Timed Petri Nets and BQOs

Parosh Aziz Abdulla and Aletta Nylén

Department of Computer Systems, Uppsala University
P.O. Box 337, SE-751 05 Uppsala, Sweden
{parosh, aletta}@docs.uu.se

Abstract. We consider (unbounded) *Timed Petri Nets (TPNs)* where each token is equipped with a real-valued clock representing the "age" of the token. Each arc in the net is provided with a subinterval of the natural numbers, restricting the ages of the tokens travelling the arc. We apply a methodology developed in [AN00], based on the theory of *better quasi orderings (BQOs)*, to derive an efficient constraint system for automatic verification of safety properties for TPNs. We have implemented a prototype based on our method and applied it for verification of a parametrized version of Fischer's protocol.

1 Introduction

One of the most widely used techniques for automatic verification of programs is that of *model checking* [CES86,QS82]. A major current challenge is to extend the applicability of model checking to the context of infinite-state systems. A program may be infinite-state since it operates on unbounded data structures, e.g. timed automata [ACD90], hybrid automata [Hen95], data-independent systems [JP93,Wol86], relational automata [Čer94], counter machines [Jan97,AČ98], pushdown processes [BS95], lossy channel systems [AJ96], completely specified protocols [Fin94] etc. A program may also be infinite-state since it has an infinite control part, e.g. Petri nets [Esp95,JM95], and parameterized systems [GS92,AJ98b,KMM$^+$97], in which the topology of the system is parameterized by the number of processes inside the system.

Petri nets are one of the most widely used models for analysis and verification of concurrent systems. Furthermore, several classes of *Timed Petri Nets (TPNs)* have been introduced in the literature for studying the behaviours of real-time systems; e.g. [RP85,MF76,BD91,GMMP91] (also see [Bow96] for a survey).

In this paper we consider verifying coverability properties of TPNs. In our model, each token in a TPN has an "age" which is represented by a real number. A marking of the net is therefore a mapping which assigns a bag of real numbers to each place. The bag represents the numbers and ages of the tokens in the corresponding place. Each arc of the net is equipped with an interval defined by two natural numbers. A transition may fire only if its input places have tokens with ages satisfying the intervals of the corresponding arcs. Tokens generated by transitions will have ages in the intervals of the output arcs. Furthermore, we assume a lazy (non-urgent) behaviour of the TPN. This means that transitions may be delayed, even if that implies that some transitions become disabled because their input tokens become too old. Observe that TPNs cannot be modelled within the context of real-time automata [AD90], as the latter operate on a finite number of clocks. In fact TPNs are infinite in two dimensions; they have an unbounded number of tokens and each token has a real-valued clock.

An instance of the coverability problem consists of an initial marking, and a upward closed set of *bad markings*. Intuitively, we do not want the bad markings to occur during the execution of the TPN, and therefore we are interested in showing that no bad marking is reachable from the initial marking. Using standard techniques [VW86,GW93], we can reduce several classes of safety properties for TPNs into the coverability problem.

To solve the coverability problem, we apply an instance of a general algorithm described in [AČJYK96a,AJ98a] for reachability analysis of infinite-state systems. We use a symbolic representation, called *existential zones* for representing (infinite) upward closed sets of markings. We perform a fixpoint iteration, in which we generate existential zones characterizing the set of markings from which a bad marking is reachable within j steps, for increasing values of j.

A main issue when using such an algorithm is to show that the fixpoint iteration always terminates. Applying the method of [AČJYK96a,AJ98a] to existential zones, we can show that the termination of our algorithm is guaranteed if we show that existential zones are *well quasi-ordered*, i.e., for each infinite sequence of zones $Z_0, Z_1, Z_2, \ldots$, there are i and j with $i < j$ where Z_j characterizes a set of markings which is a subset of the set of markings characterized by Z_i. To show the well quasi-ordering of existential zones, we follow the methodology of [AN00], and show that existential zones in fact satisfy a stronger property than well quasi-ordering, namely that they are *better quasi-ordered*. It is worth noting that the well quasi-ordering of existential zones is not possible to show with the framework of [AČJYK96a,AJ98a]. Thus, model checking of TPNs provides a strong evidence that better quasi-orderings are more suitable to use in the context of symbolic model checking than well quasi-orderings.

Based on our algorithm, we have implemented a prototype for automatic verification of safety properties for TPNs. We have used the tool for verification of a parameterized version of Fischer's protocol with encouraging results.

Related work Existential zones are variants of another symbolic representation namely that of *zones*. Zones are used in the design of existing tools for verification of real-time systems, such as KRONOS [Yov97] and UPPAAL [LPY97]. However, zones characterize finite sets of clocks, and therefore cannot be used to analyze TPNs.

In [AJ98b] we consider a model close to TPNs, namely *timed networks*. A timed network consists of an arbitrary number of timed processes, each with a single real-valued clock. However, in [AJ98b] we use *existential regions* for verification of timed networks. Existential regions are related to *regions* in the same manner as existential zones are related to zones. In the same manner as regions are less efficient than zones, existential regions are far less efficient than existential zones and explode even on very small applications. In fact, an (existential) zone is the union of a (often large) number of (existential) regions, and therefore (existential) zones offer a much more compact representation of the state space.

Most earlier work on studying decidability issues for TPNs, e.g. [RP85,BD91,GMMP91,RGdFE99] either report undecidability results or decidability under the assumption that the TPN is bounded. A work closely related to ours is [dFERA00]. The authors consider the coverability problem for a class of TPNs similar to our model. The main difference is that in [dFERA00], it is assumed that the ages of the tokens are natural numbers. Furthermore, it is not evident how efficient the constraint system is in practical applications.

In this paper, we consider only lazy TPNs. In fact it can be shown [JLL77] that very simple classes of TPNs with urgent behaviours can simulate two-counter machines, and hence almost all verification problems are undecidable for them. This is not a problem when checking coverability since the set of transitions of an urgent TPN is a subset of the set of transitions of the corresponding lazy TPN. This means that if a set of markings is not reachable in the lazy TPN, then it is certainly not reachable in the urgent TPN.

Outline In the next section we introduce timed Petri nets. In Section 3 we give an overview of our reachability algorithm. A constraint system which we call existential zones is introduced in Section 4 and in the following section we define an entailment relation on existential zones. In Section 6 we show how *Pre* is computed and in Section 7 we prove that the reachability algorithm terminates. Section 8 introduces existential DDDs, the constraint system used in our experimental work which is presented in Section 9.

2 Timed Petri Nets

We consider *Timed Petri Nets (TPNs)* where each token is equipped with a real-valued clock representing the “age” of the token. The firing conditions of a transition include the usual ones for Petri nets. Furthermore, each arc between a place and a transition is labeled with a subinterval of the natural numbers. When a transition is fired, the tokens removed from the input places of the transition and the tokens added to the output places should have ages lying in the intervals of the corresponding arcs.

We let $\mathcal{N}$, $\mathcal{Z}$, and $\mathcal{R}^{\geq 0}$ denote the sets of natural numbers, integers, and non-negative reals respectively. For a set A, we define the set $Bags(A)$ of *bags* over A to be the set of mappings from A to $\mathcal{N}$. Sometimes we write bags as lists, so e.g. $(2.4, 5.1, 5.1, 2.4, 2.4)$ represents a bag B over $\mathcal{R}^{\geq 0}$ where $B(2.4) = 3$, $B(5.1) = 2$ and $B(x) = 0$ for $x \neq 2.4, 5.1$. We may also write B as $\left(2.4^3, 5.1^2\right)$. For bags B_1 and B_2 over a set A, we say that $B_1 \leq B_2$ if $B_1(a) \leq B_2(a)$ for each $a \in A$. We define $B_1 + B_2$ to be the bag B where $B(a) = B_1(a) + B_2(a)$, and (assuming $B_1 \leq B_2$) we define $B_2 - B_1$ to be the bag B where $B(a) = B_2(a) - B_1(a)$, for each $a \in A$. We use $\emptyset$ to denote the empty bag, i.e., $\emptyset(a) = 0$ for each $a \in A$.

We use a set *Intrv* of *intervals* of the form $[a : b]$, where $a \in \mathcal{N}$ and $b \in \mathcal{N} \cup \{\infty\}$. For $x \in \mathcal{R}^{\geq 0}$, we write $x \in [a : b]$ to denote that $a \leq x \leq b$.

A *Timed Petri Net (TPN)* is a tuple $N = (P, T, In, Out)$ where P is a finite set of *places*, T is a finite set of *transitions* and $In, Out : T \times P \mapsto Bags(Intrv)$. If $In(t,p)(\mathcal{I}) \neq \emptyset$ ($Out(t,p)(\mathcal{I}) \neq \emptyset$), for some interval $\mathcal{I}$, we say that p is an *input (output) place* of t.

A *marking* M of N is a finite bag over $P \times \mathcal{R}^{\geq 0}$. The marking M defines numbers and ages of the tokens in each place in the net. That is, $M(p,x)$ defines the number of tokens with age x in place p. For example, if $M = ((p_1, 2.5), (p_1, 1.3), (p_2, 4.7), (p_2, 4.7))$, then, in the marking M, there are two tokens with ages 2.5 and 1.3 in p_1, and two tokens each with age 4.7 in the place p_2. Abusing notation, we define, for each place p, a bag $M(p)$ over $\mathcal{R}^{\geq 0}$, where $M(p)(x) = M(p,x)$. Notice that untimed Petri nets are a special case in our model where all intervals are of the form $[0 : \infty]$.

We define two types of transition relations on markings. A *timed transition* increases the age of all tokens by the same real number. Formally $M_1 \longrightarrow_T M_2$ if M_1 is of the form $((p_1, x_1), \ldots, (p_n, x_n))$, and there is $\delta \in \mathcal{R}^{\geq 0}$ such that $M_2 = ((p_1, x_1 + \delta), \ldots, (p_n, x_n + \delta))$.

We define the set of *discrete transitions* $\longrightarrow_d$ as $\bigcup_{t \in T} \longrightarrow_t$, where $\longrightarrow_t$ represents the effect of firing the transition t. More precisely, we define $M_1 \longrightarrow_t M_2$ if, for each place p with $In(t,p) = (\mathcal{I}_1, \ldots, \mathcal{I}_m)$ and $Out(t,p) = (\mathcal{J}_1, \ldots, \mathcal{J}_n)$, there are bags $B_1 = (x_1, \ldots, x_m)$ and $B_2 = (y_1, \ldots, y_n)$ over $\mathcal{R}^{\geq 0}$, such that the following holds.

- $B_1 \leq M_1(p)$.
- $x_i \in \mathcal{I}_i$, for $i : 1 \leq i \leq m$.
- $y_i \in \mathcal{J}_i$, for $i : 1 \leq i \leq n$.
- $M_2(p) = M_1(p) - B_1 + B_2$.

Intuitively, a transition t may be fired only if for each incoming arc to the transition, there is a token with the "right" age in the corresponding input place. This token will be removed from the input place when the transition is fired. Furthermore, for each outgoing arc, a token with an age in the interval will be added to the output place. We define the relation $\longrightarrow$ to be $\longrightarrow_T \cup \longrightarrow_d$, and define $\xrightarrow{*}$ to be the reflexive transitive closure of $\longrightarrow$. For markings M_1 and M_2, we say that M_1 is *reachable* from M_2 if $M_1 \xrightarrow{*} M_2$. For a marking M and a set of markings M, we write $M \xrightarrow{*} \mathsf{M}$ to denote that there is a $M' \in \mathsf{M}$ such that $M \xrightarrow{*} M'$.

For set M of markings we let $Pre(\mathsf{M})$ denote the set $\{M;\ \exists M' \in \mathsf{M}.\ M \longrightarrow M'\}$, i.e., $Pre(\mathsf{M})$ is the set of markings from which we can reach a marking in M through the application of a single (timed or discrete) transition.

A set M of markings is said to be *upward closed* if it is the case that $M \in \mathsf{M}$ and $M \leq M'$ imply $M' \in \mathsf{M}$.

Coverability The *coverability problem* is defined as follows.

Instance A TPN N, a marking M_{init} of N, and an upward closed set M_{fin} of markings of N.

Question $M_{init} \xrightarrow{*} \mathsf{M}_{fin}$?

Using standard techniques [VW86,GW93], we can show that checking several classes of safety properties for TPNs can be reduced to the coverability problem.

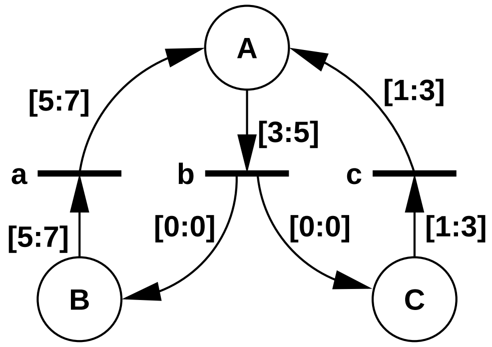

Fig. 1. A small timed Petri net.

Example Figure 1 shows an example of a TPN where $P = \{A, B, C\}$ and $T = \{a, b, c\}$. For instance, $In(a) = ((B, [5:7]))$ and $Out(b) = ((B, [0:0]), (C, [0:0]))$. The initial marking of this net is the marking $M_{init} = ((A, 0.0))$ with only one token with age 0 in place A.

Remark 1 For simplicity of presentation we use only non-strict inequalities. All the results can be generalized in a straightforward manner to include the more general case, where we also allow strict inequalities.

Remark 2 Notice that, in our definition of the operational behaviour of TPNs, we assume a lazy (non-urgent) behaviour of the net. This means that we may choose to "let time pass" instead of firing enabled transitions, even if that makes transitions disabled due to some of the needed tokens becoming "too old". Tokens that are too old to participate in firing transitions are usually called dead tokens. In an urgent TPN, timed transitions that cause dead tokens are not allowed. This means that the set of transitions of an urgent TPN is a subset of the set of transitions of the corresponding lazy TPN. Therefore, if a set of markings is not reachable in the lazy TPN it is not reachable in the urgent TPN either. In other words safety properties that hold for the lazy TPN also hold for the urgent TPN.

3 Overview of the Verification Algorithm

We give an overview of our algorithm for solving the coverability problem. The main ingredients of the algorithm are

- an instance of a symbolic algorithm described in [AČJYK96b] for checking reachability properties of infinite-state systems
- an application of a methodology based on the theory of *better quasi orderings* described in [AN00] for designing efficient data structures in the implementation of the above symbolic algorithm.

We use a symbolic representation of markings called *existential zones* (Section 4), where each zone Z characterizes an upward closed set of markings $[\![Z]\!]$. The coverability algorithm operates on finite sets of zones. Intuitively, a finite set Z of zones represents the union of the interpretations of its members, i.e., $[\![\mathsf{Z}]\!] = \bigcup_{Z \in \mathsf{Z}} [\![Z]\!]$. Given an instance of the coverability problem (Section 2), defined by M_{init} and a zone Z_0 such that $[\![Z_0]\!] = \mathsf{M}_{\mathit{fin}}$, the symbolic algorithm consists of performing a fixpoint iteration, generating a sequence $\mathsf{Z}_0, \mathsf{Z}_1, \mathsf{Z}_2, \ldots$, where each Z_i is a finite set of existential zones. The set Z_0 is defined to be the singleton $\{Z_0\}$. We define Z_{i+1} to be $\mathsf{Z}_i \cup \mathit{Pre}(\mathsf{Z}_i)$, where $[\![\mathit{Pre}(\mathsf{Z}_i)]\!] = \mathit{Pre}([\![\mathsf{Z}_i]\!])$. In other words, $\mathit{Pre}(\mathsf{Z}_i)$ characterizes exactly the markings from which we can reach a marking in $[\![\mathsf{Z}_i]\!]$ through the application of a single step of the transition relation. In Section 6, we show that the set $\mathit{Pre}(\mathsf{Z}_i)$ exists and is computable. Observe that Z_i characterizes the set of markings from which we can reach $\mathsf{M}_{\mathit{fin}}$ in i or fewer steps. We also notice that the elements of the sequence denote larger and larger sets of markings, i.e., $[\![\mathsf{Z}_0]\!] \subseteq [\![\mathsf{Z}_1]\!] \subseteq [\![Z_2]\!] \subseteq \cdots$. This implies that the procedure of generating new elements of the sequence can be terminated when we reach a point j, where $[\![\mathsf{Z}_j]\!] \supseteq [\![\mathsf{Z}_{j+1}]\!]$. In such a case we have reached the fixpoint, and Z_j characterizes the set of all markings from which $\mathsf{M}_{\mathit{fin}}$ is reachable. Consequently, the reachability of $\mathsf{M}_{\mathit{fin}}$ from M_{init} is equivalent to whether $M_{init} \in [\![\mathsf{Z}_j]\!]$. In Section 5 (Lemma 3), we show that the relation $[\![\mathsf{Z}_{j+1}]\!] \subseteq [\![\mathsf{Z}_j]\!]$ is decidable, and in Section 4 (Lemma 1), we show that the relation $M_{init} \in [\![\mathsf{Z}_j]\!]$ is decidable.

One key issue is to show that the symbolic algorithm always terminates. In [AČJYK96b] we show that, for any constraint system, the termination of the algorithm is guaranteed if the constraint system satisfies a certain property, namely that the constraint system is *well quasi-ordered.* In Section 7 we show well quasi-ordering of existential zones. We do that by applying the methodology developed in [AN00]. More precisely, we show that existential zones satisfy a stronger property than well quasi-ordering; namely that they are *better quasi-ordered.* Better quasi-ordering of existential zones follows from the fact that they can be derived starting from finite domains and then then repeatedly applying the operations of building sets, bags, strings, and taking unions.

4 Existential Zones

In this section we introduce a constraint system called *existential zones.* Intuitively, an existential zone characterizes a upward closed set of markings. An existential zone Z represents minimal conditions on markings. More precisely, Z specifies a minimum number of tokens which should be in the marking, and then imposes certain conditions on these tokens. The conditions are formulated as specifications of the places in which the tokens should reside and restrictions on their ages. The age restrictions are stated as bounds on values of clocks, and bounds on differences between values of pairs of clocks. A marking M which satisfies Z should have at least the number of tokens specified by Z. Furthermore, the places and ages of these tokens should satisfy the conditions imposed by Z. In such a case, M may have any number of additional tokens (whose places and ages are irrelevant for the satisfiability of the zone by the marking).

For a natural number n, we let n^* denote the set $\{0, 1, 2, \ldots, n\}$, and let n^+ denote the set $\{1, 2, \ldots, n\}$. We assume a TPN (P, T, In, Out).

An *existential zone* Z is a triple $\left(m, \bar{P}, D\right)$, where m is an natural number, $\bar{P}$ (called a *placing*) is a mapping $\bar{P} : m^+ \rightarrow P$, and D (called a *difference bound matrix*) is a mapping $D : m^* \times m^* \rightarrow \mathcal{N} \cup \{\infty\}$. Intuitively, m defines the minimum number of tokens in the marking, $\bar{P}$ maps each token to a place, and D defines restrictions on the ages of the tokens in forms of bounds on clock values and on differences between clock values. Difference bound matrices, or DBMs, are widely used in verification of timed automata, e.g., [Dil89,LPY95].

Consider the example from Section 2. Assume that we are interested in checking the coverability of markings with at least two tokens, one in place B and one in place C, such that the ages of the tokens are at most 8 and the token in B is at most 4 time units older than the one in C. The markings satisfying these constraints can be described by the existential zone $Z = \left(2, \bar{P}, D\right)$ where $\bar{P}(1) = B$, $\bar{P}(2) = C$ and D is described by the following table where eg.

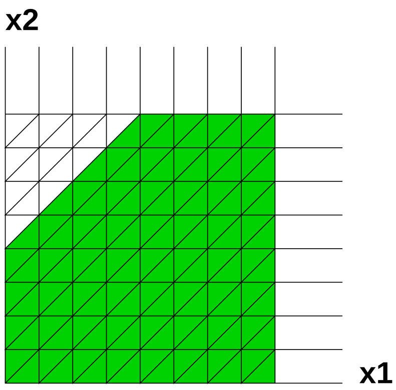

Fig. 2. Example of restrictions on ages of tokens.

$D(0, i) = 0$ and $D(2, 1) = 4$.

$$D = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & 0 & 0 \\ 1 & 8 & - & 8 \\ 2 & 8 & 4 & - \end{array}$$

Figure 2 shows an illustration of the age restrictions of Z.

Consider a marking $M = ((p_1, x_1), \ldots, (p_n, x_n))$ and an injection $h : m^+ \to n^+$ (called a *witness*). We say that M *satisfies* Z *with respect to* h, written $M, h \models Z$, if the following conditions are satisfied.

- $\bar{P}(i) = p_{h(i)}$, for each $i : 1 \leq i \leq m$.
- $x_{h(j)} - x_{h(i)} \leq D(j, i)$, for each $i, j \in m^+$ with $i \neq j$.
- $x_{h(i)} \leq D(i, 0)$ and $-D(0, i) \leq x_{h(i)}$, for each $i \in m^+$.

We say that M *satisfies* Z, written $M \models Z$, if $M, h \models Z$ for some h. Notice that if M satisfies Z then $m \leq n$ (since h is an injection), i.e., M has at least the number of tokens required by Z, and furthermore, the places and ages of the tokens satisfy the requirements of Z. We define $[\![Z]\!] = \{M;\ M \models Z\}$. Notice that the value of $D(i, i)$ is irrelevant for the satisfiability of Z.

Membership From the above definitions the following lemma is straightforward.

Lemma 1. *For an existential zone Z and a marking M, it is decidable whether $M \models Z$.*

Upward Closedness We observe that Z defines a number of minimal requirements on M, in the sense that M should contain at least m tokens whose places and ages are constrained by the functions $\bar{P}$ and D respectively. This means the set $[\![Z]\!]$ is upward closed since $M \models Z$ and $M \leq M'$ implies $M' \models Z$.

Normal and Consistent Existential Zones An existential zone $Z = \left(m, \bar{P}, D\right)$ is said to be *normal* if for each $i, j, k \in m^*$, we have $D(j,i) \leq D(j,k) + D(k,i)$. It is easy to show the following.

Lemma 2. *For each existential zone Z there is a unique (up to renaming of the index set) normal existential zone, written $\widetilde{Z}$, such that $[\![\widetilde{Z}]\!] = [\![Z]\!]$.*

This means that we can assume without loss of generality that all existential zones we work with are normal.

An existential zone Z is said to be *consistent* if $[\![Z]\!] \neq \emptyset$.

5 Entailment

Given zones Z_1 and Z_2, we say that Z_1 is *entailed* by Z_2, written $Z_1 \preceq Z_2$, if $[\![Z_2]\!] \subseteq [\![Z_1]\!]$.

We reduce checking entailment between existential zones into validity of formulas in a logic which we here call *Difference Bound Logic (DBL)*. The atomic formulas are either of the form $v \leq c$ or of the form $v - u \leq c$, where v and u are variables interpreted over $\mathcal{R}^{\geq 0}$ and $c \in \mathcal{N}$. Furthermore the set of formulas is closed under the propositional connectives. It is easy to see that validity of DBL-formulas is NP-complete.

Suppose that we are given two existential zones $Z_1 = \left(m_1, \bar{P}_1, D_1\right)$ and $Z_2 = \left(m_2, \bar{P}_2, D_2\right)$. We translate the relation $Z_1 \preceq Z_2$ into validity of a DBL-formula F as follows. We define the set of free variables in F to be $\{v_i;\ i \in {m_2}^+\}$. Let H be the set of injections from ${m_1}^+$ to ${m_2}^+$ such that $h \in H$ if and only if $\bar{P}_1(i) = \bar{P}_2(h(i))$ for each $i \in {m_1}^+$. We define $F = \left(F_1 \implies \left(\bigvee_{h \in H} F_2\right)\right)$, where $F_1 = F_{11} \wedge F_{12} \wedge F_{13}$, and $F_2 = F_{21} \wedge F_{22} \wedge F_{23}$, and

- $F_{11} = \bigwedge_{i,j \in {m_2}^+, j \neq i} \left(v_j - v_i \leq D_2(j,i)\right)$.
- $F_{12} = \bigwedge_{i \in {m_2}^+} \left(v_i \leq D_2(i,0)\right)$.
- $F_{13} = \bigwedge_{i \in {m_2}^+} \left(-D_2(0,i) \leq v_i\right)$.
- $F_{21} = \bigwedge_{i,j \in {m_1}^+, j \neq i} \left(v_{h(j)} - v_{h(i)} \leq D_1(h(j), h(i))\right)$.
- $F_{22} = \bigwedge_{i \in {m_1}^+} \left(v_{h(i)} \leq D_1(h(i), 0)\right)$.
- $F_{23} = \bigwedge_{i \in {m_1}^+} \left(-D_1(0, h(i)) \leq v_{h(i)}\right)$.

This gives the following.

Lemma 3. *The entailment relation is decidable for existential zones.*

Notice that in contrast to zones for which entailment can be checked in polynomial time, the entailment relation for existential zones can be checked only in nondeterministic polynomial time (as we have to consider exponentially many witnesses). This is the price we pay for working with an unbounded number of clocks. On the other hand, when using zones, the size of the problem grows exponentially with the number of clocks inside the system.

6 Computing Predecessors

We define a function *Pre* such that for a zone Z, the value of $Pre(Z)$ is a finite set $\{Z_1, \ldots, Z_m\}$ of zones. The set $Pre(Z)$ characterizes the set of markings from which we can reach a marking satisfying Z through the performance of a single discrete or timed transition. In other words $Pre[\![Z]\!] = [\![Z_1]\!] \cup \cdots \cup [\![Z_m]\!]$. We define $Pre = Pre_D \cup Pre_\delta$, where Pre_D corresponds to firing transitions backwards and Pre_δ corresponds to running time backwards.

We define $Pre_D = \cup_{t \in T} Pre_t$, where Pre_t characterizes the effect of running the transition t backwards. To define Pre_t, we need the following operations on zones. In the rest of the section we assume a normal existential zone $Z = \left(m, \bar{P}, D\right)$, and a timed Petri net $N = (P, T, In, Out)$. From Lemma 2 we know that assuming Z to be normal does not affect the generality of our results.

For an interval $\mathcal{I} = [a : b]$, and $i \in m^+$, we define the *conjunction* $Z \otimes (\mathcal{I}, i)$ of Z with $\mathcal{I}$ at i to be the existential zone $Z' = \left(m, \bar{P}, D'\right)$, where

- $D'(i, 0) = \min(b, D(i, 0))$.
- $D'(0, i) = \min(-a, D(0, i))$.
- $D'(k, j) = D(k, j)$, for each $j, k \in m^+$ with $k \neq j$, $(k, j) \neq (i, 0)$, and $(k, j) \neq (0, i)$.

Intuitively, the operation adds an additional constraint on the age of token i, namely that its age should be in the interval $\mathcal{I}$. For example, for a zone

$$Z = \left(2, \bar{P}, \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & 0 & 0 \\ 1 & 8 & - & 8 \\ 2 & 8 & 4 & - \end{array}\right)$$

the conjunction $Z \otimes ([1 : 6], 1)$ is the zone

$$\left(2, \bar{P}, \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & -1 & 0 \\ 1 & 6 & - & 8 \\ 2 & 8 & 4 & - \end{array}\right)$$

while the conjunction $Z \otimes ([0:10],1) = Z$

For a place p and an interval $\mathcal{I} = [a:b]$, we define the *addition* $Z \oplus (p,\mathcal{I})$ of $(p,\mathcal{I})$ to Z to be the existential zone $Z' = \left(m+1, \bar{P}', D'\right)$, and

- $D'(m+1,0) = b$, and $D'(0,m+1) = -a$.
- $D'(m+1,j) = \infty$, and $D'(j,m+1) = \infty$, for each $j \in m^+$.
- $\bar{P}'(m+1) = p$.
- $D'(k,j) = D(k,j)$, for each $j,k \in m^*$, and $\bar{P}'(j) = \bar{P}(j)$, for each $j \in m^+$.

Intuitively, the new existential zone Z' requires one additional token to be present in place p such that the age of the token is in the interval $\mathcal{I}$. For example, for a zone

$$Z = \left(2, \begin{array}{c}\bar{P}(1) = B \\ \bar{P}(2) = C\end{array}, \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & 0 & 0 \\ 1 & 8 & - & 8 \\ 2 & 8 & 4 & - \end{array}\right)$$

the addition $Z \oplus (A,[1:2])$ is the zone

$$\left(3, \begin{array}{c}\bar{P}(1) = B \\ \bar{P}(2) = C \\ \bar{P}(3) = A\end{array}, \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & - & 0 & 0 & -1 \\ 1 & 8 & - & 8 & \infty \\ 2 & 8 & 4 & - & \infty \\ 3 & 2 & \infty & \infty & - \end{array}\right)$$

For $i \in m^+$, we define the *abstraction* $Z \backslash i$ of i in Z to be the zone $Z' = \left(m-1, \bar{P}', D'\right)$, where

- $D'(j,k) = D(j,k)$, for each $j,k \in (i-1)^*$.
- $D'(j,k) = D(j,k+1)$ and $D'(k,j) = D(k+1,j)$, for each $j \in (i-1)^*$ and $k \in \{i,\ldots,m-1\}$.
- $D'(j,k) = D(j+1,k+1)$, for each $j,k \in \{i,\ldots,m-1\}$.
- $\bar{P}'(j) = \bar{P}(j)$, for each $j \in (i-1)^*$, and $\bar{P}'(j) = \bar{P}(j+1)$, for $j \in \{i,\ldots,m-1\}$.

Intuitively, the operation removes all constraints related to token i from Z, so the number of required tokens is reduced by 1 and the restrictions related to the age and place of the token disappear. For example, for a zone

$$Z = \left(3, \begin{array}{c}\bar{P}(1) = B \\ \bar{P}(2) = C \\ \bar{P}(3) = A\end{array}, \begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & - & 0 & 0 & -1 \\ 1 & 8 & - & 6 & 7 \\ 2 & 8 & 4 & - & 7 \\ 3 & 2 & 2 & 2 & - \end{array}\right)$$

the abstraction $Z \backslash 2$ is the zone

$$\left(2, \begin{array}{c}\bar{P}(1) = B \\ \bar{P}(2) = A\end{array}, \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & - & 0 & -1 \\ 1 & 8 & - & 7 \\ 2 & 2 & 2 & - \end{array}\right)$$

Notice that the existential zones we obtain as a result of performing the three operations above need not be normal.

Now, we are ready to define *Pre*.

Lemma 4. *Consider a TPN* $N = (P, T, In, Out)$*, a transition* $t \in T$*, and an existential zone* $Z = \left(m, \bar{P}, D\right)$*. Let* $In(t) = ((p_1, \mathcal{I}_1), \ldots, (p_k, \mathcal{I}_k))$*, and* $Out(t) = ((q_1, \mathcal{J}_1), \ldots, (q_\ell, \mathcal{J}_\ell))$*. Then* $Pre_t(Z)$ *is the smallest set containing each existential zone* Z' *such that there is a partial injection* $h : m^+ \longrightarrow \ell^+$ *with a domain* $\{i_1, \ldots, i_n\}$*, and an existential zone* Z_1 *satisfying the following conditions.*

- $\bar{P}(i_j) = q_{h(i_j)}$*, for each* $j \in n^+$
- $Z \otimes \left(\mathcal{J}_{h(i_1)}, i_1\right) \otimes \cdots \otimes \left(\mathcal{J}_{h(i_n)}, i_n\right)$ *is consistent.*
- $Z_1 = Z \backslash i_1 \backslash \cdots \backslash i_n$.
- $Z' = Z_1 \oplus (p_1, \mathcal{I}_1) \oplus \cdots \oplus (p_k, \mathcal{I}_k)$.

Lemma 5. *For an existential zone* $Z = \left(m, \bar{P}, D\right)$*, the set* $Pre_\delta(Z)$ *is the existential zone* $Z' = \left(m, \bar{P}, D'\right)$*,where* $D'(0, i) = 0$ *and* $D'(j, i) = D(j, i)$ *if* $j \neq 0$*, for each* $i, j \in m^*$*, with* $i \neq j$.

From Lemma 4 and Lemma 5 we get the following.

Lemma 6. *For an existential zone* Z*, the set* $Pre(Z)$ *is computable.*

7 Termination

In this section we show some results from the theories of well quasi-orderings and better quasi-orderings and explain their relation to termination of the reachability algorithm presented in Section 3.

A *quasi-ordering* or a *qo* for short, is a pair $(A, \preceq)$ where $\preceq$ is a reflexive and transitive (binary) relation on a set A. We use $a_1 \equiv a_2$ to denote that $a_1 \preceq a_2$ and $a_2 \preceq a_1$. An infinite sequence $a_1, a_2, a_3, \ldots$ of elements of A is called a *bad sequence* iff $\forall i, j : i < j \Rightarrow a_i \not\preceq a_j$. A qo $(A, \preceq)$ is a *well quasi-ordering* or a *wqo* for short, if there is no bad sequence of elements of A. Given a qo $(A, \preceq)$, we define a qo $(A^*, \preceq^*)$ on the set A^* of finite strings over A such that $x_1 \bullet \cdots \bullet x_m \preceq^* y_1 \bullet \cdots \bullet y_n$ if and only if there is a strictly monotone injection $h : \{1, \ldots, m\} \to \{1, \ldots, n\}$ where $x_i \preceq y_{h(i)}$ for $i : 1 \leq i \leq m$. A qo $\left(A^B, \preceq^B\right)$ on the set A^B of bags over A can be defined in a similar manner. We define the relation $\sqsubseteq$ on the set $\mathcal{P}(A)$ of subsets of A, so that $A_1 \sqsubseteq A_2$ if and only if $\forall b \in A_2 : \exists a \in A_1 : a \preceq b$.

In [AČJYK96a,AJ98a] we showed that the reachability algorithm is guaranteed to terminate if the constraint system is *well quasi-ordered (wqo)*. To prove well quasi-ordering of existential zones we apply a methodology presented in [AN00]. We use a tool which is more powerful than wqo, namely that of *better quasi-ordering (bqo)*. In the following theorem we state some properties of bqos.

Theorem 1.

1. *Each bqo is wqo.*
2. *If A is finite, then $(A, =)$ is bqo.*
3. *If $(A, \preceq)$ is bqo, then $(A^*, \preceq^*)$ is bqo.*
4. *If $(A, \preceq)$ is bqo, then $(A^B, \preceq^B)$ is bqo.*
5. *If $(A, \preceq)$ is bqo, then $(\mathcal{P}(A), \sqsubseteq)$ is bqo.*

A direct consequence of the last property is that bqo is closed under the operation of taking unions. Since bqo is a stronger relation than wqo it is sufficient to prove bqo of zones under entailment, to prove termination of the reachability algorithm.

In order to prove that existential zones are bqo we recall a constraint system related to existential zones, namely that of *existential regions* introduced in [AJ98b]. An existential region is a list of bags $(B_0, B_1, \ldots, B_n, B_{n+1})$ where $n \geq 0$ and B_i is a bag over $P \times \mathcal{N}$. In a similar manner to existential zones, an existential region R defines a set of conditions which should be satisfied by a configuration γ in order for γ to satisfy R. Intuitively B_0 represents tokens with ages which have fractional parts equal to 0. The bags $B_1, \ldots, B_n$ represent tokens whose ages have increasing fractional parts where ages of tokens belonging to the same bag have the same fractional part and ages of tokens belonging to B_i have a fractional part that is strictly less than the fractional part of the ages of those in B_{i+1}. Finally the bag B_{n+1} represents tokens with ages greater than the maximum natural number occuring in the enabling conditions of a given TPN (regardless of their fractional parts).

Lemma 7. *Existential zones are bqo (and hence wqo).*

1. *Existential regions are built starting from finite domains, and repeatedly building finite strings, bags, and sets. From the properties mentioned above, it follows that existential regions are bqo.*
2. *For each existential zone Z, there is a finite set Regions of existential regions such that $Z \equiv \bigcup Regions$. Since bqo is closed under union, it follows that existential zones are bqo.*

8 Existential CDDs and DDDs

CDDs [LPWY99] and DDDs [MLAH99] are constraint systems invented recently to give representations of real-time systems which are more compact than zones. In a similar manner to existential zones, we modify the definitions of CDDs

(DDDs) into *existential CDDs (DDDs)*, in order to make them suitable for verifying systems with an unbounded number of clocks. Below we give the definition of existential DDDs. The definition of existential CDDs can be stated in a similar manner.

An *existential DDD* is a tuple $Y = \left(m, \bar{P}, \mathtt{V}, \mathtt{E}\right)$, where m and $\bar{P}$ are defined as for existential zones (Section 4), and $(\mathtt{V}, \mathtt{E})$ is a finite directed acyclic graph where $\mathtt{V}$ is the set of vertices and $\mathtt{E}$ is the set of edges. We assume that $\mathtt{V}$ contains two special elements $\mathtt{v}^0$ and $\mathtt{v}^1$. The outdegrees of $\mathtt{v}^0$ and $\mathtt{v}^1$ are zero, while the outdegrees of the rest of vertices are two. Each vertex $\mathtt{v} \in \mathtt{V} - \left\{\mathtt{v}^0, \mathtt{v}^1\right\}$ has the following attributes: $pos(\mathtt{v}), neg(\mathtt{v}) \in m^*$, $op(\mathtt{v}) \in \{<, \leq\}$, $const(\mathtt{v}) \in \mathcal{Z}$, and $high(\mathtt{v}), low(\mathtt{v}) \in \mathtt{V}$. The set $\mathtt{E}$ contains the edges $(\mathtt{v}, low(\mathtt{v}))$ and $(\mathtt{v}, high(\mathtt{v}))$, where $\mathtt{v} \in \mathtt{V} - \left\{\mathtt{v}^0, \mathtt{v}^1\right\}$. In a similar manner to BDDs, the internal nodes of Y correspond to the if-then-else operator $\phi \rightarrow \phi_1, \phi_2$, defined as $(\phi \wedge \phi_1) \vee (\neg\phi \wedge \phi_2)$. Intuitively, the attributes of the node represent the DBL-formula $\phi = x_{pos(\mathtt{v})} - x_{neg(\mathtt{v})}\ op(\mathtt{v})\ const(\mathtt{v})$, and $high(\mathtt{v})$ and $low(\mathtt{v})$ are children of $\mathtt{v}$ corresponding to ϕ_1 and ϕ_2 respectively. The special vertices $\mathtt{v}^0$ and $\mathtt{v}^1$ correspond to *false* and *true*.

Consider an existential DDD $Y = \left(m, \bar{P}, \mathtt{V}, \mathtt{E}\right)$, a vertex $\mathtt{v} \in \mathtt{V}$, a marking $M = ((p_1, c_1), \ldots, (p_k, c_k))$, and an injection $h : m^+ \rightarrow k^+$. We say that M *satisfies* Y *at* $\mathtt{v}$ *with respect to* h, written $M, h \models (Y, \mathtt{v})$, if $\bar{P}(i) = p_{h(i)}$, for each $i \in m^+$, and either

- $\mathtt{v} = \mathtt{v}^1$; or
- $\left(\begin{pmatrix} x_{h(pos(\mathtt{v}))} \\ - \\ x_{h(neg(\mathtt{v}))} \end{pmatrix} \sim const(\mathtt{v})\right) \rightarrow \begin{pmatrix} M, h \models (Y, high(\mathtt{v})) \\ , \\ M, h \models (Y, low(\mathtt{v})) \end{pmatrix}$,
  where $\sim = op(\mathtt{v})$.

In a similar manner to existential zones, we can modify the operations defined in [MLAH99] to compute predecessors of existential DDDs with respect to transitions of a TPN. To check entailment we must, as we did for existential zones, take into consideration all variable permutations.

For each existential DDD Y there is a finite set Z of existential zones such that $[\![Y]\!] = [\![\mathsf{Z}]\!]$. Intuitively this means that an existential DDD can replace several existential zones, and hence existential DDDs give a more compact (efficient) representation of sets of states. Note that each existential DDD is a union of existential zones. This together with Lemma 7 and Theorem 1 (Property 5) gives us the following result.

Lemma 8. *Existential DDDs are bqo (and hence also wqo).*

9 Experimental Results

We have implemented a prototype to perform reachability analysis for TPNs. The constraints are represented by existential DDDs. The implementation is based on a DDD package developed at Technical University of Denmark [ML98]. We used the tool to verify a parameterized version of Fischer's protocol.

The purpose of the protocol is to guarantee mutual exclusion in a concurrent system consisting of an arbitrary number of processes. The example was suggested by Schneider et al. [SBK92]. The protocol analysed here is in fact a weakened version of Fischer's protocol but since the set of reachable states of the weakened version is a superset of the reachable states of the original protocol, the results of our analysis are still valid.

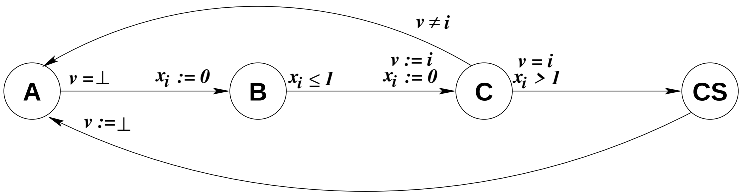

Fig. 3. Fischer's Protocol for Mutual Exclusion

The protocol consists of each process running the code which is graphically described in Figure 3. Each process i has a local clock, x_i, and a control state, which assumes values in the set $\{A, B, C, CS\}$ where A is the initial state and CS is the critical section. The processes read from and write to a shared variable v, whose value is either $\perp$ or the index of one of the processes.

All processes start in state A. If the value of the shared variable is $\perp$, a process wishing to enter the critical section can proceed to state B and reset its local clock. From state B, the process can proceed to state C within one time unit or get stuck in B forever. When making the transition from B to C, the process resets its local clock and sets the value of the shared variable to its own index. The process now has to wait in state C for more than one time unit, a period of time which is strictly greater than the one used in the timeout of state B. If the value of the shared variable is still the index of the process, the process may enter the critical section, otherwise it may return to state A and start over again. When exiting the critical section, the process resets the shared variable to $\perp$.

We will now make a model of the protocol in our TPN formalism. The processes running the protocol are modeled by tokens in the places A, B, C, CS, $A^\dagger$, $B^\dagger$, $C^\dagger$ and $CS^\dagger$. The places marked with $\dagger$ represent that the value of the shared variable is the index of the process modeled by the token in that place. We use a place *udf* to represent that the value of the shared variable is $\perp$. A straightforward translation of the description in Figure 3 yields the Petri net model in Figure 4. q is used to denote an arbitrary process state.

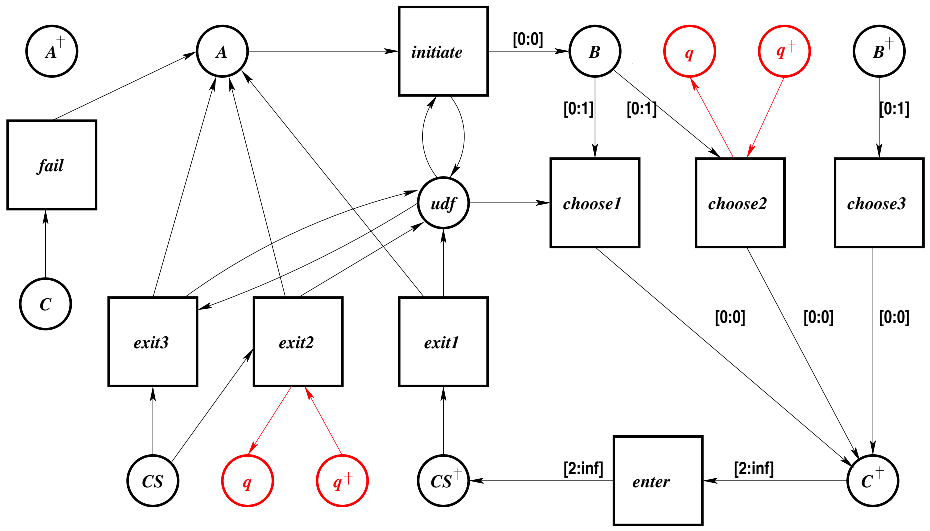

Fig. 4. TPN model of Fischer's Protocol for Mutual Exclusion

In order to prove mutual exclusion we examine the reachability of the existential zones stating that at least two processes are in the critical section, i.e., the following zones:

- $Z_1 = \big(2, \bar{P}_1, D\big)$ where $\bar{P}_1(1) = \bar{P}_1(2) = \mathit{CS}$
- $Z_2 = \big(2, \bar{P}_2, D\big)$ where $\bar{P}_2(1) = \mathit{CS}$ and $\bar{P}_2(2) = \mathit{CS}^{\dagger}$
- $Z_3 = \big(2, \bar{P}_3, D\big)$ where $\bar{P}_3(1) = \bar{P}_3(2) = \mathit{CS}^{\dagger}$

For all three zones $D(0,i) = 0$, $D(i,j) = \infty$ for $i \neq j$.

The reachable state space, represented by 45 existential DDDs, takes 3.5 seconds to compute on a Sun Ultra 60 with 512 MB memory and a 360 MHz UltraSPARC-II processor. In the process, pre was computed for 51 existential DDDs.

Acknowledgments

We are grateful to Alberto Marcone and Petr Jančar for discussions on the theory of better quasi-ordering. We would also like to thank Marc Boyer for making us aware of Ruiz, Escrig and Alonsos work on timed arc Petri nets, which is closely related to our work. Special thanks to Jesper Møller and Henrik Reif Andersen for letting us use their DDD implementation and for helping us solve the problems that arose in the process. Thanks also to the anonymous referees for many valuable comments.

References

[AČ98] Parosh Aziz Abdulla and Karlis Čerāns. Simulation is decidable for one-counter nets. In *Proc. CONCUR '98, 9^{th} Int. Conf. on Concurrency Theory*, volume 1466 of *Lecture Notes in Computer Science*, pages 253–268, 1998.

[ACD90] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for real-time systems. In *Proc. 5^{th} IEEE Int. Symp. on Logic in Computer Science*, pages 414–425, Philadelphia, 1990.

[AČJYK96a] Parosh Aziz Abdulla, Karlis Čerāns, Bengt Jonsson, and Tsay Yih-Kuen. General decidability theorems for infinite-state systems. In *Proc. 11^{th} IEEE Int. Symp. on Logic in Computer Science*, pages 313–321, 1996.

[AČJYK96b] Parosh Aziz Abdulla, Karlis Čerāns, Bengt Jonsson, and Tsay Yih-Kuen. General decidability theorems for infinite-state systems. In *Proc. 11^{th} IEEE Int. Symp. on Logic in Computer Science*, pages 313–321, 1996. To appear in the journal of Information and Computation.

[AD90] R. Alur and D. Dill. Automata for modeeling real-time systems. In *Proc. ICALP '90*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335, 1990.

[AJ96] Parosh Aziz Abdulla and Bengt Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.

[AJ98a] Parosh Aziz Abdulla and Bengt Jonsson. Ensuring completeness of symbolic verification methods for infinite-state systems, 1998. To appear in the journal of Theoretical Computer Science.

[AJ98b] Parosh Aziz Abdulla and Bengt Jonsson. Verifying networks of timed processes. In Bernhard Steffen, editor, *Proc. TACAS '98, 4^{th} Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1384 of *Lecture Notes in Computer Science*, pages 298–312, 1998.

[AN00] Parosh Aziz Abdulla and Aletta Nylén. Better is better than well: On efficient verification of infinite-state systems. In *Proc. 15^{th} IEEE Int. Symp. on Logic in Computer Science*, pages 132–140, 2000.

[BD91] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE Trans. on Software Engineering*, 17(3):259–273, 1991.

[Bow96] F. D. J. Bowden. Modelling time in Petri nets. In *Proc. Second Australian-Japan Workshop on Stochastic Models*, 1996.

[BS95] O. Burkart and B. Steffen. Composition, decomposition, and model checking of pushdown processes. *Nordic Journal of Computing*, 2(2):89–125, 1995.

[Čer94] K. Čerāns. Deciding properties of integral relational automata. In Abiteboul and Shamir, editors, *Proc. ICALP '94*, volume 820 of *Lecture Notes in Computer Science*, pages 35–46. Springer Verlag, 1994.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specification. *ACM Trans. on Programming Languages and Systems*, 8(2):244–263, April 1986.

[dFERA00] D. de Frutos Escrig, V. Valero Ruiz, and O. Marroquín Alonso. Decidability of properties of timed-arc Petri nets. In *ICATPN 2000*, number 1825, pages 187–206, 2000.

[Dil89] D.L. Dill. Timing assumptions and verification of finite-state concurrent systems. In J. Sifakis, editor, *Automatic Verification Methods for Finite-State Systems*, volume 407 of *Lecture Notes in Computer Science*. Springer Verlag, 1989.

[Esp95] J. Esparza. Petri nets, commutative context-free grammers, and basic parallel processes. In *Proc. Fundementals of Computation Theory*, volume 965 of *Lecture Notes in Computer Science*, pages 221–232, 1995.

[Fin94] A. Finkel. Decidability of the termination problem for completely specified protocols. *Distributed Computing*, 7(3), 1994.

[GMMP91] C. Ghezzi, D. Mandrioli, S. Morasca, and M. Pezzè. A unified high-level Petri net formalism for time-critical systems. *IEEE Trans. on Software Engineering*, 17(2):160–172, 1991.

[GS92] S. M. German and A. P. Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, 1992.

[GW93] P. Godefroid and P. Wolper. Using partial orders for the efficient verification of deadlock freedom and safety properties. *Formal Methods in System Design*, 2(2):149–164, 1993.

[Hen95] T.A. Henzinger. Hybrid automata with finite bisimulations. In *Proc. ICALP '95*, 1995.

[Jan97] P. Jančar. Bisimulation equivalence is decidable for one-counter processes. In *Proc. ICALP '97*, pages 549–559, 1997.

[JLL77] N. D. Jones, L. H. Landweber, and Y. E. Lyen. Complexity of some problems in Petri nets. *Theoretical Computer Science*, (4):277–299, 1977.

[JM95] P. Jančar and F. Moller. Checking regular properties of Petri nets. In *Proc. CONCUR '95*, 6^{th} *Int. Conf. on Concurrency Theory*, volume 962 of *Lecture Notes in Computer Science*, pages 348–362. Springer Verlag, 1995.

[JP93] B. Jonsson and J. Parrow. Deciding bisimulation equivalences for a class of non-finite-state programs. *Information and Computation*, 107(2):272–302, Dec. 1993.

[KMM$^+$97] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. In O. Grumberg, editor, *Proc.* 9^{th} *Int. Conf. on Computer Aided Verification*, volume 1254, pages 424–435, Haifa, Israel, 1997. Springer Verlag.

[LPWY99] K. G. Larsen, J. Pearson, C. Weise, and W. Yi. Efficient timed reachability analysis using clock difference diagrams. In *Proc.* 11^{th} *Int. Conf. on Computer Aided Verification*, 1999.

[LPY95] Kim G. Larsen, Paul Pettersson, and Wang Yi. Model-checking for real-time systems. In Horst Reichel, editor, *Proceedings of 10th International Fundamentals of Computation Theory*, number 965 in LNCS, pages 62–88, Dresden, Germany, August 1995.

[LPY97] K.G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *Software Tools for Technology Transfer*, 1(1-2), 1997.

[MF76] P. Merlin and D.J. Farber. Recoverability of communication protocols - implications of a theoretical study. *IEEE Trans. on Computers*, COM-24:1036–1043, Sept. 1976.

[ML98] Jesper Møller and Jakob Lichtenberg. Difference decision diagrams. Master's thesis, Department of Information Technology, Technical University of Denmark, Building 344, DK-2800 Lyngby, Denmark, August 1998.

[MLAH99] Jesper Møller, Jakob Lichtenberg, Henrik R. Andersen, and Henrik Hulgaard. Difference decision diagrams. Technical Report IT-TR-1999-023, Department of Information Technology, Technical University of Denmark, February 1999.

[QS82] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in cesar. In *5th International Symposium on Programming, Turin*, volume 137 of *Lecture Notes in Computer Science*, pages 337–352. Springer Verlag, 1982.

[RGdFE99] V. Valero Ruiz, F. Cuartero Gomez, and D. de Frutos Escrig. On non-decidability of reachability for timed-arc Petri nets. In *Proceedings of the 8th Int. Workshop on Petri Net and Performance Models (PNPM'99)*, pages 188–196, 1999.

[RP85] R. Razouk and C. Phelps. Performance analysis using timed Petri nets. In *Protocol Testing, Specification, and Verification*, pages 561–576, 1985.

[SBK92] F. B. Schneider, Bloom B, and Marzullo K. Putting time into proof outlines. In de Bakker, Huizing, de Roever, and Rozenberg, editors, *Real-Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*, 1992.

[VW86] M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc.* 1^{st} *IEEE Int. Symp. on Logic in Computer Science*, pages 332–344, June 1986.

[Wol86] Pierre Wolper. Expressing interesting properties of programs in propositional temporal logic (extended abstract). In *Proc.* 13^{th} *ACM Symp. on Principles of Programming Languages*, pages 184–193, Jan. 1986.

[Yov97] S. Yovine. Kronos: A verification tool for real-time systems. *Journal of Software Tools for Technology Transfer*, 1(1-2), 1997.