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

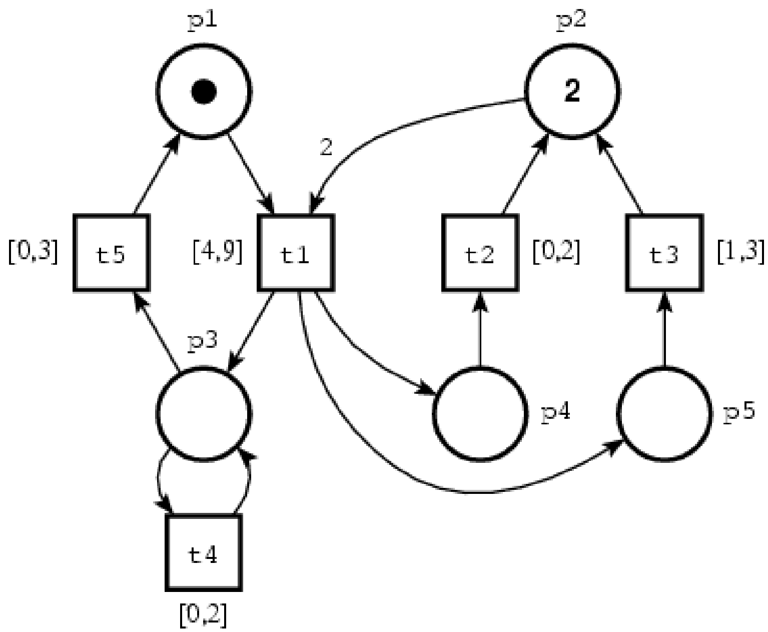
Dense semantics, state spaces

Representation of states – firing domains, clocks vectors

Basic theorems – decidability results

Logics

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

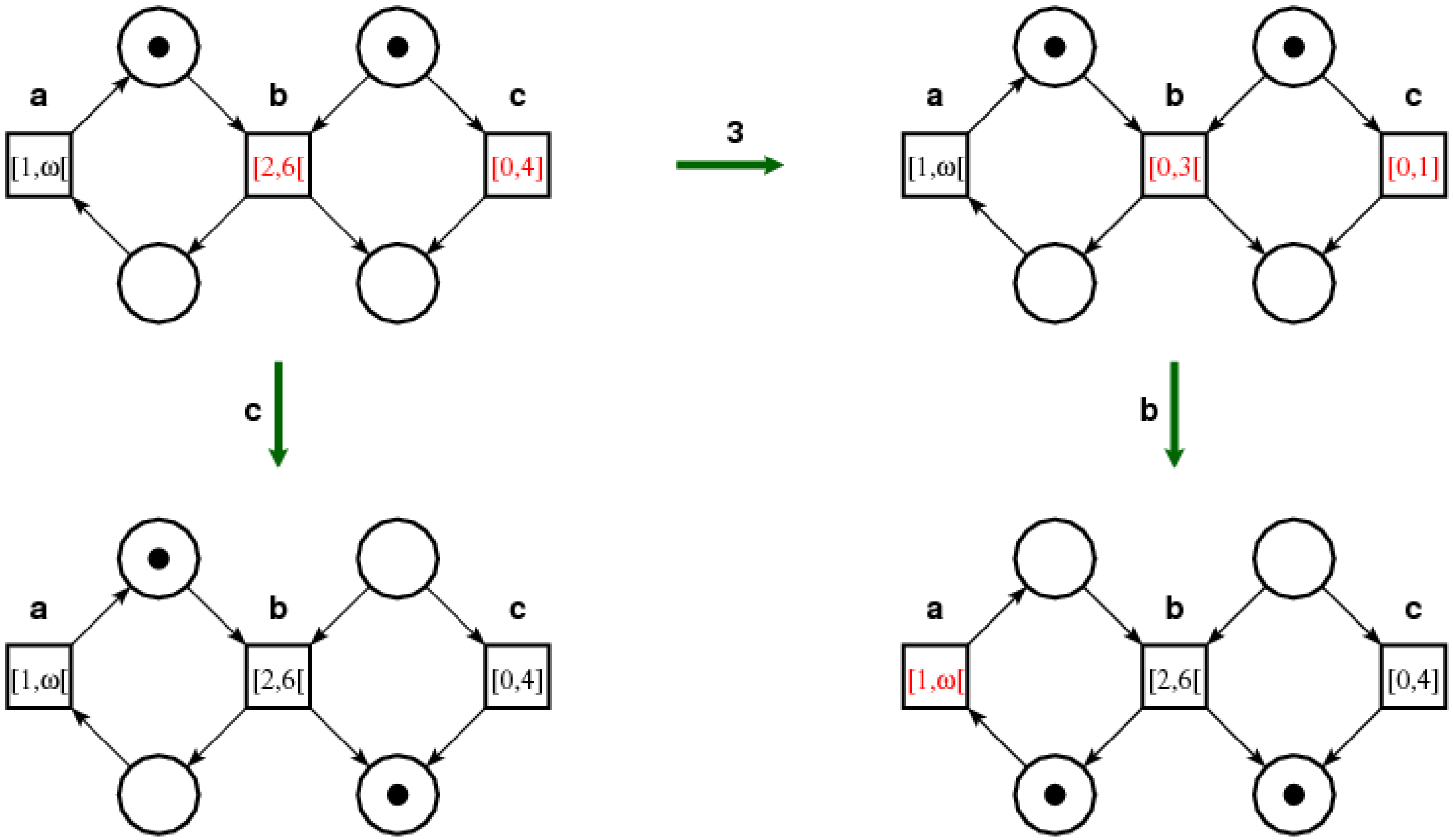


$(P, T, \mathbf{Pre}, \mathbf{Post}, m_0, I_s)$ where

- $(P, T, \mathbf{Pre}, \mathbf{Post}, m_0)$ is a Petri net
- I_s is the *Static Interval* Function

  $t \mapsto I_s(t) \subseteq \mathbb{R}^+$, 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, $t \in \mathcal{E}(m) \Leftrightarrow \mathbf{Pre}(t) \leq m$

$I, I', \ldots$: 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 ∞)

$\sigma, \sigma', \ldots$: sequences of transitions

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

$|\rho|$: *support* of ρ, $|\theta_1.t_1.\theta_2.t_2 \ldots| = t_1..t_2 \ldots$

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

$I \dot{-} \theta = \{x - \theta | x \geq \theta \wedge x \in I\}$: interval I ($I \subseteq \mathbb{R}^+$) shifted by θ 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 $s_0 = (m_0, Is \backslash \mathcal{E}(m_0))$

There are two sorts of transitions:

- discrete transitions: $(m, I) \overset{t}{\rightsquigarrow} (m', I')$ iff $t \in T$ and
  1. $m \geq \mathbf{Pre}(t)$
  2. $0 \in I(t)$
  3. $m' = m - \mathbf{Pre}(t) + \mathbf{Post}(t)$
  4. $(\forall k \in T)(m' \geq \mathbf{Pre}(k) \Rightarrow$
     $I'(k) = \ \textbf{if} \ \ k \neq t \ \wedge \ m - \mathbf{Pre}(t) \geq \mathbf{Pre}(k) \ \ \textbf{then} \ \ I(k) \ \ \textbf{else} \ \ Is(k))$
- continuous transitions: $(m, I) \overset{d}{\rightsquigarrow} (m, I')$ iff

  $(\forall k \in T)(m \geq \mathbf{Pre}(k) \Rightarrow d \leq\uparrow I(k) \wedge I'(k) = I(k) \ \dot{-} \ d)$

State spaces

With all continuous and discrete transitions:

$$SG = (S, \overset{t}{\rightsquigarrow} \cup \overset{d}{\rightsquigarrow}, s_0)$$

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:

$$DSG = (S, \xrightarrow{t}, s_0)$$

where

$$s \xrightarrow{t} s' \Leftrightarrow (\exists\theta)(\exists s'')(s \overset{\theta}{\rightsquigarrow} s'' \wedge s'' \overset{t}{\rightsquigarrow} s')$$

State graphs are typically infinite, dense.

Direct "Discrete" semantics (DSG)

Let $s \xrightarrow{t@\theta} s' \Leftrightarrow (\exists s'')(s \overset{\theta}{\rightsquigarrow} s'' \wedge s'' \overset{t}{\rightsquigarrow} s')$

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

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

1. $\mathbf{Pre}(t) \leq m$ (t is enabled at m)

   $\theta \geq \downarrow I(t)$

   $(\forall k)(\mathbf{Pre}(k) \leq m \Rightarrow \theta \leq \uparrow I(k))$

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

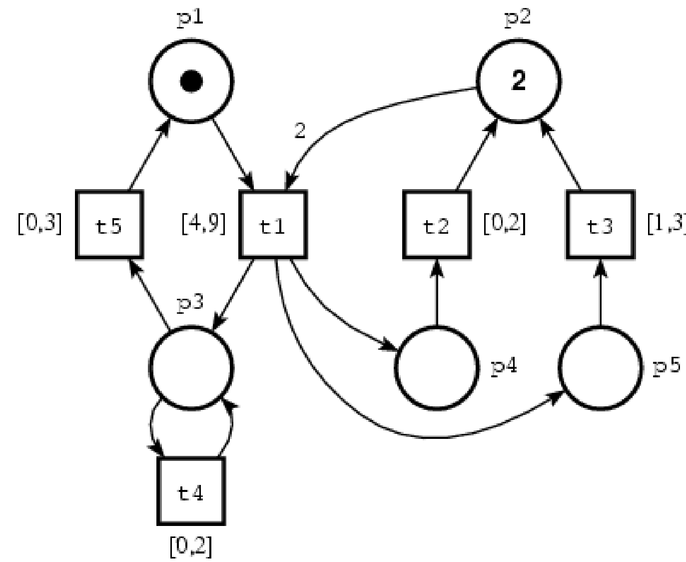
3. $(\forall k)(\mathbf{Pre}(k) \leq m \Rightarrow I'(k) =$

   if $k \neq t \wedge m - \mathbf{Pre}(t) \geq \mathbf{Pre}(k)$

   then $I(k) \dot{-} \theta$

   else $I_S(k))$

Example



$E_0 = (m_0, I_0)$

m_0 : $p_1, p_2(2)$
I_0 : solutions in t_1 of

$$4 \leq t_1 \leq 9$$

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

m_1 : p_3, p_4, p_5
I_1 : solutions in (t_2, t_3, t_4, t_5) of

$$0 \leq t_2 \leq 2$$
$$1 \leq t_3 \leq 3$$
$$0 \leq t_4 \leq 2$$
$$0 \leq t_5 \leq 3$$

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

m_2 : p_2, p_3, p_5
I_2 : solutions in (t_3, t_4, t_5) 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 represented by $\{\underline{\phi} \mid \underline{\phi} \in I(t_1) \times I(t_2) \times I(t_3)\}$

$$s = (m, \{\underline{\phi} \in \mathbb{R}^3 \mid 2 \leq \underline{\phi}_{t_1} \leq 3 \ \wedge \ 2 \leq \underline{\phi}_{t_2} \ \wedge \ 0 < \underline{\phi}_{t_3} \leq 5\})$$

By clock vectors (surjection, relative to I_s)

I represented by $\underline{\gamma}$, where $(\forall t \in \mathcal{E}(m))(I(t) = I_s(t) \dot{-} \underline{\gamma}_t\}$

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

By "total" clock vectors (cf. Louchka, # means "undefined"):

$s = (m, \underline{\gamma})$, with $\underline{\gamma} \in (\mathbb{R} \cup \{\#\})^{|T|}$, 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 : $(\forall s \in R)(\forall t \in T)(\exists \rho)(\exists s')(s \xrightarrow{\rho.t} s')$

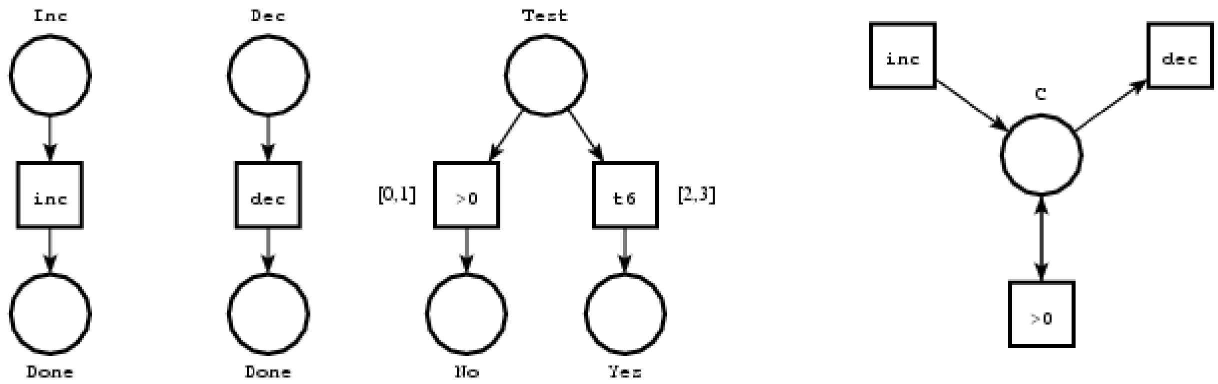
Boundedness : $(\exists b \in \mathbb{N})(\forall (m, I) \in R)(\forall p \in P)(m(p) \leq b)$

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)
ϕ	ϕ true at the first state
$\bigcirc\phi$	ϕ true at next state
$\Box\phi$	ϕ always true
$\Diamond\phi$	ϕ eventually true
$\phi\ U\ \psi$	ϕ true until ψ does and ψ eventually true
$\Box\Diamond\phi$	ϕ true infinitely often (fairness requirements)
$\Box(\phi \Rightarrow \Diamond\psi)$	ϕ always results in ψ (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 μ-calculus

Logics

Branching time

CTL (Computational tree logic)

Interpreted at the states of a transition system

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

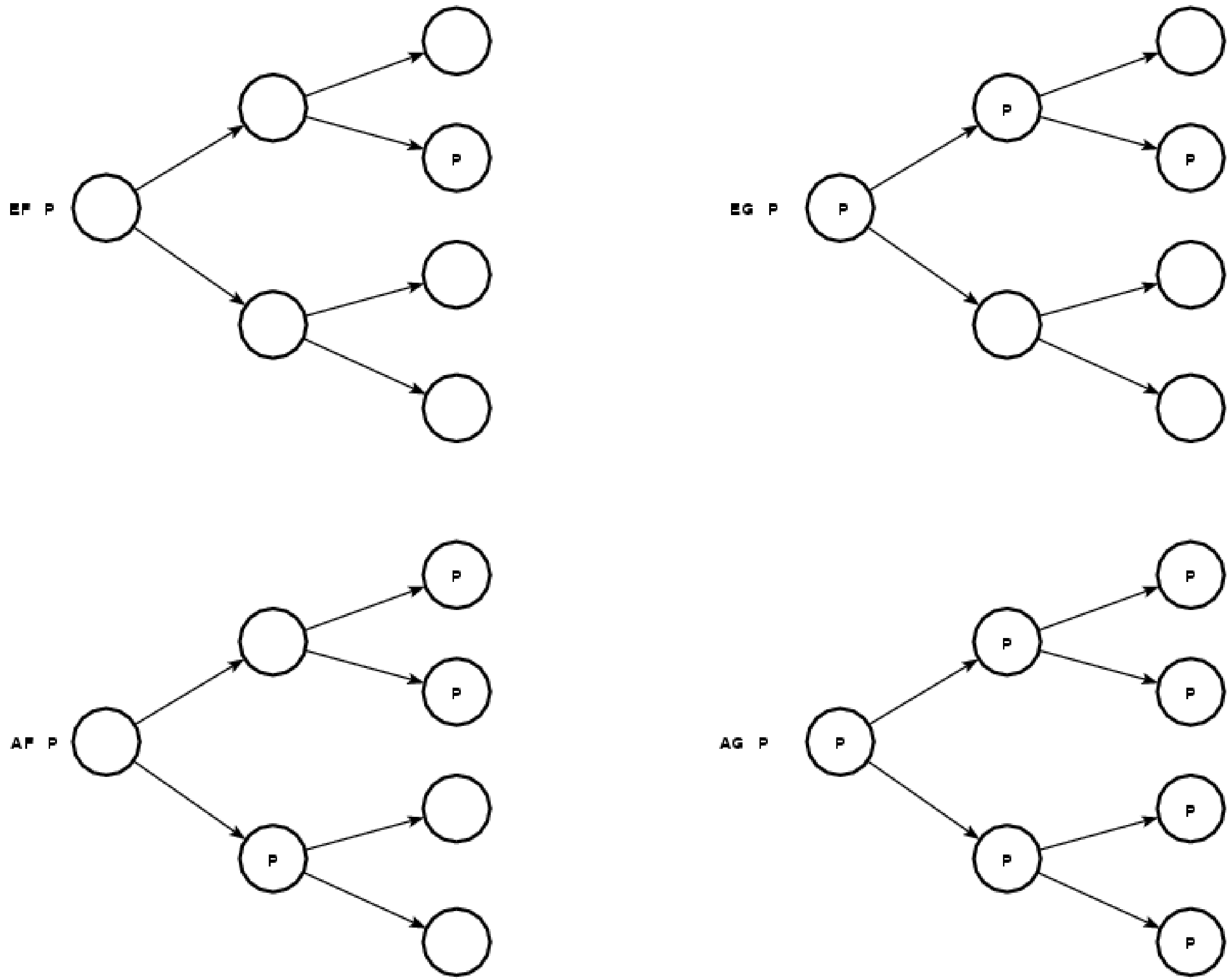
$EF\ \phi = E[true\ U\ \phi]$
$AF\ \phi = A[true\ U\ \phi]$
$EG\ \phi = \neg(AF(\neg\phi))$
$AG\ \phi = \neg(EF(\neg\phi))$

Fixpoint calculi

Modal μ-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 μ-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

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

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

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

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

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

Abstract states are linked ($\longrightarrow_A$) iff concrete states are ($\longrightarrow$):

$$\text{EE} = (\forall t)(\forall s, s')(\forall c, c')(c \xrightarrow{t}_A c' \Leftrightarrow s \xrightarrow{t} s')$$

Weaker EE, if $\mathbf{C}$ is a cover of $\mathbf{S}$ rather than a partition:

$$\begin{aligned}\text{EE'} = (\forall t)((&\forall c, c')(c \xrightarrow{t}_A c' \Rightarrow (\exists s \in c)(\exists s' \in c')(s \xrightarrow{t} s') \\ \wedge \;\; &(\forall s, s')(s \xrightarrow{t} s' \Rightarrow (\exists c \ni s)(\exists c' \ni s')(c \xrightarrow{t}_A c'))\end{aligned}$$

Theorems

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

EE is a soundness condition on $\mathbf{C}$ wrt $\mathbf{S}$

Assuming $\mathbf{C}$ is a partition of $\mathbf{S}$: (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' ($\longrightarrow_A$ simply written $\longrightarrow$)

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 $t \in T$, $\theta \in \mathbf{R}^+$ and:

1. $\mathbf{Pre}(t) \leq m \quad$ (t is enabled at m)

   $\theta \geq \downarrow I(t)$

   $(\forall k)(\mathbf{Pre}(k) \leq m \Rightarrow \theta \leq \uparrow I(k))$

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

3. $(\forall k)(\mathbf{Pre}(k) \leq m \Rightarrow I'(k) =$

   $\quad\quad \mathbf{if}\ k \neq t \wedge m - \mathbf{Pre}(t) \geq \mathbf{Pre}(k)\ \mathbf{then}\ I(k) \dot{-} \theta\ \mathbf{else}\ I_S(k))$

Idea: abstract parameter θ

State classes

States

$(m, \{\underline{\phi} \mid \underline{\phi} \in I(t_1) \times \ldots \times I(t_n)\})$ 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 $W\underline{\phi} \leq \underline{q}$

Equality of classes:

$(m, W) \cong (m', W')$ iff $m = m'$ and W and W' have same solution set

Computing State Classes

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

- $C_\epsilon = (m_0, \{\downarrow I_s(t) \leq \underline{\phi}_t \leq \uparrow I_s(t) \mid \mathbf{Pre}(t) \leq m_0\})$

- t is firable from some state of C_σ iff:

  (i) $m \geq \mathbf{Pre}(t)$ (t is enabled at m)

  (ii) W augmented with the following is consistent:
  $\{\underline{\phi}_t \leq \underline{\phi}_i \mid i \neq t \wedge m \geq \mathbf{Pre}(i)\}$

- If so, then $m' = m - \mathbf{Pre}(t) + \mathbf{Post}(t)$, and W' is obtained by:

  1. add inequations (ii) to W;
  2. $\forall i$ enabled at m', add variable $\underline{\phi}'_i$ and inequations:

     $\underline{\phi}'_i = \underline{\phi}_i - \underline{\phi}_t$, if $i \neq t$ and $m - \mathbf{Pre}(t) \geq \mathbf{Pre}(i)$

     $\downarrow I_s(i) \leq \underline{\phi}_i \leq \uparrow I_s(i)$, otherwise
  3. Eliminate variables $\underline{\phi}$

- $(m, W) \cong (m', W')$ iff $m = m'$ and W and W' have equal solution sets

In terms of states

Let:

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

Then:

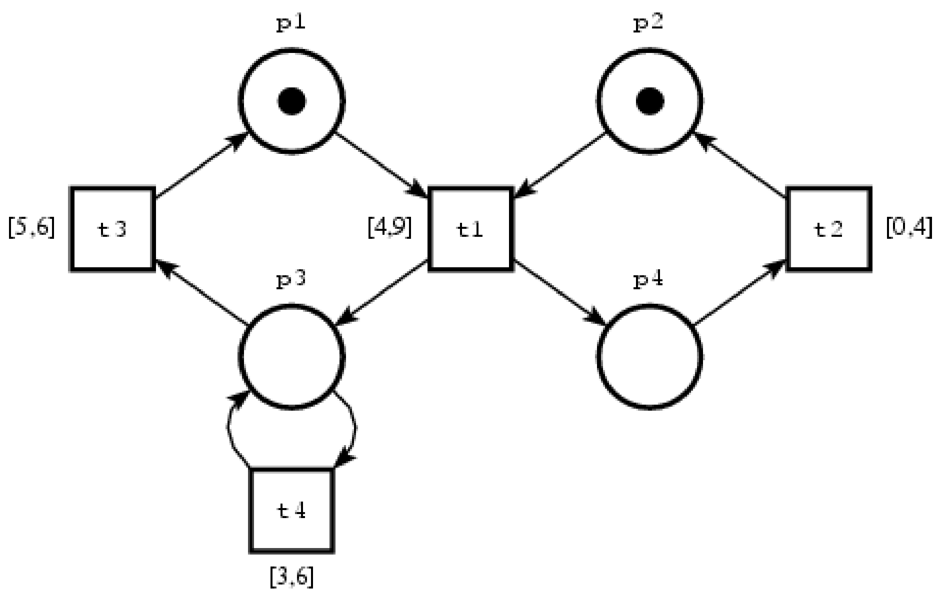
$$SCG = (C/\cong, \xrightarrow{t}, [\{s_0\}]_\cong)$$

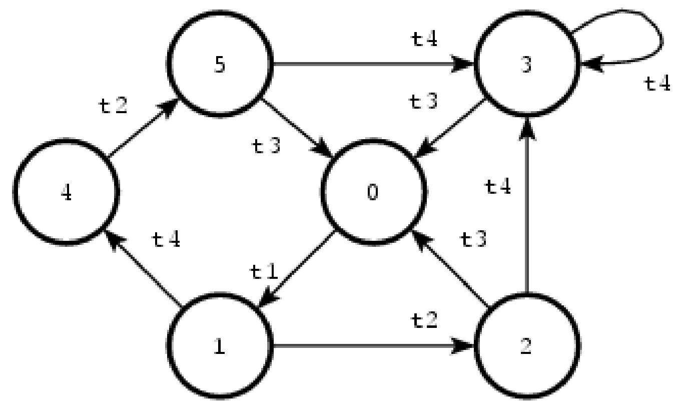
$$c \cong c' \quad \text{iff} \quad (\forall((m,I),(m',I')) \in c \times c')(m = m') \quad \wedge \quad \bigcup_{s \in c}(\mathcal{F}(s)) = \bigcup_{s' \in c'}(\mathcal{F}(s'))$$

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





$C_0 = (p_1\ p_2, \{4 \leq t_1 9\})$

$C_1 = (p_3\ p_4, \{0 \leq t_2 \leq 4, 5 \leq t_3 \leq 6, 3 \leq t_4 \leq 6\})$

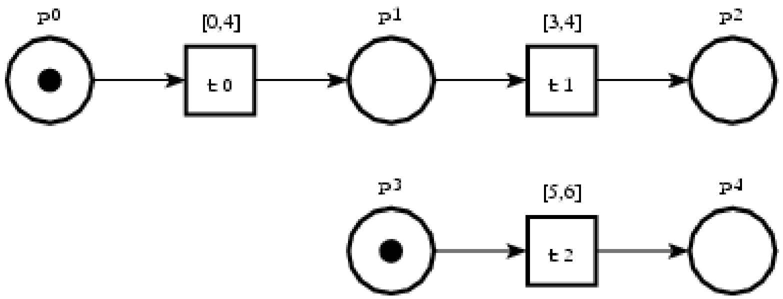
$C_2 = (p_2\ p_3, \{1 \leq t_3 \leq 6, 0 \leq t_4 \leq 6, t_3 - t_4 \leq 3, t_4 - t_3 \leq 1\})$

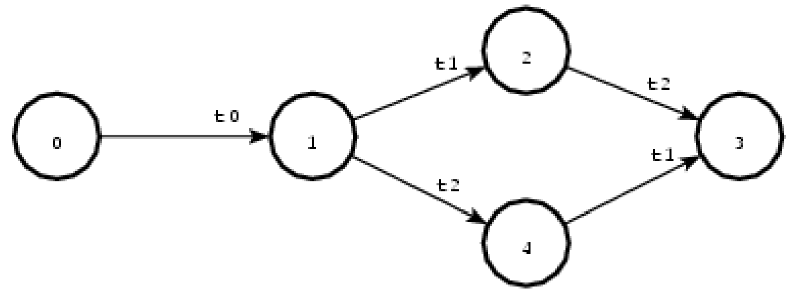
$C_3 = (p_2\ p_3, \{5 \leq t_3 \leq 6, 3 \leq t_4 \leq 6\})$

$C_4 = (p_3\ p_4, \{0 \leq t_2 \leq 1, 5 \leq t_3 \leq 6, 3 \leq t_4 \leq 6\})$

$C_5 = (p_2\ p_3, \{4 \leq t_3 \leq 6, 2 \leq t_4 \leq 6, t_3 - t_4 \leq 3, t_4 - t_3 \leq 1\})$

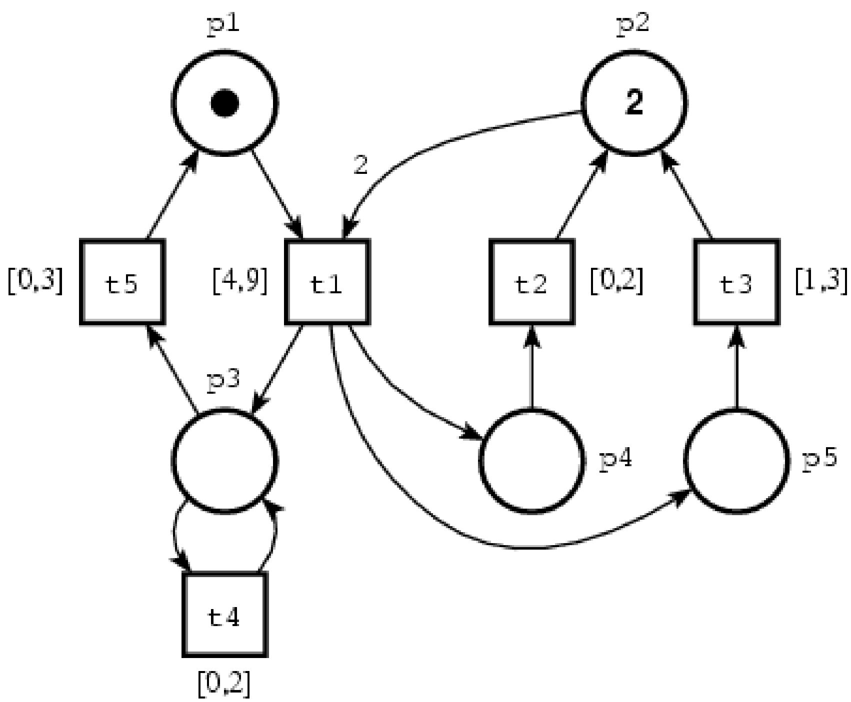
Example 2

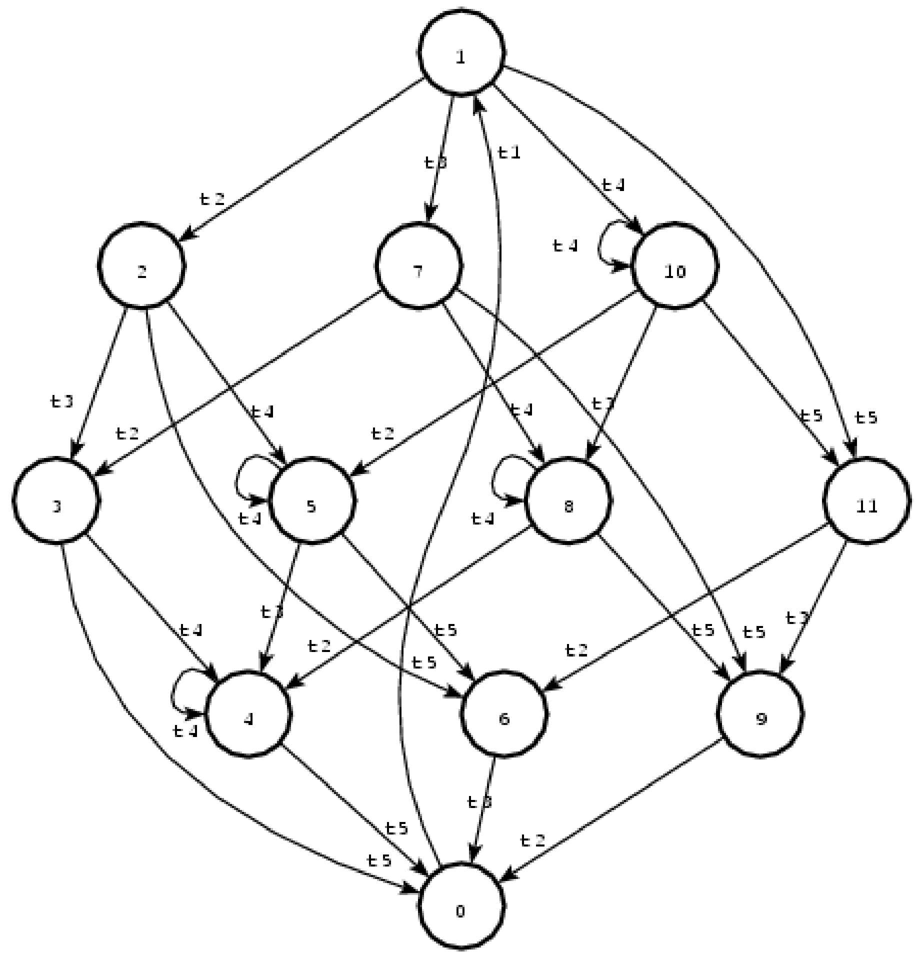




$$C_0 = (p_0\ p_3, \{0 \leq t_0 \leq 4, 5 \leq t_2 \leq 6\})$$
$$C_1 = (p_1\ p_3, \{3 \leq t_1 \leq 4, 1 \leq t_2 \leq 6\})$$
$$C_2 = (p_2\ p_3, \{0 \leq t_2 \leq 3\})$$
$$C_3 = (p_2\ p_4, \{\})$$
$$C_4 = (p_1\ p_4, \{0 \leq t_1 \leq 3\})$$

TPN example





Properties of the abstraction

State sets equivalent by $\cong$ have same futures

SCG Finite iff the TPN is bounded

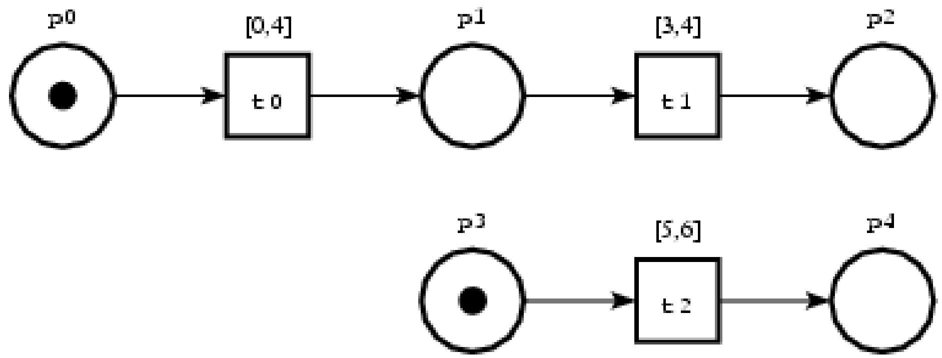
Preserves markings and firing sequences (LTL)

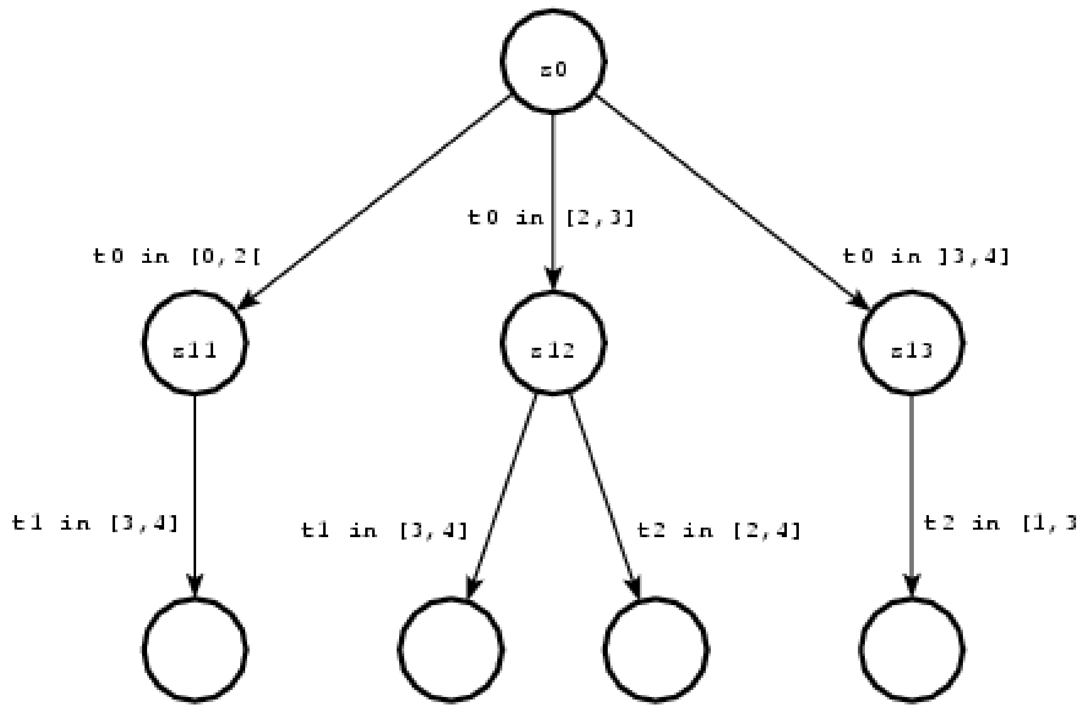
Decides k-boundedness, marking reachability (if bounded)

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):

$$\begin{array}{ccc} t_3 & \leq & 6 \\ 4 & \leq & t_3 \\ 2 & < & t_4 \\ t_3 - t_4 & < & 3 \\ t_4 - t_3 & \leq & 1 \end{array} \qquad \begin{array}{ccc} t_3 - \iota & \leq & 6 \\ t_4 - \iota & \leq & \infty \\ \iota - t_3 & \leq & -4 \\ \iota - t_4 & < & -2 \\ t_3 - t_4 & < & 3 \\ t_4 - t_3 & \leq & 1 \end{array}$$

$x - y$	ι	t_3	t_4
ι	$(\leq, 0)$	$(\leq, -4)$	$(<, -2)$
t_3	$(\leq, 6)$	$(\leq, 0)$	$(<, 3)$
t_4	$(\leq, \infty)$	$(\leq, 1)$	$(\leq, 0)$

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': $M'_{t'0} = min(M'_{t0}, M'_{tt'})$

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 $M'_{tt'} = min(M_{tt'}, M'_{t0} + M'_{ot'})$

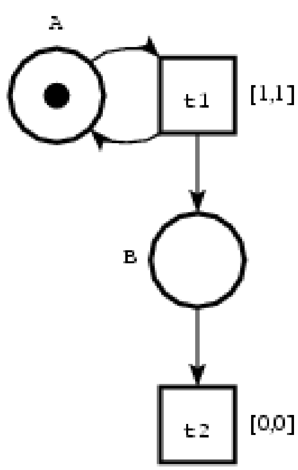
Checking boundedness

Sufficient conditions for boundedness:

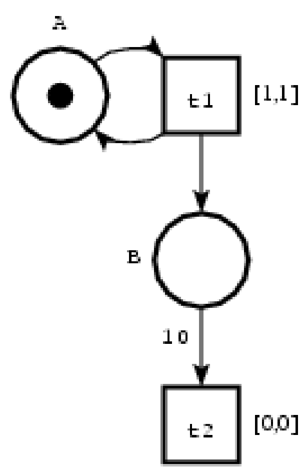
No $c = (m, D)$ and $c' = (m', D')$ such that:

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

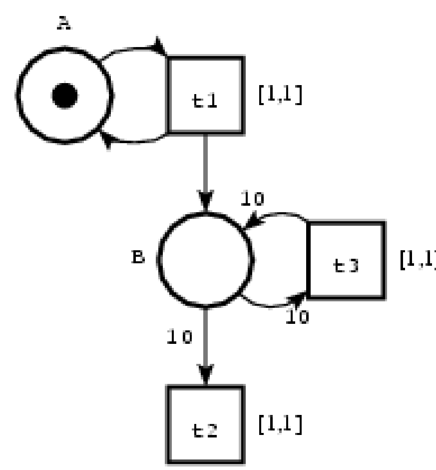
But not necessary:



passes with 1,2,3
fails with 1,2



passes with 1,2,3,4
fails with 1,2,3



fails with 1,2,3,4

LTL model checking of Time Petri Nets

Obtaining a Kripke transition system:

- Build the SCG
- Add loops to deadlock states
- Add loops to temporarilly 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

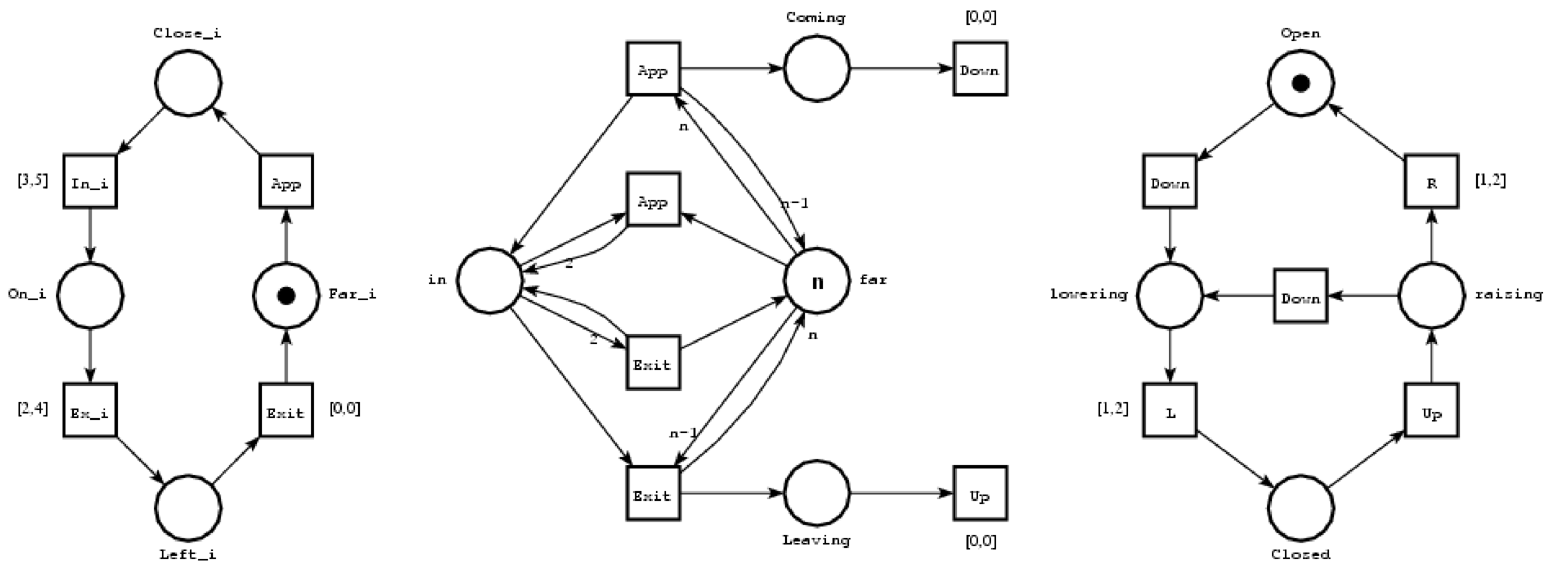
If $Sol(D) \subseteq Sol(D')$ then any schedule firable from (m, D) is firable from (m, D'), so we won't find new markings by storing (m, D)

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

Preserves markings but NOT firing sequences

Often much smaller than SCG

Example : Level crossing



		SCG	$SSG_{\subseteq}$
(1 $train$)	$Classes$	11	10
	$Edges$	14	13
	$CPU(s)$	0.00	0.00
(2 $trains$)	$Classes$	123	37
	$Edges$	218	74
	$CPU(s)$	0.00	0.00
(3 $trains$)	$Classes$	3101	172
	$Edges$	7754	492
	$CPU(s)$	0.07	0.01
(4 $trains$)	$Classes$	134501	1175
	$Edges$	436896	4534
	$CPU(s)$	5.85	0.07
(5 $trains$)	$Classes$	8557621	10972
	$Edges$	34337748	53766
	$CPU(s)$	1254.92	1.20

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:

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

Then: [BV03]

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

Clocks, equivalence $\equiv$

Clock systems

$\underline{\gamma}_t$ = time elapsed since t was last enabled

Clock vector $\underline{\gamma}$ denotes the interval I such that $(\forall t)(I(t) = Is(t) \dot{-} \underline{\gamma}_t)$

NOTE: infinitely many clock vectors may denote the same state

Strong Classes

Represented by a marking and a clock system

$(m, G\underline{\gamma} \leq \underline{g})$ 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 $(m, Q) \equiv (m', Q') \Leftrightarrow m = m' \wedge Sol(Q) = Sol(Q')$

Computing Strong State Classes

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

- $C_\epsilon = (m_0, \{0 \leq \underline{\gamma}_t \leq 0 \mid \mathbf{Pre}(t) \leq m_0\})$
- t is firable from some state of C_σ iff:

  (i) $m \geq \mathbf{Pre}(t)$ (t is enabled at m)

  (ii) Q augmented with the following is consistent:
  $0 \leq \theta$
  $\downarrow I_s(t) \leq \underline{\gamma}_t + \theta$
  $\{\theta + \underline{\gamma}_i \leq \uparrow I_s(i) \mid m \geq \mathbf{Pre}(i)\}$

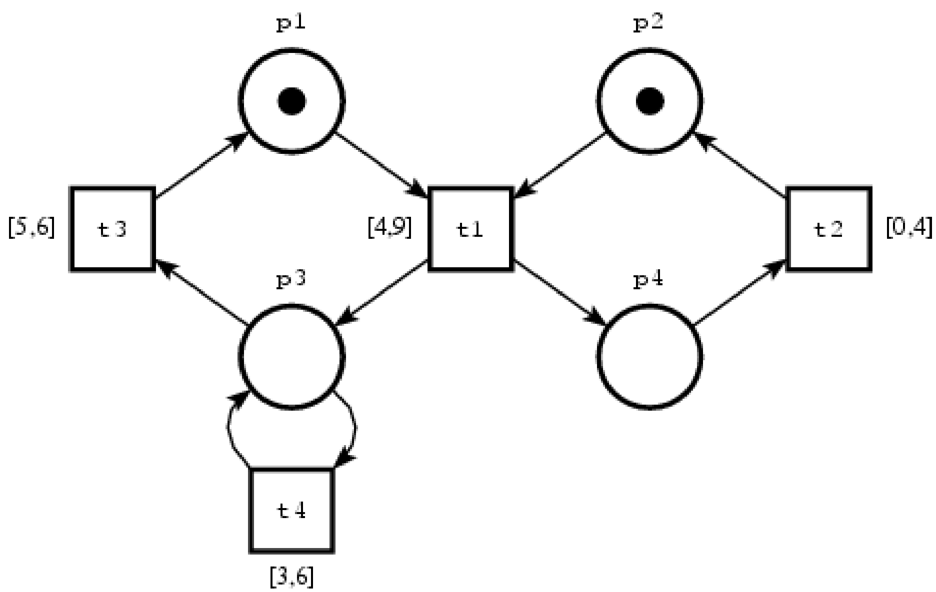
- If so, then $m' = m - \mathbf{Pre}(t) + \mathbf{Post}(t)$, and Q' is obtained by:
  1. add inequations (ii) to Q;
  2. $\forall i$ enabled at m', add $\underline{\gamma}'_i$ and inequations:

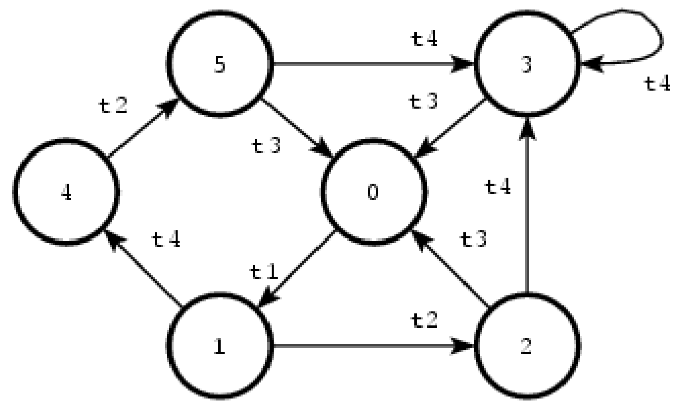
     $\underline{\gamma}'_i = \underline{\gamma}_i + \theta$, if $i \neq t$ and $m - \mathbf{Pre}(t) \geq \mathbf{Pre}(i)$

     $0 \leq \underline{\gamma}'_i \leq 0$, otherwise
  3. Eliminate variables $\underline{\gamma}$ and θ

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

Example





$C_0 = (p_1\ p_2, \{0 \le t_1 0\})$

$C_1 = (p_3\ p_4, \{0 \le t_2 \le 0, 0 \le t_3 \le 0, 0 \le t_4 \le 0\})$

$C_2 = (p_2\ p_3, \{0 \le t_3 \le 4, 0 \le t_4 \le 4, t_3 - t_4 \le 0, t_4 - t_3 \le 0\})$

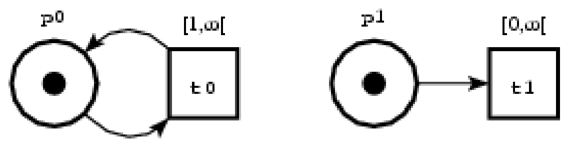
$C_3 = (p_2\ p_3, \{0 \le t_3 \le 0, 0 \le t_4 \le 0\})$

$C_4 = (p_3\ p_4, \{3 \le t_2 \le 4, 0 \le t_3 \le 0, 0 \le t_4 \le 0\})$

$C_5 = (p_2\ p_3, \{0 \le t_3 \le 1, 0 \le t_4 \le 1, t_3 - t_4 \le 0, t_4 - t_3 \le 0\})$

Handling Unbounded Intervals

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



$$C_\epsilon = (m_0, \{0 \leq \underline{\gamma}_{t_0} \leq 0,\ 0 \leq \underline{\gamma}_{t_1} \leq 0\})$$

$$C_{(t_0)^k} = (m_0, \{0 \leq \underline{\gamma}_{t_0} \leq 0,\ k \leq \underline{\gamma}_{t_1}\})$$

But $C_{(t_0)^k} \equiv (m_0, \{0 \leq \underline{\gamma}_{t_0} \leq 0,\ 0 \leq \underline{\gamma}_{t_1}\})$

Solution: Relax clock systems in Strong Classes

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

Partition Q by $\underline{\gamma}_k \geq Eft_s(k)$, for k s.t. $Lft_s(k) = \infty$

In half space $\underline{\gamma}_k \geq Eft_s(k)$, relax upper bound of $\underline{\gamma}_k$

Theorem:

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

Implementations

Assume Q denotes the set of states E

Relaxation [BV03]:

computes the largest set of clock vectors denoting set E

fragments classes ($\widehat{Q}$ 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 $SSCG$

Checking state reachability (in the DSG)

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

$$(\forall t \in \mathcal{E}(m))(I(t) = I_s(t) \dot{-} \underline{\gamma}_t)$$

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

LTL model checking with the $SSCG$

As for the SCG

But SCG is a better choice since typically smaller

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 ($O(n^2)$)

$\equiv$ implemented as equality of canonical forms
after relaxation or normalization

Preserving states only, $SSCG_{\subseteq}$

Similar to the SCG:

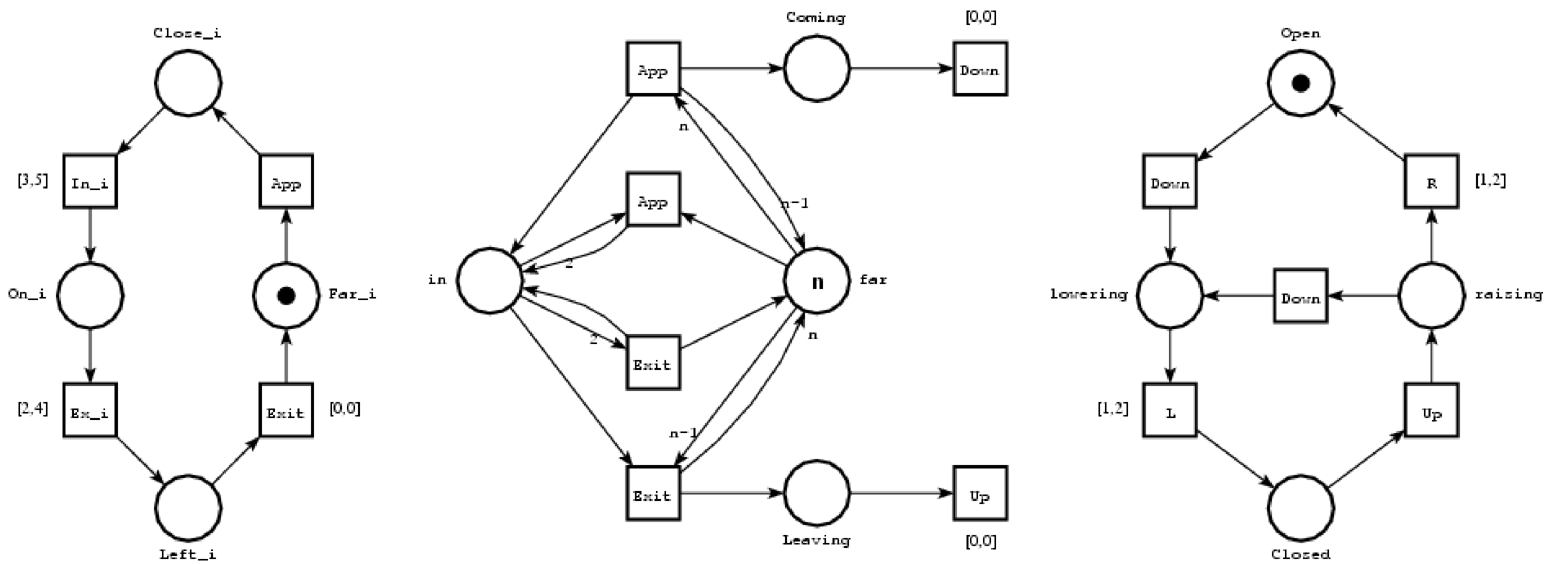
If $Sol(Q) \subseteq Sol(Q')$ then any schedule firable from (m, Q) is firable from (m, Q'), so we won't find new states by storing (m, Q)

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

Preserves states but NOT firing sequences

Often much smaller than $SSCG$

Example : Level crossing



		SCG	$SSCG$
(1 *train*)	*Classes*	11	11
	Edges	14	14
	$CPU(s)$	0.00	0.00
(2 *trains*)	*Classes*	123	141
	Edges	218	254
	$CPU(s)$	0.00	0.00
(3 *trains*)	*Classes*	3101	5051
	Edges	7754	13019
	$CPU(s)$	0.07	0.13
(4 *trains*)	*Classes*	134501	351271
	Edges	436896	1193376
	$CPU(s)$	5.85	20.14
(5 *trains*)	*Classes*	8557621	35945411
	Edges	34337748	151908273
	$CPU(s)$	1254.92	7439.25

		$SCG_{\subseteq}$	$SSCG_{\subseteq}$
(1 *train*)	*Classes*	10	10
	Edges	13	13
	$CPU(s)$	0.00	0.00
(2 *trains*)	*Classes*	37	41
	Edges	74	82
	$CPU(s)$	0.00	0.00
(3 *trains*)	*Classes*	172	232
	Edges	492	672
	$CPU(s)$	0.01	0.01
(4 *trains*)	*Classes*	1175	1807
	Edges	4534	7062
	$CPU(s)$	0.07	0.15
(5 *trains*)	*Classes*	10972	18052
	Edges	53766	89166
	$CPU(s)$	1.20	3.70

State Classes

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 $SSCG_{\subseteq}$

Enforce AE using partition refinement

The ASCG and DSG will be bisimilar

First such construction proposed in [YR98] (Atomic state classes)

Partition refinement

[Paige et Tarjan, 1987]

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:

> $Q = P$
> **while** $(\exists A, B \in Q)(A$ is not Stable wrt $B)$
> **do** replace A by $A_1 = A \cap B^{-1}$ 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

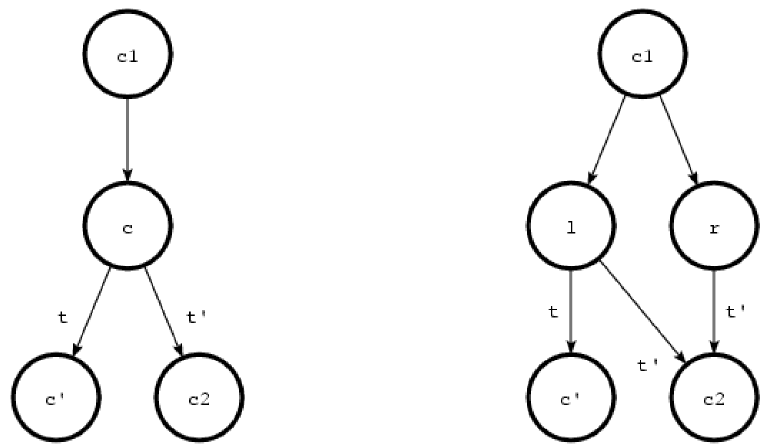
Partitionning $SSCG$ classes

Partition Technique

If $c\!=\!(m,Q) \stackrel{t}{\rightarrow} c'$ and c is unstable wrt c' then some constraint ρ is:

- necessary for $s \in c$ to have a successor in c'
- nonredundant in Q

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



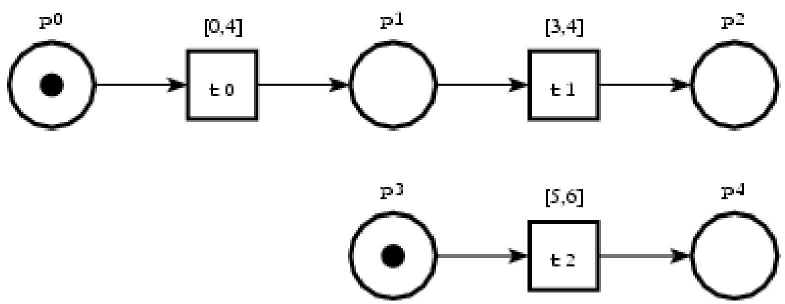
Computing ρ constraints

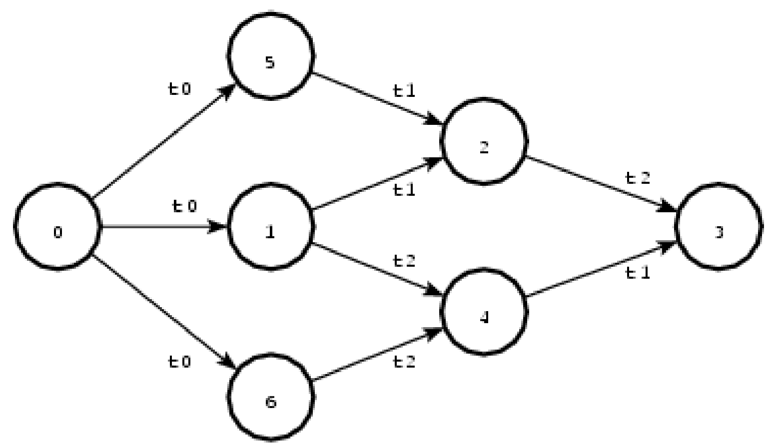
Compute predecessors P by t of states in c' (by reverse SSCG rule)

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

Otherwise take any constraint of P nonredundant in Q

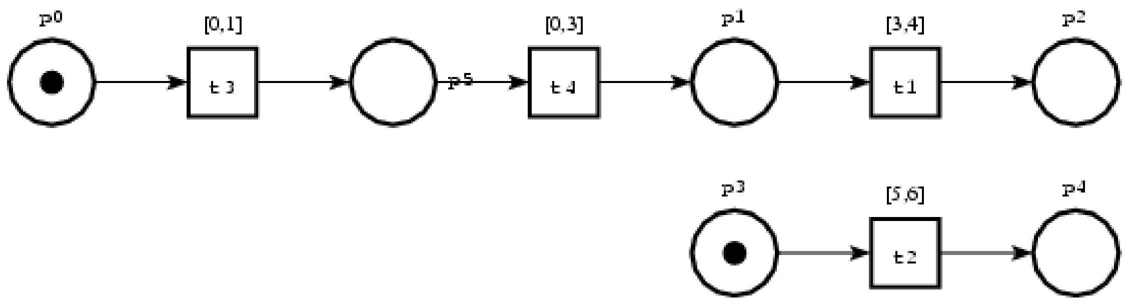
Example 1



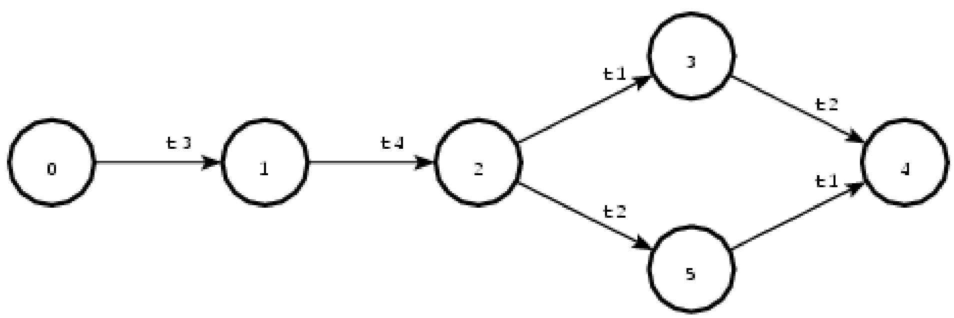


$C_0 = (p_0\ p_3, \{0 \leq t_0 \leq 0, 0 \leq t_2 \leq 0\})$
$C_1 = (p_1\ p_3, \{0 \leq t_1 \leq 0, 1 \leq t_2 \leq 3\})$
$C_2 = (p_2\ p_3, \{3 \leq t_2 \leq 6\})$
$C_3 = (p_2\ p_4, \{\})$
$C_4 = (p_1\ p_4, \{1 \leq t_1 \leq 4\})$
$C_5 = (p_1\ p_3, \{0 \leq t_1 \leq 0, 0 \leq t_2 < 1\})$
$C_6 = (p_1\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

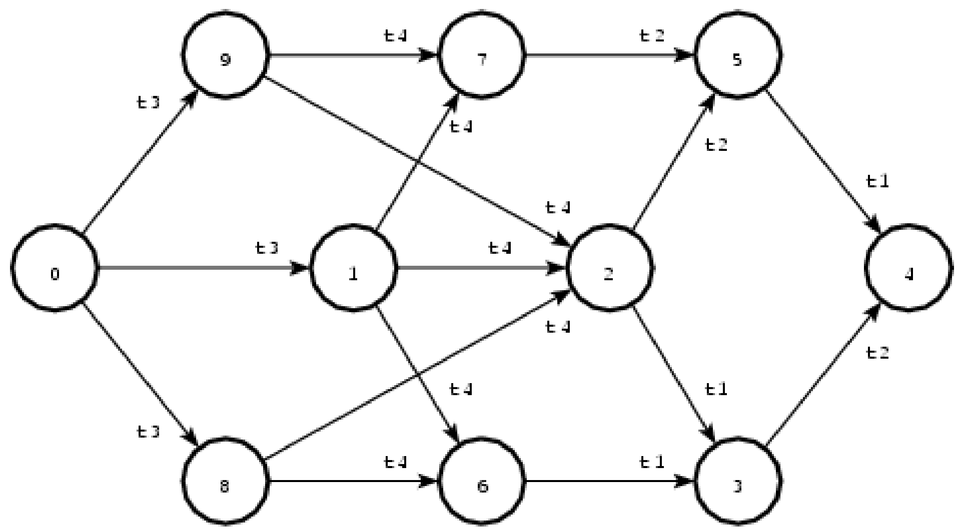
Example 2



p0
[0,1]
t3
p5
[0,3]
t4
p1
[3,4]
t1
p2
p3
[5,6]
t2
p4



0
t3
1
t4
2
t1
3
t2
4
t2
5
t1



9
t4
7
t2
5
t3
t4
t2
t1
t4
0
t3
1
t4
2
4
t4
t2
t4
t1
t3
8
t4
6
t1
3

Properties:

Finite iff the TPN is bounded

Abstraction preserves states and firing sequences (LTL)

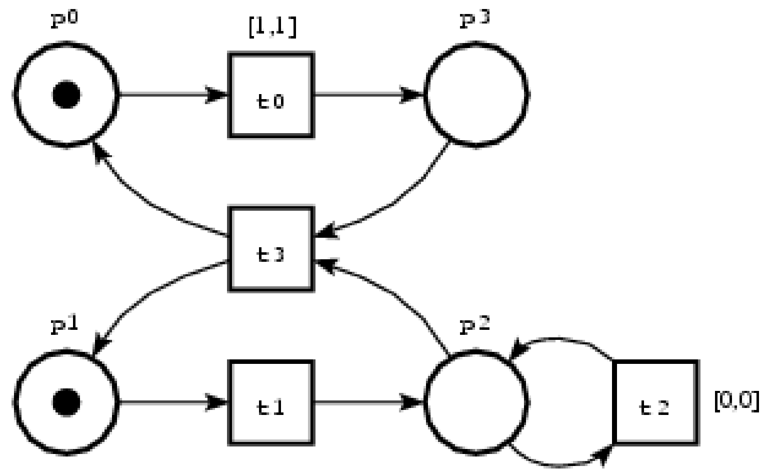
Decides k-boundedness, marking and state reachability

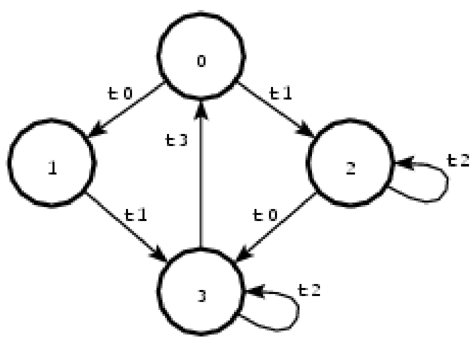
Refinement restores AE, hence ASCG preserve branching properties and liveness (suitable for CTL modelchecking)

Notes:

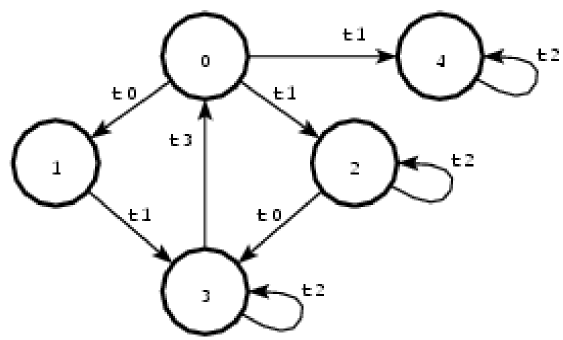
- ASCG is a cover rather than a partition $\Rightarrow$ not minimal
- ASCG is bisimilar to the DSG, but not to the SG

Liveness analysis





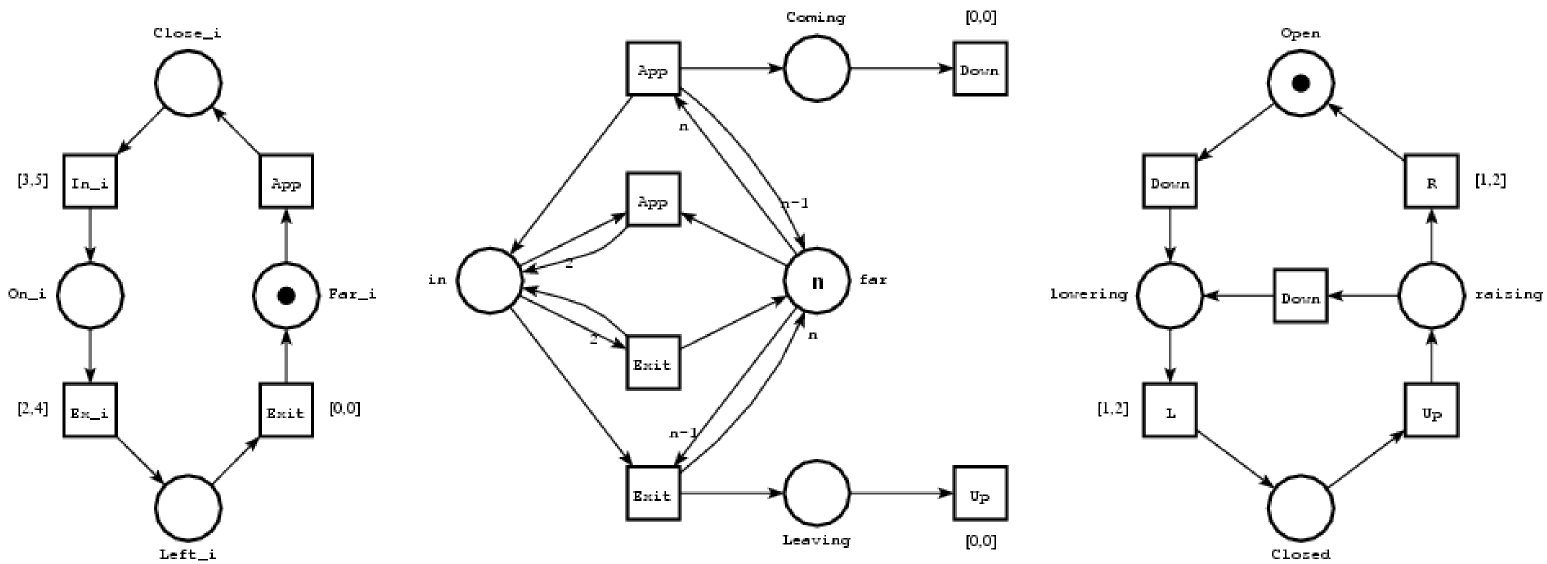
SCG, SSCG



ASCG

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

Example : Level crossing



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

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

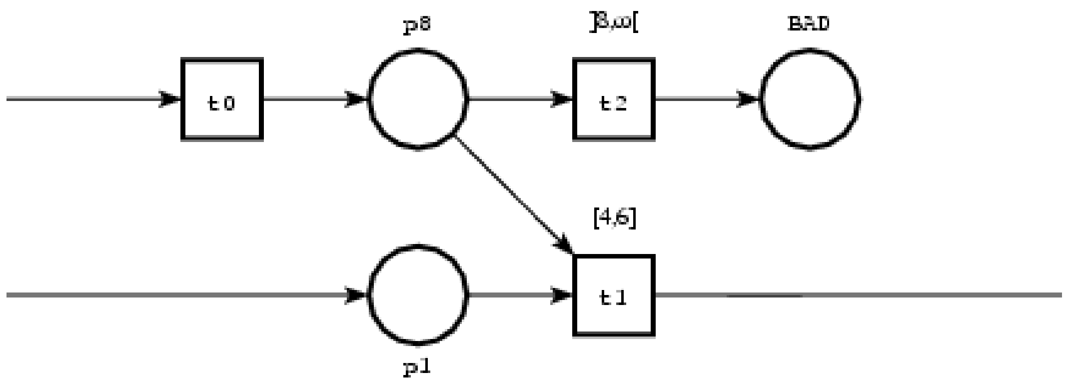
Model checkers for timed logics:

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 σ:

Characterize firing schedules over σ

Check existence of time constrained schedules

Find fastest/slowest schedule

...

Computing Path Systems

As for SSCG, but without elimination of θ:

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

- $K_\epsilon = (m_0, \{0 \le \underline{\gamma}_t \le 0 \mid \mathbf{Pre}(t) \le m_0\})$
- t is firable from some state of K_σ iff:
  - (i) $m \ge \mathbf{Pre}(t)$ (t is enabled at m)
  - (ii) Q augmented with the following is consistent:
    $0 \le \theta$
    $\downarrow I_s(t) \le \underline{\gamma}_t + \theta$
    $\{\theta + \underline{\gamma}_i \le \uparrow I_s(i) \mid m \ge \mathbf{Pre}(i)\}$
- If so, then $m' = m - \mathbf{Pre}(t) + \mathbf{Post}(t)$, and Q' is obtained by:
  1. add inequations (ii) to Q;
  2. $\forall i$ enabled at m', add $\underline{\gamma}'_i$ and inequations:

     $\underline{\gamma}'_i = \underline{\gamma}_i + \theta$, if $i \ne t$ and $m - \mathbf{Pre}(t) \ge \mathbf{Pre}(i)$

     $0 \le \underline{\gamma}'_i \le 0$, otherwise
  3. Eliminate variables $\underline{\gamma}$

Path Systems ...

K_σ **Links firing times along** σ **with state reached**

$$P(\underline{\theta}|\underline{\gamma}) \leq \underline{p}$$

Projecting on $\underline{\theta}$ **yields path system**

$$T(\underline{\theta}) \leq \underline{t}$$

Characterizes times at which transitions can fire along σ

in delays ($\underline{\theta}$, relative times)

or dates ($\underline{\delta}$, absolute times) using:

$$\underline{\delta}_i = \underline{\theta}_1 + \ldots + \underline{\theta}_i$$

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

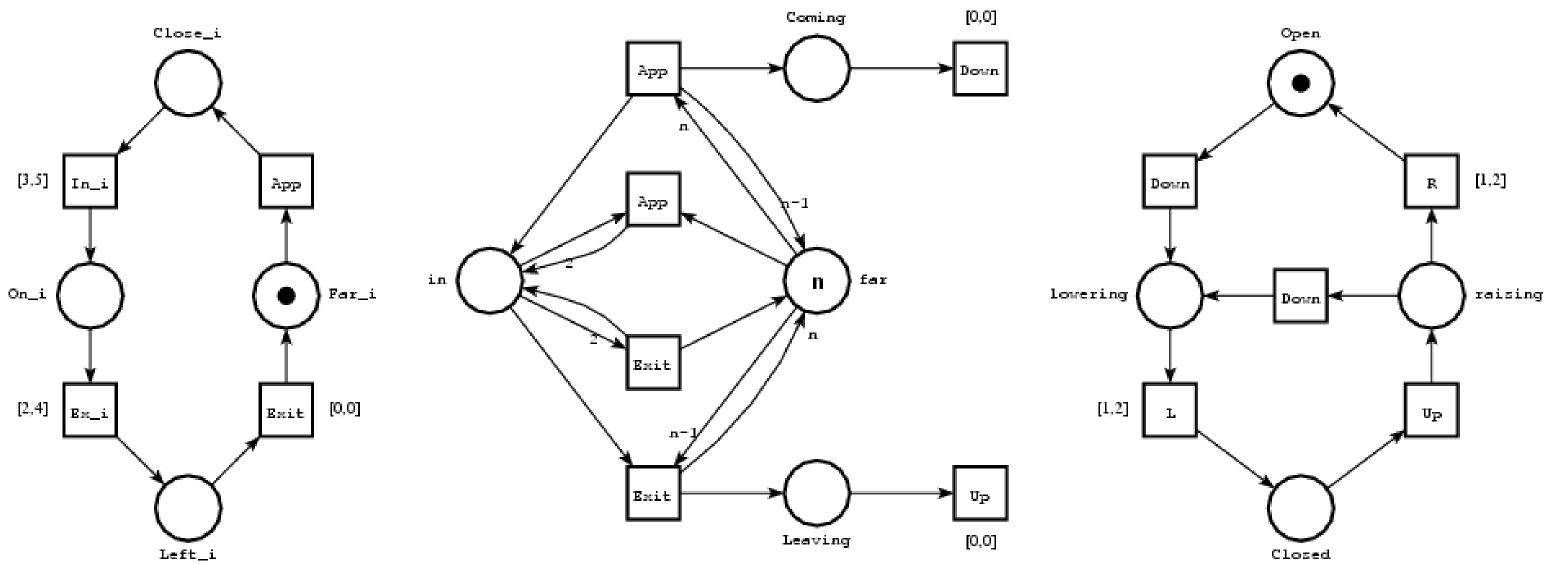
some progress for dense timed systems

Translation into Timed Automata

Structural translation [CR06] preserves weak timed bisimilarity

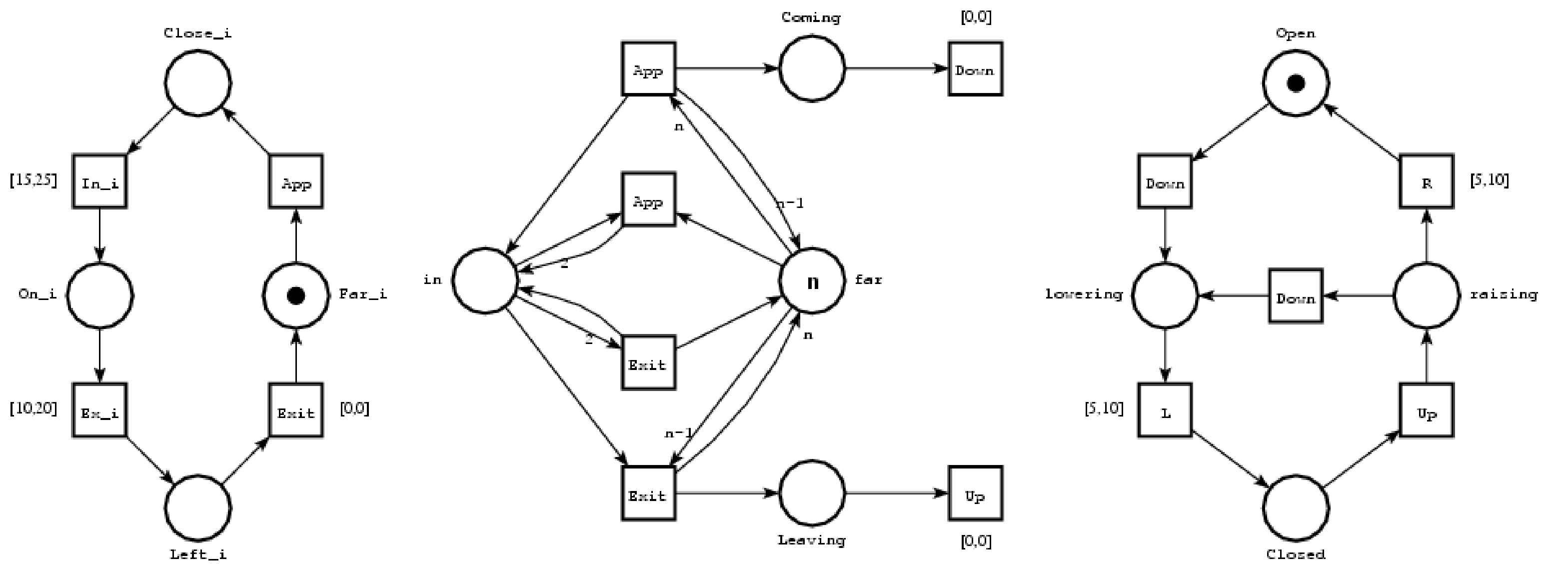
Provided by Roméo toolbox

State classes % Essential states



		SCG	*ES*	*ES + delays*
(1 *train*)	*Classes*	11	13	24
	Edges	14	27	37
	CPU(s)	0.00	0.00	0.00
(2 *trains*)	*Classes*	123	116	203
	Edges	218	382	378
	CPU(s)	0.00	0.00	0.00
(3 *trains*)	*Classes*	3101	1550	2299
	Edges	7754	5823	5294
	CPU(s)	0.07	0.03	0.03
(4 *trains*)	*Classes*	134501	22268	28895
	Edges	436896	91256	81142
	CPU(s)	5.85	0.671	0.600
(5 *trains*)	*Classes*	8557621	313214	372475
	Edges	34337748	1397517	1245566
	CPU(s)	1254.92	15.12	12.92

Same example, intervals scaled by 5



		SCG	ES	$ES + delays$
(1 *train*)	*Classes*	11	25	80
	Edges	14	123	129
	$CPU(s)$	0.00	0.00	0.00
(2 *trains*)	*Classes*	123	564	3110
	Edges	218	8154	6107
	$CPU(s)$	0.00	0.03	0.02
(3 *trains*)	*Classes*	3101	27950	119479
	Edges	7754	315629	273782
	$CPU(s)$	0.07	1.65	1.48
(4 *trains*)	*Classes*	134501	1680212	5785743
	Edges	436896	18328768	15813462
	$CPU(s)$	5.85	133.09	114.61
(5 *trains*)	*Classes*	8557621	?	?
	Edges	34337748	?	?
	$CPU(s)$	1254.92	?	?

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 is k-enabled at m if $m \geq k * \mathbf{Pre}(t)$ ($k \geq 0$)

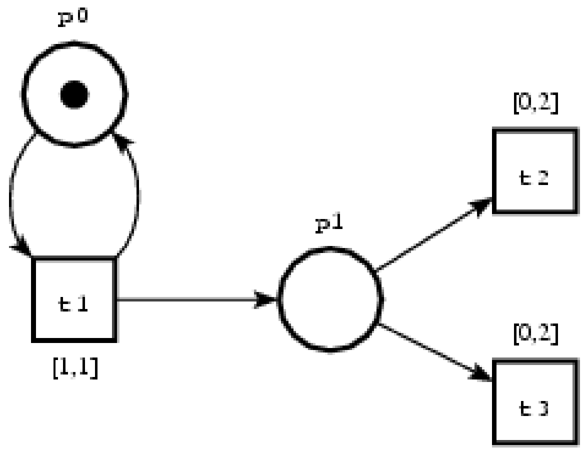
So far: One temporal variable per transition, whether or not multi-enabled (single-server semantics)

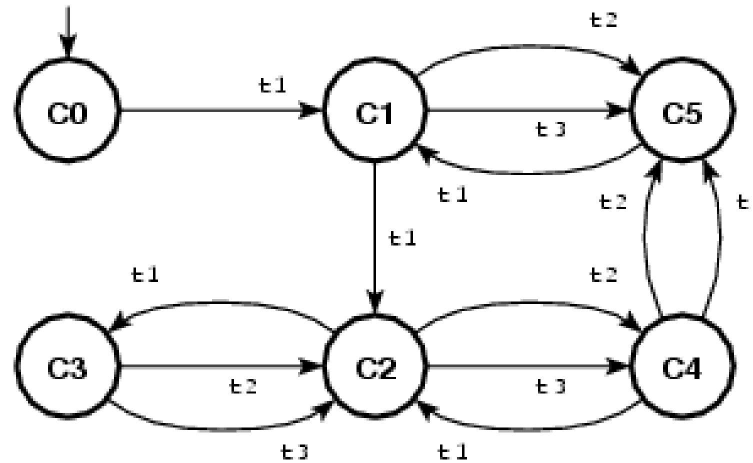
Consider: If t is k-enabled, then k temporal variables associated with t (multi-server semantics)

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

$\Rightarrow$ State class constructions can be adapted

Multi-enabledness example (oldest fires first SCG)





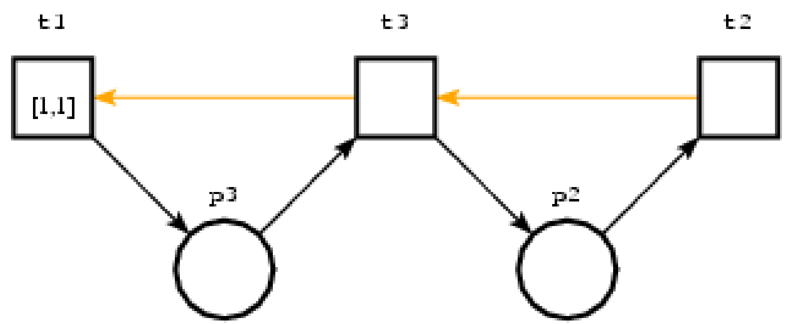
C_0 M_0 $p_0(1)$ D_0 $1 \le t_1 \le 1$	C_1 M_1 $p_0(1), p_1(1)$ D_1 $1 \le t_1 \le 1$ $0 \le t_2 \le 2$ $0 \le t_3 \le 2$	C_2 M_2 $p_0(1), p_1(2)$ D_2 $1 \le t_1 \le 1$ $0 \le t_2^0 \le 1$ $0 \le t_2^1 \le 2$ $0 \le t_3^0 \le 1$ $0 \le t_3^1 \le 2$
C_3 M_3 $p_0(1), p_1(3)$ D_3 $1 \le t_1 \le 1$ $0 \le t_2^0 \le 0$ $0 \le t_2^1 \le 1$ $0 \le t_2^2 \le 2$ $0 \le t_3^0 \le 0$ $0 \le t_3^1 \le 1$ $0 \le t_3^2 \le 2$	C_4 M_4 $p_0(1), p_1(1)$ D_4 $0 \le t_1 \le 1$ $0 \le t_2 \le 2$ $0 \le t_3 \le 2$ $t_2 - t_1 \le 1$ $t_3 - t_1 \le 1$	C_5 M_5 $p_0(1)$ D_5 $0 \le t_1 \le 1$

Time Petri nets with Priorities ($PrTPN$)

$\langle P, T, \mathbf{Pre}, \mathbf{Post}, m_0, Is, \succ \rangle$ in which:

- $\langle P, T, \mathbf{Pre}, \mathbf{Post}, m_0 \rangle, \mathbf{I}^+$ 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, Is_0)

- discrete transitions: $(m, I) \xrightarrow{t} (m', I')$ iff $t \in T$ and

  1. $m \geq \mathbf{Pre}(t)$
  2. $0 \in I(t)$
  3. $(\forall k \in T)(m \geq \mathbf{Pre}(k) \wedge 0 \in I(k) \Rightarrow \neg(k \succ t))$
  4. $m' = m - \mathbf{Pre}(t) + \mathbf{Post}(t)$
  5. $(\forall k \in T)(m' \geq \mathbf{Pre}(k) \Rightarrow$
     $I'(k) =$ **if** $k \neq t \wedge m - \mathbf{Pre}(t) \geq \mathbf{Pre}(k)$ **then** $I(k)$ **else** $Is(k))$

- continuous transitions: $(m, I) \xrightarrow{d} (m, I')$ iff

  $$(\forall k \in T)(m \geq \mathbf{Pre}(k) \Rightarrow d \leq\uparrow I(k) \wedge I'(k) = I(k) \dot{-} d)$$

Expressiveness

In terms of timed language acceptance:

$TPN = TA$ [BCHRL05, BHR06]

In terms of weak timed bisimulation:

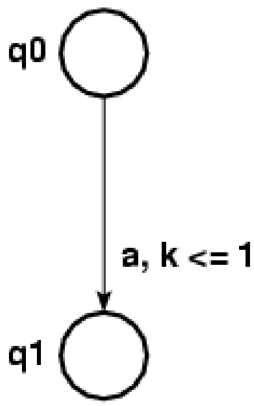
$TPN < TA$ [CR06]

$TPN = TA^{-}$ [BCHRL05]

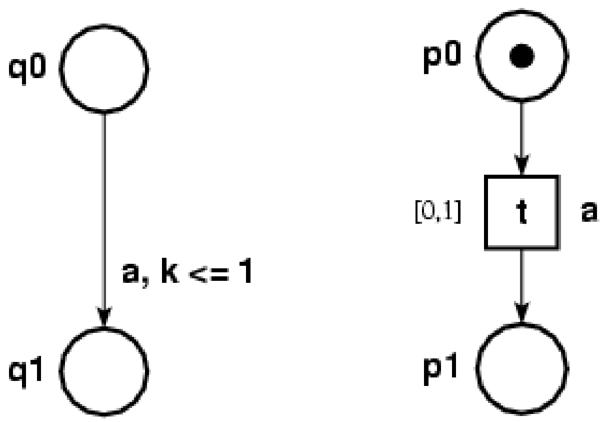
$TA + \{\leq, \wedge\} = PrTPN$ with right-closed or unbounded intervals [BPV06]

Note: Priorities enable compositional design

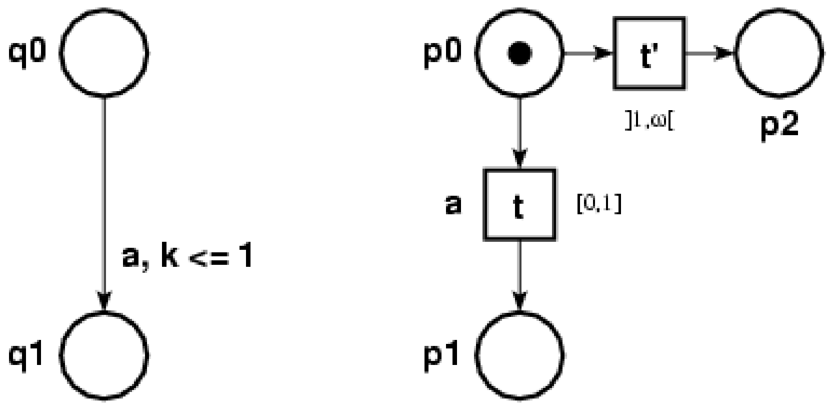
Priorities add expressiveness to TPN



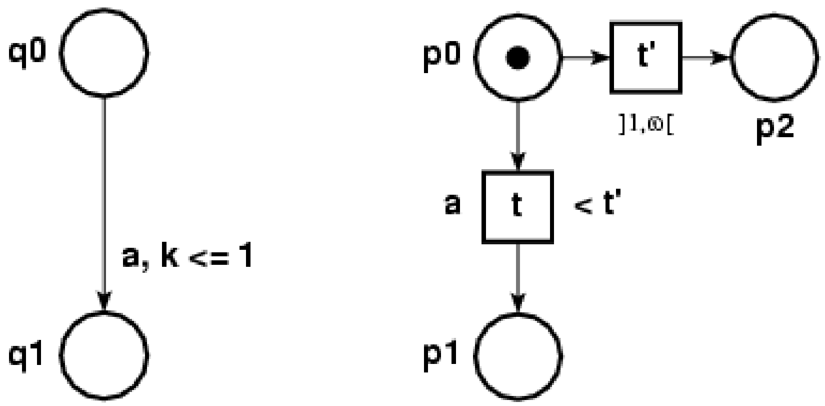
Priorities add expressiveness to TPN



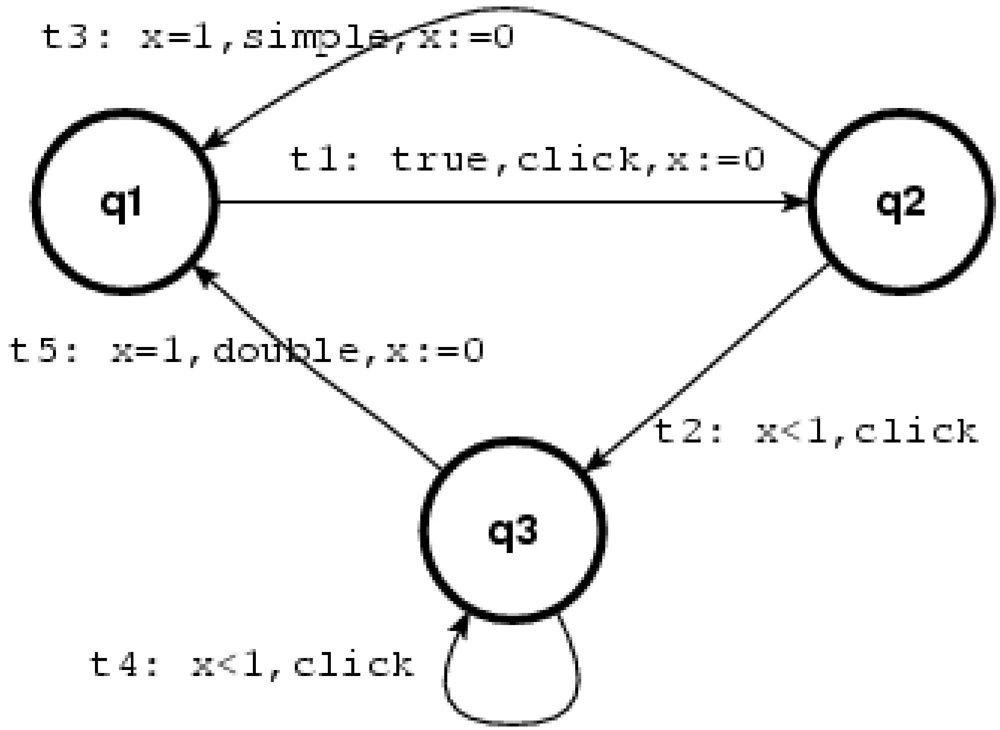
Priorities add expressiveness to TPN



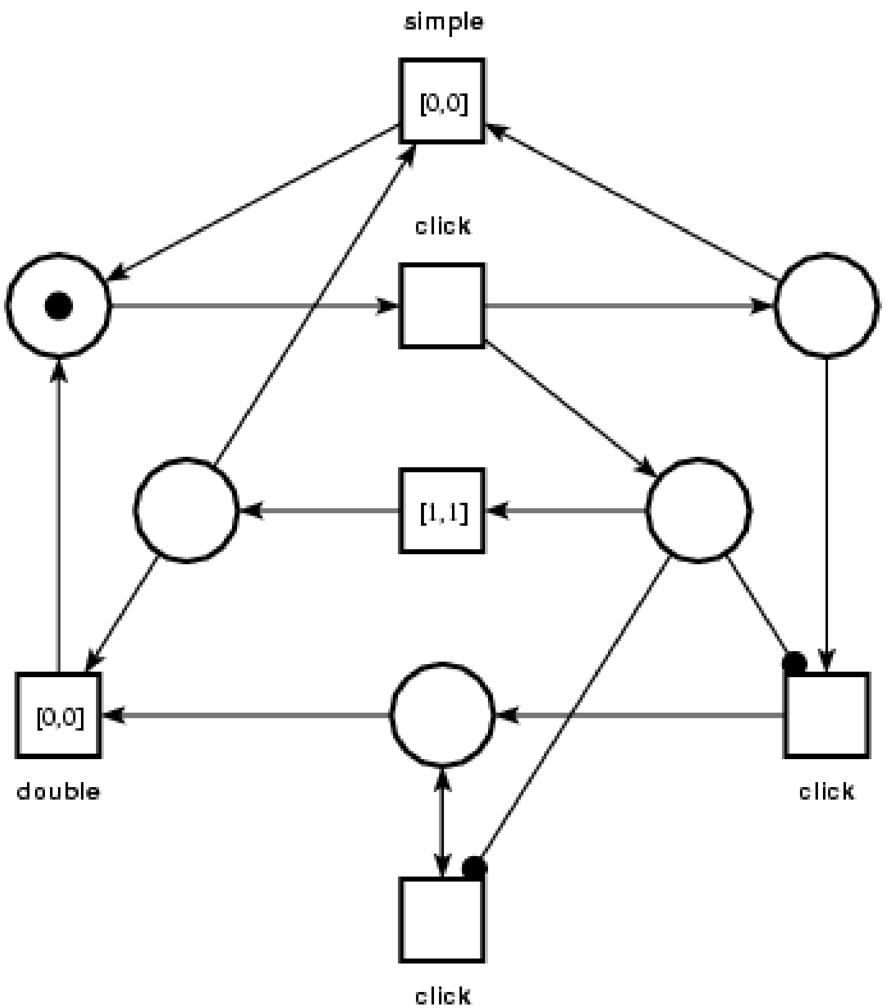
Priorities add expressiveness to TPN



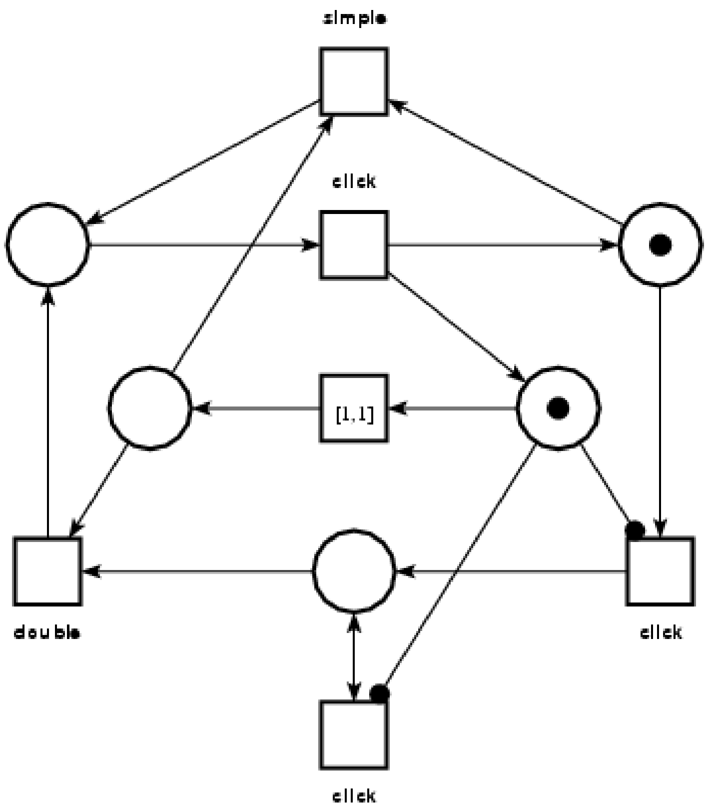
Double click TA

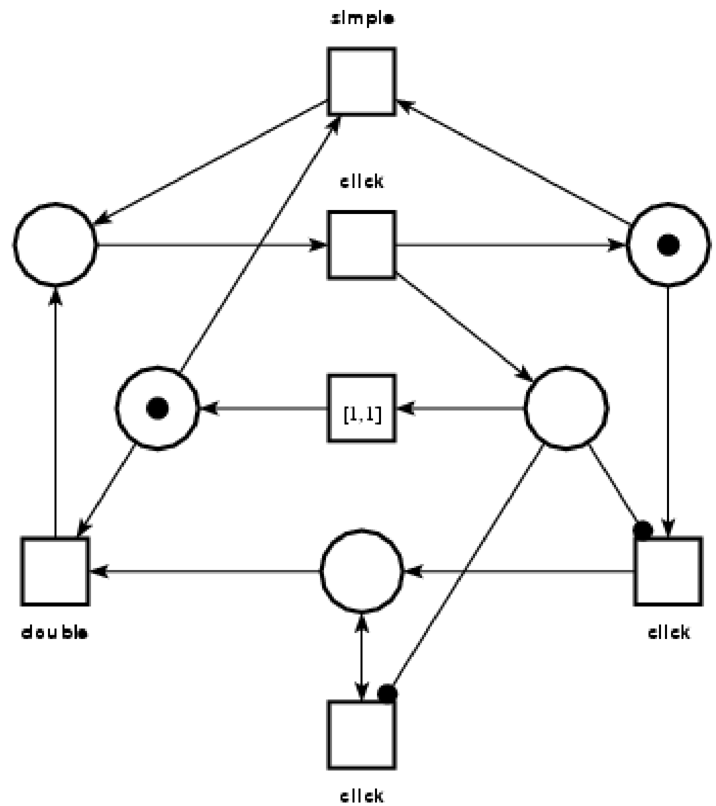


Not quite double click in TPN



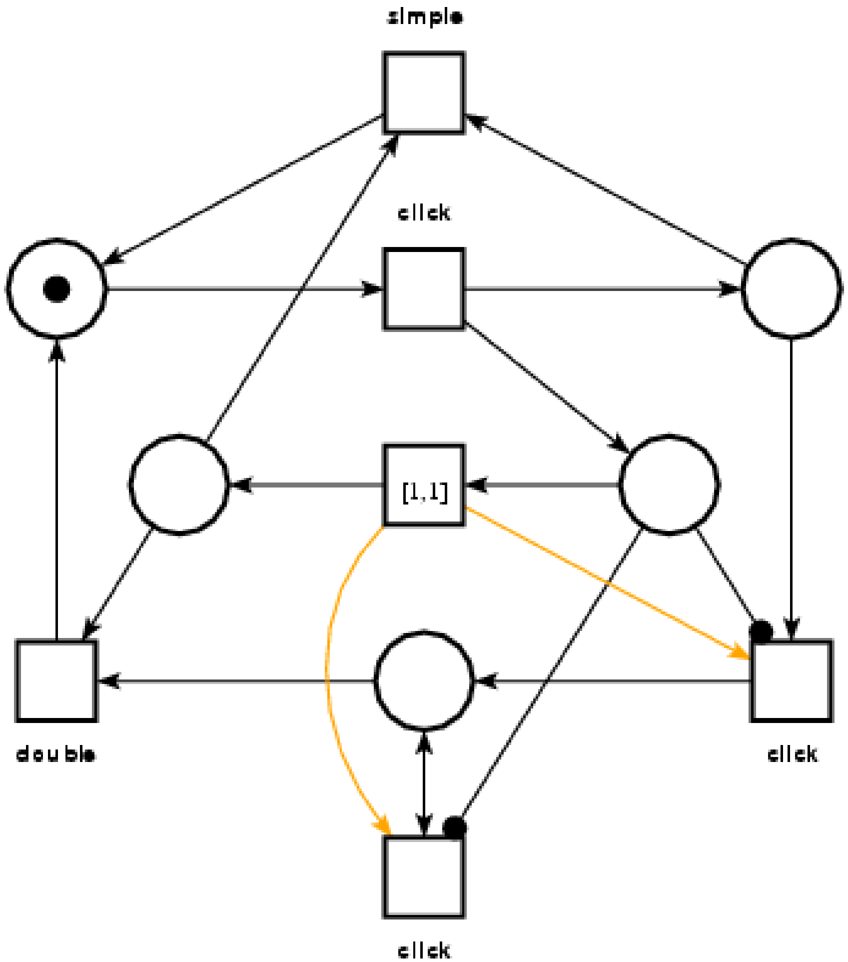
At time 1:





Incorrect: simple enabled

Double click in $PrTPN$

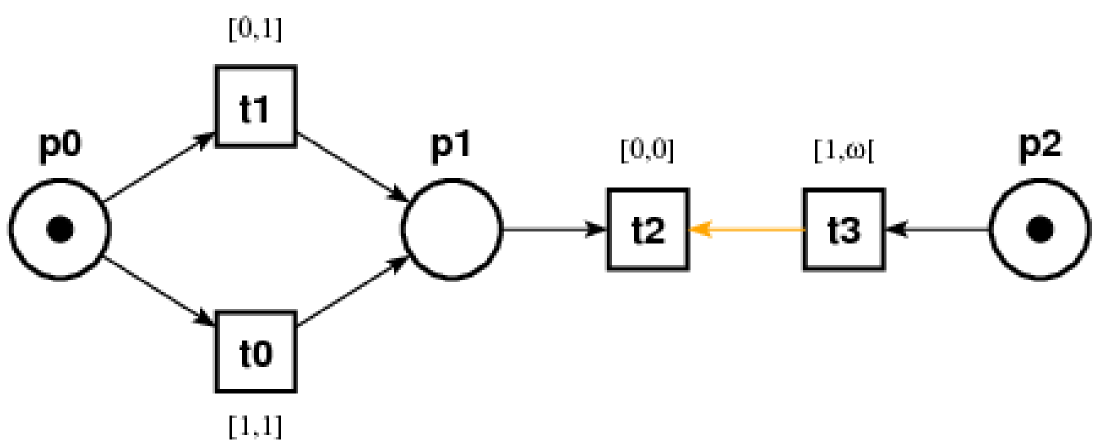


SCG and priorities

Founding observation for SCG:

Classes equivalent by $\cong$ 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 $C_\sigma = (m, Q)$:

- $C_\epsilon = (m_0, \{0 \leq \underline{\gamma}_t \leq 0 \mid \mathbf{Pre}(t) \leq m_0\})$

- t is firable from some state of C_σ iff:

  (i) $m \geq \mathbf{Pre}(t)$ (t is enabled at m)

  (ii) Q augmented with the following is consistent:

  $0 \leq \theta$

  $\downarrow I_s(t) \leq \underline{\gamma}_t + \theta$

  $\{\theta + \underline{\gamma}_i \leq \uparrow I_s(i) \mid m \geq \mathbf{Pre}(i)\}$

  $\{\theta + \underline{\gamma}_j < \uparrow I_s(j) \mid m \geq \mathbf{Pre}(j) \wedge j \succ t\}$

- If so, then $m' = m - \mathbf{Pre}(t) + \mathbf{Post}(t)$, and Q' is obtained by:

  1. add inequations (ii) to Q;

  2. $\forall i$ enabled at m', add $\underline{\gamma}'_i$ and inequations:

     $\underline{\gamma}'_i = \underline{\gamma}_i + \theta$, if $i \neq t$ and $m - \mathbf{Pre}(t) \geq \mathbf{Pre}(i)$

     $0 \leq \underline{\gamma}'_i \leq 0$, otherwise

  3. Eliminate variables $\underline{\gamma}$ and θ

Updated firability conditions

Firability conditions (ii) rephrased:

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

(ii.2) $\theta + \underline{\gamma}_t \in I_s(t)$

(ii.3) $(\forall i \neq t)(m \geq \mathbf{Pre}(i) \Rightarrow \theta + \underline{\gamma}_i \leq \uparrow I_s(i))$

(ii.4) $(\forall j)(m \geq \mathbf{Pre}(j) \wedge j \succ t \Rightarrow \theta + \underline{\gamma}_j \notin I_s(j))$

In (ii.4):

$$\theta + \underline{\gamma}_i \notin I_s(i) \Leftrightarrow \theta + \underline{\gamma}_j < \downarrow I_s(j) \vee \theta + \underline{\gamma}_j > \uparrow I_s(j)$$

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

$$\theta + \underline{\gamma}_i \notin I_s(i) \Leftrightarrow \theta + \underline{\gamma}_j < \downarrow I_s(j)$$

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]

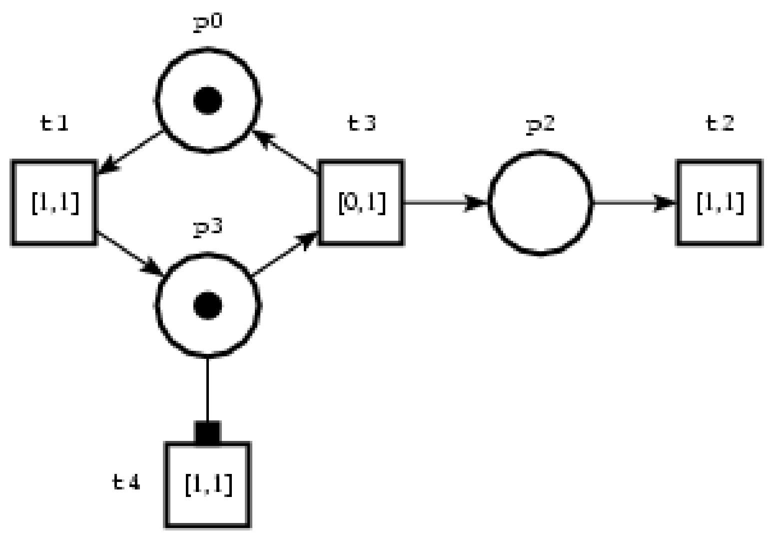
Stopwatch Time Petri Nets [BLRV07]

Time Petri nets with Stopwatches ($SwTPN$)

[BLRV07]

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

- $\langle P, T, \mathbf{Pre}, \mathbf{Post}, m_0 \rangle, \mathbf{I}^+$ is a Time Petri net
- $\mathbf{Sw}$ is the *Stopwatch incidence function*



An enabled transition is either Active or Suspended

Semantics

- Initial state: (m_0, Is_0)

- discrete transitions: $(m, I) \xrightarrow{t} (m', I')$ iff $t \in T$ and

  1. $m \geq \mathbf{Pre}(t) \wedge m \geq \mathbf{Sw}(t)$

  2. $0 \in I(t)$

  3. $m' = m - \mathbf{Pre}(t) + \mathbf{Post}(t)$

  4. $(\forall k \in T)(m' \geq \mathbf{Pre}(k) \Rightarrow$
     $I'(k) =$ **if** $k \neq t \ \wedge \ m - \mathbf{Pre}(t) \geq \mathbf{Pre}(k)$ **then** $I(k)$ **else** $Is(k))$

- continuous transitions: $(m, I) \xrightarrow{d} (m, I')$ iff

  $(\forall k \in T)(m \geq \mathbf{Pre}(k) \Rightarrow$
  $d \leq\uparrow I(k) \wedge I'(k) = \mathbf{if}\ m \geq \mathbf{Sw}(k)\ \mathbf{then}\ I(k) \dot{-} d\ \mathbf{else}\ I(k))$

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

Undecidability [BPV07]

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

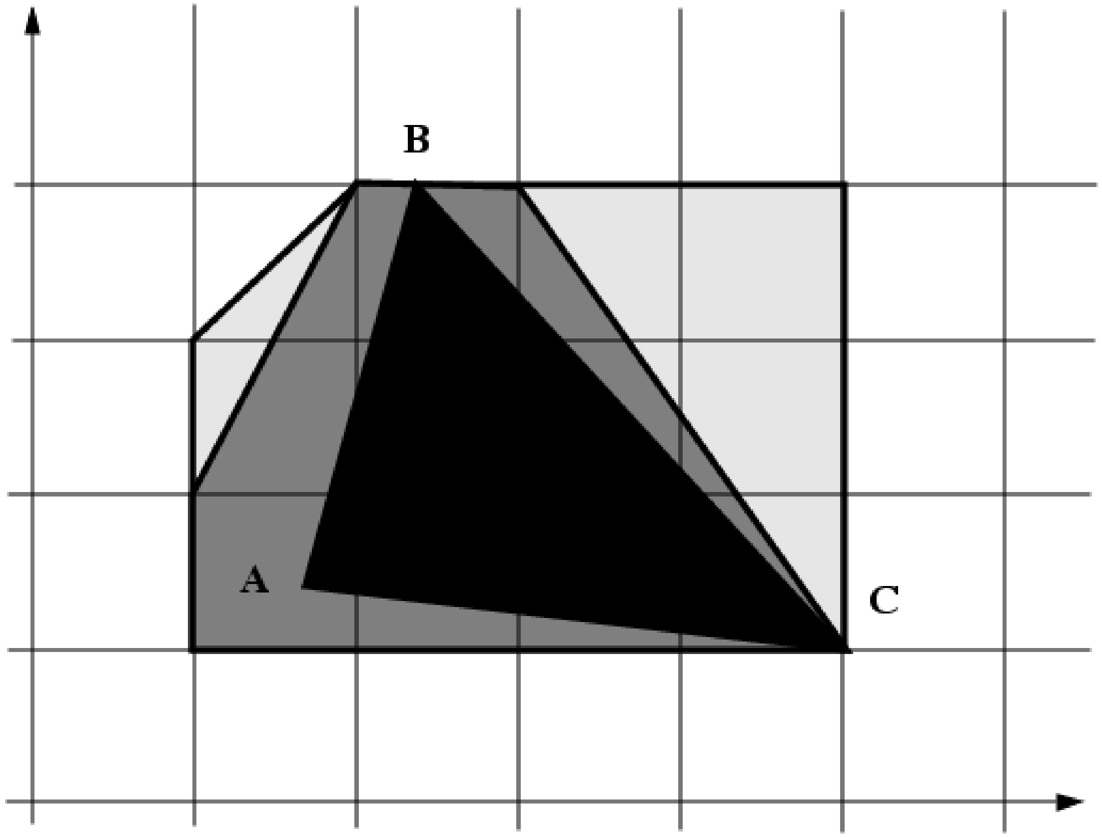
A single transition with non singular interval

Hence:

State/marking reachability undecidable for bounded SwTPN

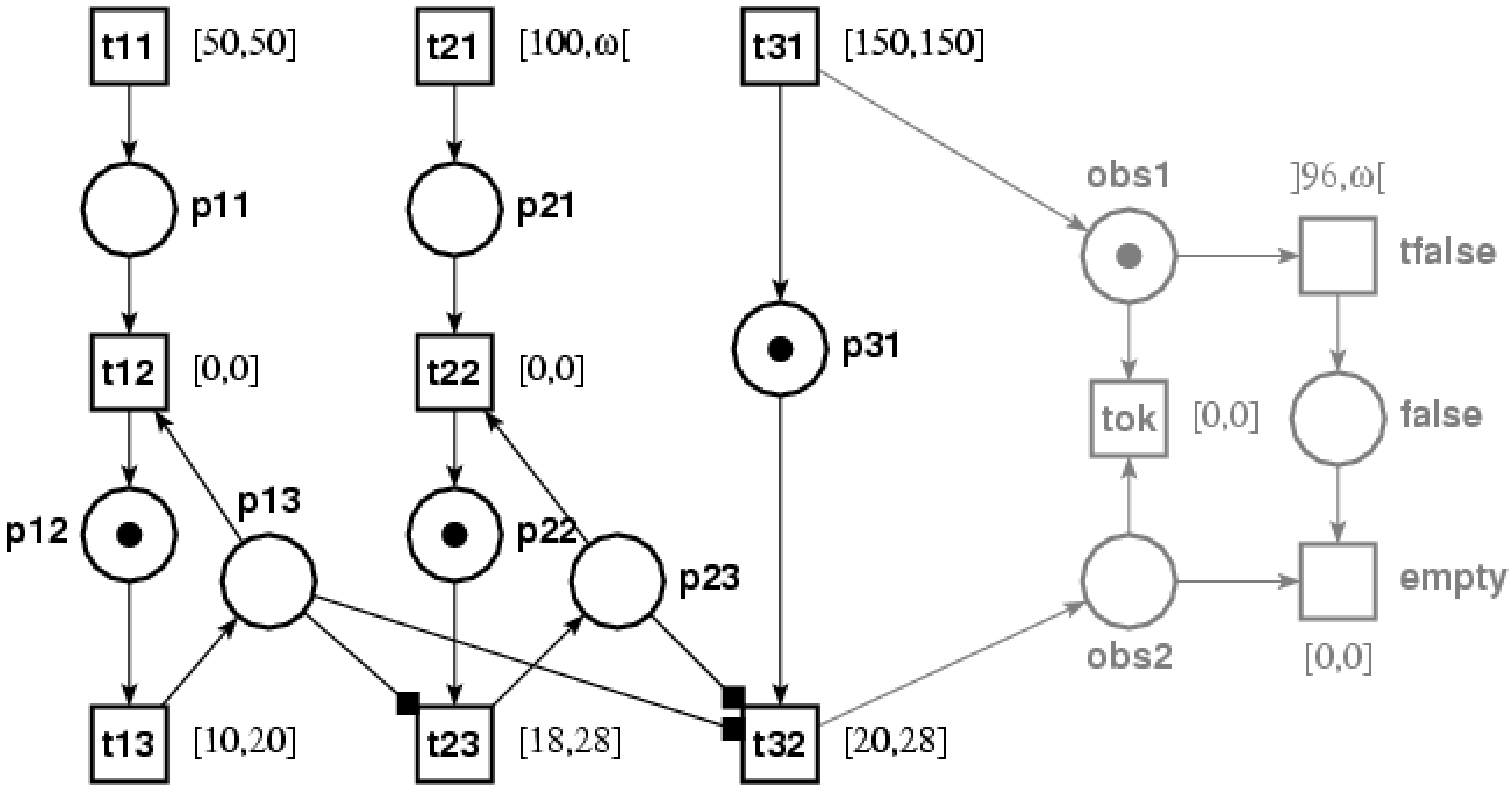
k-boundedness undecidable for SwTPN

Overapproximations



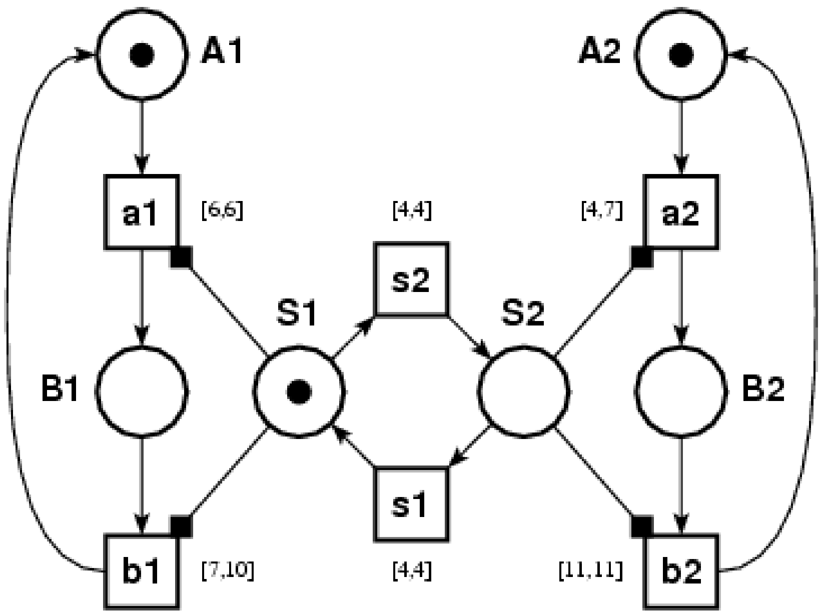
exact polyhedra $\subseteq$ quantized polyhedra $\subseteq$ smallest enclosing DBM

Example, task system [BFSV04]

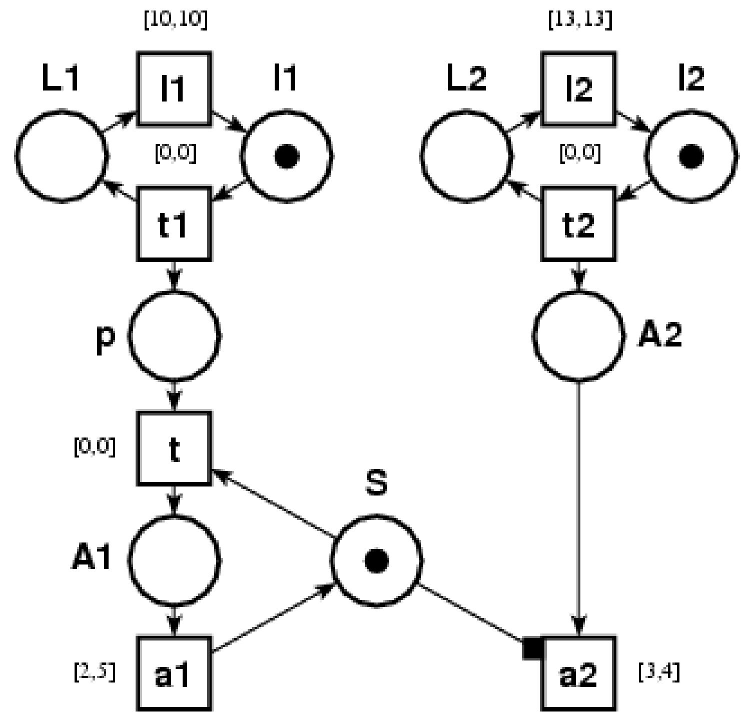


(observer in grey for the property "task 3 achieved in $\leq$ 96s")

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`)

Avionics software

Cotre language

TOPCASED project (`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

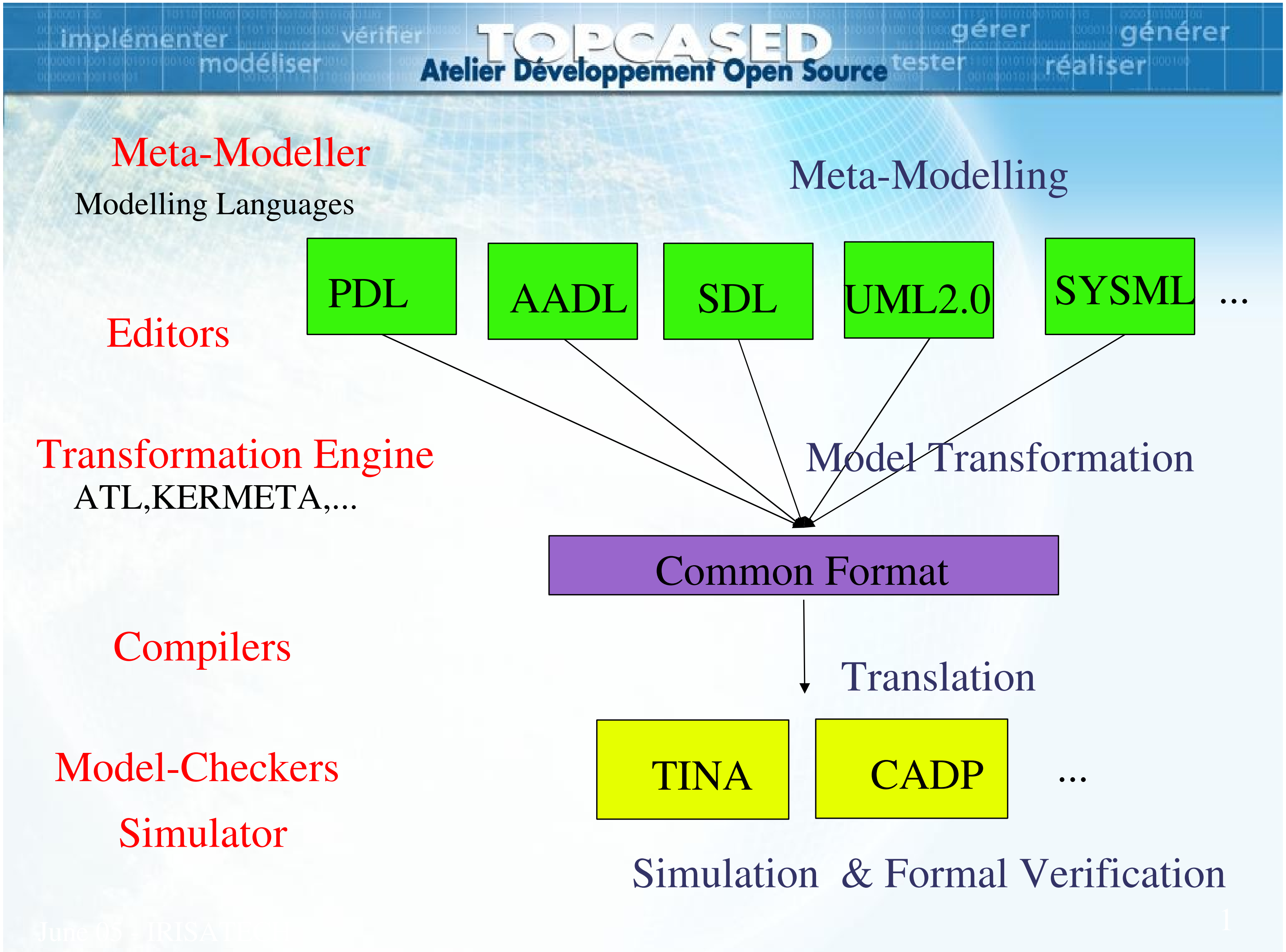
Hardware systems

8.2. Tools

Some tools using state classes

The TINA toolbox

Topcased Project



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 $\Rightarrow$ geometric overapproximations

Handling Data

High level description languages $\Rightarrow$ discrete overapproximations

Main components

tina (**TI**me petri **N**et **A**nalyzer)

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

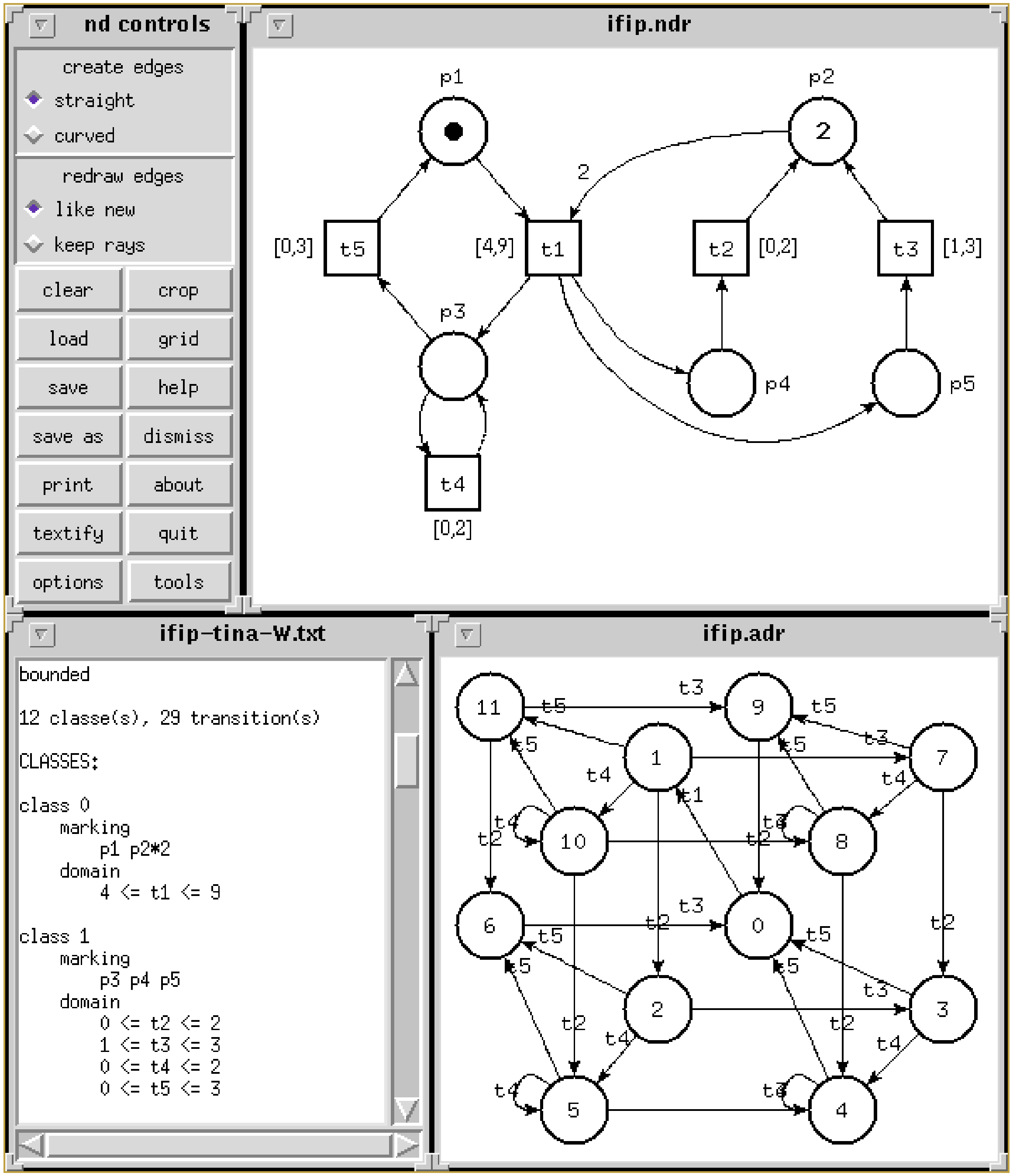
Drawing, printing functions

Interfaced with tina tool and selt model-checker

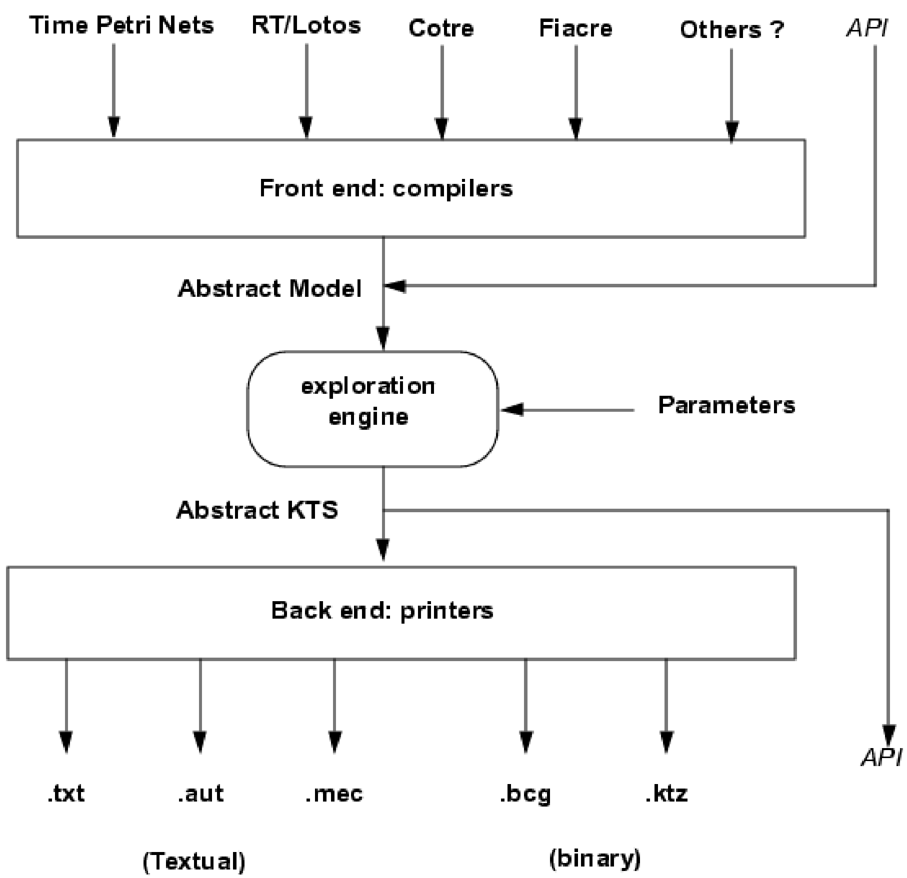
struct, plan, setl, muse, ktzio, ndrio, ...

Structural analysis, path analysis, SE-LTL model-checker, converters . . .

nd


nd controls
create edges
straight
curved
redraw edges
like new
keep rays
clear
crop
load
grid
save
help
save as
dismiss
print
about
textify
quit
options
tools
ifip.ndr
p1
p2
2
2
[0,3]
t5
[4,9]
t1
t2
[0,2]
t3
[1,3]
p3
p4
p5
t4
[0,2]
ifip-tina-W.txt
bounded
12 classe(s), 29 transition(s)
CLASSES:
class 0
marking
p1 p2*2
domain
4 <= t1 <= 9
class 1
marking
p3 p4 p5
domain
0 <= t2 <= 2
1 <= t3 <= 3
0 <= t4 <= 2
0 <= t5 <= 3
ifip.adr

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

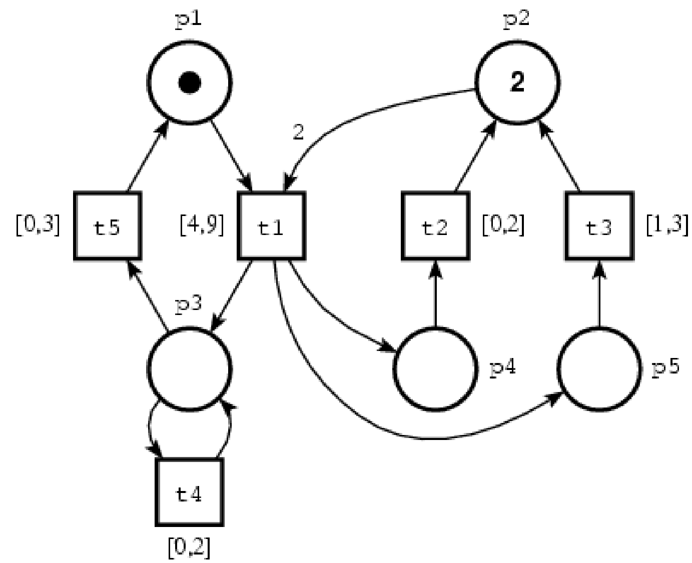
Stubborn sets

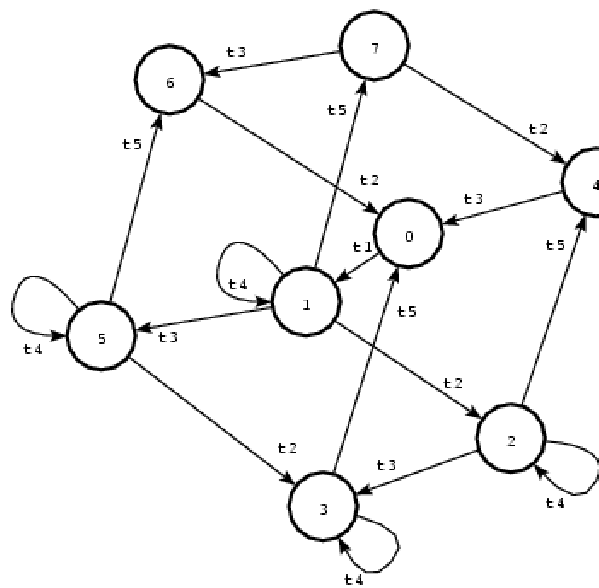
Stubborn steps

KTS, example

Marking graph

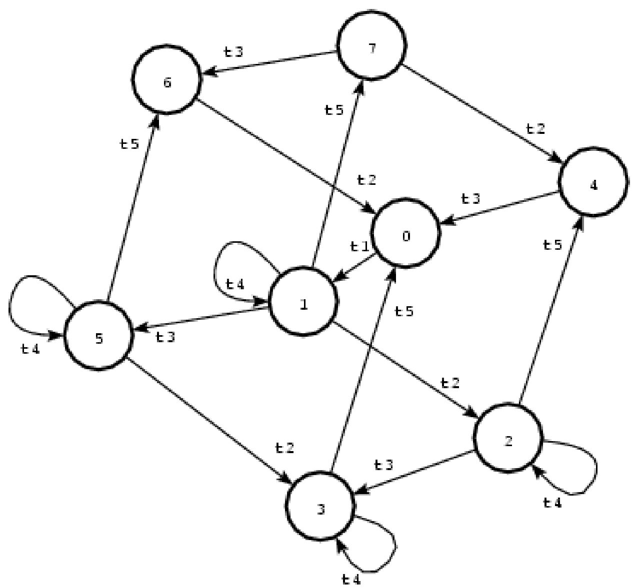
```
MARKINGS:
0 : p1 p2*2
1 : p3 p4 p5
2 : p2 p3 p5
3 : p2*2 p3
4 : p1 p2 p5
5 : p2 p3 p4
6 : p1 p2 p4
7 : p1 p4 p5

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 ($SCG_{\subseteq}$)

Preserving markings and LTL properties (SCG)

Multi-enabledness SCG

Preserving states ($SSCG_{\subseteq}$)

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

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[BCHRL05] B. Bérard, F. Cassez, S. Haddad, O. H. Roux, and D. Lime. Comparison of the Expressiveness of Timed Automata and Time Petri Nets. In *Formal Modeling and Analysis of Timed Systems (FORMATS'05), LNCS 3829*, pages 211–225, 2005.

[BM82] B. Berthomieu, M. Menasche, A State Enumeration Approach for Analyzing Time Petri Nets, *3rd European Workshop on Petri Nets*, Varenna, Italy, 1982.

[BM83] B. Berthomieu, M. Menasche, An Enumerative Approach for Analyzing Time Petri Nets, *IFIP Congress 1983*, Paris, France, 1983.

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

[BRV04] B. Berthomieu, P-O. Ribet, F. Vernadat, The tool TINA – Construction of Abstract State Spaces for Petri Nets and Time Petri Nets, *International Journal of Production Research*, Vol 42, Number 14, July 2004

[BLRV07] B. Berthomieu, D. Lime, O. H. Roux, F. Vernadat, Reachability Problems and Abstract State Spaces for Time Petri Nets with Stopwatches, *Journal of Discrete Event Dynamic Systems*, 2007.

[BPV06] B. Berthomieu, F. Peres, F. Vernadat, Bridging the gap between Timed Automata and Bounded Time Petri Nets, FORMATS 2006. Springer LNCS 4202, 2006

[BPV07] B. Berthomieu, F. Peres, F. Vernadat, Model-checking Bounded Prioriterized Time Petri Nets, ATVA 2007. Springer LNCS 4762, 2007

Some references ...

[BH04 H. Boucheneb and R. Hadjidj. Towards optimal CTL^* model checking of time Petri nets. *Proceedings of 7th Workshop on Discrete Events Systems*, Reims, France, September 2004.

[BM03] H. Boucheneb and J. Mullins. Analyse des réseaux temporels : Calcul des classes en o(n[2]) et des temps de chemin en o(mn). *Technique et Science Informatiques*, 22:435–459, 2003.

[BHR06] P. Bouyer, S. Haddad, and P-A. Reynier. Extended timed automata and time Petri nets. In *Proc. of 6th International Conference on Application of Concurrency to System Design (ACSD'06)*, Turku, Finland, June 2006. IEEE Computer Society Press.

[BR06] M. Boyer and O. H. Roux. Comparison of the expressiveness of arc, place and transition time Petri nets. *Application and Theory of Petri Nets 2007*, Siedlce, Poland, Springer LNCS 4546.

[BFSV04] G. Bucci, A. Fedeli, L. Sassoli, and E. Vicario. Timed State Space Analysis of Real-Time Preemptive Systems. *IEEE Transactions on Software Engineering*, 30(2):97–111, February 2004.

[CR06] F. Cassez and O. H. Roux. Structural translation from time petri nets to timed automata. *Journal of Systems and Software*, 2006.

[Ha06] R. Hadjidj. *Analyse et validation formelle des systèmes temps réel*. PhD Thesis, Ecole Polytechnique de Montréal, Université de Montréal, February 2006.

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

Some references ...

[LR03] D. Lime and O. H. Roux. Expressiveness and analysis of scheduling extended time Petri nets. *5th IFAC International Conference on Fieldbus Systems and their Applications*. Elsevier Science, July 2003.

[Me74] P. M. Merlin. *A Study of the Recoverability of Computing Systems.* PhD Thesis, Univ. of California, Irvine, 1974.

[MF76] P. M. Merlin and D. J. Farber. Recoverability of communication protocols: Implications of a theoretical study. *IEEE Tr. Comm.*, 24(9):1036–1043, Sept. 1976.

[PP04] W. Penczek and A. Półrola. Specification and Model Checking of Temporal Properties in Time Petri Nets and Timed Automata. *Applications and Theory of Petri Nets 2004*, Bologna, Italy, Springer LNCS 3099.

[RL04] O. (H.) Roux and Didier Lime. Time Petri nets with inhibitor hyperarcs. Formal semantics and state space computation. *Application and Theory of Petri Nets 2004*, Bologna, Italy, Springer LNCS 3099

[Ro93] T. G. Rokicki. *Representing and Modeling Circuits.* PhD Thesis, Stan-. ford Univ., Stanford, CA, 1993

[RM94] T. Rokicki, C. Myers, Automatic Verification of Timed Circuits. *6th Conference Computer Aided Verification, CAV'94, Springer LNCS 818*

[Vi01] E. Vicario. Static Analysis and Dynamic Steering of Time-Dependent Systems. *IEEE Transactions on Software Engineering*, 27(8):728–748, August 2001.

[YR98] T. Yoneda and H. Ryuba. CTL model checking of Time Petri nets using geometric regions. *IEEE Transactions on Information and Systems*, E99-D(3):1–10, 1998.