Time Petri Nets
Part II: State Class based methods

Bernard Berthomieu
LAAS/CNRS, Université de Toulouse
7 avenue du Colonel Roche, 31077 Toulouse, France
Bernard.Berthomieu@laas.fr

ATPN
XIAN – June 2008
Plan

1. Background

2. State Class graphs as abstract state spaces

3. State Classes Preserving markings and traces

4. Preserving states and traces

5. Preserving states and branching properties

6. Quantitative properties, Other techniques

7. Subclasses, extensions, alternatives

8. Application areas, Tools
Background

1. Background

2. State Class graphs as abstract state spaces

3. State Classes Preserving markings and traces

4. Preserving states and traces

5. Preserving states and branching properties

6. Quantitative properties, Other techniques

7. Subclasses, extensions, alternatives

8. Application areas, Tools
1. Background

Time Petri Nets

Dense semantics, state spaces

Representation of states – firing domains, clocks vectors

Basic theorems – decidability results

Logics
Time Petri Nets (Merlin 1974) [Me74,MF76]

\[(P, T, \text{Pre}, \text{Post}, m_0, I_S)\] where

- \((P, T, \text{Pre}, \text{Post}, m_0)\) is a Petri net
- \(I_S\) is the Static Interval Function

\[t \mapsto I_S(t) \subseteq \mathbb{R^+}, \text{ rational bounds}\]
Behaviour

States characterize sets of time-transition sequences
Terminology, Notations

\( p, p', \ldots \) : places

\( t, t', \ldots \) : transitions

\( m, m', \ldots \) : markings, map places to nonnegative integers

\( \mathcal{E}(m) \) : transitions enabled at \( m \), \( t \in \mathcal{E}(m) \Leftrightarrow \text{Pre}(t) \leq m \)

\( I, I', \ldots \) : interval functions, map enabled transitions to real intervals

\( \downarrow I(t) \) : earliest firing time of \( t \) (left endpoint of \( I(t) \))

\( \uparrow I(t) \) : latest firing time of \( t \) (right endpoint of \( I(t) \), or \( \infty \))

\( \sigma, \sigma', \ldots \) : sequences of transitions

\( \rho, \rho', \ldots \) : time-transition sequences (or firing schedules) \( \theta_1.t_1.\theta_2.t_2\ldots \)

\( |\rho| \) : support of \( \rho \), \( |\theta_1.t_1.\theta_2.t_2\ldots| = t_1..t_2\ldots \)

\( f \setminus D = \{(x,y) \in f \mid x \in D\} \) : restriction of function \( f \) to domain \( D \)

\( I \sim \theta = \{x - \theta \mid x \geq \theta \land x \in I\} \) : interval \( I \ (I \subseteq \mathbb{R}^+) \) shifted by \( \theta \) and truncated
Semantics

A state is a pair $s = (m, I) \in S$, where:

- $m$ is a marking
- $I$ is an interval function with domain $\mathcal{E}(m)$

The initial state is $s_0 = (m_0, Is \setminus \mathcal{E}(m_0))$

There are two sorts of transitions:

- **discrete transitions**: $(m, I) \overset{t}{\rightsquigarrow} (m', I')$ iff $t \in T$ and
  1. $m \geq \text{Pre}(t)$
  2. $0 \in I(t)$
  3. $m' = m - \text{Pre}(t) + \text{Post}(t)$
  4. $(\forall k \in T)(m' \geq \text{Pre}(k) \Rightarrow I'(k) = \begin{cases} \text{if } k \neq t \land m - \text{Pre}(t) \geq \text{Pre}(k) \quad \text{then } I(k) \\ \text{else } Is(k) \end{cases})$

- **continuous transitions**: $(m, I) \overset{d}{\rightsquigarrow} (m, I')$ iff
  $(\forall k \in T)(m \geq \text{Pre}(k) \Rightarrow d \leq \uparrow I(k) \land I'(k) = I(k) \downarrow d)$
State spaces

With all continuous and discrete transitions:

\[ SG = (S, \sim_t \cup \sim_d, s_0) \]

Any state is reachable from the initial state by some sequence alternating delays and discrete transitions (a time-transition sequence, or firing schedule).

Restricted to the targets of discrete transitions, delays abstracted:

\[ DSG = (S, \sim_t, s_0) \]

where

\[ s \sim_t s' \iff (\exists \theta)(\exists s'')(s \sim s'' \land \theta \sim t \sim s') \]

State graphs are typically infinite, dense.
Direct “Discrete” semantics ($DSG$)

Let $s \xrightarrow{t@\theta} s' \iff (\exists s'')(s \xrightarrow{\theta} s'' \land s'' \xrightarrow{t} s')$

Then $s \xrightarrow{t} s' \iff (\exists \theta)(s \xrightarrow{t@\theta} s')$

With $(m, I) \xrightarrow{t@\theta} (m', I')$ iff $t \in T$, $\theta \in \mathbb{R}^+$ and:

1. $\text{Pre}(t) \leq m$ (t is enabled at $m$)
   \[ \theta \geq \downarrow I(t) \]
   \[ (\forall k)(\text{Pre}(k) \leq m \Rightarrow \theta \leq \uparrow I(k)) \]

2. $m' = m - \text{Pre}(t) + \text{Post}(t)$

3. $(\forall k)(\text{Pre}(k) \leq m \Rightarrow I'(k) =$
   
   if $k \neq t \land m - \text{Pre}(t) \geq \text{Pre}(k)$
   then $I(k) \sim \theta$
   else $I_S(k))$
Example

\[ E_0 = (m_0, I_0) \]

\[ m_0 : p_1, p_2(2) \]
\[ I_0 : \text{solutions in } t_1 \text{ of } 4 \leq t_1 \leq 9 \]

\[ E_0 \xrightarrow{t_1@\theta_1} E_1 = (m_1, I_1) \text{ with } (\theta_1 \in [4, 9]) : \]

\[ m_1 : p_3, p_4, p_5 \]
\[ I_1 : \text{solutions in } (t_2, t_3, t_4, t_5) \text{ of } \]
\[ 0 \leq t_2 \leq 2 \]
\[ 1 \leq t_3 \leq 3 \]
\[ 0 \leq t_4 \leq 2 \]
\[ 0 \leq t_5 \leq 3 \]

\[ E_1 \xrightarrow{t_2@\theta_2} E_2 = (m_2, I_2) \text{ with } (\theta_2 \in [0, 2]) : \]

\[ m_2 : p_2, p_3, p_5 \]
\[ I_2 : \text{solutions in } (t_3, t_4, t_5) \text{ of } \]
\[ \max(0, 1 - \theta_2) \leq t_3 \leq 3 - \theta_2 \]
\[ 0 \leq t_4 \leq 2 - \theta_2 \]
\[ 0 \leq t_5 \leq 3 - \theta_2 \]

The schedule, or time-transition sequence, 5.t_1.0.t_2 is firable.
Representing states

By *Interval functions* (canonical)

\[ s = (m, \{(t_1, [2, 3]), (t_2, [2, \infty[), (t_3, ]0, 5]\}) \]

By firing domains (canonical)

\[ I \text{ represented by } \{\phi \mid \phi \in I(t_1) \times I(t_2) \times I(t_3)\} \]

\[ s = (m, \{\phi \in \mathbb{R}^3 \mid 2 \leq \phi_{t_1} \leq 3 \land 2 \leq \phi_{t_2} \land 0 < \phi_{t_3} \leq 5\}) \]

By clock vectors (surjection, relative to \( I_s \))

\[ I \text{ represented by } \underline{\gamma}, \text{ where } (\forall t \in \mathcal{E}(m))(I(t) = I_s(t) \rightarrow \underline{\gamma}_t) \]

\[ s = (m, \underline{\gamma}), \text{ with } \underline{\gamma} \in \mathbb{R}^3, \text{ indexed over } \{t_1, t_2, t_2\} \]

By “total” clock vectors (cf. Louchka, \# means “undefined”):

\[ s = (m, \underline{\gamma}), \text{ with } \underline{\gamma} \in (\mathbb{R} \cup \{\#\})^{|T|}, \text{ indexed over all transitions} \]
“General” Properties

Let \( R = \{ s \mid (\exists \rho)(s_0 \xrightarrow{\rho} s) \} \)

Problems:

**State reachability**: \( s \in R \)

**Marking reachability**: \((\exists I)((m, I) \in R)\)

**Liveness**: \((\forall s \in R)(\forall t \in T)(\exists \rho)(\exists s')(s \xrightarrow{\rho, t} s')\)

**Boundedness**: \((\exists b \in \mathbb{N})(\forall (m, I) \in R)(\forall p \in P)(m(p) \leq b)\)

**k-boundedness**: \((\forall (m, I) \in R)(\forall p \in P)(m(p) \leq k)\)
Decidability results

Marking reachability: undecidable [JLL77]

TPNs can encode 2-counter machines:

State reachability, Boundedness, Liveness: undecidable

k-boundedness: decidable [BM82]

For bounded TPNs: all problems decidable
Logics

Linear time

Propositional LTL (e.g. SPIN)

Interpreted over runs (infinite sequence of states)

(For each run)

\( \phi \) \( \phi \) true at the first state
\( \Diamond \phi \) \( \phi \) true at next state
\( \Box \phi \) \( \phi \) always true
\( \Diamond \Box \phi \) \( \phi \) eventually true
\( \phi U \psi \) \( \phi \) true until \( \psi \) does and \( \psi \) eventually true

\( \Box \Diamond \phi \) \( \phi \) true infinitely often (fairness requirements)
\( \Box (\phi \Rightarrow \Diamond \psi) \) \( \phi \) always results in \( \psi \) (later)

State/Event LTL (e.g. SELT/TINA)

Both state and event properties

A run is an infinite sequence alternating states and transitions

Linear time \( \mu \)-calculus
Logics

Branching time

CTL (Computational tree logic)

Interpreted at the states of a transition system

- $\phi$ holds at the current state
- $EX \; \phi$ some transition leads to a state at which $\phi$ holds
- $AX \; \phi$ all transitions lead to a state at which $\phi$ holds
- $E[\phi \; U \; \psi]$ $\psi$ true at current state or for some path ...
- $A[\phi \; U \; \psi]$ $\psi$ true at current state or for all paths ........

- $EF \; \phi = E[true \; U \; \phi]$
- $AF \; \phi = A[true \; U \; \phi]$
- $EG \; \phi = \neg(AF(\neg\phi))$
- $AG \; \phi = \neg(EF(\neg\phi))$

Fixpoint calculi

Modal $\mu$-calculus (Hennessy-Milner logic + least/greatest fixpoints)
(e.g. Evaluator/CADP, MEC5/Altarica)

Dicky/Arnold calculus ($src, tgt, rsrc, rtgt$ + systems of equations)
(MEC4/Altarica)
Useful CTL derived modalities
“Timed” Logics

Temporal or fixpoint operators tagged with clock expressions

(e.g. \( k \leq 5 \))

Linear time

MTL, MITL (Metric Temporal Logics)

Branching time

TCTL (e.g. Kronos, Uppaal (fragment), Romeo (fragment))

Timed \( \mu \)-calculi
State Classes

1. Background

2. State Class graphs as abstract state spaces

3. State Classes Preserving markings and traces

4. Preserving states and traces

5. Preserving states and branching properties

6. Quantitative properties, Other techniques

7. Subclasses, extensions, alternatives

8. Application areas, Tools
State space abstractions

Concrete state space infinite dense $\Rightarrow$ unsuitable representation

Abstraction required

state space is partitionned into abstract states

concrete states in an abstract state considered collectively

many possible partitions
Properties of abstract state spaces

$s \in S$ : concrete states, $c \in C$ : abstract states

All states in $c$ have a successor in all successors of $c$:

$$AE = (\forall c, c')(\forall t)(c \xrightarrow{t} A c' \Rightarrow (\forall s \in c)(\exists s' \in c')(s \xrightarrow{t} s'))$$

All states in $c$ have a predecessor in all predecessors of $c$:

$$EA = (\forall c, c')(\forall t)(c' \xrightarrow{t} A c \Rightarrow (\forall s \in c)(\exists s' \in c')(s' \xrightarrow{t} s))$$

Abstract states are linked ($\xrightarrow{A}$) iff concrete states are ($\xrightarrow{}$):

$$EE = (\forall t)(\forall s, s')(\forall c, c')(c \xrightarrow{t} A c' \iff s \xrightarrow{t} s')$$

Weaker $EE$, if $C$ is a cover of $S$ rather than a partition:

$$EE' = (\forall t)((\forall c, c')(c \xrightarrow{t} A c' \Rightarrow (\exists s \in c)(\exists s' \in c')(s \xrightarrow{t} s')) \wedge (\forall s, s')(s \xrightarrow{t} s' \Rightarrow (\exists c \ni s)(\exists c' \ni s')(c \xrightarrow{t} A c')))$$
Theorems

\[ s \in S : \text{concrete states}, \ c \in C : \text{abstract states} \]

EE is a soundness condition on \( C \) wrt \( S \)

Assuming \( C \) is a partition of \( S \): (see e.g. [PP04])

- AE ensures preservation of branching properties (bisimilarity)
- EA ensures preservation of linear properties (LTL)
State Class graphs

State : \( E = (m, I) \) : marking \( \times \) firing interval vector

State class graphs

Covers of the state space by convex (wrt time info) subsets of states

all states in a class share the same marking

satisfying \( EE' \) (\( \rightarrow_A \) simply written \( \rightarrow \))

Several partitions possible

Preserving markings

Preserving markings and \( LTL \) properties [BM 82, BM83, BD91]

Preserving states

Preserving states and \( LTL \) properties [BV03]

Preserving states and \( CTL \) properties [YR98, BV03]
State Classes

1. Background

2. State Class graphs as abstract state spaces

3. State Classes Preserving markings and traces

4. Preserving states and traces

5. Preserving states and branching properties

6. Quantitative properties, Other techniques

7. Subclasses, extensions, alternatives

8. Application areas, Tools
State classes

Recall direct discrete semantics:

\[ s \xrightarrow{t} s' \Leftrightarrow (\exists \theta)(s \xrightarrow{t@\theta} s') \]

With \((m, I) \xrightarrow{t@\theta} (m', I')\) iff \(t \in T\), \(\theta \in \mathbb{R}^+\) and:

1. \(\text{Pre}(t) \leq m\) (\(t\) is enabled at \(m\))
   \[
   \theta \geq \downarrow I(t)
   \]
   \[
   (\forall k)(\text{Pre}(k) \leq m \Rightarrow \theta \leq \uparrow I(k))
   \]

2. \(m' = m - \text{Pre}(t) + \text{Post}(t)\)

3. \((\forall k)(\text{Pre}(k) \leq m \Rightarrow I'(k) =\)
   \[
   \text{if } k \neq t \land m - \text{Pre}(t) \geq \text{Pre}(k) \text{ then } I(k) \div \theta \text{ else } I_S(k))
   \]

Idea: abstract parameter \(\theta\)
State classes

States

$$(m, \{\phi \mid \phi \in I(t_1) \times \ldots \times I(t_n)\}) \text{ where } \{t_1, \ldots, t_n\} = \mathcal{E}(m)$$

Representation of classes:

Marking $+$ firing domain

where

Marking of class $=$ marking of any state in the class

Domain of class $=$ solution set of inequality system $W\phi \leq q$

Equality of classes:

$$(m, W) \cong (m', W') \text{ iff } m = m' \text{ and } W \text{ and } W' \text{ have same solution set}$$
Computing State Classes

**Algorithm 1:** Computes $C_{\sigma,t} = (m', W')$ from $C_\sigma = (m, W)$:

- $C_\epsilon = (m_0, \{\downarrow I_s(t) \leq \phi_t \leq \uparrow I_s(t) \mid \text{Pre}(t) \leq m_0\})$
- $t$ is firable from some state of $C_\sigma$ iff:
  (i) $m \geq \text{Pre}(t)$ (t is enabled at m)
  (ii) $W$ augmented with the following is consistent:
    \[
    \{\phi_t \leq \phi_i \mid i \neq t \land m \geq \text{Pre}(i)\}
    \]
- If so, then $m' = m - \text{Pre}(t) + \text{Post}(t)$, and $W'$ is obtained by:
  1. add inequations (ii) to $W$;
  2. $\forall i$ enabled at $m'$, add variable $\phi'_i$ and inequations:
    \[
    \phi'_i = \phi_i - \phi_t, \text{ if } i \neq t \text{ and } m - \text{Pre}(t) \geq \text{Pre}(i)
    \]
    \[
    \downarrow I_s(i) \leq \phi_i \leq \uparrow I_s(i), \text{ otherwise}
    \]
  3. Eliminate variables $\phi$
- $(m, W) \cong (m', W')$ iff $m = m'$ and $W$ and $W'$ have equal solution sets
In terms of states

Let:

\[ C = \bigcup_{\sigma \in T^*} \{C_\sigma\}, \quad \text{where} \quad C_\varepsilon = \{s_0\}, \quad C_{\sigma.t} = \{s | (\exists s' \in C_\sigma)(s' \xrightarrow{t} s)\} \]

Then:

\[ SCG = (C/\cong, \xrightarrow{t}, [\{s_0\}]\cong) \]

\[ c \cong c' \iff (\forall((m, I), (m', I')) \in c \times c')(m = m') \land \bigcup_{s \in c}(\mathcal{F}(s)) = \bigcup_{s' \in c'}(\mathcal{F}(s')) \]

where \( \mathcal{F}(m, I) = I(t_1) \times \ldots \times I(t_n) \quad (t_1, \ldots, t_n \in \mathcal{E}(m)) \)

Note: SCG is an abstract state space
Example 1

\(C_0 = (p_1 p_2, \{4 \leq t_19\})\)
\(C_1 = (p_3 p_4, \{0 \leq t_2 \leq 4, 5 \leq t_3 \leq 6, 3 \leq t_4 \leq 6\})\)
\(C_2 = (p_2 p_3, \{1 \leq t_3 \leq 6, 0 \leq t_4 \leq 6, t_3 - t_4 \leq 3, t_4 - t_3 \leq 1\})\)
\(C_3 = (p_2 p_3, \{5 \leq t_3 \leq 6, 3 \leq t_4 \leq 6\})\)
\(C_4 = (p_3 p_4, \{0 \leq t_2 \leq 1, 5 \leq t_3 \leq 6, 3 \leq t_4 \leq 6\})\)
\(C_5 = (p_2 p_3, \{4 \leq t_3 \leq 6, 2 \leq t_4 \leq 6, t_3 - t_4 \leq 3, t_4 - t_3 \leq 1\})\)
Example 2

\[ C_0 = (p_0 \ p_3, \{0 \leq t_0 \leq 4, 5 \leq t_2 \leq 6\}) \]

\[ C_1 = (p_1 \ p_3, \{3 \leq t_1 \leq 4, 1 \leq t_2 \leq 6\}) \]

\[ C_2 = (p_2 \ p_3, \{0 \leq t_2 \leq 3\}) \]

\[ C_3 = (p_2 \ p_4, \{\}) \]

\[ C_4 = (p_1 \ p_4, \{0 \leq t_1 \leq 3\}) \]
TPN example
Properties of the abstraction

State sets equivalent by $\cong$ have same futures

SCG Finite iff the $TPN$ is bounded

Preserves markings and firing sequences ($LTL$)

Decides k-boundedness, marking reachability (if bounded)

Does not preserve states (state membership cannot be inferred)

Does not preserve branching properties nor liveness
Branching properties **not** preserved
Computing classes

Firing domains of classes are difference systems

Represented by Difference Bounds Matrices (DBM’s):

\[
\begin{align*}
 t_3 &< 6 & t_3 - \mu &< 6 & x - y &< \infty \\
 4 &< t_3 & t_4 - \mu &< -4 & \iota &< -2 \\
 2 &< t_4 & \iota - t_3 &< -2 & t_3 &< 3 \\
 t_3 - t_4 &< 3 & t_3 - t_4 &< 3 & t_3 &< 1 \\
 t_4 - t_3 &< 1 & t_4 - t_3 &< 1 & t_4 &< 0 \\
 \end{align*}
\]

Canonical forms (tightest constraints) computed in \(O(n^3)\)

\[\cong\] implemented as equality of canonical forms
\(O(n^2)\) **Firing rule** [Ro93, Vi01, BM03]

\((m, M)\) is the current class, \(M\) canonical.

- Transition \(f\) is firable iff \((\forall i \neq f)(-M_{if} \leq 0)\)

- The canonical \(M'\) at the target class \((m', M')\) is obtained by:
  
  \[
  M'_{00} = 0
  \]
  
  Foreach \(t\) enabled at \(m'\):
  \[
  M'_{tt} = 0
  \]
  
  if \(t\) is newly enabled then
  \[
  M'_{t0} = -\downarrow (I_s(t)), \ M'_0t = \uparrow (I_s(t))
  \]
  
  else
  \[
  M_{t0} = 0, \ M'_ot = M'_ft
  \]
  
  Foreach \(t'\) enabled at \(m'\): \(M'_{t0} = \min(M'_{t0}, M'_{tt'})\)

  Foreach \(t\) enabled at \(m'\)
  
  Foreach \(t' \neq t\) enabled at \(m'\)
  
  if \(t\) or \(t'\) is newly enabled
  
  then \(M'_{tt'} = M'_{t0} + M'_{ot'}\)
  
  else \(M'_{tt'} = \min(M_{tt'}, M'_{t0} + M'_{ot'})\)
Checking boundedness

Sufficient conditions for boundedness:

No \( c = (m, D) \) and \( c' = (m', D') \) such that:

1. \( c' \) reachable from \( c \)
2. \( m' \geq m \land m' \neq m \)
3. \( D' = D \)
4. \( (\forall p)(m'(p) > m(p) \Rightarrow m'(p) \geq \max_t \{\text{Pre}(p, t)\}) \)

But not necessary:

35
LTL model checking of Time Petri Nets

Obtaining a Kripke transition system:

- Build the SCG
- Add loops to deadlock states
- Add loops to temporarily diverging states (those at which all enabled transitions have unbounded intervals)

Atomic properties are the places marked and transitions fired

Check property (standard):

- Synchronize KTS with Buchi automaton obtained from the negation of formula
- Find a strong connected component containing an accepting state (of the automaton)

Check can be done on the fly while building the SCG
Preserving markings only \((SCG_{\subseteq})\)

If \(Sol(D) = Sol(D')\) then \((m, D)\) and \((m, D')\) have same futures

If \(Sol(D) \subseteq Sol(D')\) then any schedule firable from \((m, D)\) is firable from \((m, D')\), so we won’t find new markings by storing \((m, D)\)

\(SCG_{\subseteq} = SCG\) except a class is identified with any including it

Preserves markings but NOT firing sequences

Often much smaller than \(SCG\)
Example: Level crossing

<table>
<thead>
<tr>
<th>(n trains)</th>
<th>SCG</th>
<th>SSG ≤</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Edges</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU(s)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1 train)</td>
<td>11</td>
<td>10</td>
</tr>
<tr>
<td>(2 trains)</td>
<td>14</td>
<td>13</td>
</tr>
<tr>
<td>(3 trains)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>(4 trains)</td>
<td>123</td>
<td>37</td>
</tr>
<tr>
<td>(5 trains)</td>
<td>218</td>
<td>74</td>
</tr>
<tr>
<td>(3 trains)</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>(4 trains)</td>
<td>3101</td>
<td>172</td>
</tr>
<tr>
<td>(5 trains)</td>
<td>7754</td>
<td>492</td>
</tr>
<tr>
<td>(6 trains)</td>
<td>0.07</td>
<td>0.01</td>
</tr>
<tr>
<td>(7 trains)</td>
<td>134501</td>
<td>1175</td>
</tr>
<tr>
<td>(8 trains)</td>
<td>436896</td>
<td>4534</td>
</tr>
<tr>
<td>(9 trains)</td>
<td>5.85</td>
<td>0.07</td>
</tr>
<tr>
<td>(10 trains)</td>
<td>8557621</td>
<td>10972</td>
</tr>
<tr>
<td>(11 trains)</td>
<td>34337748</td>
<td>53766</td>
</tr>
<tr>
<td>(12 trains)</td>
<td>1254.92</td>
<td>1.20</td>
</tr>
</tbody>
</table>
Strong state classes

1. Background

2. State Class graphs as abstract state spaces

3. State Classes Preserving markings and traces

4. Preserving states and traces

5. Preserving states and branching properties

6. Quantitative properties, Other techniques

7. Subclasses, extensions, alternatives

8. Application areas, Tools
**Strong State classes**

**SCG:**

Do not preserve branching properties (no AE)
Cannot decide state reachability ($\cong$ too coarse)

Let:

$$C = \bigcup_{\sigma \in T^*} \{C_\sigma\}, \text{ where } C_\epsilon = \{s_0\}, \quad C_{\sigma,t} = \{s | (\exists s' \in C_\sigma) (s' \xrightarrow{t} s)\}$$

Then: [BV03]

$$SSCG = (C, \xrightarrow{t}, \{s_0\})$$
Clocks, equivalence \(\equiv\)

Clock systems

\[ \gamma_t = \text{time elapsed since } t \text{ was last enabled} \]

Clock vector \(\gamma\) denotes the interval \(I\) such that \((\forall t)(I(t) = Is(t) - \gamma_t)\)

NOTE: infinitely many clock vectors may denote the same state

Strong Classes

Represented by a marking and a clock system

\((m, G\gamma \leq g)\) denotes a set of states

Clock system equivalence

\((m, Q) \equiv (m', Q')\) iff they denote the same set of states

special case: If all transitions have bounded static intervals

Then \((m, Q) \equiv (m', Q') \iff m = m' \land Sol(Q) = Sol(Q')\)
Computing Strong State Classes

Algorithm 2: Computes $C_{\sigma,t} = (m', Q')$ from $C_{\sigma} = (m, Q)$:

- $C_\emptyset = (m_0, \{0 \leq \gamma_t \leq 0 \mid \text{Pre}(t) \leq m_0\})$

- $t$ is firable from some state of $C_{\sigma}$ iff:
  
  (i) $m \geq \text{Pre}(t)$ \hspace{1em} (t is enabled at $m$)
  
  (ii) $Q$ augmented with the following is consistent:
  
  $0 \leq \theta$
  
  $\downarrow I_s(t) \leq \gamma_t + \theta$
  
  $\{\theta + \gamma_i \leq \uparrow I_s(i) \mid m \geq \text{Pre}(i)\}$

- If so, then $m' = m - \text{Pre}(t) + \text{Post}(t)$, and $Q'$ is obtained by:
  
  1. add inequations (ii) to $Q$;
  2. \forall $i$ enabled at $m'$, add $\gamma'_i$ and inequations:
     
     $\gamma'_i = \gamma_i + \theta$, if $i \neq t$ and $m - \text{Pre}(t) \geq \text{Pre}(i)$
     
     $0 \leq \gamma'_i \leq 0$, otherwise
  3. Eliminate variables $\gamma$ and $\theta$

- $(m, Q) \equiv (m', Q')$ iff $m = m'$ and $Q$ and $Q'$ have equal solution sets
Example

\begin{align*}
C_0 &= (p_1, p_2, \{0 \leq t_{10}\}) \\
C_1 &= (p_3, p_4, \{0 \leq t_2 \leq 0, 0 \leq t_3 \leq 0, 0 \leq t_4 \leq 0\}) \\
C_2 &= (p_2, p_3, \{0 \leq t_3 \leq 4, 0 \leq t_4 \leq 4, t_3 - t_4 \leq 0, t_4 - t_3 \leq 0\}) \\
C_3 &= (p_2, p_3, \{0 \leq t_3 \leq 0, 0 \leq t_4 \leq 0\}) \\
C_4 &= (p_3, p_4, \{3 \leq t_2 \leq 4, 0 \leq t_3 \leq 0, 0 \leq t_4 \leq 0\}) \\
C_5 &= (p_2, p_3, \{0 \leq t_3 \leq 1, 0 \leq t_4 \leq 1, t_3 - t_4 \leq 0, t_4 - t_3 \leq 0\})
\end{align*}
Handling Unbounded Intervals

Problem: If $\equiv$ implemented as said, then $SSCG$ may be infinite

Given $C \in \mathbb{R}_{\geq 0}^4 = (m_0, 0 \leq \gamma_t \leq 0, 0 \leq \gamma_t \leq 0)$

But $C_{(t_0)} \equiv (m_0, 0 \leq \gamma_t \leq 0, 0 \leq \gamma_t \leq 0)$

Solution: Relax clock systems in Strong Classes

$\hat{Q}$ obtained by, recursively:

- Partition $Q$ by $\gamma_k \geq Eft_s(k)$, for $k$ s.t. $Lft_s(k) = \infty$
- In half space $\gamma_k \geq Eft_s(k)$, relax upper bound of $\gamma_k$

Theorem:

$(m, Q) \equiv (m', Q')$ iff $m = m'$ and $Sol(\hat{Q}) = Sol(\hat{Q}')$
Implementations

Assume $Q$ denotes the set of states $E$

**Relaxation** [BV03]:

computes the largest set of clock vectors denoting set $E$

fragments classes ($\hat{Q}$ is not convex)

**Normalization** [Had06]:

compute the largest clock DBM denoting set $E$

faster, avoids fragmentation
Properties

SSCG Finite iff the $TPN$ is bounded

Preserves EA, hence firing sequences ($LTL$)

Decides k-boundedness, marking and state reachability (if bounded)

Does not preserve branching properties nor liveness
Analysis with the \textit{SSCG}

\textbf{Checking state reachability} (in the DSG)

From $s = (m, I)$, compute the smallest $\gamma$ such that

$$(\forall t \in \mathcal{E}(m))(I(t) = I_s(t) - \gamma_t)$$

Then $s$ is reachable if $\gamma$ belongs to some (relaxed) strong class

\textbf{LTL model checking with the \textit{SSCG}}

As for the SCG

But SCG is a better choice since typically smaller

\textbf{Checking boundedness}

As for the SCG
Computation of the SSCG

Clock domains of classes are difference systems (DBM’s)

Same complexity as SCG for class computations ($O(n^2)$)

≡ implemented as equality of canonical forms
after relaxation or normalization
Preserving states only, $SSCG_{\subseteq}$

Similar to the SCG:

If $Sol(Q) \subseteq Sol(Q')$ then any schedule firable from $(m, Q)$ is firable from $(m, Q')$, so we won’t find new states by storing $(m, Q)$

$SSCG_{\subseteq} = SSCG$ except a class is identified with any including it

Preserves states but NOT firing sequences

Often much smaller than $SSCG$
Example: Level crossing

<table>
<thead>
<tr>
<th></th>
<th>SCG</th>
<th>SSCG</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1 train)</td>
<td>Classes</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Edges</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>CPU(s)</td>
<td>0.00</td>
</tr>
<tr>
<td>(2 trains)</td>
<td>Classes</td>
<td>123</td>
</tr>
<tr>
<td></td>
<td>Edges</td>
<td>218</td>
</tr>
<tr>
<td></td>
<td>CPU(s)</td>
<td>0.00</td>
</tr>
<tr>
<td>(3 trains)</td>
<td>Classes</td>
<td>3101</td>
</tr>
<tr>
<td></td>
<td>Edges</td>
<td>7754</td>
</tr>
<tr>
<td></td>
<td>CPU(s)</td>
<td>0.07</td>
</tr>
<tr>
<td>(4 trains)</td>
<td>Classes</td>
<td>134501</td>
</tr>
<tr>
<td></td>
<td>Edges</td>
<td>436896</td>
</tr>
<tr>
<td></td>
<td>CPU(s)</td>
<td>5.85</td>
</tr>
<tr>
<td>(5 trains)</td>
<td>Classes</td>
<td>8557621</td>
</tr>
<tr>
<td></td>
<td>Edges</td>
<td>34337748</td>
</tr>
<tr>
<td></td>
<td>CPU(s)</td>
<td>1254.92</td>
</tr>
</tbody>
</table>
State Classes

1. Background

2. State Class graphs as abstract state spaces

3. State Classes Preserving markings and traces

4. Preserving states and traces

5. Preserving states and branching properties

6. Quantitative properties, Other techniques

7. Subclasses, extensions, alternatives

8. Application areas, Tools
SSCG analysis

Satisfies EA (hence preserves LTL) but not AE

Does not preserve branching properties

**ASCG**: (Atomic State class graph revisited):

Start from the SSCG or $\text{SSCG}_\subseteq$

Enforce AE using partition refinement

The ASCG and DSG will be bisimilar

First such construction proposed in [YR98] (Atomic state classes)
Consider a structure \((P, \rightarrow)\) and two subsets \(A\) and \(B\) of \(P\)

\(A\) is **Stable** wrt \(B\) if no \(s \in A\) has a successor in \(B\) or all have one.

\(B^{-1} = \{A | A \rightarrow B\}\)

Partitions \((P, \rightarrow)\) according to bisimulation:

\[
Q = P \\
\text{while } (\exists A, B \in Q)(A \text{ is not Stable wrt } B) \\
\text{do replace } A \text{ by } A_1 = A \cap B^{-1} \text{ and } A_2 = A - B^{-1}
\]
Revisited Atomic state classes

$SCG$ inadequate as initial partition (too coarse)

$SSCG$ or $SSCG_{\subseteq}$ are adequate

Algorithm 3

Start from the $SSCG$ [BV03] (or $SSCG_{\subseteq}$ [BH04])

while some class $c$ is unstable wrt one of its successor classes $c'$

do partition $c$ such that is stable wrt $c'$

Collect all classes reachable from the initial one
Partitioning SSCG classes

Partition Technique

If \( c = (m, Q) \xrightarrow{t} c' \) and \( c \) is unstable wrt \( c' \) then some constraint \( \rho \) is:

- **necessary** for \( s \in c \) to have a successor in \( c' \)
- **nonredundant** in \( Q \)

\( c \) is partitionned into \((m, Q \cap \{\rho\}), (m, Q \cap \{\neg\rho\})\):

Computing \( \rho \) constraints

Compute predecessors \( P \) by \( t \) of states in \( c' \) (by reverse SSCG rule)

\( Q \) is stable iff \( Sol(Q) \subseteq Sol(P) \)

Otherwise take any constraint of \( P \) nonredundant in \( Q \)
Example 1

\[ C_0 = (p_0 \ p_3, \{0 \leq t_0 \leq 0, 0 \leq t_2 \leq 0\}) \]
\[ C_1 = (p_1 \ p_3, \{0 \leq t_1 \leq 0, 1 \leq t_2 \leq 3\}) \]
\[ C_2 = (p_2 \ p_3, \{3 \leq t_2 \leq 6}\) \]
\[ C_3 = (p_2 \ p_4, \{\}) \]
\[ C_4 = (p_1 \ p_4, \{1 \leq t_1 \leq 4\}) \]
\[ C_5 = (p_1 \ p_3, \{0 \leq t_1 \leq 0, 0 \leq t_2 < 1\}) \]
\[ C_6 = (p_1 \ p_3, \{0 \leq t_1 \leq 0, 3 < t_2 \leq 4\}) \]
Example 2
Properties:

Finite iff the $TPN$ is bounded

Abstraction preserves states and firing sequences ($LTL$)

Decides $k$-boundedness, marking and state reachability

Refinement restores AE, hence ASCG preserve branching properties and liveness (suitable for CTL model checking)

Notes:

- ASCG is a cover rather than a partition $\Rightarrow$ not minimal
- ASCG is bisimilar to the DSG, but not to the SG
Liveness analysis

Theorem: A TPN is live if each of its transitions labels some arc in all pending SCCs of its ASCG.
### Example: Level crossing

<table>
<thead>
<tr>
<th></th>
<th>SCG</th>
<th>SSCG</th>
<th>ASCG</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>(1 train)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classes</td>
<td>11</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>Edges</td>
<td>14</td>
<td>14</td>
<td>15</td>
</tr>
<tr>
<td>CPU(s)</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td><strong>(2 trains)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classes</td>
<td>123</td>
<td>141</td>
<td>192</td>
</tr>
<tr>
<td>Edges</td>
<td>218</td>
<td>254</td>
<td>844</td>
</tr>
<tr>
<td>CPU(s)</td>
<td>0.00</td>
<td>0.00</td>
<td>0.02</td>
</tr>
<tr>
<td><strong>(3 trains)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classes</td>
<td>3101</td>
<td>5051</td>
<td>6966</td>
</tr>
<tr>
<td>Edges</td>
<td>7754</td>
<td>13019</td>
<td>49802</td>
</tr>
<tr>
<td>CPU(s)</td>
<td>0.07</td>
<td>0.13</td>
<td>2.24</td>
</tr>
<tr>
<td><strong>(4 trains)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classes</td>
<td>134501</td>
<td>351271</td>
<td>356940</td>
</tr>
<tr>
<td>Edges</td>
<td>436896</td>
<td>1193376</td>
<td>3447624</td>
</tr>
<tr>
<td>CPU(s)</td>
<td>5.85</td>
<td>20.14</td>
<td>291.478</td>
</tr>
<tr>
<td><strong>(5 trains)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Classes</td>
<td>8557621</td>
<td>35945411</td>
<td>23081275</td>
</tr>
<tr>
<td>Edges</td>
<td>34337748</td>
<td>151908273</td>
<td>279572133</td>
</tr>
<tr>
<td>CPU(s)</td>
<td>1254.92</td>
<td>7439.25</td>
<td>54:30:07</td>
</tr>
</tbody>
</table>
State Classes

1. Background
2. State Class graphs as abstract state spaces
3. State Classes Preserving markings and traces
4. Preserving states and traces
5. Preserving states and branching properties
6. Quantitative properties, Other techniques
7. Subclasses, extensions, alternatives
8. Application areas, Tools
6. Quantitative properties, Other techniques

Checking “Timed” properties

Path analysis

State classes % alternative techniques
Checking “Timed” properties

Model checkers for timed logics:

- e.g. Romeo, technique adapted from Timed Automata

Observers technique:

Reduce property to reachability using an observer composed with TPN

- e.g. $t_1$ fires at most 8 ut after $t_0 \Rightarrow$ no reachable marking marks $BAD$:

A large class of formulas can be reduced to reachability
Path Analysis

Problem:

Given a firing sequence $\sigma$:

- Characterize firing schedules over $\sigma$
- Check existence of time constrained schedules
- Find fastest/slowest schedule

...
Computing Path Systems

As for SSCG, but without elimination of $\theta$:

Algorithm 4: Computes $K_{\sigma,t} = (m', Q')$ from $K_{\sigma} = (m, Q)$:

- $K_{\epsilon} = (m_0, \{0 \leq \gamma_t \leq 0 \mid \text{Pre}(t) \leq m_0\})$

- $t$ is firable from some state of $K_{\sigma}$ iff:
  (i) $m \geq \text{Pre}(t)$ (t is enabled at $m$)
  (ii) $Q$ augmented with the following is consistent:
    \[
    \begin{align*}
    0 &\leq \theta \\
    \downarrow I_s(t) &\leq \gamma_t + \theta \\
    \{\theta + \gamma_i &\leq \uparrow I_s(i) \mid m \geq \text{Pre}(i)\}
    \end{align*}
    \]

- If so, then $m' = m - \text{Pre}(t) + \text{Post}(t)$, and $Q'$ is obtained by:
  1. add inequations (ii) to $Q$;
  2. $\forall i$ enabled at $m'$, add $\gamma'_i$ and inequations:
    \[
    \gamma'_i = \gamma_i + \theta, \text{ if } i \neq t \text{ and } m - \text{Pre}(t) \geq \text{Pre}(i)
    \]
    \[
    0 \leq \gamma'_i \leq 0, \text{ otherwise}
    \]
  3. Eliminate variables $\gamma$
Path Systems ...

\( K_\sigma \) Links firing times along \( \sigma \) with state reached

\[
P(\theta|\gamma) \leq p
\]

Projecting on \( \theta \) yields path system

\[
T(\theta) \leq t
\]

Characterizes times at which transitions can fire along \( \sigma \)

in delays (\( \theta \), relative times)

or dates (\( \delta \), absolute times) using:

\[
\delta_i = \theta_1 + \ldots + \theta_i
\]
Tools ...

Implementation: PLAN/TINA

  Computes all paths (system) or one path

  In delays or dates

Applications:

  Path analysis (existence, fastest, . . . )

  Timing counter-examples returned by LTL modelchecker
Alternative methods

Essential states methods
  easier implementation
  build nondeterministic graphs (may be much smaller than deterministic)
  preserve LTL
  no open intervals

  sensitive to scaling of intervals (may blow up)

Unfolding methods
  mature for untimed nets

  some progress for dense timed systems

Translation into Timed Automata
  Structural translation [CR06] preserves weak timed bisimilarity

  Provided by Roméo toolbox
State classes % Essential states

<table>
<thead>
<tr>
<th></th>
<th>SCG</th>
<th>ES</th>
<th>ES + delays</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1 train)</td>
<td>11</td>
<td>13</td>
<td>24</td>
</tr>
<tr>
<td>Classes</td>
<td>11</td>
<td>13</td>
<td>24</td>
</tr>
<tr>
<td>Edges</td>
<td>14</td>
<td>27</td>
<td>37</td>
</tr>
<tr>
<td>CPU (s)</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>(2 trains)</td>
<td>123</td>
<td>116</td>
<td>203</td>
</tr>
<tr>
<td>Classes</td>
<td>123</td>
<td>116</td>
<td>203</td>
</tr>
<tr>
<td>Edges</td>
<td>218</td>
<td>382</td>
<td>378</td>
</tr>
<tr>
<td>CPU (s)</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>(3 trains)</td>
<td>3101</td>
<td>1550</td>
<td>2299</td>
</tr>
<tr>
<td>Classes</td>
<td>3101</td>
<td>1550</td>
<td>2299</td>
</tr>
<tr>
<td>Edges</td>
<td>7754</td>
<td>5823</td>
<td>5294</td>
</tr>
<tr>
<td>CPU (s)</td>
<td>0.07</td>
<td>0.03</td>
<td>0.03</td>
</tr>
<tr>
<td>(4 trains)</td>
<td>134501</td>
<td>22268</td>
<td>28895</td>
</tr>
<tr>
<td>Classes</td>
<td>134501</td>
<td>22268</td>
<td>28895</td>
</tr>
<tr>
<td>Edges</td>
<td>436896</td>
<td>91256</td>
<td>81142</td>
</tr>
<tr>
<td>CPU (s)</td>
<td>5.85</td>
<td>0.671</td>
<td>0.600</td>
</tr>
<tr>
<td>(5 trains)</td>
<td>8557621</td>
<td>313214</td>
<td>372475</td>
</tr>
<tr>
<td>Classes</td>
<td>8557621</td>
<td>313214</td>
<td>372475</td>
</tr>
<tr>
<td>Edges</td>
<td>34337748</td>
<td>1397517</td>
<td>1245566</td>
</tr>
<tr>
<td>CPU (s)</td>
<td>1254.92</td>
<td>15.12</td>
<td>12.92</td>
</tr>
<tr>
<td></td>
<td>SCG</td>
<td>ES</td>
<td>ES + delays</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>-----</td>
<td>-------------</td>
</tr>
<tr>
<td>(1 train)</td>
<td>11</td>
<td>25</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>123</td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>(2 trains)</td>
<td>123</td>
<td>564</td>
<td>3110</td>
</tr>
<tr>
<td></td>
<td>218</td>
<td>8154</td>
<td>6107</td>
</tr>
<tr>
<td></td>
<td>0.00</td>
<td>0.03</td>
<td>0.02</td>
</tr>
<tr>
<td>(3 trains)</td>
<td>3101</td>
<td>27950</td>
<td>119479</td>
</tr>
<tr>
<td></td>
<td>7754</td>
<td>315629</td>
<td>273782</td>
</tr>
<tr>
<td></td>
<td>0.07</td>
<td>1.65</td>
<td>1.48</td>
</tr>
<tr>
<td>(4 trains)</td>
<td>134501</td>
<td>1680212</td>
<td>5785743</td>
</tr>
<tr>
<td></td>
<td>436896</td>
<td>18328768</td>
<td>15813462</td>
</tr>
<tr>
<td></td>
<td>5.85</td>
<td>133.09</td>
<td>114.61</td>
</tr>
<tr>
<td>(5 trains)</td>
<td>8557621</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>34337748</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>1254.92</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
State Classes

1. Background

2. State Class graphs as abstract state spaces

3. State Classes Preserving markings and traces

4. Preserving states and traces

5. Preserving states and branching properties

6. Quantitative properties, Other techniques

7. Subclasses, extensions, alternatives

8. Application areas, Tools
7. Subclasses, extensions, alternatives

7.1. Subclasses

7.2. Extensions

Open time intervals
Inhibitor arcs, read arcs, flush arcs
Priorities
Stopwatches
High level notations – Time transition systems

7.3. Other models for real-time systems

The variety of TPN’s
Timed Automata
**Subclasses**

All intervals singular (reduced to a point)

> have finite state spaces

All intervals unbounded

> state class graph = marking graph

Poor expressiveness
7. Subclasses, extensions, alternatives

7.1. Subclasses

7.2. Extensions

   Open time intervals

   Inhibitor arcs, read arcs, flush arcs

   Multi-enabledness

   Priorities

   Stopwatches

   High level notations – Time transition systems

7.3. Other models for real-time systems

   The variety of TPN's

   Timed Automata
“Light” extensions

Open time intervals

- e.g. $[1, 3]$ $[3, 6]$ $[4, 5]$ $[6, \infty[$

Read arcs, Inhibitor arcs:

- Do not transfer tokens
- Positive ($m(p) \geq k$) or Negative ($m(p) < k$) conditions
- Only impacts enabledness (and resets of intervals)

Flush arcs:

- Transfer as many tokens as found in the source place
- Only impacts computation of markings

$\Rightarrow$ Can be handled
Multi-enabledness

\[ t \text{ is } k\text{-enabled at } m \text{ if } m \geq k \times \text{Pre}(t) \quad (k \geq 0) \]

So far: One temporal variable per transition, whether or not multi-enabled (single-server semantics)

Consider: If \( t \) is \( k \)-enabled, then \( k \) temporal variables associated with \( t \) (multi-server semantics)

Instances considered independent or not (e.g. oldest fires first)

\[ \Rightarrow \text{State class constructions can be adapted} \]
Multi-enabledness example (oldest fires first SCG)
Time Petri nets with Priorities ($PrTPN$)

$\langle P, T, Pre, Post, m_0, I_s, \succ \rangle$ in which:

- $\langle P, T, Pre, Post, m_0 \rangle, I^+$ is a Time Petri net

- $\succ \subseteq T \times T$ is the Priority relation

$\succ$ assumed irreflexive, asymmetric and transitive
Semantics

- Initial state: \((m_0, I_{s_0})\)

- discrete transitions: \((m, I) \xrightarrow{t} (m', I')\) iff \(t \in T\) and
  
  1. \(m \geq \text{Pre}(t)\)
  2. \(0 \in I(t)\)
  3. \((\forall k \in T)(m \geq \text{Pre}(k) \land 0 \in I(k) \Rightarrow \neg (k \succ t))\)
  4. \(m' = m - \text{Pre}(t) + \text{Post}(t)\)
  5. \((\forall k \in T)(m' \geq \text{Pre}(k) \Rightarrow I'(k) = \text{if } k \neq t \land m - \text{Pre}(t) \geq \text{Pre}(k) \text{ then } I(k) \text{ else } I_{s}(k))\)

- continuous transitions: \((m, I) \xrightarrow{d} (m, I')\) iff
  
  \((\forall k \in T)(m \geq \text{Pre}(k) \Rightarrow d \leq \uparrow I(k) \land I'(k) = I(k) \downarrow d)\)
Expressiveness

In terms of timed language acceptance:

\[ TPN = TA \] [BCHRL05, BHR06]

In terms of weak timed bisimulation:

\[ TPN < TA \] [CR06]

\[ TPN = TA^- \] [BCHRL05]

\[ TA^+\{\leq, \land\} = PrTPN \text{ with right-closed or unbounded intervals} \] [BPV06]

Note: Priorities enable compositional design
Priorities add expressiveness to $TPN$
Priorities add expressiveness to $TPN$
Priorities add expressiveness to $TPN$
Priorities add expressiveness to $TPN$
Double click $TA$

t3: $x=1$, simple, $x:=0$

t1: true, click, $x:=0$

t5: $x=1$, double, $x:=0$

t2: $x<1$, click

t4: $x<1$, click
Not quite double click in $TPN$
At time 1:

Incorrect: simple enabled
Double click in *PrTPN*
SCG and priorities

Founding observation for \( SCG \):

**Classes equivalent by \( \simeq \) have same future**

Is no more true with priorities:

Firing \( t_0 \) or \( t_1 \) leads to equal classes
but \( t_2 \) may fire only if less than 1 unit of time elapsed ... 

\( \Rightarrow \) \( SCG \) inapplicable
Computing Strong State Classes with priorities

Algorithm 2: Computes $C_{\sigma,t} = (m',Q')$ from $C_\sigma = (m,Q)$:

- $C_\epsilon = (m_0, \{0 \leq \gamma_t \leq 0 \mid \text{Pre}(t) \leq m_0\})$

- $t$ is firable from some state of $C_\sigma$ iff:
  
  (i) $m \geq \text{Pre}(t)$ (t is enabled at $m$)

  (ii) $Q$ augmented with the following is consistent:

  \[
  0 \leq \theta \\
  \downarrow I_s(t) \leq \gamma_t + \theta \\
  \{\theta + \gamma_i \leq \uparrow I_s(i) \mid m \geq \text{Pre}(i)\} \\
  \{\theta + \gamma_j < \uparrow I_s(j) \mid m \geq \text{Pre}(j) \land j \succ t\}
  \]

- If so, then $m' = m - \text{Pre}(t) + \text{Post}(t)$, and $Q'$ is obtained by:
  1. add inequations (ii) to $Q$;
  2. $\forall i$ enabled at $m'$, add $\gamma'_i$ and inequations:

     \[
     \gamma'_i = \gamma_i + \theta, \text{ if } i \neq t \text{ and } m - \text{Pre}(t) \geq \text{Pre}(i) \\
     0 \leq \gamma'_i \leq 0, \text{ otherwise}
     \]
  3. Eliminate variables $\gamma$ and $\theta$
**Updated firability conditions**

Firability conditions (ii) rephrased:

(ii.1) \( \theta \geq 0 \)

(ii.2) \( \theta + \gamma_t \in I_s(t) \)

(ii.3) \( (\forall i \neq t)(m \geq \text{Pre}(i) \Rightarrow \theta + \gamma_i \leq \uparrow I_s(i)) \)

(ii.4) \( (\forall j)(m \geq \text{Pre}(j) \land j > t \Rightarrow \theta + \gamma_j \not\in I_s(j)) \)

In (ii.4):

\[
\theta + \gamma_i \not\in I_s(i) \Leftrightarrow \theta + \gamma_j < \downarrow I_s(j) \lor \theta + \gamma_j > \uparrow I_s(j)
\]

But last subcondition would contradict (ii.3), hence:

\[
\theta + \gamma_i \not\in I_s(i) \Leftrightarrow \theta + \gamma_j < \downarrow I_s(j)
\]

Hence no cost penalties \( (O(n^2)) \)

(No \( O(n^4) \) polyhedra differences required)
Modeling temporal preemption

Why

Verification of task scheduling in realtime systems (e.g. Avionics)

How

Scheduling extended TPNs [LR03]
Preemptive TPNs [BFSV04]
TPNs with inhibitor hyperarcs [RL04]
Stopwatch Time Petri Nets [BLRV07]
Time Petri nets with Stopwatches ($SwTPN$)

[BLRV07]

$\langle P, T, Pre, Sw, Post, m_0, Is \rangle$ in which:

- $\langle P, T, Pre, Post, m_0 \rangle, I^+$ is a Time Petri net

- $Sw$ is the *Stopwatch incidence function*

An enabled transition is either *Active* or *Suspended*
Semantics

- Initial state: \((m_0, I_0)\)

- discrete transitions: \((m, I) \xrightarrow{t} (m', I')\) iff \(t \in T\) and
  1. \(m \geq \text{Pre}(t) \land m \geq \text{Sw}(t)\)
  2. \(0 \in I(t)\)
  3. \(m' = m - \text{Pre}(t) + \text{Post}(t)\)
  4. \((\forall k \in T) (m' \geq \text{Pre}(k) \Rightarrow I'(k) = \text{if } k \neq t \land m - \text{Pre}(t) \geq \text{Pre}(k) \text{ then } I(k) \text{ else } I_s(k))\)

- continuous transitions: \((m, I) \xrightarrow{d} (m, I')\) iff
  \[(\forall k \in T) (m \geq \text{Pre}(k) \Rightarrow d \leq \uparrow I(k) \land I'(k) = \text{if } m \geq \text{Sw}(k) \text{ then } I(k) - d \text{ else } I(k))\]
State classes

All state class constructions remain applicable, but

- May yield infinite graphs, even for bounded nets
- In fact: state reachability with stopwatches is undecidable

Overapproximations of state spaces

- Identify state spaces containing the exact one
- Finite iff the net is bounded
- Yield sufficient conditions for verification
Counters can be encoded as phase differences between two periodic events

Any 2-counter machine can be encoded into a safe (1-bounded) SwTPN with:

- A single stopwatch arc
- A single transition with non singular interval

Hence:

State/marking reachability undecidable for bounded SwTPN

k-boundedness undecidable for SwTPN
Overapproximations

exact polyhedra $\subseteq$ quantized polyhedra $\subseteq$ smallest enclosing DBM
Example, task system \[\text{[BFSV04]}\]

(observer in grey for the property “task 3 achieved in \(\leq 96s\)”

\[\text{[BFSV04]}\]
More examples, scheduling policies

Round-Robin

Rate-monotonic
Handling Data

From Petri nets to Keller transition systems:

- markings $\Rightarrow$ vectors of integers
- “additives” transitions $\Rightarrow$ arbitrary transitions

Higher expressiveness but:

- reachability and boundedness undecidable

From Keller systems to Time transition systems:

- Time Transition System = Keller TS + temporal intervals
- State class techniques remain applicable
High level Notations

**Cotre Project** ([http://www.laas.fr/COTRE](http://www.laas.fr/COTRE))

Avionics software

**Cotre** language

**TOPCASED project** ([http://www.topcased.org](http://www.topcased.org))

Toolkit in OPen source for Critical Applications and SystEms Development

**Fiacre** language:

- intermediate form language for RTS;
- end-user formalisms (AADL, SDL, etc) translated into **Fiacre**;
- **Fiacre** programs translated into Tina and CADP input (mid 2008).
Fiacre example

type index is 0..3

type request is union get_sum, get_value of index end

type data is array 4 of nat

process ATM [req : in request, resp : out nat] is
    states ready, send_sum, send_value
    var c : request, i : index, sum : nat, val : data := [6, 2, 7, 9]
    init to ready
    from ready
        req ?c;
        case c of get_sum -> to send_sum
            | get_value (i) -> to send_value
        end
    from send_value
        resp !val[i]; to ready
    from send_sum
        sum, i := 0, 0;
        while i < 3 do sum, i := sum + val[i], i + 1 end;
        sum := sum + val[i];
        resp !sum;
    to ready

component C [p : in nat] (&X : read nat) is
    port q : none in [2, 8]
    var Y : bool := false
    par p -> C1 [p,q] (X, Y)
    || p -> C2 [p,q] (X, Y)
end
7. Subclasses, extensions, alternatives

7.1. Subclasses

7.2. Extensions

- Open time intervals
- Inhibitor arcs, read arcs, flush arcs
- Priorities
- Stopwatches
- High level notations – Time transition systems

7.3. Other models for real-time systems

- The variety of TPN's
- Timed Automata
The variety of TPNs

Intervals on transitions (TPNs)

- Oldest, and most widely used
- Established convenient analysis methods, tools available
- Good expressiveness
- Extensions available (priorities, stopwatches)

Intervals on places (p-TPNs)

- Tokens have age of creation attached
- Places bear intervals, filtering tokens according to their age

Intervals on arcs (Timed arcs TPNs)

- Tokens have age of creation attached
- Arcs from places bear intervals, filtering tokens according to age
- More expressive than above both

Some relative expressiveness results can be found in [BR06]
Timed Automata

Timed Automata

Without progress conditions
With progress conditions (invariants, urgency, etc)
Extensions available (priorities, stopwatches, linear hybrid, etc)
Widely used, extensively studied, tools available [Uppaal, Kronos, Hytech]

Same semantic model (timed transition systems)

TPN to TA translators available [Romeo]
Analyzing TPNs by translation into TAs
Adapting TA methods to TPNs (e.g. TCTL model checking)

Expressiveness

In terms of language acceptance: $TA = TPN$
In terms of weak timed bisimilarity: $TA > TPN$
But $TA + \{\leq, \wedge\} < PrTPN$
State Classes

1. Background

2. State Class graphs as abstract state spaces

3. State Classes Preserving markings and traces

4. Preserving states and traces

5. Preserving states and branching properties

6. Quantitative properties, Other techniques

7. Subclasses, extensions, alternatives

8. Application areas, Tools
8. Applications, Tools

8.1. Application areas

- Communication protocols (Merlin)
- Embedded software systems
- Hardware systems

8.2. Tools

- Some tools using state classes
- The TINA toolbox
Topcased Project

Meta-Modeller
Modelling Languages

Editors
PDL  AADL  SDL  UML2.0  SYSML  ...

Transformation Engine
ATL,KERMETA,...

Compilers
Model-Checkers
Simulator

Meta-Modelling
Model Transformation
Common Format
Translation
TINA  CADP  ...

Simulation & Formal Verification
Some tools (state class based)

Tina, http://www.laas.fr/tina

Oris, http://www.stlab.dsi.unifi.it/oris

Romeo, http://romeo.rts-software.org
TINA (TIme petri Net Analyzer)

Handles

- Time Petri Nets (+ read arcs, inhibitor arcs, open intervals)
- Priorities (Priority TPNs)
- Data (Time Transition Systems)
- Suspension/Resumption (Stopwatch TPNs)
- High level notations (Fiacre language, forthcoming)
State space abstractions

**Exact state spaces**

When possible . . .

**Managing combinatorial explosion**

Partial order methods (Covering steps, Stubborn/Persistent sets)

**Handling time constraints**

Finite abstractions by State Class methods

**Handling Suspension/Resumption**

State reachability undecidable ⇒ geometric overapproximations

**Handling Data**

High level description languages ⇒ discrete overapproximations
Main components

**tina** *(TIme petri NNet Aalyzer)*

- Input nets in graphical or textual form
- Builds behavior abstractions, Preserving some classes of properties
- Output in verbose form or for popular transition system analyzers

**nd**

- Graphic and textual editor
- Of Time Petri Net or Transition Systems
- Drawing, printing functions
- Interfaced with tina tool and selt model-checker

**struct, plan, setl, muse, ktzio, ndrio, ...**

- Structural analysis, path analysis, SE-LTL model-checker, converters ...
tina — exploration module
Untimed constructions

Covering graphs (Karp/Miller)

Detection of unbounded places, several heuristics

Marking graphs (Classical constructions)

Various stopping conditions
Liveness analysis

Partial order constructions (Classical constructions)

Covering steps
Stubborn sets
Stubborn steps
KTS, example

Marking graph

MARKINGS:
0 : p1 p2*2
1 : p3 p4 p5
2 : p2 p3 p5
3 : p2*2 p3
4 : p1 p2 p5
5 : p2 p3 p4
6 : p1 p2 p4
7 : p1 p4 p5

REACHABILITY GRAPH:
0 -> t1/1
1 -> t2/2, t3/5, t4/1,
2 -> t3/3, t4/2, t5/4
3 -> t4/3, t5/0
4 -> t3/0
5 -> t2/3, t4/5, t5/6
6 -> t2/0
7 -> t2/4, t3/6
Or in CADP format

```
des(0,17,8)
(0, "t1", 1)
(1, "t2", 2)
(1, "t3", 5)
(1, "t4", 1)
(1, "t5", 7)
(2, "t3", 3)
(2, "t4", 2)
(2, "t5", 4)
(3, "t4", 3)
(3, "t5", 0)
(4, "t3", 0)
(5, "t2", 3)
(5, "t4", 5)
(5, "t5", 6)
(6, "t2", 0)
(7, "t2", 4)
(7, "t3", 6)
```
Or in binary formats

Compact storage and exchange formats

**BCG** (CADP Toolbox, INRIA Grenoble)

Access to CADP tools

**KTZ** (Compressed Kripke Transition Systems)

State AND transition properties

  e.g. packs 135000 states and 450000 transitions into 1Mb
Timed constructions

State class graphs

Preserving markings (SCG⊆)

Preserving markings and LTL properties (SCG)

Multi-enabledness SCG

Preserving states (SSCG⊆)

Preserving states and LTL properties (SSCG)

Preserving CTL* properties (ASCG)
Model checking

Native *State/Event* – *LTL* model checker (*selt*)

Exports to external equivalence or model checkers (CADP, MEC)

Path analysis by the *plan* tool

In progress:

More native model-checkers (*μ*-calculus, MITL, . . . )

Parallel model checkers, for very large state spaces

High level descriptions (Fiacre)
# Some references

<table>
<thead>
<tr>
<th>Reference</th>
<th>Title</th>
</tr>
</thead>
</table>
Some references ...


### Some references …

<table>
<thead>
<tr>
<th>Reference</th>
<th>Title</th>
</tr>
</thead>
</table>