v) \neq \$ \wedge v \in Enabled(M) \\ 0 & I(v) = \$ \wedge v \in Enabled(M) \\ \$ & \text{otherwise.} \end{cases}$$

   These transitions are intended to model updating the clocks of newly activated or deactivated transitions. The set of these transitions is denoted as *Update*, and the transitions are referred to as *update-transitions*.

Hence, altogether we have $P_X = P \cup P^c \cup Clock$, $T_X = Tic \cup Fire \cup Update$, F_X as described above, and $M_{X_0} = M_0 \cup M_0^c \cup PL(I_0)$. $\diamond$

Definition 3. Let TN be a DT-net and $X(TN)$ its time-expansion. A marking $\overline{M}$ of $X(TN)$ is called *consistent* iff $\mid \overline{M} \cap \{p, p^c\} \mid \; = 1$ for each $p \in P$ and $\mid \overline{M} \cap \{p_t^{\$}, p_t^0, ... p_t^{lft(t)}\} \mid \; = 1$ and $(p_t^{\$} \in \overline{M} \Leftrightarrow t \notin Enabled(\overline{M} \cap P))$ for all $t \in T$. $\diamond$

Theorem 1. *Let $TN = (P, T, F, \chi, M_0)$ be a DT-net. Then the following hold:*

(i) $(M, I) \rightarrow^{\{t_1,\ldots,t_n\}} (M', I')$ *in TN iff either*
$M \cup M^c \cup PL(I) \rightarrow^{\{t_1(I(t_1)),\ldots,t_n(I(t_n))\}} M' \cup M'^c \cup PL(I')$ *or*
$M \cup M^c \cup PL(I) \rightarrow^{\{t_1(I(t_1)),\ldots,t_n(I(t_n))\}} \overline{M} \cup \overline{M}^c \cup PL(\overline{I})$ *and*
$\overline{M} \cup \overline{M}^c \cup PL(\overline{I}) \rightarrow^{update(\overline{M},\overline{I})} M' \cup M'^c \cup PL(I')$ *in $X(TN)$.*

(ii) The consistent state (M, I) is reachable in TN iff the consistent marking $(M \cup M^c \cup PL(I))$ is reachable in $X(TN)$. In particular, M is a reachable marking of TN iff $M = M' \cap P$ for some reachable marking M' of $X(TN)$.

Figure 4 (i) shows a simple time Petri net TN_2. The time expansion $X(TN_2)$ (see figure 4 (ii)) shows up the strong connection between the 'clock-net' (on the right) and the 'normal-net' (on the left). Some update-transitions are omitted, c.f. *update*$(\{p_0\}, 1\$\$)$ because the state $(M, I) = (\{p_0\}, 1\$\$)$ is not consistent.

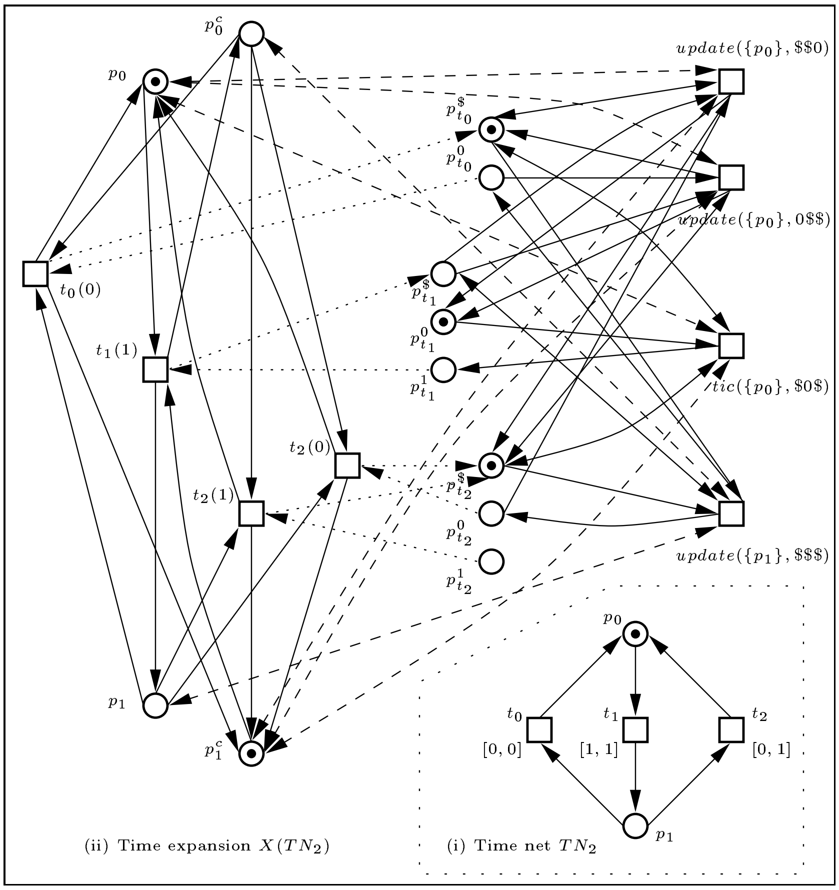

Fig. 4. Time expansion example

4 Partial Order Model Checking of Time Petri Nets

In section 2.3 we have extended the temporal logic $\mathcal{L}$ to time Petri nets. In this section we will show that, for a time Petri net TN, the model checking problem w.r.t. $\mathcal{L}$ can be solved by applying Esparza's model checking algorithm to the time expansion $X(TN)$ of TN.

Let e be a cut off event of a P/T-net N, and e^0 an event such that $|\,[e^0]\,| < |\,[e]\,|$ and $Mark([e]) = Mark([e^0])$. As pointed out in [9], in this case the possible future after $Cut([e])$ is isomorphic to the possible future after $Cut([e^0])$. This property is crucial for the proof of Esparza's theorem.

Figure 1 (iii) shows, that for a time Petri net TN the possible future after some finite configuration C also depends on the timing associated to C. It follows, that for a time Petri net TN that part of the McMillan-unfolding $McM(N)$ of the underlying P/T-net N, which is realizable under the time restrictions, does not contain sufficient information to serve as a basis for model checking (cf. Figure 1 (ii)).

The time-expansion $X(TN)$ of a time Petri net TN, on the other hand, has all necessary information about future behaviour, cf. theorem 1. The connection of TN and $X(TN)$ w.r.t. satisfaction of formulae is given by the following lemma, which states that a temporal logic formula ϕ is time-satisfied by a state of a time Petri net TN iff it is satisfied by the corresponding marking of the time-expansion $X(TN)$ of TN.

Lemma 1. *Let TN be a DT-net, (M, I) a reachable state of TN, and ϕ a formula over TN. Let $\overline{M} = M \cup M^c \cup PL(I)$ be the time-expansion of (M, I). Then*

$$(M, I) \models_t^{TN} \phi \text{ iff } \overline{M} \models'^{X(TN)} \phi.$$

Corollary 1. *Let TN be a DT-net and ϕ a formula over TN. Then*

$$TN \models_t^{TN} \phi \text{ iff } X(TN) \models^{X(TN)} \phi.$$

Hence, by Esparza's theorem, in order to check if TN satisfies a given formula ϕ, it is sufficient to consider the McMillan-unfolding of the time expansion $X(TN)$. On the other hand, $X(TN)$ – and therefore also its McMillan-unfolding – contains also some unnecessary information; e.g., conditions labeled by clock-places or complementary places which do not occur in formulas. If these are removed, also events labeled by update-transitions become meaningless, because then these just reproduce their presets. Events labeled by tic-transitions, on the other hand, will still be relevant because these are representing timed ordering of conditions. According to these considerations, we will simplify $McM(X(TN))$.

Definition 4 (Reduced unfolding).
Let TN be a DT-net and let $\beta = (B, E, F, \pi)$ be any branching process of its time-expansion. The *reduced* branching process $R(\beta)$ of β is constructed by performing the following steps:

1. Remove from β all conditions labeled by clock-places or complementary places and all incidental arcs, resulting in a process β_1.

2. Remove from β_1 all parts which are in conflict with an update-transition resulting in a process β_2.
3. Remove from β_2 all events labeled by update-transitions and all incidental arcs. Each time such an event e is taken out, ${}^\bullet e$ and $e^\bullet$ are fused. This results in a process β_3.
4. Rename the labels of all events: Label $t(i)$ of event e is replaced by t. This results in a process β_4.
5. For a configuration C of β the reduction $R(C)$ is defined by $R(C) = C \cap E_{R(\beta)}$.

$\diamond$

It is easy to see that reduction of a process β of $X(TN)$ is well defined, and that reduction preserves inclusion of configurations i.e.,

$$C \subseteq C' \text{ in } \beta \text{ iff } R(C) \subseteq R(C') \text{ in } R(\beta)$$

for all configurations C, C' in β.

Theorem 2. *Let TN be a DT-net, β a process of $X(TN)$, and ϕ a $\mathcal{L}$-formula over TN. Then it holds that*

$$\beta \models^{X(TN)} \phi \textit{ iff } R(\beta) \models^{X(TN)} \phi.$$

Theorem 2 motivates the following definition:

Definition 5 (McMillan-unfolding). Let TN be a DT-net. The McMillan-unfolding of TN is defined by $McM(TN) = R(McM(X(TN)))$. $\diamond$

The McMillan-unfolding $McM(TN_1)$ of the time Petri net TN_1 is shown in Figure 1 (iii).

Next we will prove that, for a time Petri net TN, $McM(TN)$ satisfies exactly the same formulae as the McMillan-unfolding of $X(TN)$. Let $Sat_{rf}(\phi)_N$ denote the set of configurations of the reduced McMillan-unfolding of $X(TN)$ that satisfy the $\mathcal{L}$-formula ϕ over TN. The following proposition is an easy consequence of theorem 2:

Proposition 2. *Let TN be a time Petri net and ϕ be a $\mathcal{L}$-formula over TN. Then*

$$Sat_{rf}(\phi)_{X(TN)} = \emptyset \textit{ iff } Sat_f(\phi)_{X(TN)} = \emptyset.$$

Finally, putting together the puzzle, we can establish the key result of this paper, which states that a formula ϕ is time-satisfied by a time Petri net TN iff it is satisfied by the McMillan-unfolding of the time-expansion of TN.

Theorem 3. *Let TN be a time Petri net, $X(TN)$ its time expansion, and ϕ a formula over TN. Then*

$$TN \models_t \phi \textit{ iff } Sat_{rf}(\neg\phi)_{X(TN)} = \emptyset.$$

Consequently, to check of an $\mathcal{L}$-formula ϕ against a time Petri net TN one has to compute the McMillan-unfolding $McM(TN)$ and to check ϕ against $McM(TN)$.

Note, that neither the explicit construction of $X(TN)$ nor of $McM(X(TN))$ is needed for the construction of $McM(TN)$. In [11] we present an algorithm, which is a refinement of the algorithm for the finite prefix given in [10]. We prove that with this algorithm, $McM(TN)$ may be computed in $O(n \cdot log_n)$ time, where n denotes the size (number of nodes) of $McM(TN)$. It is shown that the number of events in $McM(TN)$ is bounded by the number of reachable states in TN, i.e. that the size of $McM(TN)$ is at most linear in the size of the reachability graph of TN. Moreover, some examples of time Petri nets are presented whose finite prefix is substantially smaller than the reachability graph.

5 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) [23, 24], or on the timed branching process [3, 15, 23].

In [24], 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 represented by at least one node, this appraoch faces the state explosion problem.

The time state graph construction of [24] is improved in [15] 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 [23]. Here, a region graph construction is used to compute a finite representation of the timed state space, cf. the construction for timed automata in [1].

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 [21], 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 [24], 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.

The approach presented in this paper may also be compared to timed automata (cf. e.g. [1]). 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 insription of a transition $M \to^t M' \in E$ consists of the guard $eft(t) \leq c_1 \leq lft(t)$ and the set of clocks to be reset is given by $\{t' \mid t' \in Enabled(M - {}^\bullet t)\}$. 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.

6 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.*

In [12], the language SDL/R was introduced which extends SDL (Specification and Description Language) by adding real-time requirements. For SDL/R, a denotational semantics was defined in terms of time M-nets, which form a class of compositional timed high level Petri nets. As pointed out in [12], time M-nets may be unfolded into safe time Petri nets. Hence, the result of this paper also suggests a new method for proving functional and qualitative temporal properties of SDL/R-specifications. Currently, the new method is being integrated into the MOBY-tool (cf. [2]) and the PEP-tool (cf. [5]).

Acknowledgements

The authors would like to thank Javier Esparza, Eike Best, Alexander Lavrov, and an anonymous referee for helpful comments on preliminary versions of the paper.

References

1. R. Alur, C. Courcoubetis, and D. Dill: Model Checking for Real-time Systems. *Proc. 5th Symp. LICS* (1990).
2. P. Amthor, H. Fleischhack, and J. Tapken: *MOBY – more than a Tool for the Verification of SDL-Specifications.* Technical Report, University of Oldenburg (1996).
3. T. Aura, J. Lilius: Time Processes for Time Petri Nets. *Proc. ICATPN*, Toulouse (1997).

4. B. Berthomieu, M. Diaz: Modelling and Verification of Time Dependent Systems Using Time Petri Nets. *IEEE Transactions on Software Engineering*, Volume 17/3, 259–273 (1991).
5. E. Best: Partial Order Verification with PEP. Proc. *POMIV'96, Partial Order Methods in Verification*, G. Holzmann, D. Peled, and V. Pratt (eds.), American Mathematical Society (1996).
6. B. Bieber, H. Fleischhack: *Model Checking of Time Petri Nets Based on Partial Order Semantics*. Techn. Rep., CvO University of Oldenburg (1999). URL: http://theoretica.informatik.uni-oldenburg.de/~max/paper/
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, 244–263 (1986).
8. J. Engelfriet: Branching Processes of Petri Nets, *Acta Informatica*, Volume 28, pages 575–591 (1991).
9. J. Esparza: Model Checking Using Net Unfoldings. *Science of Computer Programming*, Volume 23, 151–195, Elsevier (1994).
10. J. Esparza, S. Römer, and W. Vogler: An Improvement of McMillan's Unfolding Algorithm. *Proc. of TACAS'96* (1996).
11. H. Fleischhack: *Efficient computation of processes for time Petri nets*. University of Oldenburg (1999). (in preparation)
12. H. Fleischhack, and J. Tapken: An M-Net Semantics for a Real-Time Extension of μSDL, *Proc. FME'97*, Graz (1997).
13. P. Godefroid: Using partial orders to improve automatic verification methods. in *Proc. Workshop on Computer Aided Verification* (1990).
14. B. Graves: Identification of Specific Processes Contained in McMillan's Finite Unfolding. in *Proceedings of FSTTCS17*, Kharagpur, India (1997).
15. J. Lilius: Efficient state space search for time Petri nets. *MFCS Workshop on Concurrency* (1998).
16. K. MacMillan: Symbolic Model Checking – an Approach to the State Explosion Problem. PhD Thesis, SCS, Carnegie Mellon University (1992).
17. K.L. McMillan: Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. *Proc. CAV '92, Fourth Workshop on Computer-Aided Verification*, Vol. 663 of Lecture Notes in Computer Science, 164–174 (1992).
18. P. Merlin, D. Farber: Recoverability of Communication Protocols – Implication of a Theoretical Study. *IEEE Transactions on Software Communications*, Vol. 24, 1036–1043 (1976).
19. X. Nicollin, J. Sifakis, and S. Yovine: From ATP to Timed Graphs and Hybrid Systems. Lecture Notes in Computer Science, *Real Time: Theory in Practice* (1992).
20. L. Popova: On Time Petri Nets. *Journal of Information Processing and Cybernetics*, Volume 1055 of *Lecture Notes in Computer Science*, Springer (1991).
21. A. Semenov, A. Yakovlev, and A. Koelmans: Time Petri net unfoldings and hardware verification. University of Newcastle (1998) (Draft).
22. P. Starke: Analyse von Petri-Netz-Modellen. Teubner Verlag, Stuttgart (1990) (in German).
23. I. Virbitskaite, E. Pokozy: On Partial Order Verification of Time Petri Nets. University of Novosibirsk (1998) (Draft).
24. T. Yoneda, B.-H. Schlingloff: Efficient Verification of Parallel Real-Time Systems. *Journal of Formal Methods in System Design*, Vol. 11-2, 187–215 (1997).