Specification and Model Checking of Time Petri Nets and Timed Automata

WOJCIECH PENCZEK
ICS PAS, Warsaw, Poland
Outline

- Petri nets (PNs)
- Time Petri nets (TPNs)
- Timed automata (TA)
- Timed temporal logics: TCTL
- Verification methods for TPNs: state class approaches
- From TPNs to TA
- Verification methods for TA: partitioning and SAT-based approaches
- Experimental results for verifying TPNs directly and TPNs via TA
Petri nets are directed weighted graphs of two types of nodes: places (representing conditions) and transitions (representing events). The arcs are assigned positive weights.
A Petri net is a four-element tuple $\mathcal{P} = (P, T, F, m^0)$, where

- $P = \{p_1, \ldots, p_{n_P}\}$ is a finite set of places,
- $T = \{t_1, \ldots, t_{n_T}\}$ is a finite set of transitions, where $P \cap T = \emptyset$,
- $F : (P \times T) \cup (T \times P) \rightarrow N$ is the flow function, and
- $m^0 : P \rightarrow N$ is the initial marking of $\mathcal{P}$. 
Some history

Timed extensions of Petri nets:

Timed Petri nets  [Ramchandani’74]

Time Petri nets  [Merlin, Farber’76]

Timed extensions of automata theory:

Timed automata  [Alur,Dill’90]

Hybrid automata  
[Alur, Courcoubetis, Henzinger, Ho’93;  Nicollin, Olivero, Sifakis, Yovine’93]
Time Petri nets - an example
A **time Petri net** (TPN): \( \mathcal{N} = (P, T, FR, Eft, Lft, m_0) \), where

- \( P = \{p_1, \ldots, p_{n_P}\} \) - a finite set of **places**, 
- \( T = \{t_1, \ldots, t_{n_T}\} \) - a finite set of **transitions**, 
- \( FR \subseteq (P \times T) \cup (T \times P) \) - the **flow relation**, 
- \( Eft : T \rightarrow \mathbb{N}, Lft : T \rightarrow \mathbb{N} \cup \{\infty\} \) - the **earliest** and the **latest firing time** of the transitions; \( Eft(t) \leq Lft(t) \),
- \( m_0 \subseteq P \) - the **initial marking** of \( \mathcal{N} \).
TPNs - some definitions

\[ \bullet t = \{ p \in P \mid (p, t) \in FR \} \text{ - the preset of } t \in T, \]

\[ t\bullet = \{ p \in P \mid (t, p) \in FR \} \text{ - the postset of } t \in T, \]

a marking of \( \mathcal{N} \) - any subset \( m \subseteq P \),

a transition \( t \in T \) is enabled at \( m \) (\( m[t] \) for short) if

\[ \bullet t \subseteq m \text{ and } t \bullet \cap (m \setminus \bullet t) = \emptyset, \]

\[ \text{en}(m) = \{ t \in T \mid m[t] \}. \]
TPN: Mutual Exclusion Protocol

Advanced Course on Petri Nets 2010, Rostock, September 2010 – p.9/55
Concrete states of TPNs: clock approach

A *concrete state* of a net - a pair $\sigma = (m, \text{clock})$, where $m$ - a marking, $\text{clock}$ - values of clocks.

$\sigma^0 = (m_0, (0, \ldots, 0))$ - an initial state
Concrete states of TPNs: clock approach

A *concrete state* of a net - a pair $\sigma = (m, \text{clock})$, where
$m$ - a marking, $\text{clock}$ - values of clocks.

$\sigma^0 = (m_0, (0, \ldots, 0))$ - an initial state

Clocks can be associated with:

- transitions, places, or processes of a distributed net.
Concrete states of TPNs: clock approach

A concrete state of a net - a pair \( \sigma = (m, \text{clock}) \), where 

\( m \) - a marking, \( \text{clock} \) - values of clocks.

\( \sigma^0 = (m_0, (0,\ldots,0)) \) - an initial state

Clocks can be associated with:

- transitions, places, or processes of a distributed net.

Concrete states change because of:

- firing of a transition \( (\sigma \xrightarrow{t} c \sigma', t \in T) \),

- passing some time which does not disable any enabled transition \( (\sigma \xrightarrow{\tau} c \sigma') \).
Concrete states of TPNs: clock approach

A **concrete state** of a net - a pair \( \sigma = (m, \text{clock}) \), where

- \( m \) - a marking,
- \( \text{clock} \) - values of clocks.

\( \sigma^0 = (m_0, (0, \ldots, 0)) \) - an initial state

Clocks can be associated with:

- transitions, places, or processes of a distributed net.

Concrete states change because of:

- firing of a transition \( (\sigma \xrightarrow{t} \sigma') \),
- passing some time which does not disable any enabled transition \( (\sigma \xrightarrow{\tau} \sigma') \).

**Discrete transition relation:** \( \sigma \xrightarrow{t} d \sigma' \) iff \( \sigma \xrightarrow{\tau} c \xrightarrow{t} c \xrightarrow{\tau} c \sigma' \), \( t \in T \)
A **concrete state** of a net - a pair \( \sigma^F = (m, f) \), where

- \( m \) - a marking,
- \( f \) - **firing interval function** assigning to each \( t \in \text{en}(m) \) the timing interval in which \( t \) can fire.

\[
(\sigma^0)^F = (m_0, f_0)
\]

- an initial state,

where \( f_0(t) = [Eft(t), Lft(t)] \) for all \( t \in \text{en}(m_0) \)
Concrete states of TPNs: firing interval approach

A **concrete state** of a net - a pair $\sigma^F = (m, f)$, where
$m$ - a marking, and $f$ - **firing interval function** assigning to
each $t \in en(m)$ the timing interval in which $t$ can fire.

$$(\sigma^0)^F = (m_0, f_0)$$ - an initial state,
where $f_0(t) = [Eft(t), Lft(t)]$ for all $t \in en(m_0)$

**Concrete states change** because of:

- firing of a transition $(\sigma^F \xrightarrow{t} \sigma'^F, t \in T)$.
- passing some time which does not disable any enabled
  transition $(\sigma^F \xrightarrow{\tau} \sigma'^F)$. 
Concrete models for TPNs

\[ \Sigma \] - a set of all the concrete states of \( \mathcal{N} \)

\[ PV = \{ \emptyset_p \mid p \in P \} \] - a set of propositional variables

\[ V_c : \Sigma \rightarrow PV \] - a valuation function s.t.

\[ V_c((m, \cdot)) = \{ \emptyset_p \mid p \in m \} \]

\[ M_c(\mathcal{N}) = ((\Sigma, \sigma^0, \rightarrow), V_c), \text{ where } \rightarrow \in \{ \rightarrow_c, \rightarrow_d \} \]

- a concrete model of \( \mathcal{N} \) (usually infinite)
\[ \mathcal{X} = \{x_1, \ldots, x_n\} \] - a set of variables (*clocks*).

**Zone** - each convex polyhedron in \( \mathbb{R}^n \) which can be described by a finite set of inequalities of the form \( x_i \sim c \) or \( x_i - x_j \sim c \), where \( \sim \in \{\leq, <, >, \geq\} \) and \( c \in \mathbb{N} \).

\( Z(n) \) - the set of all the zones in \( \mathbb{R}^n \)
Timed automata - an example

0

approach

x := 0

1

exit

x <= 500

3

in

x >= 300

2

out

x <= 500

x-y <= 500

x <= 500
Timed automata - definition

A timed automaton $\mathcal{A}$ is a tuple $(A, L, \mathcal{X}, l^0, E, \mathcal{I})$, where:

- $A$ - a finite set of actions;
- $L$ - a finite set of locations;
- $\mathcal{X} = \{x_1, \ldots, x_n\}$ - a finite set of clocks;
- $l^0 \in L$ - an initial location;
- $E \subseteq L \times A \times Z(n) \times 2^\mathcal{X} \times L$ - a transition relation;
- $\mathcal{I} : L \rightarrow Z(n)$ - a location invariant.
A timed automaton $\mathcal{A}$ is a tuple $(A, L, X, l^0, E, \mathcal{I})$, where:

- $A$ - a finite set of actions;
- $L$ - a finite set of locations;
- $X = \{x_1, \ldots, x_n\}$ - a finite set of clocks;
- $l^0 \in L$ - an initial location;
- $E \subseteq L \times A \times Z(n) \times 2^X \times L$ - a transition relation;
- $\mathcal{I} : L \rightarrow Z(n)$ - a location invariant.

To reason about properties:

$V_A : L \rightarrow 2^{PV}$ - a valuation function for a set of propositional variables $PV$. 

Advanced Course on Petri Nets 2010, Rostock, September 2010 – p.15/55
Train–Gate–Controller example
Fischer’s mutual exclusion protocol for two processes
A *concrete state* of $\mathcal{A}$ is a pair $q = (l, v)$, where $l \in L$, and $v \in \mathbb{R}^n$.

$q^0 = (l^0, (0, \ldots, 0))$ - the initial state
Concrete states of TA

A **concrete state** of $\mathcal{A}$ is a pair $q = (l, v)$, where $l \in L$, and $v \in \mathbb{R}^n$.

$q^0 = (l^0, (0, \ldots, 0))$ - the initial state

Concrete states can change because of:

- a transition between locations ($q \xrightarrow{e} q'$, $e \in E$),
- passage of time ($q \xrightarrow{\tau} q'$).
A concrete state of $\mathcal{A}$ is a pair $q = (l, v)$, where $l \in L$, and $v \in \mathbb{R}^n$.

$q^0 = (l^0, (0, \ldots, 0))$ - the initial state

Concrete states can change because of:

- a transition between locations $(q \xrightarrow{e_{lc}} q', e \in E)$,
- passage of time $(q \xrightarrow{\tau_{lc}} q')$.

Discrete transition relation:

$q \xrightarrow{d_{lc}} q'$ iff $q \xrightarrow{\tau_{lc}} e \xrightarrow{e_{lc}} \xrightarrow{\tau_{lc}} q'$, $e \in E$
Concrete models for TA

\( Q \) - a set of all the concrete states of \( \mathcal{A} \)

\( PV \) - a set of propositional variables

\( V_c : Q \to PV \) - a *valuation function* which extends \( V_{\mathcal{A}} \)

\( V_c((l, \cdot)) = V_{\mathcal{A}}(l) \) (assigns the same propositions to the states with the same locations)

\( M_c(\mathcal{A}) = ((Q, q^0, \to), V_c), \) where \( \to \in \{ \to_c, \to_d \} \)

- a *concrete model* of \( \mathcal{N} \) (*usually infinite*)
Abstract models

\[ M_a = ((W, w^0, \rightarrow), V) \] - an **abstract model** for a concrete model \[ M_c = ((S, s^0, \rightarrow), V_c) \]

- each node \( w \in W \) is a set of states of \( S \) and \( s^0 \in w^0 \),
- \( V(w) = V_c(s) \) for each \( s \in w \),
- \( EE) \ w_1 \xrightarrow{b} w_2 \) if \( (\exists s_1 \in w_1) (\exists s_2 \in w_2) \) s.t. \( s_1 \xrightarrow{b} s_2 \).

Other conditions depend on the properties to be preserved.
Examples of abstract models

**Surjective models:**

EA) \( w_1 \xrightarrow{b} w_2 \) iff \( (\forall s_2 \in w_2) \ (\exists s_1 \in w_1) \ s_1 \xrightarrow{b} s_2 \).
Examples of abstract models

**Surjective models:**

EA) \( w_1 \overset{b}{\rightarrow} w_2 \text{ iff } (\forall s_2 \in w_2) \ (\exists s_1 \in w_1) \ s_1 \overset{b}{\rightarrow} s_2. \)

**Bisimulating (b-) models:**

AE) \( w_1 \overset{b}{\rightarrow} w_2 \text{ iff } (\forall s_1 \in w_1) \ (\exists s_2 \in w_2) \ s_1 \overset{b}{\rightarrow} s_2. \)
Examples of abstract models - cont’d

Simulating (s-) models:
for each \( w \in W \) there is a nonempty \( w^{cor} \subseteq w \) s.t.

\[ s^0 \in (w^0)^{cor}, \text{ and} \]

\[ \bigcup \ (w_1 \xrightarrow{b} w_2 \iff (\forall s_1 \in w_1^{cor}) (\exists s_2 \in w_2^{cor}) s_1 \xrightarrow{b} s_2. \]
Temporal logics: CTL*

$P_V = \{\varphi_1, \varphi_2 \ldots\}$ - a set of propositional variables.

Syntax of CTL*:  
the state formulas $\varphi_s$, defined using path formulas $\varphi_p$:

$$
\begin{align*}
\varphi_s & ::= \varphi \mid \neg \varphi \mid \varphi_s \land \varphi_s \mid \varphi_s \lor \varphi_s \mid A \varphi_p \mid E \varphi_p \\
\varphi_p & ::= \varphi_s \mid \varphi_p \land \varphi_p \mid \varphi_p \lor \varphi_p \mid X \varphi_p \mid \varphi_p U \varphi_p \mid \varphi_p R \varphi_p
\end{align*}
$$

$A$ (’for all paths’) and $E$ (’there exists a path’) are path quantifiers,  

$X$ (’next’), $U$ (’Until’), and $R$ (’Release’) are state operators.
Temporal operators of CTL

\[ s \models AX\varphi \]

\[ s \models A(\varphi U\psi) \]

\[ s \models A(\varphi R\psi) \]
\[ PV = \{ \varphi_1, \varphi_2, \ldots \} \] - a set of propositional variables.

**Syntax of TCTL:**
the formulas defined by the grammar:

\[ \varphi ::= \varphi \mid \neg \varphi \mid \varphi \lor \varphi \mid \varphi \land \varphi \mid E(\varphi U_I \varphi) \mid E(\varphi R_I \varphi), \]

where \( \varphi \in PV \) and \( I \) is an interval in \( \mathbb{N} \).
Temporal operators of TCTL

\[ s \models E(\varphi U_{[2,3]} \psi) \]
Temporal operators of TCTL

\[ s \models E(\varphi U_{[2,3]} \psi) \]

\[ s \models E(\varphi R_{[2,3]} \psi) \]

\[ s \models E(\varphi [2,3] \psi) \]
A state class of a TPN is a pair $C = (m, I)$, where $m$ is a marking, and $I$ is a set of inequalities built over variables corresponding to transitions.

State classes can be defined for both the clock and firing intervals approach.
Abstract models of timed systems

surjective models (LTL, reachability)

TPN:  State Class Graph (SCG): Berthomieu, Menasche (IFIP WCC’83), Berthomieu, Diaz 1991, Strong SCG: Berthomieu, Vernadat (TACAS’03), Geometric Region Graph: Yoneda, Ryuba 1998, Gardey, O. H. Roux, O. F. Roux (FORMATS’03) and others

TA: Bouajjani, Tripakis, Yovine (RTSS’97)
Abstract models of timed systems

surjective models (LTL, reachability)
b-models, discrete semantics (CTL*)

**TPN:** Atomic SCG - Yoneda, Ryuba 1998, Strong Atomic SCG - Berthomieu, Vernadat (TACAS’03), Improved SCG - Hadjidj, Boucheneb (STTT’08)

**TA:** ≈ Alur, Courcoubetis, Dill, Halbwachs, Wong-Toi (CONCUR’92), Dembiński, Penczek, Półroła (Fundamenta Informaticae 2002)
Abstract models of timed systems

surjective models (LTL, reachability)  TPN ✓  TA ✓
b-models, discrete semantics (CTL*)  TPN ✓  TA ✓
b-models, dense semantics (CTL*$_X$, TCTL)  TPN ✓  TA ✓

TA: Alur, Courcoubetis, Dill, Halbwachs, Wong-Toi (RTSS’92); Yannakakis, Lee (CAV’93);
Tripakis, Yovine (CAV’96)
Abstract models of timed systems

surjective models (LTL, reachability)

b-models, discrete semantics (CTL*)

b-models, dense semantics (CTL*$_X$, TCTL)

s-models, discrete semantics (ACTL*)

TPN: Pseudo-Atomic SCG - Penczek, Półrola (ICATPN’01)

TA: Dembiński, Penczek, Półrola (Fundamenta Informaticae, 2002)
Abstract models of timed systems

- surjective models (LTL, reachability)
- b-models, discrete semantics (CTL*)
- b-models, dense semantics (CTL*$_X$, TCTL)
- s-models, discrete semantics (ACTL*)
- s-models, dense semantics (ACTL*$_X$, TACTL)

TPN: Dembiński, Penczek, Półrola (Fundamenta Informaticae 2002)
Abstract models of timed systems

- surjective models (LTL, reachability) ✔ ✔
- b-models, discrete semantics (CTL*) ✔ ✔
- b-models, dense semantics (CTL$_X^*$, TCTL) ✔ ✔
- s-models, discrete semantics (ACTL*) ✔
- s-models, dense semantics (ACTL$_X^*$, TACTL) — ✔
- pb-models (reachability) — ✔

TPN: Tórola, Penczek, Szreter (Fundamenta Informaticae 2002)
surjective models (LTL, reachability)

b-models, discrete semantics (CTL*)

b-models, dense semantics (CTL_{X,} TCTL)

s-models, discrete semantics (ACTL*)

s-models, dense semantics (ACTL_{X,} TACTL)

pb-models (reachability)

ps-models (reachability)

**TPN:**

<table>
<thead>
<tr>
<th>TPN</th>
<th>TA</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>–</td>
<td>✓</td>
</tr>
<tr>
<td>–</td>
<td>✓</td>
</tr>
</tbody>
</table>

**TA:** Półrola, Penczek, Szreter (FORMATS’03)
Abstract models of timed systems

TPN: Okawa, Yoneda 1997, Virbitskaite, Pokozy 1999
TA: Alur, Courcoubetis, Dill (LISC’90)

surjective models (LTL, reachability) ✓ ✓
b-models, discrete semantics (CTL*) ✓ ✓
b-models, dense semantics (CTL*$_{-X}$, TCTL) ✓ ✓
s-models, discrete semantics (ACTL*) ✓ ✓
s-models, dense semantics (ACTL*$_{-X}$, TACTL) — ✓
pb-models (reachability) — ✓
ps-models (reachability) — ✓
detailed region graph (CTL*$_{-X}$, TCTL) ✓ ✓
Verifying TPNs via a translation to TA

To adapt TA-specific verification methods to TPNs, we need:

- clocks
- locations and invariants
- guards and resets.
“Transitions–as–clocks” approach

clock_t1 < 2

1 < clock_t1 < 2

0 < clock_t2 < 3

clock_t2 < 3

clock_t3 < 2

clock_t5 < 2

clock_t5 ≤ 2

clock_t5 ≤ 2

clock_t5 ≤ 2
**Other translations TPN → TA**

- **Sifakis, Yovine (STACS’96)**
  translating time stream Petri nets (TSPNs) to TA with disjunctions of clock constraints. TPNs are a subclass of TSPNs.

- **Cortés, Eles, Peng (RTCSA’02)**
  translating extended TPNs (called PRES+ models) to a network of (extended) TA, exploiting “clusters” (sets of sequentially enabling transitions).

- **Lime, Roux (PNPM’03)**
  translation based on building SCG for the net (“state class automaton”).

- **Gu, Shin (DIPES’02), Cassez, Roux (MSR’03)**
  translations to TA with shared variables and urgency modelling.
Partitioning algorithms

\[ \Pi \subseteq 2^S \] - a partition of the state space \( S \) into classes

for TA, classes are represented by \((location, zone, \cdots)\)

(additional components depend on the kind of the model to be built)

The partitioning (minimization) algorithms generate models whose states are classes of a partition:

- start from an initial partition \( \Pi_0 \) of the state space,
- successively refine the partition until all the classes of \( \Pi \) satisfy an appropriate condition (\( AE, EA, U, \cdots \)).
Partitioning algorithm: how it works
(an example for b-models)
Partitioning algorithm: how it works
(an example for b-models)
Partitioning algorithm: how it works
(an example for b-models)
Partitioning algorithm: how it works
(an example for b-models)
Partitioning algorithm: how it works
(an example for b-models)
Partitioning algorithm: how it works
(an example for b-models)
Partitioning algorithm: how it works
(an example for b-models)
Partitioning algorithm: how it works
(an example for b-models)
Partitioning algorithm: how it works
(an example for b-models)
...and how partitioning works for s-models

pseudo-$e$-stable  pseudo-$e$-unstable  semi-$e$-unstable  $e$-unstable

modify $X^{cor}$  split $\hat{X}$  split $\hat{Y}$ and modify $X^{cor}$  split $\hat{X}$ and $\hat{Y}$
Symbolic data structures

Difference Bound Matrices (DBM) [Dill’89] - for representing state classes of TPNs or regions of TA.

Clock Difference Diagrams (CDD) [Behrmann et al.’99]  
Clock Restriction Diagrams (CRD) [Wang’00],  
Difference Decision Diagrams (DDD) [Møller et al.’99] - for representing sets of regions.

Propositional Logic (PL) - for representing sets of detailed regions.
Detailed zones and regions

$c_{max}$ - the largest constant used in $A$

**Detailed zones** - the equivalence classes of the zone equivalence relation $\simeq$ in the set of the clock valuations.

The set of all the detailed zones is denoted by $DZ(n)$.

$DZ(2)$

$c_{max} = 1$
Detailed zones and regions

\( c_{\text{max}} \) - the largest constant used in \( A \)

*Detailed zones* - the equivalence classes of the zone equivalence relation \( \sim \) in the set of the clock valuations.

The set of all the detailed zones is denoted by \( DZ(n) \).

\[
DZ(2)
\]

\( c_{\text{max}} = 1 \)
*Detailed zones and regions*

\[ \text{\textcolor{blue}{c}}_{\text{max}} \text{ - the largest constant used in } A \]

**Detailed zones** - the equivalence classes of the zone equivalence relation \( \sim \) in the set of the clock valuations.

The set of all the detailed zones is denoted by \( DZ(n) \).

\[ DZ(2) \]

\[ c_{\text{max}} = 1 \]

**time steps**

\[ \text{by transition } s \xrightarrow{x_1:=0} s' \]
$c_{\text{max}}$ - the largest constant used in $\mathcal{A}$

**Detailed zones** - the equivalence classes of the zone equivalence relation $\sim$ in the set of the clock valuations.

The set of all the detailed zones is denoted by $DZ(n)$.

$$DZ(2)$$

$c_{\text{max}} = 1$

**Time steps**

**Action steps**

by transition $s \xrightarrow{x_1 := 0} s'$

$(l, Z)$ - a (detailed) region, where $l \in L$ and $Z \in DZ(n)$. 
BMC: exploiting a part of the model

Selecting submodels of the $k$-model
Discretization scheme

[Asarin, Bozga, Kerbrat, Maler, Pnueli, Rasse, "Data-Structures for the Verification of Timed Automata"]

Discretizing $[0, 1)^2$: the circle points are the elements of the discretization.
Discretization scheme

[Asarin, Bozga, Kerbrat, Maler, Pnueli, Rasse, "Data-Structures for the Verification of Timed Automata"]

Discretizing $[0, 1)^2$: the circle points are the elements of the discretization.
Discretization scheme

Asarin, Bozga, Kerbrat, Maler, Pnueli, Rasse, "Data-Structures for the Verification of Timed Automata"

Discretizing $[0, 1)^2$:
the circle points are the elements of the discretization.
A - a timed automaton with \( n \) clocks

\[
\Delta = \frac{1}{m} \text{ as a discretization step, where } m = 2^{\lceil \log_2(2 \cdot n) \rceil}
\]

\[
\mathcal{D} = \{l \cdot \Delta \mid 0 \leq l \cdot \Delta < 2 \cdot c_{\max} + 2\} \text{ - the set of discretized values, and}
\]

\[
E = \{l \cdot \Delta \mid 0 \leq l \cdot \Delta < c_{\max} + 1\} \text{ - the set of labels.}
\]

The discrete representatives:

\[
U = \{u \in \mathcal{D}^n \mid (\forall x \in \mathcal{X})(\exists l \in \mathbb{N})u(x) = 2l\Delta \quad \lor
\]

\[
(\forall x \in \mathcal{X})(\exists l \in \mathbb{N})u(x) = (2l + 1)\Delta\}
\]
Discrete abstract model for a timed automaton:

\[ DM(\mathcal{A}) = ((S, (l^0, (0, \ldots, 0)), \rightarrow), V) \]

where \( S = L \times \mathbb{U} \) is the set of states, and the transition relation \( \rightarrow \) has two types of transitions:

"SMART" time transitions
(the transitive closure of timed steps)

"SMART" action transitions
(action steps combined with 'adjust' steps to remain within \( \mathbb{U} \)).
A **special $k$-path** is a finite sequence $\pi = \langle u_0, \cdots, u_k \rangle$ of states of $DM(A)$ such that

\[
u_0 = (l^0, (0, \ldots, 0))\] is the initial state of $DM(A)$,

\[
u_i \longrightarrow \nu_{i+1} \text{ for each } 0 \leq i < k,
\]

the transition $\nu_0 \longrightarrow \nu_1$ is a **time transition**,

each **time transition** is followed by an **action transition**,

each **action transition** is followed by a **time transition**.

$(l^0, (0, \ldots, 0))$
Checking reachability

Each state \((l, (v_1, \ldots, v_n))\) is represented by a vector \(u_i = (u_{i,1}, \ldots, u_{i,m})\) of propositional variables, where \(m\) depends on the number of locations, clocks, and \(c_{\text{max}}\).

\(udp(u)\) - a propositional formula encoding an undesirable property.

\(path_k(u_0, \ldots, u_k)\) - a propositional formula encoding all the special \(k\)-paths.

\[
\alpha = path_k(u_0, \ldots, u_k) \land \bigvee_{i=0}^{k} udp(u_i)
\]

The property is reachable \(\iff\) \(\alpha\) is satisfiable.
Checking unreachability - an intuition

To prove unreachability of states satisfying $udp(u)$:

Search for a longest path from an arbitrary state via states satisfying $\neg udp(u)$ to a state satisfying $udp(u)$

if such a path $\pi$ exists, then

a path from the initial state to a state satisfying $udp(u)$ cannot be longer than $\pi$,

therefore it is sufficient to test reachability only for $k = length(\pi)$
A **free special \( k \)-path** is a finite sequence \( \pi = \langle u_0, \cdots, u_k \rangle \) of states of \( DM(A) \) such that

- \( u_i \rightarrow u_{i+1} \) for each \( 0 \leq i < k \),
- the transition \( u_0 \rightarrow u_1 \) is a **time transition**, each **time transition** is followed by an **action transition**, each **action transition** is followed by a **time transition**.
Searching for the longest witness

Find the length of a longest free special $k$-path $\pi$ s.t.:

- the last transition of $\pi$ is an action transition
- the undesirable property is true in the last state and false in all the previous states of $\pi$

$\text{freepath}_k(u_0, \ldots, u_k)$ - a propositional formula encoding all the free special $k$-paths.

Check satisfiability of $\beta$:

$$\beta = \text{freepath}_k(u_0, \ldots, u_k) \land \text{udp}(u_k) \land \bigwedge_{i=0}^{k-1} \neg \text{udp}(u_i)$$
Checking unreachability

\[
\beta = freepath_k(u_0, \ldots, u_k) \land udp(u_k) \land \bigwedge_{i=0}^{k-1} \neg udp(u_i)
\]

If \( \beta \) is unsatisfiable for some \( k_0 \in \{2, 4, 6, \ldots\} \), to prove unreachability of \( udp(u) \), it is sufficient to verify satisfiability of the formula

\[
\alpha = path_k(u_0, \ldots, u_k) \land \bigvee_{i=0}^{k} udp(u_i)
\]

only for \( k = k_0 - 2 \).
Selected tools for TPNs

**Tina** - a toolbox for analysis of (time) Petri nets. It constructs (atomic) state class graphs and performs (CTL) LTL or reachability verification.

**Romeo** - provides several methods for translating TPNs to TA and computation of state class graphs.

**Petri Net Toolbox** - a tool for simulation, analysis and synthesis of discrete event systems based on (Time) Petri net models.
Selected tools for TPNs - cont’d

PEP (Programming Environment based on Petri nets) - various verification algorithms (e.g., reachability and deadlock-freeness checking, partial-order based model checking).

INA (Integrated Net Analyser) - a Petri net analysis tool. INA provides verification by analysis of paths for TPNs.

CPN Tools - a software package for modelling and analysis of both timed and untimed Coloured Petri Nets, enabling their simulation, generating occurrence (reachability) graph, and analysis by place invariants.
Selected tools for TA

**Kronos** - uses DBM’s to perform verification of TCTL using partitioning algorithms.

**UppAal2k** uses CDD to represent unions of convex clock regions for modelling, simulation and verification of timed automata.

**Red** is a model checker based on CRD. It supports TCTL model checking.
**Selected tools for TA**

**Rabbit** - a tool for BDD-based verification of extended timed automata, called Cottbus Timed Automata. It provides reachability analysis.

**VerICS** implements partition refinement algorithms and SAT-based BMC for verifying TCTL and reachability for timed automata and Estelle programs.
## Experimental results

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Net 5a</td>
<td></td>
<td>Net 5b</td>
<td></td>
<td>Net 5c</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>states</td>
<td>edges</td>
<td>states</td>
<td>edges</td>
<td>states</td>
<td>edges</td>
<td></td>
</tr>
<tr>
<td>obtained by TPN - specific methods</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tina</td>
<td>SCG</td>
<td>18</td>
<td>26</td>
<td>34</td>
<td>58</td>
<td>50</td>
<td>76</td>
</tr>
<tr>
<td>Tina</td>
<td>SSCG</td>
<td>21</td>
<td>29</td>
<td>39</td>
<td>63</td>
<td>60</td>
<td>93</td>
</tr>
<tr>
<td>Tina</td>
<td>SASCG</td>
<td>36</td>
<td>61</td>
<td>62</td>
<td>163</td>
<td>80</td>
<td>204</td>
</tr>
<tr>
<td>implem. of [YR98]</td>
<td>atomic</td>
<td>53</td>
<td>95</td>
<td>64</td>
<td>179</td>
<td>168</td>
<td>363</td>
</tr>
<tr>
<td>implem. of [YR98]</td>
<td>geometric</td>
<td>16</td>
<td>25</td>
<td>32</td>
<td>57</td>
<td>105</td>
<td>170</td>
</tr>
<tr>
<td>obtained by TPN to TA translations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kronos</td>
<td>bis. dense</td>
<td>51</td>
<td>77</td>
<td>134</td>
<td>229</td>
<td>185</td>
<td>321</td>
</tr>
<tr>
<td>Kronos</td>
<td>forw-ai-ax</td>
<td>37</td>
<td>42</td>
<td>37</td>
<td>42</td>
<td>26</td>
<td>40</td>
</tr>
<tr>
<td>VerICS</td>
<td>bis. dense</td>
<td>54</td>
<td>80</td>
<td>135</td>
<td>230</td>
<td>186</td>
<td>323</td>
</tr>
<tr>
<td>VerICS</td>
<td>bis. discr.</td>
<td>26</td>
<td>47</td>
<td>46</td>
<td>135</td>
<td>80</td>
<td>204</td>
</tr>
<tr>
<td>VerICS</td>
<td>ps- discr.</td>
<td>21</td>
<td>34</td>
<td>13</td>
<td>22</td>
<td>53</td>
<td>121</td>
</tr>
</tbody>
</table>

Abstract models for nets of [YR98] by some different tools
### Experimental results - cont’d

<table>
<thead>
<tr>
<th></th>
<th>noP</th>
<th>states</th>
<th>edges</th>
<th>noP</th>
<th>states</th>
<th>edges</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>obtained by TPN - specific methods</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tina</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SASCG</td>
<td>9</td>
<td>81035</td>
<td>280170</td>
<td>7</td>
<td>73600</td>
<td>200704</td>
</tr>
<tr>
<td>SCG</td>
<td>9</td>
<td>81035</td>
<td>280170</td>
<td>7</td>
<td>73600</td>
<td>200704</td>
</tr>
<tr>
<td>SSCG</td>
<td>9</td>
<td>81035</td>
<td>280170</td>
<td>7</td>
<td>73600</td>
<td>200704</td>
</tr>
<tr>
<td><strong>implem. of [YR98]</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>atomic</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not supported</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>geometric</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>not supported</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>obtained by TPN → TA translations</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Kronos</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bis. dense</td>
<td>5</td>
<td>807</td>
<td>1590</td>
<td>4</td>
<td>1008</td>
<td>1856</td>
</tr>
<tr>
<td>forw-ai-ax</td>
<td>5</td>
<td>33451</td>
<td>62223</td>
<td>4</td>
<td>12850</td>
<td>27848</td>
</tr>
<tr>
<td>VerICS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bis. dense</td>
<td>3</td>
<td>77</td>
<td>108</td>
<td>3</td>
<td>200</td>
<td>312</td>
</tr>
<tr>
<td>bis. discr.</td>
<td>3</td>
<td>65</td>
<td>96</td>
<td>3</td>
<td>152</td>
<td>240</td>
</tr>
<tr>
<td>ps- discr.</td>
<td>3</td>
<td>65</td>
<td>96</td>
<td>3</td>
<td>152</td>
<td>204</td>
</tr>
</tbody>
</table>

Abstract models for Fischer’s protocol by some different tools

**parameters:** $\Delta = 1, \delta = 2$ or $\Delta = 2, \delta = 1$, max. 128 MB RAM and max. 1800 s
Experimental results - cont’d

<table>
<thead>
<tr>
<th>NoP</th>
<th>TPN to TA</th>
<th>TA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>vars</td>
<td>clauses</td>
</tr>
<tr>
<td>8</td>
<td>61530</td>
<td>176319</td>
</tr>
<tr>
<td>8</td>
<td>22552</td>
<td>64442</td>
</tr>
<tr>
<td>10</td>
<td>29918</td>
<td>86002</td>
</tr>
<tr>
<td>50</td>
<td>378203</td>
<td>1118763</td>
</tr>
<tr>
<td>104</td>
<td>1411156</td>
<td>4200809</td>
</tr>
<tr>
<td>310</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

BMC of VerICS for Fischer’s protocol modelled by TPN and TA
parameters: $\Delta = 1, \delta = 2$ (k=44) or $\Delta = 2, \delta = 1$ (k=17)
Main References

References

